Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ragnar: Ferramenta para Pentest em dispositivos da Internet das Coisas

Cristoffer Leite da Silva

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. João José Costa Gondim

Brasília
2017

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ragnar: Ferramenta para Pentest em dispositivos da Internet das Coisas

Cristoffer Leite da Silva

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. João José Costa Gondim (Orientador)
CIC/UnB

Prof. Dr. Robson de Oliveira Albuquerque
ENE/UnB

Dr. Dino Macedo do Amaral
Banco do Brasil

Prof. Dr. Rodrigo Bonifácio de Almeida
Coordenador do Bacharelado em Ciência da Computação

Brasília, 08 de dezembro de 2017

Dedicatória

Dedico este trabalho a mim mesmo, por todo meu esforço e pela minha perseverança. Só eu sei como foi chegar aqui e, por isso, valorizo o meu empenho.

Agradecimentos

Agradeço ao meu orientador, João Gondim, por ter me orientando durante os estudos e por ter me apresentado a área de segurança;

Ao professor Remis Balaniuk, por ter me instruído a buscar a ciência e, por consequência, a ir para a Universidade de Brasília;

À minha sobrinha, Isabella Maria, por ser sempre o meu elo com coisas mais simples e importantes do mundo;

À minha irmã mais velha, Ketheleen Leite, pelo apoio e pelo companheirismo nos momentos difíceis;

À Confraria, por ter sido meu lar nesses últimos dois anos, pelo acolhimento e pela convivência;

À UnB, pela formação dada, pelos professores que me guiaram e pelos conteúdos disponibizados para realização desse trabalho;

Ao meu amigo, João Versiani, pelo auxílio na correção deste trabalho e pelos momentos de amizade proporcionados nesse último ano;

A todos os demais amigos e amigas, pelo estímulo, paciência e companheirismo durante esse período.

Resumo

A Internet das Coisas é tida como o novo paradigma dos sistemas de computação. Ela provê uma maior integração entre dispositivos e usuários e facilita a obtenção de dados para diversas áreas. Com a premissa de que tudo pode ser conectado, ela possibilita serviços anteriormente difíceis de serem fornecidos e permite um avanço na conexão entre dispositivos. Porém, a presença desses dispositivos interconectados na extração de informações críticas ou pessoais acaba sendo constante e torna-se necessário garantir a segurança implementada para evitar vazamentos de dados. Esse este estudo traz, portanto, uma análise dos maiores problemas de segurança relatados para esse paradigma e propõe uma ferramenta que realiza testes de intrusão com o objetivo de assegurar que vulnerabilidades estão sendo tratadas. Espera-se que a ferramenta atenda à demanda por testes mais precisos e que seja continuamente atualizada conforme os cenários sejam alterados.

Palavras-chave: pentest, internet das coisas, ferramenta, iot

Abstract

The Internet of Things is regarded as the new paradigm of computer systems. It provides a better integration between devices and users and makes it easier to obtain data for several areas. With the premise that everything can be connected, it enables previously difficult services to be provided and allows a breakthrough in the connection between devices. However, the presence of these interconnected devices in the extraction of critical or personal information ends up being constant and it becomes necessary to guarantee the security implemented to avoid data leakage. This study therefore provides an analysis of the major security problems reported for this paradigm and proposes a tool that performs penetration tests with the purpose of ensuring that vulnerabilities are being addressed. The tool is expected to meet the demand for more accurate testing and to continually update as the scenarios change.

Keywords: pentest, internet of things, tool, iot

Sumário

1 Introdução **1**
1.1 Motivação 2
1.2 Problema 2
1.3 Objetivos 3
1.3.1 Objetivo Geral 3
1.3.2 Objetivos Específicos 3
1.4 Organização do Trabalho 3

2 A Internet das Coisas **5**
2.1 Dispositivos IoT 5
2.1.1 Definição 6
2.2 IoT e Suas Visões 6
2.2.1 Orientada à Semântica 7
2.2.2 Orientada às Coisas 8
2.2.3 Orientada à Internet 8
2.3 Utilidade e Perspectivas 9
2.3.1 Perspectivas de Utilização 9

3 Segurança em IdC **10**
3.1 Privacidade e Segurança 10
3.2 O Caso da Botnet Mirai 11
3.3 Mapeamento de Componentes 12
3.3.1 Dispositivo 13
3.3.2 Software e Aplicações 13
3.3.3 Sistemas de Comunicação de Radiofrequência 13
3.4 Principais Projetos e Soluções Propostas para Segurança em IoT 14
3.4.1 OTA 14
3.4.2 I am the Cavalry 15
3.4.3 OWASP 16

3.5 Análise das Maiores Vulnerabilidades de IoT de Acordo com a Lista da OWASP . . . 17
3.5.1 Interface *Web* Insegura . . . 17
3.5.2 Autenticação e Autorização Insuficientes . . . 17
3.5.3 Serviços de Rede Inseguros . . . 18
3.5.4 Falta de Criptografia no Transporte . . . 18
3.5.5 Problemas de Privacidade . . . 19
3.5.6 Interfaces de Nuvem Inseguras . . . 19
3.5.7 Interfaces Móveis Inseguras . . . 20
3.5.8 Insuficiência na Configurabilidade de Segurança . . . 20
3.5.9 *Softwares/Firmwares* Inseguros . . . 20
3.5.10 Péssima Segurança Física . . . 21
3.6 Considerações Finais . . . 21

4 Ferramenta para Testes de Invasão 22
4.1 Mapeamento da Ferramenta . . . 22
4.1.1 Análise das Categorias e Subdivisão dos Módulos . . . 24
4.2 Estruturação e Desenvolvimento . . . 32
4.2.1 Linguagem Utilizada e Pré-requisitos . . . 33
4.2.2 Björn - Módulo de Interface *Web* . . . 34
4.2.3 Ubba - Módulo de Serviço de Rede . . . 40
4.2.4 Hvitserk - Módulo de Encriptação . . . 44
4.2.5 Ivar - Módulo de *Firmware* . . . 47
4.3 Considerações Finais . . . 48

5 Resultados Obtidos 50
5.1 Objetivos dos Testes . . . 50
5.2 Ambiente de Testes . . . 50
5.2.1 Configuração do Ambiente . . . 50
5.3 Teste de Detecção . . . 52
5.3.1 Interface *Web* . . . 53
5.3.2 Serviços de Rede . . . 54
5.3.3 Encriptação . . . 54
5.3.4 Análise de *Firmware* . . . 55
5.3.5 Resultados . . . 55
5.4 Considerações Finais . . . 56

6 Conclusão e Trabalhos Futuros 57

Referências **58**

Apêndice **62**

A Modelos de Redes de Computadores **63**

A.1 Modelo OSI 63

A.2 Funções das Camadas no Modelo OSI 63

A.2.1 Física 63

A.2.2 Enlace 63

A.2.3 Redes 64

A.2.4 Transporte 64

A.2.5 Sessão 64

A.2.6 Apresentação 64

A.2.7 Aplicação 64

A.3 Modelo TCP/IP 65

A.4 Funções das Camadas no Modelo TCP/IP 65

A.4.1 Enlace 65

A.4.2 *Internet* 65

A.4.3 Transporte 65

A.4.4 Aplicação 65

Lista de Figuras

2.1 IoT vista como a intersecção entre a visão orientada à Internet e às Coisas com a Semântica servindo de apoio [1]. 7

3.1 Arquitetura utilizada pela botnet Mirai para conseguir novos bots e realizar ataques através deles. 11
3.2 Crescimento dos scans de telnet realizados pela botnet Mirai em comparação com scans do mesmo tipo realizados por outras redes [2]. 12

4.1 Informações de Conexão Fornecidas pela Ferramenta. 34
4.2 Verificação de HTTPS e ClickJacking. 36
4.3 *Scan* de Caminhos e Páginas Comuns de Autenticação. 37
4.4 Teste de Injeção SQL Indicando Campos e Caminhos Testados. 38
4.5 Resultado de um Teste de XSS Realizado na Interface, com o Detalhamento dos Caminhos. 39
4.6 Resultado da Verificação de Portas e da Análise dos Servidor. 41
4.7 Testes de Resistência a DDoS Utilizando HTTP Flood e SYN Flood. . . . 43
4.8 Exemplo de teste de *Fuzzing*. 44
4.9 Lista de Protocolos Utilizados e Análise do Protocolo Padrão do Servidor. . 45
4.10 Resultado de uma Análise do uso de PFS. 46
4.11 Resultado do Teste de Cifragem Fraca. 46
4.12 Exemplo da Primeira Análise Mostrando o Tipo do Arquivo e a Validação da Assinatura MD5. 47
4.13 Conteúdo Gerado em um *Hexdump* nos Testes. 48

Lista de Tabelas

4.1 Guia de Testes da OWASP para Dispositivos IoT 23
4.2 Divisão de Testes do Módulo de Interface *Web* (Björn) 25
4.3 Divisão de Testes de Autenticação 26
4.4 Divisão de Testes do Módulo de Serviços de Rede (Ubba) 27
4.5 Divisão de Testes do Módulo de Encriptação (Hvitserk) 28
4.6 Divisão de Testes de Interface em Nuvem 29
4.7 Disivão de Testes de Interface Móvel 30
4.8 Divisão de Testes de Configurabilidade 31
4.9 Divisão dos Testes do Módulo de *Firmware* (Ivar) 31
4.10 Divisão de Testes de Segurança Física 32
4.11 Divisão da Ferramenta para Testes de Invasão 33

5.1 Configuração dos Computadores Utilizados nos Testes 51
5.2 Resultados dos Testes Executados em cada Cenário 56
5.3 Resultados dos Testes de *Firmware* 56

Capítulo 1

Introdução

> "Nós apenas podemos ver a uma curta distância, mas já vemos o bastante que precisa para ser feito."
>
> Alan Turing

Internet das Coisas (do inglês, Internet of Things) ou também IoT, é um novo paradigma que está crescendo bastante no ambiente computacional contemporâneo. A definição de IoT depende de qual objetivo se quer atingir com sua implementação, podendo englobar uma série de dispositivos que possuem capacidade de conexão e poder computacional para realizar tarefas bem como outros que apenas interagem através de estímulos externos e que não possuem capacidade alguma de processamento.

Dispositivos IoT surgem todo dia com propostas de automação de tarefas ou como um aprimoramento de algum objeto de uso rotineiro e, com isso, ganham espaço nos lares e em empresas que querem simplificar algum processo ou ter informações em tempo real desses objetos. Conforme novos dispositivos são lançados, surgem com eles novos problemas relativos à heterogeneidade, à padronização e principalmente, à segurança aplicada neles.

Apesar da grande adoção desse novo paradigma, pouca atenção tem sido dada à segurança, permitindo que brechas sejam utilizadas para realizar invasões e tomar posse de sistemas completos. Casos de ataques utilizando a infraestrutura fornecida por esses aparelhos já estão surgindo e são preocupantes.

Há uma série de ferramentas que auxiliam na realização de testes de intrusão, também conhecidos como Pentest (do inglês, *penetration test*), porém são poucas que possuem a IoT como foco, o que faz com que os testes tenham que ser feitos utilizando adaptações e uma série de ferramentas para funcionalidades pequenas.

Diante do exposto, a proposta deste trabalho é o desenvolvimento de uma ferramenta que auxilie na proteção de ambientes com dispositivos IoT, realizando os testes de intrusão mais comuns, de forma a diminuir a necessidade de uma gama de ferramentas auxiliares. Dessa forma, passa-se a ter um ambiente de testes mais coeso, pois as análises realizadas são orientadas para aparelhos da Internet das Coisas. Além disso, elimina-se a maioria das limitações geradas pelo uso de ferramentas que não foram desenvolvidas tendo a IoT como foco.

1.1 Motivação

Atualmente há ferramentas que podem ser utilidas para realizar a análise da segurança desses dispositivos. Algumas delas são: Nmap [1], TestSSL[2] e Binwalk[3], que fornecem desde alguns testes mais complexos como a análise de portas e verificação da segurança aplicada nos serviços fornecidos, até tarefas mais específicas como extração e análise de dados de um arquivo binário, cada qual com vantagens e desvantagens em determinados tipos de verificações.

Porém, não há uma ferramenta que centralize os testes de vulnerabilidades mais comuns específicos de dispostivos IoT. Diante deste contexto, a motivação deste trabalho é implementar uma ferramenta de testes de intrusão para dispositivos da Internet das Coisas, tendo como base os problemas identificados como os mais comuns.

1.2 Problema

Atualmente, para realizar os testes de segurança mais simples em dispositivos IoT, é necessário uma gama de ferramentas que disponibilizam dados sem que as análises tenham pensadas e orientadas a esses aparelhos. Isto é ineficiente porque, da maneira como são conduzidos os testes atualmente, é necesário um esforço enorme para coletar uma quantidade de informações pequenas. Isso significa que nesse momento ainda não há uma ferramenta que disponibilize informações suficientes para que se possa realizar testes de uma forma precisa e sem recorrer a diversos outros métodos de análise. Além disso, com esta abordagem, existe uma dispersão dos dados coletados e uma falta de coesão nos relatórios gerados. Com o tipo de ferramenta de análise atual, a única maneira de testar determinadas vulnerabilidades nesses ambientes é gerando testes específicos com cada ferramenta e compilando relatórios que são obtidos.

[1] https://nmap.org/
[2] https://testssl.sh
[3] https://github.com/ReFirmLabs/binwalk

1.3 Objetivos

1.3.1 Objetivo Geral

Este projeto tem como objetivo implementar uma ferramenta de testes de intrusão em dispositivos da Internet das Coisas que, fazendo uso de uma lista com os maiores problemas relatados em estudos sobre o tema, tenha análises específicas e ofereça resultados mais precisos.

1.3.2 Objetivos Específicos

Para que o objetivo geral seja cumprido, faz-se necessário atingir os seguintes objetivos específicos:

- Determinar os maiores problemas de segurança em Dispositivos IoT atualmente;
- Dominar cada etapa dos ciclos de ataque;
- Definir os métodos de análise utilizados para se encontrar esses problemas;
- Analisar quais métodos são passíveis de automatização;
- Implementar a ferramenta com base nas análises feitas;
- Realizar experimentos para avaliar a ferramenta;
- Comparar com as ferramentas disponíveis hoje.

1.4 Organização do Trabalho

Este trabalho foi dividido em mais cinco capítulos, além deste introdutório. No Capítulo 2 são apresentadas as definições e os conceitos fundamentais para a compreensão do que é a Internet das Coisas, suas visões, que dispostivos se enquadram nesse conceito e a utilização de acordo com a perspectiva analisada.

O Capítulo 3 apresenta as questões de segurança envolvidas na adoção desse paradigma e os problemas de privacidade e segurança envolvidos, fazendo um mapeamento básico das superfícies e mostrando os principais projetos de segurança dessa área. Em seguida, é feita uma análise do projeto escolhido para servir de base no desenvolvimento da ferramenta, categorizando os problemas apresentados pela instituição e a maneira indicada de tratamento.

No Capítulo 4 é mostrado o processo de mapeamento da ferramenta, com a definição dos módulos e das vulnerabilidades tratadas. Em seguida, é detalhada a estruturação

e o desenvolvimento das funcionalidades, mostrando os benefícios do uso da linguagem escolhida, bem como dos pacotes auxiliares. Por fim, é detalhado cada módulo e a forma que ele faz a verificação de cada problema apontado.

No Capítulo 5 são apresentados os resultados obtidos nos testes realizados com a ferramenta de testes de vulnerabilidades Ragnar. O objetivo destes testes é comprovar o funcionamento da ferramenta proposta e verificar se eles condizem com o que é obtido em outras ferramentas já estabelecidas.

Por fim, o Capítulo 6 apresenta as conclusões obtidas da implementação e teste da ferramenta e propõe trabalhos futuros, que podem expandir o que foi proposto neste trabalho.

Capítulo 2

A Internet das Coisas

O objetivo deste capítulo é apresentar e definir o ambiente onde se situa a Internet das Coisas (IoT). A Seção 2.1 contém uma explanação sobre o que são dispositivos IoT. Em seguida, na seção 2.2, serão apresentadas as visões conceituais da Internet das Coisas e a classificação utilizada para definir objetos inteligentes. Por último, na seção 2.3, será dada uma breve explicação das perspectivas de utilização desses objetos.

2.1 Dispositivos IoT

A expressão Internet das Coisas pode ser um tanto vaga e deixar confuso o real significado por trás do que ela representa, por isso é necessário definir o que é e para que serve um dispositivo desse tipo. Segundo Kevin Ashton, cofundador do Auto-ID Center no Massachusetts Institute of Technology (MIT) e que diz ser o primeiro a utilizar o termo Internet das Coisas em 1999 numa apresentação para a Procter & Gambler [3], IoT pode ser descrito como a ligação entre dispositivos com capacidade de Identificação por radiofrequência ou RFID (do inglês "Radio-Frequency IDentification") em uma cadeia logística e a internet. Ashton descrevia o caso da empresa para qual ele palestrava, mas acabou dando os primeiros detalhes do que eventualmente seria esse novo paradigma.

Desde a criação da internet até o dia em que a apresentação de Ashton ocorreu, o ser humano ainda era o único responsável por toda a informação que havia sido inserida na rede global, sendo o único responsável por gerar conteúdo para a web. Só após alguns anos que aparelhos passaram a coletar dados, analisá-los e enviá-los entre si, sendo que, conforme novas funcionalidades iam sendo incluídas, os dispositivos ficavam cada vez mais autossuficientes e passavam a ser intitulados como *smart-objects*. Tornou-se necessária, portanto, a conceituação do que é IoT e o que é preciso para que um objeto seja considerado inteligente.

2.1.1 Definição

Segundo [4], sistemas IoT são comumente definidos a partir de algo que possui um identificador único e que pode ser acessado de qualquer lugar e a qualquer hora, sendo que o nível de informação obtida ao se conectar ao objeto pode ser tão baixo quanto qualquer dado estático que é armazenado em RFIDs. Porém, para ser considerado integrante da Internet das Coisas no escopo deste trabalho, será exigido que o dispositivo tenha também uma capacidade própria de processamento, tornando-o um objeto inteligente.

Miorandi et. al [5] definem objetos inteligentes como entidades que: têm um corpo físico, têm capacidade de comunicação, possuem ID único, são associados a pelo menos um nome ou um endereço, possuem capacidades computacionais básicas e têm meios de detectar algum fenômeno físico. Objetos inteligentes são as coisas da Internet das Coisas.

Assim, dispositivos IoT são objetos que obtêm, processam e fornecem dados, seja por um meio físico ou não, em conexões com usuários ou com outras entidades de uma rede.

2.2 IoT e Suas Visões

A Internet das Coisas cria um ambiente novo de interação humana com a tecnologia, possibilitando inovadoras formas de diversão, comunicação e convívio social. Tudo isso se dá por um sistema de dispositivos inteligentes conectados a uma infraestrutura e comunicando entre si. Ela não acaba com a ideia da internet como conhecemos hoje em dia, mas incrementa essa última com diversos novos componentes integrados, mudando o foco de conexão entre usuários para conexão entre objetos e usuários. Assim, de um ponto conceitual, ela faz com que ambos se tornem atores que geram e consomem dados do mundo físico.

IoT é sobre entidades agindo como provedores e consumidores de dados relativos ao ambiente em que estão inseridos. O foco principal deve ser nos dados ao invés da comunicação ponto a ponto, por isso as *coisas* acabam sendo o ponto-chave de qualquer sistema IoT. São elas que produzem o conteúdo do sistema e que geralmente fazem a primeira análise dos dados coletados. Para entender mais o que são as "coisas" deve-se primeiro analisar as visões de IoT.

Segundo Luigi Atzori [1], a Internet das Coisas pode ser ser dividida em três visões: Orientada à Internet, Orientada às Coisas e Semântica. Cada uma expressa o conceito do que é IoT de uma perspectiva diferente e provê uma forma única de estudar a aplicabilidade das tecnologias disponíveis, além de servir de base para o desenvolvimento de novos meios de utilização. A junção das *coisas* com a internet representa a parte central da IoT, porém a semântica é necessária para dar sentido e um contexto aos embientes conforme mostrado na Figura 2.1 [1].

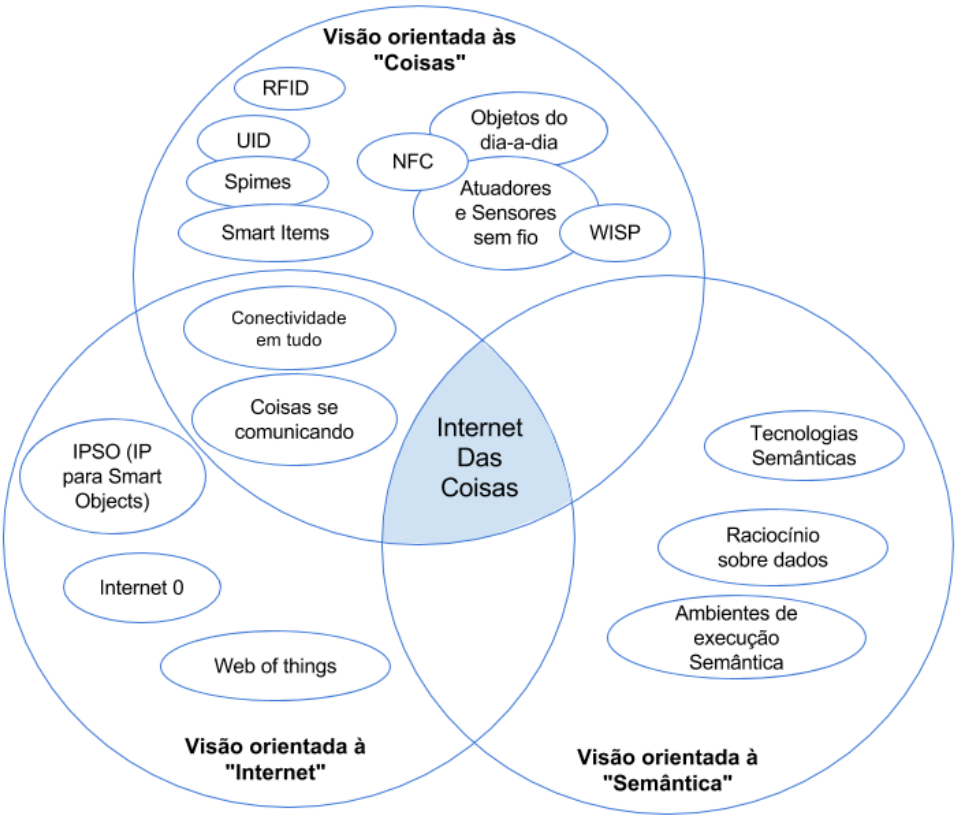

Figura 2.1: IoT vista como a intersecção entre a visão orientada à Internet e às Coisas com a Semântica servindo de apoio [1].

2.2.1 Orientada à Semântica

Essa visão é apresentada como a composição dos desafios de interconexão e organização da informação gerada pela em sistemas de IoT [6]. Conforme os ambientes vão se tornando mais robustos, a necessidade de processamento se torna maior, e como na Internet das Coisas há inúmeras fontes de informação numa mesma rede, é necessário que todos esses dados coletados sejam armazenados e transmitidos sem que isso afete a capacidade de processamento geral.

Com a escalabilidade, surgem problemas ao armazenar o conteúdo resultante desse processo. Informações como tipo de dispositivo, características físicas e localização do objeto são alguns dos dados a serem manipulados. E para solucionar esse tipo de problema há sistemas semânticos tais como a linguagem de dados baseada em gráficos chamada RDF (em inglês, *Resource Description Framework*), que permite a associação de dados e descrições de propriedades e relações entre objetos, possibilitando buscas inteligentes de objetos com características em comum [7].

2.2.2 Orientada às Coisas

Tida como a visão central do IoT, já que um futuro próximo é previsto que as "coisas" passem a gerar mais conteúdo que os próprios usuários [8]. É o que faz com que a Internet das Coisas seja a arquitetura natural para permitir a implementação de sistemas conjuntos e aplicações independentes, pois qualquer objeto com um mínimo de capacidade computacional pode ser integrado ao ambiente e, assim, passar a gerar e analisar conteúdo.

Como coisa deve-se considerar qualquer item que seja passível de identificabilidade e endereçabilidade na rede, porém IoT implica uma visão bem maior que somente identificação de objetos [1]. Dispositivos de identificação por radiofrequência, objetos com comunicação por campo de proximidade (NFC) e o grupo de sensores identificados como WSAN, compõem o grupo central de componentes que ligam o mundo real com o mundo digital [9]. Eles são o elo entre as coisas e o mundo físico.

E para além da conectividade, ainda há a prospecção de que os objetos passarão a ter um histórico e que comecem a ser analisados não só de forma estática, mas como um ente que evolui conforme absorve informações do ambiente ao seu redor, gerando uma linha do tempo através de uma aproximação mais proativa do dispositivo. E que assim começarão a ter identidades e personalidades que operam em ambientes inteligentes enquanto se conectam e comunicam com contextos diversos.

2.2.3 Orientada à Internet

As aplicações e sistemas de IoT devem possuir alto grau de autonomia em captura de dados, transferência de eventos, conectividade e interoperabilidade. Essas características permitem a junção das *coisas* com a internet e tornam possível que o conteúdo adquirido seja analisado, trabalhado e aplicado efetivamente na rede.

Sendo assim, a visão de Internet está orientada mais para a rede. Porém, é necessário padronizar os métodos de comunicação utilizados, destacando-se nesse meio algumas iniciativas como a IPSO (IP for Smart Objects) [10] para promover o Protocolo de Internet (IP) como a tecnologia de rede utilizada para a conexão padrão de objetos ao redor do mundo. Um forte argumento é o fato de que o IP já se mostrou altamente capaz de realizar esse tipo de tarefa por ser estável e possuir uma escalabilidade já testada, além de ter evoluído bem durante os anos para se adaptar às novas tecnologias. Uma ideia similar é proposta pela Internet Ø com o "IP em tudo" [11]. As duas propostas propõem uma simplificação do IP como é utilizado atualmente, de forma a adaptá-lo à nova realidade da IoT e tornar possível o endereçamento de qualquer tipo de objeto.

2.3 Utilidade e Perspectivas

IoT proporciona novas formas de interação e, com isso, modifica ambientes de trabalho, permite a criação de novos meios de diversão e facilita o dia a dia através de automação de tarefas. Baseando-se na noção de objetos inteligentes interconectados formando ambientes computacionais comuns (difusivos), faz com que os espaços sejam modificados pela sua presença. E as possibilidades são diversas de tal forma que não há como prever todas as implementações que podem ser feitas com esses objetos. Mas é possível analisar as perspectivas de utilização deles, o que essas perspectivas permitem e quais são os problemas gerados por elas.

2.3.1 Perspectivas de Utilização

Segundo a Gartner[1] há estimativas de que, até 2020, algo em torno de 13,5 bilhões de novos dispositivos IoT estarão operando. Tudo isso se dará pela junção de máquinas intercomunicáveis (M2M) e serviços em nuvem integrados. O que resultará na interação de serviços domésticos, veículos automotores inteligentes e automação industrial a níveis altíssimos. Com esse panorama e toda a definição já apresentada podem ser apreendidas as perspectivas possíveis para as quais os dispositivos são úteis.

Perspectiva de Nível de Sistema

IoT pode ser visto como uma rede altamente dinâmica e radicalmente distribuida, composta de um número muito grande de objetos inteligentes produzindo e consumindo informação. Capacidades autônomas e autogerência são esperadas. Essas características são intrísecas ao software e à rede.

Perspectiva de Nível de Serviço

O maior problema é como integrar as funcionalidades e os recursos providos pelos objetos inteligentes e transformar isso em serviço.

Perspectiva de Nível de Usuário

Permitir novas formas de serviços cada vez mais responsivos às necessidades do usuário e dar suporte nas atividades diárias.

[1] Gartner. http://www.gartner.com/newsroom/id/3165317, acesso em 06/04/2017

Capítulo 3

Segurança em IdC

Com a rápida expansão do uso de dispositivos IoT, surgem questões sérias concernentes à segurança e privacidade dos usuários desses aparelhos. Ataques a esses equipamentos estão se tornando cada vez mais comuns e se aproveitam da baixa segurança fornecida. Conforme eles são inseridos em ambientes cotidianos para facilitar tarefas, a pouca atenção dada à proteção coloca em risco a integridade das informações que circulam naquele ambiente e deixa os usuários vulneráveis a ataques externos ou até mesmo internos, expondo dados dos mesmos sem que eles saibam.

Pouca ou nenhuma medida de segurança é tomada visando reduzir os custos e com o objetivo tornar esses sistemas mais portáteis, o que gera problemas gravíssimos quando inseridos no dia a dia de usuários comuns ou na cadeia de produção de grandes empresas. Há casos conhecidos em que a baixa segurança de dispositivos IoT permitiu ataques de larga escala sem que fosse necessário muito esforço. Esse tipo de problema tende a crescer mais ainda conforme novos sistemas vão sendo desenvolvidos e adotados.

3.1 Privacidade e Segurança

Os dois principais pontos a serem observados em questão de vulnerabilidade são a privacidade (acesso garantido, exclusivo e confidencial) e a segurança (confiabilidade e persistência) da informação transmitida. Em [12] os problemas de privacidade são definidos como "limitações de acesso de outros a um indivíduo". E Ross Anderson [13] se refere à engenharia de segurança como "construção de sistemas confiáveis ante a malícia, erros e o acaso". Ambos descrevem preocupações centrais do acesso a dispositivos na internet como um todo, mas em IoT esses problemas tendem a escalar exponencialmente devido à falta de zelo e importância dada a essas questões e ao fato de que esses dispositivos estão a todo momento coletando informações do ambiente ao seu redor ou dos usuários.

3.2 O Caso da Botnet Mirai

Um exemplo de como a baixa segurança em IoT pode gerar grandes perdas aconteceu em outubro de 2016 com um ataque conhecido como Mirai Botnet [14]. Um processo rodava continuamente em cada bot que já compunha a rede e através do protocolo telnet tentava logar em endereços IPs aleatórios. As tentativas eram feitas utilizando senhas padrões de fábrica (pares de login e senha) e quando obtinham sucesso no login, os dados dos pares de acesso junto com o IP eram enviados para uma central de comando e controle (CnC) que utilizava dos bots já existentes para logar na máquina informada e infectá-la com seu código. Após isso ela adicionava esses dados à base e transformava os dispositivos invadidos em novos bots, assim conseguiria orientar os ataques através dos novos e dos antigos vetores.

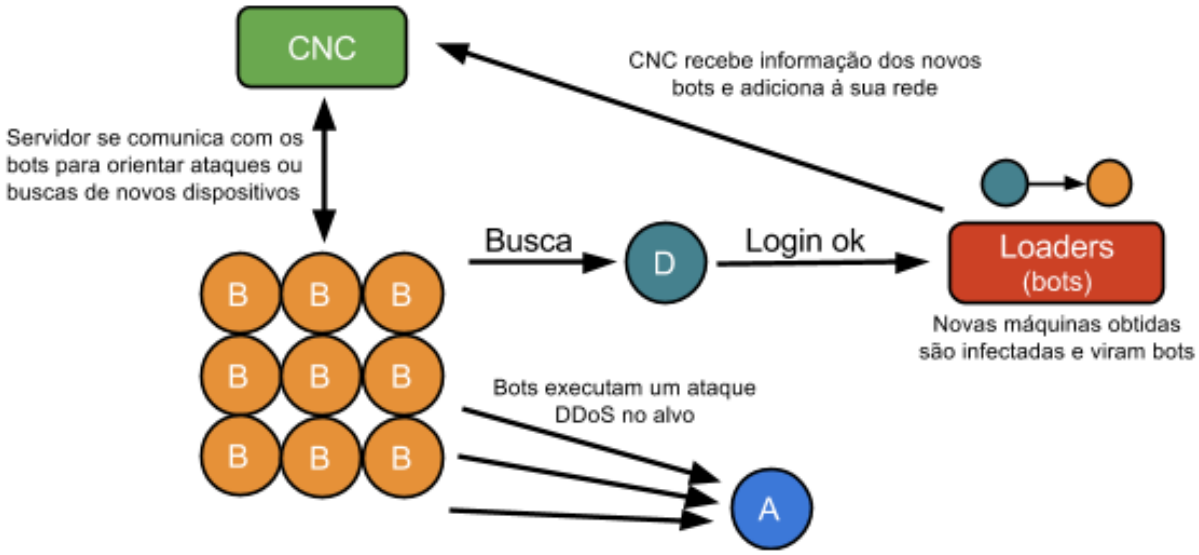

Figura 3.1: Arquitetura utilizada pela botnet Mirai para conseguir novos bots e realizar ataques através deles.

Feito para várias arquiteturas, o vírus utilizava métodos de camuflagem para não ser detectado e gerava cópias de si mesmo nos novos alvos que tinham sido encontrados. Com essa rede criada, foi lançado um ataque de negação de serviço (conhecido como DDoS) em uma empresa de infra-estrutura da costa leste dos Estados Unidos que fornecia serviços de gerenciamento de nomes e endereços virtuais (DNS), o que impediu o acesso a sites como Spotify, Twitter, New York Times e outros. O ataque foi feito utilizando principalmente chips infectados de webcams, sistemas de vigilância e roteadores, aproveitando do baixo nível de segurança desses dispositivos e a escalabilidade que um ataque poderia ter ao se direcionar milhares desses microchips para o mesmo alvo.

Esse tipo de ataque se aproveita da pouca importância que é dada à segurança em dispositivos embarcados. E é essa uma das brechas que deve ser analisada ao se fazer testes de invasão (Pentest, em inglês) em uma rede, o que se expande para dispositivos IoT. É importante ressaltar que no caso da Mirai foram usados aparelhos do dia-a-dia como vetores de ataque. Se imaginarmos um cenário em que várias residências utilizem os mesmos sistemas de automação inseguros, a capacidade de processamento torna absurda a amplitude de um ataque que se use desses dispositivos.

Ofensivas como essa tem um grau de escalabilidade muito alto devido ao grande número de implementações que usam como base hardwares populares ou até mesmo sistemas quase prontos que podem ser facilmente adaptáveis para usos específicos. A Mirai, por exemplo, começou apenas com 1 único dispositivo realizando buscas e testando logins e atingiu em 10 minutos de tempo de scan, 11 mil dispositivos [2], passando em 3 dias a ser responsável por metades dos scans de telnet (porta 23) na internet, conforme mostrado na Figura 3.2.

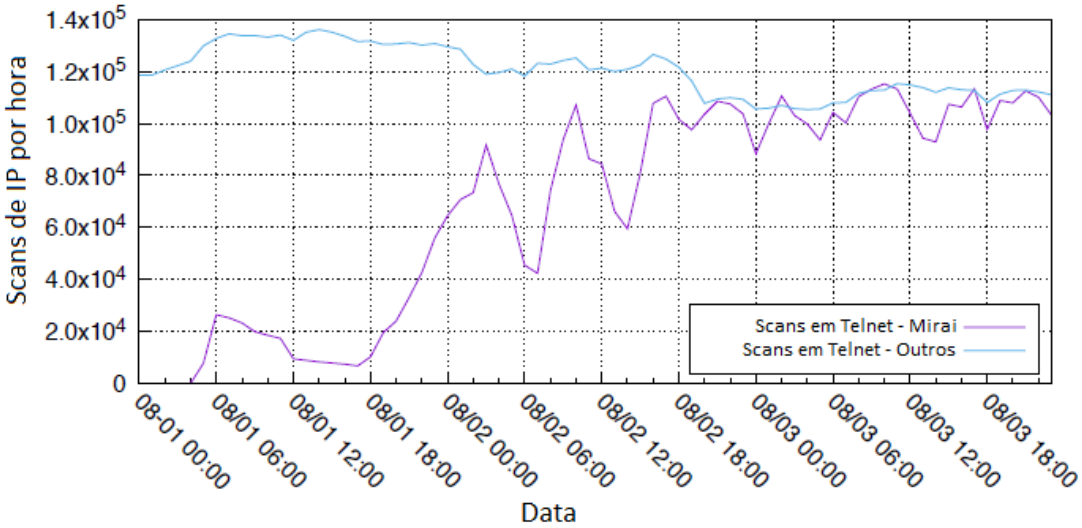

Figura 3.2: Crescimento dos scans de telnet realizados pela botnet Mirai em comparação com scans do mesmo tipo realizados por outras redes [2].

3.3 Mapeamento de Componentes

Para avaliar a segurança na IoT, é importante entender a base estrutural que engloba os aparelhos disponíveis e mapear as interfaces de ataque comumente utilizadas.

Pode-se dividir a infraestrutura de dispositivos IoT em três principais componentes: (1) Dispositivos Embarcados, (2) Software e Aplicações e (3) Sistemas de Comunicação

de Radiofrequência. Cada um possui suas respectivas vulnerabilidades e facilidades de exploração.

3.3.1 Dispositivo

O componente central em qualquer arquitetura IoT é o dispositivo fisico. Dele vêm os dados coletados e/ou as informações necessárias à aplicação. Como dispositivo pode ser considerado qualquer hardware que esteja no centro da operação e que gere o conteúdo em uma infraestrutura. Pode ser central, situação em que é utilizado como conexão ou ponte entre dispositivos de uma mesma rede ou um dispositivo "monitor", que faz geralmente uso de sensores para obter as informações desejadas e informa para outro dispositivo o que foi coletado.

As vulnerabilidades conhecidas nesse componente geralmente estão relacionadas a um acesso de alto nível sendo dado a uma porta sem um devido filtro ou também a capacidade de extrair o firmware sem grande controle, permitindo que informações importantes possam ser alteradas.

3.3.2 Software e Aplicações

Engloba o Firmware, aplicações web e softwares mobile utilizados para controlar, configurar ou monitorar os dispositivos embarcados. Qualquer software que faça comunicação com um dispositivo IoT acrescenta uma nova ameaça à integridade da infraestrutura a partir do momento em que possui acesso a dados importantes sem que possua uma segurança estável.

Os maiores problemas aqui são o uso de Firmware desatualizado, conexões sem criptografia de dados e o uso de protocolos de comunicação não padronizados.

3.3.3 Sistemas de Comunicação de Radiofrequência

A área mais crítica de todo dispositivo IoT. Qualquer comunicação que aconteça entre dispositivos pode ser incluída nessa categoria. As mais comuns são WiFi, BLE, Zigbee, ZWave, 6LoWPAN e redes de celular.

Uma grande quantidade de ataques ocorre por esse componente, já que é por ele que se dá a autenticação e transmissão de dados importantes.

3.4 Principais Projetos e Soluções Propostas para Segurança em IoT

Os principais causadores de acessos indevidos a dispositivos são: protocolos desatualizados, senhas padrão não modificadas, pouco ou nenhum tipo de criptografia nas informações transmitidas e esquemas mínimos de proteção visando o baixo custo. Todos esses problemas geram brechas e permitem algum tipo de violação à segurança dos aparelhos.

Visando a elaboração de respostas para os problemas comuns de segurança, algumas instituições possuem projetos de padronização e boas práticas voltados para a Internet das Coisas. Essas iniciativas unem diversos grupos e visam a disseminação de conhecimento para que questões relativas à segurança sejam amplamente difundidas e solucionadas. Aqui estão listados alguns desses grupos e seus projetos, mostrando o foco de atuação, as soluções apresentadas e o que eles têm a oferecer em segurança quando se trata de IoT.

3.4.1 OTA

Criada em 2005, a *Online Trust Alliance* (OTA) tem como objetivo auxiliar grupos e empresas a desenvolver boas práticas de desenvolvimento e aprimorar a atenção dada às práticas que visam proteger a privacidade e a identidade dos usuários [15]. Surgiu informalmente com um grupo que reunia várias empresas com o objetivo de aumentar a confiança dos usuários nas aplicações e serviços online.

Atualmente possui uma iniciativa própria para a Internet das Coisas chamada IoT Trustworthy Working Group (ITWG), cujo foco é desenvolver padrões para melhorar a segurança, a privacidade e a sustentabilidade em projetos de IoT [16]. Disponibiliza modelos de elaboração, planejamento e execução de projetos, bem como um Framework [17] com vários princípios que visam orientar e auxiliar no desenvolvimento de aplicações mais seguras, tendo como premissas em sua versão 2.1:

1 - Assegurar a segurança do dispositivo, dos aplicativos e dos serviços de nuvem.

2 - Garantir um bom esquema de autenticação de usuário e credenciais de acesso.

3 - Privacidade dos dados, boa divulgação do tempo de suporte das funcionalidades e transparência quanto aos dados que são coletados pelo dispositivo.

4 - Sistema de notificação ao usuário final

Possui outras ferramentas de auxílio, como guias, resultados de pesquisas focadas em segurança para IoT e algumas listas com critérios que desenvolvedores podem utilizar e comparar com suas aplicações. Todo esse ferramental é constantemente atualizado e revisado para que os erros mais comuns estejam sempre de acordo com a realidade atual.

3.4.2 I am the Cavalry

Lançada oficialmente em 2013 em dois eventos, o DEFCON e o Bsides Las Vegas. Seu nome significa "Eu sou a Cavalaria", sendo uma referência aos grandes grupos de cavaleiros que defendiam reinos de ataques e eram chamados de Cavalaria. Tem como objetivo conscientizar os usuários de vulnerabilidades encontradas, tornando-os agentes na proteção de seus próprios dispostivos, sem que eles tenham que esperar por soluções, passando a serem proativos em sua própria segurança *online* [18].

Um de seus criadores, Josh Corman, cita como motivo para formação do projeto o fato de que a gigantesca adoção de tencologias computacionais está afetando o corpo, a mente e a alma dos usuários [19]. Tendo a habilidade de afetar a vida humana e a saúde pública - o corpo - no momento em que a Internet das Coisas se torna algo presente no cotidiano e passa a compor carros, dispositivos médicos, etc. O recente crescimento na crítica aos pesquisadores de segurança e na criminalização dos especialistas é visto como uma ameaça à mente. E, segundo ele, afeta a alma quando os reforços em vigilância e monitoração pública são transformados sem que haja um debate justo se esses fatores são importantes para a sociedade.

Apesar de relativamente recente, conta com vários especialistas de segurança, jornalistas, pesquisadores e políticos como seus integrantes e, por ter o foco no usuário, consegue atingir problemas pontuais com mais facilidade. O grupo trata de questões como a necessidade de uma tecnologia de IoT na sociedade, não apenas analisando a possibilidade de existência, mas questionando a implicação que a criação de uma determinada inovação irá ter na privacidade na segurança dos indivíduos.

Tem quatro áreas como foco:

Automotiva : Com vários projetos visando o desenvolvimento de carros autônomos, há a possibilidade real de ataques que invadam veículos para bloquear ruas, causar acidentes [20] ou praticar atos terroristas [21]. Tecnologias de estacionamento automático, possibilidade de desativar o sistema em caso de roubo e piloto automático estão sendo incluídos nos automóveis sem a devida atenção às implicações que se tem ao abrir uma porta de acesso e expor os usuários a possíveis falhas de segurança.

Infraestrutura Pública : Devido à quantidade de dados que poderiam ser coletados, cidades têm investido em projetos para automatizar tarefas, aumentar a transparência e orientar as ações do governo de acordo com as necessidades dos cidadãos de forma mais simples e objetiva. Mesmo ainda não tendo se tornado realidade por questões políticas ou financeiras [22], projetos de cidades inteligentes estão sendo analisados e têm como principais temas a segurança pública, sistemas de aviação, coleta de lixo e transporte público.

Médica : Mercado promissor no desenvolvimento dispositivos inteligentes, principalmente por possibilitar a redução nos custos de tratamentos [23] e por tornar algumas análises disponíveis a qualquer momento, facilitando o monitoramento de pacientes. É constante o surgimento de novas aplicações médicas que trazem mais conexão e algum tipo de evolução nas análises computadorizadas. Implantes, imagens em tempo real, diagnóticos computadorizados e asistência remota são as áreas mais promissoras e, consequentemente, são as mais abordadas pelo grupo.

Residencial : Automação de atividades rotineiras, eletrodomésticos inteligentes e controle de iluminação, temperatura e de trancas eletrônicas com integração a sistemas de alarmes são as novas ambições de um ambiente domiciliar. Sistemas de monitoramento de baixo custo já são realidade [24] e aos poucos estão sendo adotados em lares e é importante que sejam observadas questões de privacidade e segurança dos usuários dos sistemas em qualquer implementação.

3.4.3 OWASP

Do inglês, Open Web Application Security Project. É uma comunidade *online* sem fins lucrativos, na qual indivíduos e organizações podem discutir e criar artigos, motodologias, ferramentas e tecnologias que possibilitem a outros tomar decisões seguras ao identificar e tratar riscos de segurança em aplicações. Possui diversos recursos para projetos, testes e validação na área de segurança. As principais categorias do site são: documentação, ferramentas, códigos e projetos próprios [25].

As listas produzidas pela organização, que representam um consenso a respeito das falhas de segurança em aplicações *web* críticas, são geradas após pesquisas com profissionais da indústria e visam oferecer uma forma prática de se consultar os maiores problemas sobre segurança. Os erros reportados são geralmente fáceis de encontrar e simples de serem explorados para fins maliciosos, o que faz com que seu tratamento seja necessário.

O projeto possui uma área exclusiva para o estudo de IoT chamada OWASP Internet of Things Project [26] que disponibiliza vários recursos para auxílio no desenvolvimento seguro, sendo que o principal, que será abordado logo abaixo, traz uma lista atualizada periodicamente com os problemas mais comuns de segurança para essa área no mundo [27] e um segundo artefato que, de acordo com os dez problemas mais comuns levantados, auxilia no processo de testes para cada área [28] e que será utilizado para guiar o trabalho proposto. Também são fornecidos: Uma lista com as principais superfícies de ataque em IoT; um guia com padrões a serem seguidos no desenvolvimento de software seguro em dispositivos e na nuvem; um manual com princípios de design seguro; um manual de auxílio ao desenvolvedor com informações da comunidade; um guia de análise de firmware.

3.5 Análise das Maiores Vulnerabilidades de IoT de Acordo com a Lista da OWASP

Com base na análise da OWASP, elencam-se as 10 maiores vulnerabilidades em IoT. Nessa lista são avaliados os agentes da ameaça, os riscos envolvidos e quais tipos de impactos possíveis que essa falha pode gerar na aplicação e no negócio. A última lista gerada foi do ano de 2014 e nela estão os maiores problemas percebidos na época; que estão ainda atuais se considerarmos o quão recorrentes são esses ataques ao analisarmos as outras listas geradas pela própria organização [29].

3.5.1 Interface *Web* Insegura

Os principais problemas aqui são: credenciais fracas, informações de acesso enviadas por texto entre a aplicação e o servidor e enumeração de contas de autenticação. Injeção SQL (SQLi), Cross-Site Scripting (XSS) e Cross-Site Request Forgery (CRSF) são as formas de ataque mais comuns que exploram esse tipo de vulnerabilidade, com o primeiro correspondendo por 25% das vulnerabilidades encontradas em aplicações Web entre 2011 e 2013 segundo relatório da Cenzic [30]. Esses problemas são facilmente identificáveis por uma verificação manual ou uma ferramenta de análise automatizada. A negligência no tratamento desse tipo de vulnerabilidade pode gerar perda de dados importantes, falta de credibilidade da informação e a privação do controle de acesso total do dispositivo.

Para evitar esses problemas é importante garantir possibilidade de modificar nomes de usuário e senhas padrão, oferecendo mecanismos de recuperação de senha robustos e que não forneçam informações de uma conta válida e que a interface não esteja suscetível aos ataques mais comuns. Também é necessário garantir que credenciais não estejam expostas no tráfego de informações, senhas fracas não sejam permitidas e que o acesso à conta seja bloqueado depois de 3 a 5 tentativas.

3.5.2 Autenticação e Autorização Insuficientes

Os principais meios de ataque nesse item são: senhas fracas, mecanismos não seguros de recuperação de senha, pouca ou nenhuma proteção de credenciais e a inexistência de um acesso granular no sistema de autenticação. Podendo vir tanto de usuários internos ou externos através das interfaces *web*, *cloud* e *mobile*. Esses problemas são comuns [30] e podem ser identificados utilizando as mesmas verificações que são feitas para evitar uma interface web insegura e o não tratamento pode acarretar os problemas semelhantes, bem como a perda dos dados de todos os usuários e uma base de credenciais completamente comprometida.

Deve-se garantir que senhas fracas não sejam permitidas, podendo-se utilizar de listas obtidas com as senhas mais comuns ou com algoritmos de análise de força de senhas. É necessário que controle de acesso granular seja feito quando for indispensável, com as credenciais devidamente protegidas e que haja autenticação de "dois fatores" quando for possível. O mecanismo de recuperação de senha deve ser seguro para evitar tomada de controle total de contas de usuários por terceiros, uma segunda autenticação deve ser feita para acesso a conteúdos críticos ou sensíveis e deve-se fornecer opções de configuração de controle de senhas para limitar permissões de acesso.

3.5.3 Serviços de Rede Inseguros

Com a presença de mecanismos de comunicação inseguros, torna-se possível atacar o dispositivo ou até mesmo utilizá-lo com o intuito de direcionar ataques para outros aparelhos. Negação de serviço em massa, conhecido como DDoS, e *Buffer Overflow* são as principais formas de ataque utlizadas, por isso é necessário focar na identificação de possíveis ameaças que se utilizem desses métodos. Ferramentas automatizadas de análise de portas ou *Fuzzers* podem ser utilizadas para auxiliar na detecção desses problemas, sendo que casos de Buffer Overflow necessitam de um tratamento mais detalhado por não serem facilmente identificáveis. Não tratar esses pontos pode acarretar na perda de dados, negação do serviço para usuários ou até mesmo a facilitação de ataques a outras redes se utilizando da estrutura computacional comprometida como ocorreu no caso da Mirai [14].

É importante garantir que somente as portas necessárias estão disponíveis e expostas, serviços não estão vulneráveis a buffer overflow e nem ataques fuzzing e que exista uma verificação de pedidos ou que se utilize de uma estrutura intermediária de conexão para controlar ataques de negação de serviço. Também é necessário garantir que a rede não está vulnerável a ataques que possam afetar o próprio dispositivo ou outros dispositivos e/ou usuários na rede local ou em outras redes e que portas ou serviços não estão expostos na internet por UPnP (Universal Plug and Play)[31] por exemplo ou que portas de teste não estejam abertas.

3.5.4 Falta de Criptografia no Transporte

Ataques podem se utilizar da falta de criptografia na comunicação de dados pela rede para visualizar informações que estão sendo passadas. Em redes locais é facilmente assumido que não é necessário um tipo de encriptação nos dados, pois os mesmos só estão sendo visualizados localmente. Porém, qualquer alteração nas configurações de acesso pode dar permissão de visualização para quem estiver na área próxima à da transmissão. Esses problemas são facilmente detectáveis ao se analisar o tráfego da rede e ao se procurar

dados que possam ser lidos facilmente. Ferramentas automatizadas podem procurar por problemas em sistemas comuns de encriptação, como TLS e SSL. A não observância desses problemas pode acarretar na perda de dados e o comprometimento completo das informações de acesso dos usuários.

Deve-se garantir que toda a informação está sendo encriptada utilizando protocolos como SSL ou TLS enquanto transita na rede, algum outro tipo de padrão de encriptação está sendo utilizado para proteger os dados enquanto eles são transportados se SSL ou TLS não estiverem disponíveis, que somente padrões aceitáveis de encriptação são utilizados e que protocolos proprietários são evitados.

3.5.5 Problemas de Privacidade

A falta de uma autenticação boa, de encriptação no transporte de dados ou até mesmo serviços não seguros de rede podem permitir com que agentes não pertencentes ao fluxo de informações consigam dados que não estão devidamente protegidos ou que estão sendo coletados sem necessidade. É comum que isso ocorra no momento que o usuário acessa o dispositivo, então a forma mais fácil de analisar possíveis ameaças é monitorando o momento de acesso e verificando os dados que são gerados e o que está sendo exibido. Algumas ferramentas podem automatizar esse processo e indicar dados sensíveis ou privados que estão sendo expostos. A falta de cuidado com esse quesito pode comprometer a privacidade dos dados pessoais do usuário.

Para evitar esse tipo de problema, somente dados intrinsecamente importantes para o funcionamento do dispositivo devem ser coletados, tendo como princípio que qualquer dado coletado e que seja de uma natureza não tão privada deve ser mostrado de forma anônima, garantindo assim a despersonificação quando possível e que qualquer informação obtida esteja devidamente encriptada. Além disso, o dispositivo e todos os seus componentes devem proteger informações pessoais e somente indivíduos autorizados devem ter permissão de coletar dados dos usuários, impondo também um limite na quantidade de informação que pode ser coletada. Por último, deve-se fornecer aos usuários finais uma opção de selecionar o que eles querem receber no caso de o dispositivo ter a possibilidade de coletar mais dados do que esperado.

3.5.6 Interfaces de Nuvem Inseguras

Possuem os mesmos problemas de uma interface *web* insegura: Credenciais fracas, informações de acesso enviadas por texto entre a aplicação e o servidor (sem encriptação) e enumeração de contas de autenticação. É facilmente identificável ao se testar uma conexão entre o servidor e a interface de nuvem e monitorar o uso de SSL ou a possiblidade

de enumeração de contas. Esse tipo de falha pode acarretar o comprometimento de toda informação intrínseca ao usuário e também o controle do dispositivo por terceiros.

Para evitar problemas nessas interfaces deve-se garantir que nomes de usuário e senhas padrão sejam modificados na configuração inicial, fornecer mecanismos de recuperação de senha robustos e que não exponham informações de uma conta válida, prover uma interface que não esteja suscetível aos ataques mais comuns e não expor credenciais no tráfego de informações. Para mitigar essas falhas de segurança e evitar ataques por força bruta, a autenticação de dois fatores deve estar disponível quando for necessária, não permitindo senhas fracas e bloqueando o acesso à conta depois de 3 a 5 tentativas.

3.5.7 Interfaces Móveis Inseguras

Problemas e tratamento são similares aos de interfaces de nuvem inseguras. Assim, os problemas acarretados são os mesmos e as mesmas formas de detecção podem ser aplicadas. É importante garantir também os mesmos tipos de cuidado para evitar a ocorrência desse problema.

3.5.8 Insuficiência na Configurabilidade de Segurança

A falta de permissão granular de acesso a dados e controles do dispositivo, bem como a falta de criptografia ou opções para senhas são as maiores falhas dessa área. Nesse cenário, os usuários não possuem muitas opções e recursos de configuração nos controles de segurança, o que padroniza ainda mais os ataques ao não forçar o usuário a ter senhas fortes por exemplo. Esse tipo de vulnerabilidade pode ocasionar a perda de dados e acesso indevido intencional ou até mesmo não-intencional por parte de outros usuários.

É importante garantir que haja separação entre usuários comuns e outros com permissões maiores, armazenando e transmitindo dados sempre de forma encriptada, com disponibilidade de relatórios de atividades na conta dos usuários e uma política de senhas fortes sendo implementada.

3.5.9 *Softwares/Firmwares* Inseguros

Um ataque pode vir da captura de arquivos de atualização enviados por conexões não encriptadas ou de um arquivo que não esteja ele mesmo encriptado. Ambos podem levar à alteração do sistema, com injeção de código no firmware ou no software utilizado, fornecendo ao responsável pelo ataque o controle do dispositivo e o acesso às informações presentes na rede bem como os dados trafegados. Esse tipo de ameaça pode comprometer toda uma infraestrutura e os dados de usuário presentes, tornando vulnerável toda a informação contida e permitindo o roubo e até mesmo a alteração dos dados sem que o

ataque seja devidamente detectado. Ferramentas de monitoramento de tráfego são úteis na análise desse problema e na detecção do mesmo.

É importante garantir que o dispositivo tenha a capacidade de se atualizar, que o servidor utilizado seja seguro e que o arquivo de atualização esteja utilizando métodos aceitáveis de encriptação, esteja sendo transmitido por uma conexão encriptada, não esteja expondo dados sensíveis e que a atualização seja verificada antes de autorizar seu envio e aplicação.

3.5.10 Péssima Segurança Física

O ataque pode partir do próprio dispositivo, utilizando o acesso por meio de uma porta USB, cartões SD ou memória física para entrar no sistema operacional. Esses problemas se tornam presentes quando um usuário pode desmontar o dispositivo e acessar componentes cruciais ou até mesmo quando permissões de manutenção e configuração são dadas por acesso via alguma porta. Isso pode comprometer tanto o próprio dispositivo, quanto os dados contidos nele.

É importante garantir que o meio de armazenamento não possa ser removido facilmente, os dados armazenados estejam encriptados, portas não possam ser utilizadas para permitir acessos indevidos, o dispositivo não seja facilmente desmontável, somente as portas intrisicamente necessárias do dispositivo estejam em funcionamento e que o aparelho tenha a capacidade de limitar as modificações feitas por usuários administrativos.

3.6 Considerações Finais

Este capítulo abordou os principais pontos em relação à segurança na Internet das Coisas. Para isto, foram mapeadas as superfície de ataque comuns em IoT, mostrando os maiores problemas encontrados em cada uma. Em seguida, foram apresentados os principais projetos que lidam com esses problemas e uma breve apresentação da história e dos objetivos de cada um. E, por último, foi feita uma análise da lista da OWASP, detalhando as categorias e suas formas indicadas de prevenção de ataques.

Todos os projetos mostrados possuem suas vantagens na análise de problemas de segurança, alcançando isso através de metodologias próprias muito bem conceituadas e embasadas em experiências reais. Porém, por estar bem atual, ter uma comunidade ampla e visar a difusão do conhecimento sem viés na informação, a lista da OWASP sobre segurança na Internet das Coisas será utilizada como base no desenvolvimento da ferramenta proposta neste trabalho. O Capítulo 4 irá apresentar com detalhes a proposta deste trabalho para a ferramenta de Pentest em dispositivos da Internet das Coisas.

Capítulo 4

Ferramenta para Testes de Invasão

Aqui, serão analisadas as categorias já mencionadas na lista da OWASP [27], visitando cada uma e detalhando as superfícies de ataque em que os testes possam ser automatizados e erros sejam passíveis de verificação prática. Como a ferramenta tem o objetivo de atender o maior número de dispositivos possível e dar suporte para o analista de vulnerabilidades, o foco será utilizar métodos que consigam extrair informações importantes, testar problemas de modo geral e apontar erros sem que, para isso, venha comprometer o dispositivo ou os dados contidos nele.

Montar um diagrama de comunicação completo e que apresente os diversos tipos de interação externa que o dispositivo faz (por radiofrequência ou de forma física) e que tipo de Application Programming Interface (API) ele utiliza (no caso de usar) pode ajudar a entender quais são os pontos mais críticos. Para a ferramenta, o diagrama deve compreender, de maneira mais ampla, superfícies comuns de diversos dispositivos. Assim, o Guia de Testes da OWASP [28] será utilizado para nortear a diagramção dos módulos e quais funcionalidades são necessárias.

4.1 Mapeamento da Ferramenta

O Guia de Testes de IoT apresentado na Tabela 4.1 mapeia, de uma forma mais geral, os pontos importantes e fornece um conjunto básico de premissas para os testes. Alguns ensaios propostos pelo guia não são passíveis de automação por necessitarem de uma verificação mais precisa por parte do analista ou que o desenvolvedor apresente o código-fonte para testes. Os itens que se enquadram nessa categoria não serão analisados de forma automática pela ferramenta, mas sim apresentados como algo a se verificar, facilitando o trabalho do especialista nos pontos em que é possível. Há também verificações que podem ser feitas em conjunto e que, por isso, serão agrupadas em um mesmo módulo ou em um mesmo teste.

Tabela 4.1: Guia de Testes da OWASP para Dispositivos IoT

Guia de Testes para Dispositivos IoT		
Categoria	Código	Descrição
Interface *Web*	A1	Senhas fracas não devem ser permitidas
	A2	Uso de mecanismo de bloqueio de contas é necessário
	A3	Vulnerabilidade a XSS, SQLi, CSRF e outros
	A4	Uso de HTTPS para proteger informações transmitidas
	A5	Possibilidiade de mudar o nome de usuário e a senha
	A6	Uso de Web Application Firewall (WAF)
Autenticação e Autorização	B1	Proibição de senhas fracas na autenticação
	B2	Separação de perfis em ambientes com vários usuários
	B3	Autenticação por dois fatores onde for possível
	B4	Mecanismo de recuperação de senha deve funcionar
	B5	Opção de solicitar senhas fortes
	B6	Opção de forçar que a senha expire após um período
	B7	Opção de modificar nome de usuário e senha padrão
Serviços de Rede	C1	Resiliência Buffer Overflow, Fuzzing e DDos
	C2	Portas de teste não devem estar habilitadas
Encriptação no Transporte de Dados	D1	Comunicação entre dispositivos feita forma encriptada
	D2	Práticas de encriptação aceitáveis devem ser utilizadas
	D3	Opção de habilitar *Firewall*
Privacidade	E1	Avaliar a quantidade de dados pessoais coletados
	E2	Informação pessoal coletada é encriptada no armazenamento e em trânsito
	E3	Dados de verificação devem estar sem identificação
	E4	Usuários devem ter a opção de limitar os dados coletados somente ao que for estritamente necessário
Interface de Nuvem	F1	Vulnerabilidades de Segurança (Interfaces de APIs e Interfaces *Web* baseadas na nuvem)
	F2	Proibição de senhas fracas na autenticação
	F3	Uso de mecanismo de bloqueio de contas é obrigatório
	F4	Uso de autenticação por dois fatores onde for possível
	F5	Vulnerabilidade a XSS, SQLi, CSRF e outros
	F6	Encriptação no transporte de dados
	F7	Opção de solicitar senhas fortes
	F8	Opção de forçar que a senha espire após um período
	F9	Possibildade de modificar nome de usuário e senha
Interface Móvel	G1	Proibição de senhas fracas onde autenticação é necessária
	G2	Uso de mecanismo de bloqueio de contas
	G3	Autenticação por dois fatores onde for possível
	G4	Encriptação no transporte de dados
	G5	Opção de solicitar senhas fortes

	G6 G7 G8	Opção de forçar que a senha expire após um período Possibildade de modificar nome de usuário e senha Avaliar a quantidade de dados pessoais coletados
Configurabilidade de Segurança	H1 H2 H3 H4	Disponibilidade de opções de segurança Disponibilidade de opções de encriptação Necessidade de novo *login* para acesso a funções críticas Alertas ao acessar funções críticas
Software e *Firmware*	I1 I2 I3	Atualização rápida contra vulnerabilidades detectadas Encriptação no armazenamento e envio dos arquivos de atualização Validação dos arquivos antes da atualização
Segurança Física	J1 J2 J3 J4	Somente estritamente portas necessárias disponíveis Acesso somente por meio de meios previstos Capacidade de desativar portas que não estão em uso Habilidade de transmitir capacidades administrativas somente para interfaces locais

4.1.1 Análise das Categorias e Subdivisão dos Módulos

A metodologia utilizada para verificar a possibilidade de automação será divida em três partes. Primeiro, deve-se determinar o objeto de verificação de cada categoria, o que será analisado e quais os pré-requisitos que devem ser cumpridos. Em seguida, deve-se determinar se a categoria em geral pode ser automatizada e testada, notando quais os problemas relativos à implementação. Por último, é necessário verificar pontualmente quais dos testes são passíveis de automação sem causar danos reais ao dispositivo e quais verificações dessa categoria podem ser feitas em conjunto com outras já abordadas.

Algumas categorias têm pré-requisitos específicos que não podem ser cumpridos sem que isso desvie o programa das premissas de automação e universalidade. Por isso, só serão implementadas como um guia indicativo dentro da ferramenta. Outras, por possuírem pontos em comum com outros testes, serão analisadas por um mesmo módulo. Há também casos em que um teste pode ser feito por outro já pensado, estes serão agrupadas ou incorporados a uma mesma funcionalidade.

Interface Web

Uma Interface *Web* é definida como o painel de controle que está entre os usuários e o software rodando em um servidor *online* [32]. É qualquer interface homem-máquina que consista em páginas da *internet* e que permita, em alguns casos, utilizar funcionalidades disponibilizadas por aplicativos. Geralmente, é necessário um cliente ou navegador para se visualizar o conteúdo dessa interface. Muitos dispositivos possuem uma interface própria que permite o acesso a seus conteúdos administrativos.

Aqui será analisada a conexão homem-máquina através de um teste que simula ação do cliente. Para isso, é necessário um servidor rodando no dispositivo alvo dos testes. Dentre as verificações que podem ser automatizadas com esse pré-requisito, tem-se: verificação de vulnerabilidade a XSS, SQLi e outras vulnerabilidades (A3), análise do uso de HTTPS para proteger as informações sendo transmitidas (A4) e verificação da presença de um WAF (A6).

Não é possível verificar se senhas fracas não são permitidas (A1) sem que o teste seja feito de forma estática ou sem monitorar a comunicação dos dados de acesso dos usuários para o servidor, o que necessitaria de uma conexão intermediária. A habilidade de verificação dos dados de acesso (A5) requer que o analista acesse a funcionalidade manualmente e verifique a configurabilidade do serviço. O mesmo caso se aplica ao uso de um mecanismo de bloqueio de contas (A2). O teste de vulnerabilidade a CSRF foi excluído da análise devido à especificidade que o ataque tem para cada sistema, onde as informações enviadas devem ser analisadas e modificadas de acordo com a intenção do atacante [33]. Assim, recomenda-se que sejam feitos ensaios manuais para esse teste .

Outro teste importante a ser feito é verificar a proteção do site contra *Clickjacking* (do inglês, "furto de clique"), onde a página permite a execução de operações vindas de páginas externas dentro do seu código através de *frames*, deixando assim o usuário vulnerável a ataques como injeção e extração de dados [34]. O tratamento dessa vulnerabilidade diminui inclusive a superfície de ataques para métodos de invasão como o CSRF [35]. Além disso, deve-se também analisar a possibilidade de enumeração das páginas de autenticação através de métodos de comparação ou de força bruta [36]. Ficando assim a divisão dos testes dessas interfaces no primeiro módulo da ferramenta (Björn) conforme a Tabela 4.2.

Tabela 4.2: Divisão de Testes do Módulo de Interface *Web* (Björn)

Testes em Interface *Web*			
Passíveis de Automatização		Necessário Esforço Manual	
Código	Descrição	Código	Descrição
A3*	Vulnerabilidade a XSS e SQLi	A1	Senhas fracas não devem ser permitidas
A4	Uso de HTTPS para proteger informações transmitidas	A2	Uso de mecanismo de bloqueio de contas é necessário
A6	Uso de WAF (*Web Application Firewall*)	A3*	Vulnerabilidade a CSRF e outros
N1	Proteção contra *Clickjacking*	A5	Possibilidiade de mudar o nome de usuário e a senha
N2	Enumeração de páginas de autenticação		

Autenticação e Autorização Insuficientes

Essa categoria engloba todos os testes relativos à autenticação de usuários no sistema. Requer um sistema de controle de acesso com dados de usuário e senha, com uma relação imposta entre eles. Há sistemas de autenticação tanto locais quanto em rede, cada um com suas especificidades.

Para ser testada, essa categoria requer uma interface de conexão, por intermédio da qual os dados serão comunicados. O foco aqui está na configurabilidade do sistema de autenticação e, por isso, requer uma atenção manual e mais técnica do analista, sendo específica para cada sistema. Não é possível automatizar esse tipo de verificação e, por isso, será incluída na ferramenta somente como uma lista a ser analisada. Todos os testes se enquadram na mesma situação, conforme descrito na Tabela 4.3

Tabela 4.3: Divisão de Testes de Autenticação

Testes de Autenticação - Necessário Esforço Manual	
Código	Descrição
B1	Proibição de senhas fracas na autenticação
B2	Separação de perfis em ambientes com vários usuários
B3	Autenticação por dois fatores onde for possível
B4	Mecanismo de recuperação de senha deve funcionar
B5	Opção de solicitar senhas fortes
B6	Opção de forçar que a senha expire após um período
B7	Opção de modificar nome de usuário e senha padrão

Serviços de Rede

Um serviço de rede pode ser definido como um processo sendo executado em uma rede e que utiliza uma porta específica aguardando a conexão de processos de clientes rodando em outros computadores[37]. Tem como objetivo final prover algum serviço na rede. Cada serviço é provido por algum componente do servidor que está sendo executado em um ou mais computadores (é comum, porém, que um mesmo computador ofereça vários serviços). É dado pela tripla (IP, porta, protocolo) que define a máquina em que o serviço está sendo executado, o tipo de aplicação e qual protocolo de internet está sendo utilizado (TCP ou UDP).

Neste caso, a análise é feita a respeito de quais serviços estão sendo fornecidos e se eles cumprem as validações de segurança. Para isso, é necessário um servidor rodando no dispositivo alvo dos testes e que os serviços que precisam ser testados estejam disponíveis. Conforme descrito pela Tabela 4.4, dentre as verificações que podem ser automatizadas com esse pré-requisito, tem-se: teste de resiliência a *Fuzzing* e DDos (C1) e verificação da presença de portas de teste (C2).

Não é possível verificar de forma automatizada a resiliência do serviço a *Buffer Overflow* (C1) devido à especificidade de cada sistema quanto ao gerenciamento de memória [38]. Esse tipo de teste será incluído na ferramenta como um ponto a ser analisado.

Tabela 4.4: Divisão de Testes do Módulo de Serviços de Rede (Ubba)

Testes de Serviços de Rede			
Passíveis de Automatização		Necessário Esforço Manual	
Código	Descrição	Código	Descrição
C1* C2	Resiliência a *Fuzzing* e DDoS Verificação de portas de teste	C1*	Resiliência a *Buffer Overflow*

Encriptação no Transporte de Dados

Aqui, é necessário verificar os serviços de rede com interfaces que utilizam protocolos de proteção no transporte de informações. Isso se faz analisando o uso de TLS (do inglês, *Transport Layer Security*), SSL (*Secure Sockets Layer*) e protocolos proprietários. Para isso, é necessário um servidor rodando no dispositivo alvo dos testes.

Dentre as verificações que podem ser automatizadas com esse pré-requisito, tem-se: comunicação entre dispositivos feita de forma encriptada (D1), práticas de encriptação aceitáveis sendo utilizadas (D2). Ambas podem ser executadas de forma automática ao simular uma conexão entre um cliente e o servidor e, por isso, serão incluídas no módulo de Encriptação da ferramenta (Hvitserk).

Para se testar se há a opção de habilitar um *Firewall* (D3) na conexão é necessário uma análise mais profunda da configurabilidade do sistema de comunicação, o que requer uma atenção manual e uma verificação mais técnica do analista a respeito de como as opções estão sendo apresentadas aos usuários do serviço. Não é possível automatizar esse tipo de apuração e, por isso, será incluída na ferramenta somente como um ponto a ser analisado, conforme mostra a Tabela 4.5.

Privacidade

Essa categoria tem os mesmos pré-requisitos que a anterior, somando-se a eles, um serviço de autenticação para ser avaliado. Deve ser avaliada a quantidade de informação pessoal coletada e quais os métodos de proteção utilizados para esses dados. Configurabilidade também é importante aqui.

Para ser testada, essa categoria requer uma interface de conexão, por intermédio da qual os dados serão comunicados. O foco aqui está na configurabilidade do sistema de autenticação e, por isso, requer uma atenção manual e mais técnica do analista, sendo específica para cada sistema. O teste para verificar se a informação pessoal coletada é encriptada no armazenamento e em trânsito (E2) é a única análise passível de automatização e, por ter uma premissa em comum com os testes realizados na categoria de Encriptação no Transporte, será incluída na ferramenta em conjunto com as verificações deste último, conforme descrito na Tabela 4.5

Tabela 4.5: Divisão de Testes do Módulo de Encriptação (Hvitserk)

Testes de Encriptação no Transporte de Dados			
Passíveis de Automatização		Necessário Esforço Manual	
Código	Descrição	Código	Descrição
D1	Comunicação entre dispositivos feita de forma encriptada	D3	Opção de habilitar *Firewall*
D2	Práticas de encriptação aceitáveis devem ser utilizadas		
Testes de Privacidade			
Passíveis de Automatização		Necessário Esforço Manual	
Código	Descrição	Código	Descrição
E2	Informação pessoal coletada é encriptada no armazenamento e em trânsito	E1	Avaliar a quantidade de dados pessoais coletados
		E3	Dados de verificação devem estar sem identificação
		E4	Opção de limitar os dados coletados somente ao que for estritamente necessário

Interface de Nuvem

Requer um provedor rodando serviços em um computador externo conectado à *internet*, o qual provê uma interface de acesso. Pode ser separado em dois cenários para análise neste estudo: acesso de serviço por APIs de *back-end* de terceiros; servidor em nuvem que

fornece interface *Web* [39]. O primeiro caso não é passível de avaliação automatizada, pois requer uma análise de ferramentas proprietárias ou que não seguem o mesmo padrão, por isso será somente incluído como uma lista de questões a se observar.

No segundo caso, temos um cenário idêntico ao que já será tratado no Módulo de Interfaces *Web* (Björn), assim, as verificações de vulnerabilidades de segurança (F1) e vulnerabilidade a XSS e SQLi (F5*) já estão contempladas, com a exclusão do teste de vulnerabilidade a CSRF. Das outras exigências, a análise na encriptação de dados (F6) é feita pelo módulo de Encriptação (Hvitserk).

Todas as outras verificações se enquadram no mesmo tipo de análise feita na categoria de Autenticação. Assim, os testes de proibição de senhas fracas na autenticação (F2), uso de mecanismo de bloqueio de contas (F3), método de autenticação por dois fatores (F4), opção de solicitar senhas fortes (F7), opção de forçar que a senha expire após um período (F8) e a possibilidade de modificar nome de usuário e senha (F9) serão incluídos como observações. A organização dos testes dessa categoria está descrita na Tabela 4.6.

Tabela 4.6: Divisão de Testes de Interface em Nuvem

Testes de Interface em Nuvem			
Já Contempladas (Björn e Hvitserk)		Necessário Esforço Manual	
Código	Descrição	Código	Descrição
F1	Vulnerabilidades de segurança	F2	Senhas fracas na autenticação
F5*	Vulnerabilidade a XSS e SQLi	F3	Mecanismo de bloqueio de contas
F6	Encriptação de dados	F4	Autenticação por dois fatores
		F5	Vulnerabilidade a CSRF
		F7	Opção de solicitar senhas fortes
		F8	Opcão de prazo de expiração de senha
		F9	Possibilidade de modificar nome de usuário e senha

Interface Móvel

Requer um servidor rodando serviços em um computador externo conectado à *internet*, o qual provê uma interface de acesso utilizada por dispositivos móveis. Possui uma gama de padrões, altamente dependendtes do tipo de dispositivo em questão, na forma como lida com autenticação e gerenciamento de contas. São estes os dois principais pontos a serem analisados.

Levando em consideração que a comunicação do dispositivo com a interface móvel se dá por um serviço de rede e que a verificação de um mecanismo de encriptação no

transporte de dados (G4) já está prevista pelo Módulo de Encriptação (Ubba), não há a necessidade de implementação dessa funcionalidade de forma separada.

Todas as outras verificações se enquadram no mesmo tipo de análise feita na categoria de Autenticação. Assim, os testes de proibição de senhas fracas na autenticação (G1), uso de mecanismo de bloqueio de contas (G2), método de autenticação por dois fatores (G3), opção de solicitar senhas fortes (G5), opção de forçar que a senha expire após um período (G6), a possibilidade de modificar nome de usuário e senha (G7) e avaliação da quantidade de dados pessoais coletados (G8) serão incluídos como observações. A organização dos testes dessa categoria está descrita na Tabela 4.7.

Tabela 4.7: Disivão de Testes de Interface Móvel

Testes de Interface Móvel			
Já Contempladas (Ubba)		Necessário Esforço Manual	
Código	Descrição	Código	Descrição
G4	Encriptação no Trasnporte de Dados	G1	Senhas fracas na autenticação
		G2	Mecanismo de bloqueio de contas
		G3	Autenticação por dois fatores
		G5	Opção de solicitar senhas fortes
		G6	Opcão de prazo de expiração de senha
		G7	Possibilidade de modificar nome de usuário e senha
		G8	Avaliação da quantidade de dados pessoais coletados

Configurabilidade de Segurança

Essa categoria se assemelha à de Autenticação, pois foca em testes relativos às configurações na autenticação de usuários. Requer um sistema de controle de acesso com dados de usuário e senha, com uma relação imposta entre eles.

Para ser testada, essa categoria requer uma interface de conexão, por intermédio da qual os dados serão comunicados. Além disso, necessita que o analista verifique as mensagens de aviso que são emitidas, sendo que não há um padrão de notificação. Assim, requer uma atenção manual específica para cada sistema. Não é possível automatizar esse tipo de verificação e, por isso, será incluída na ferramenta somente como uma lista a ser analisada. Todos os testes se enquadram na mesma situação, conforme descrito na Tabela 4.8

Tabela 4.8: Divisão de Testes de Configurabilidade

Testes de Configurabilidade - Necessário Esforço Manual	
Código	Descrição
H1	Disponibilidade de opções de segurança
H2	Disponibilidade de opções de encriptação
H3	Necessidade de novo *login* para acesso a funções críticas
H4	Alertas ao acessar funções críticas

Software* e *Firmware

O foco deste tipo de análise está em verificar o sistema do dispositivo. Por isso, é necessário que o analista possua o dispositivo localmente e/ou consiga ter a posse do arquivo do sistema-base na página do fabricante, capturando-o durante uma atualização ou extraindo-o direto do aparelho.

Os testes de encriptação no armazenamento dos arquivos de atualização (I2*) e validação dos arquivos antes de a atualização ser executada (I3) podem ser verificados com uma análise do sistema de *Firmware* ou na transmissão dos dados, sendo o teste de transmissão dos arquivos de forma segura (I2*) já está sendo visto no Módulo de Encriptação, enquanto o teste no armazenamento e o de validação serão verificados por um novo módulo que terá como foco a extração de dados de arquivos binários.

Já a atualização rápida contra vulnerabilidades detectadas (I1) requer um estudo do tempo que se demora para que uma atualização específica que corrija um erro seja liberada pelo fabricante, esse tipo de verificação necessita de análise ao longo do tempo e não pode ser automatizada pelo que é oferecido aqui. Assim, as análises ficam conforme descritas na Tabela 4.9

Tabela 4.9: Divisão dos Testes do Módulo de *Firmware* (Ivar)

Testes de *Software* e *Firmware*			
Passíveis de Automatização		Necessário Esforço Manual	
Código	Descrição	Código	Descrição
I2*	Encriptação no armazenamento dos arquivos de atualização	I1	Atualização rápida contra vulnerabilidades detectadas
I3	Validação dos arquivos antes da atualização		

Segurança Física

Essa categoria trata dos problemas relativos aos acessos indevidos por meio de portas físicas presentes no dispositivo. Há formas de verificar quais dessas portas estão habilitadas, porém não de um modo automatizado geral, pois, dependendo do tipo de sistema operacional instalado, a forma de se lidar com dispositivos externos muda [40].

Para ser testado, requer um dispositivo presente no local de testes e também permissões administrativas para fazer verificações de acesso específicas. Além disso, necessita que o analista saiba quais são as portas que devem estar presentes. Devido a esses fatores, será incluída na ferramenta somente como uma lista a ser analisada, conforme descrito na Tabela 4.10

Tabela 4.10: Divisão de Testes de Segurança Física

Testes de Segurança Física - Necessário Esforço Manual	
Código	Descrição
J1	Somente portas estritamente necessárias disponíveis
J2	Acesso somente por meio de meios previstos
J3	Capacidade de desativar portas que não estão em uso
J4	Habilidade de transmitir capacidades administrativas somente para interfaces locais

4.2 Estruturação e Desenvolvimento

Conforme analisado, a ferramenta possui quatro módulos principais: Björn (Interface Web), Ubba (Serviços de Rede), Hvitserk (Encriptação) e Ivar (Firmware). Cada um contempla pontos importantes na análise de vulnerabilidades em um dispositivo IoT. As verificações finais realizadas por cada módulo estão descritas na Tabela 4.11. Conforme demonstrado na análise, eles abordam os testes que têm entradas e saídas em comum ou que possuem objetivos finais semelhantes.

Para cada módulo, serão gerados em tela ou em arquivo de saída resultados específicos dos testes que são propostos. Aqui será descrito como cada módulo funciona, com um detalhamento do tipo de vulnerabilidade tratada, a causa do problema e como a ferramenta faz para detectá-la. As entradas necessárias para que os testes sejam realizados serão especificadas junto aos métodos de detecção utilizados. Há também, quando for possível, um detalhamento de quais envios e recebimentos são feitos e quais decisões de desenvolvimento foram tomadas ao longo do processo para cada teste.

Tabela 4.11: Divisão da Ferramenta para Testes de Invasão

Ferramenta para Testes de Invasão - *Ragnar*	
Módulo	Funcionalidades
Björn Interface *Web*	Vulnerabilidade a XSS e SQLi Uso de HTTPS para proteger informações transmitidas Uso de *Web Application Firewall* (WAF) Proteção contra *Clickjacking* Enumeração de páginas de autenticação
Ubba Serviços de Rede	Resiliência a *Fuzzing* e DDos Portas de teste não devem estar habilitadas
Hvitserk Encriptação	Comunicação entre dispositivos feita de forma encriptada Práticas de encriptação aceitáveis devem ser utilizadas Informação é encriptada no armazenamento e em trânsito
Ivar *Firmware*	Encriptação no armazenamento dos arquivos de atualização Validação dos arquivos antes da atualização
Utils Módulo de apoio	Manter lista de serviços oferecidos em portas para consulta e de assinaturas de *User Agents* comuns Construir respostas e organização de interface Validação dos arquivos antes da atualização Gerenciar paradas propositais no sistema

4.2.1 Linguagem Utilizada e Pré-requisitos

Foi escolhida a linguagem Python para o desenvolvimento da ferramenta devido à possibilidade de uso em diferentes sistemas operacionais e também pela facilidade e simplicidade de código utilizado para se realizar algumas tarefas. Além disso, a linguagem permite a importação de módulos e *scripts* para serem utilizados no desenvolvimento. A versão 2.7.12 foi escolhida para permitir os testes em SSLv2 e SSLv3 de forma mais controlada, pois a possibilidade de utilizar esses dois protocolos foi retirada da biblioteca padrão de SSL na versão 2.7.13. Para se executar os testes é necessário também a instalação dos pacotes abaixo bem como suas dependências:

- BeautifulSoup
- Crawler (já incluído)
- OpenSSL
- python-magic
- Scapy
- uefi-firmware

4.2.2 Björn - Módulo de Interface *Web*

Neste módulo, a ferramenta testa um serviço de rede HTTP ou HTTPS com uma interface fornecida para utilização, sendo necessária assim a inserção do endereço ou IP de acesso ao serviço, que será responsável por identificar a interface a ser testada. Antes de realizar os testes, é necessária uma verificação de disponibilidade, podendo esta ser feita através de uma tentativa de acesso automatizada.

Teste de conexão inicial

De posse o endereço de acesso, a ferramenta realiza inicialmente uma verificação através de um pedido de conexão com o servidor. No caso de o usuário não ter entrado com a porta de acesso, o sistema automaticamente infere como 80 por ser o padrão utilizado [41]. A forma mais simples de se realizar essa conexão é através de uma mensagem de *Request* com o padrão HTTP/1.1.

Uma *Request* deve incluir em suas informações o tipo de requisição através da definição de um método [42]. Sendo possíveis os seguintes tipos: OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE e CONNECT. Na ferramenta foi feita a utilização do método GET por permitir simular uma conexão em que a resposta possui informações como hora de conexão, servidor utilizado e outros dados que possam auxiliar na análise dos testes [1]. A resposta do servidor é então mostrada conforme a Figura 4.1.

```
=============================================================
[!] Interface esta online.
 www.aaa.com.br
- Hora:          Tue, 14 Nov 2017 02:03:40 GMT
- IP:            012.34.888.88
- Hostname:      www.aaa.com.br
- Porta:         80 [http]
- Servidor:      cloudflare-nginx
=============================================================
```

Figura 4.1: Informações de Conexão Fornecidas pela Ferramenta.

Em caso de falha da conexão, seja por erro de *socket* ou problema na requisição, é motrada uma mensagem informando que a interface não respondeu à soliticação, inferindo que esta não está disponível ou que a URL ou o IP fornecido é inválido.

[1] Na transmissão de informações mais críticas não é recomendada a utilização do método GET

Teste Básico de HTTPS

Também é executado em forma de *Request*, dessa vez inferindo o serviço fornecido como sendo HTTPS e modificando o método GET para atender à solicitação. Caso a porta de acesso não tenha sido fornecida, é inferida como 443 por ser a porta padrão [43]. Nesse teste, o esperado é o *Status-Code* 200 como resposta, indicando que o GET foi aceito e atendido corretamente pelo servidor.

Em caso de sucesso, a ferramenta exibe uma mensagem confirmando o uso de HTTPS conforme mostrado pela Figura 4.2. Se ocorrer algum erro é mostrada em tela a resposta obtida na tentativa de conexão para que, com ela, seja possível verificar o problema que inviabilizou a requisição.

Proteção contra *ClickJacking*

Ataques de *Clickjacking* utilizam *frames* inseridos em páginas e que se apresentam como elementos escondidos na interface da aplicação. Também conhecido como UI Redressing [44], esse tipo de fraude tenta influenciar o usuário a clicar em uma janela do site, fazendo uso de permissões que o próprio endereço acessado fornece para produzir resultados fora do contexto da página.

Huan et. al [45] classificam esse ataque em três tipos de acordo com a forma que eles fazem o usuário clicar em comandos fora do contexto do site: (1) comprometendo a integridade de visualização ao impedir que quem acessa o domínio consiga reconhecer o elemento alvo antes de entrar com uma ação; (2) compromentendo a integridade do ponteiro, tornando nula a garantia que o usuário consiga utilizar o cursor normalmente; e (3) comprometendo a integridade temporal, evitando que o usuário tenha tempo suficiente para compreender o que ele está executando.

Como configuração mínima de proteção, é necessário que a interface esteja com o campo *x-frame-options* do cabeçalho configurado como DENY para proibir qualquer domínio externo de inserir conteúdos de execução externa em *frames* ou, no mínimo, como SAMEORIGIN, permitindo que somente a próprio página possa incluir esses *frames*. Há também a possibilidade de configurar para apenas alguns sites de origem específica terem essa permissão, configurando a opção como "ALLOW-FROM URI", sendo que URI é a página que será autorizada a realizar inserções, porém nem todos os navegadores *web* aceitam essa opção [46].

A ferramenta trata o caso básico, enviando um *Request* com o método GET e analisando o cabeçalho retornado pela interface. Na qual o *x-frame-options* deve estar habilitado como SAMEORIGIN ou DENY. É emitido um alerta para o caso de não estar configurado com uma dessas duas opções. E no caso de estar, é mostrada uma mensagem validando o teste conforme a Figura 4.2

```
==============================================================
[1] Uso de HTTPS
Resposta: <Response [200]>
Https OK
==============================================================
[2] Protecao contra Clickjack/UI Redress
Resposta: <Response [200]>
X-Frame-Options OK
==============================================================
```

Figura 4.2: Verificação de HTTPS e ClickJacking.

Enumeração de páginas de autenticação

Interfaces de administração de conteúdos devem estar protegidas contra fácil acesso por parte de pessoas não autorizadas. A localização desse tipo de conteúdo na organização da página deve ser feita de forma a coibir e limitar a possibilidade de que certas vias de acesso providas possam ser utilizadas para testes, diminuindo a superfície de ataque.

Páginas e caminhos com nomes comuns como: *admin*, *login*, *user*; padrões utilizados em linguagens com PHP ou ASP; e painéis de administração de *Frameworks* para *blogs* como o Wordpress tornam a enumeração de interfaces de acesso muito mais fácil [36]. Isso possibilita a um atacante analisar uma quantidade maior de entradas procurando por possíveis falhas.

Para verificar a possibilidade de se realizar enumeração, a ferramenta utiliza uma lista de páginas que aborda os casos apresentados como comuns e, para cada caminho, envia um *Request* para o endereço "URL+caminho". Nos os casos em que a resposta for *Status-Code* 200, conclui-se que a presença do caminho em questão foi verificada e confirmada, incluindo assim essa informação a uma lista de caminhos comuns encontrados (tratada aqui como LCE para referências posteriores). Há também o caso de a resposta ser *Status-Code* 302, que indica redirecionamento para outra página, denotada aqui como URL2. Neste caso, o subdomínio da URL2, relativo ao novo caminho testado, é verificado e então adicionado à LCE caso seja válido.

Ao fim da análise de todos os caminhos, no caso de pelo menos um teste ter retornado sucesso, é apresentada uma mensagem informando que páginas de adminstração foram encontradas e também uma lista com todos os caminhos que retornaram sucesso conforme pode ser visto na Figura 4.3. Caso nenhuma página comum tenha sido encontrada, o sistema emite um aviso informando o resultado sem uma lista para mostrar.

```
==============================================================
[3] Paginas de Login e Admin comuns
[+] Escaneando  www.aaa.com.br...
Resposta: Caminhos comuns encontrados!
[+] Paginas de Login contem url comum:
[||]     /admin/
[||]     /login/
==============================================================
```

Figura 4.3: *Scan* de Caminhos e Páginas Comuns de Autenticação.

Vulnerabilidade a Injeção SQL

Ataques de Injeção SQL (SQLi) representam um grande risco para páginas e conteúdos disponibilizados na Internet por permitirem que atacantes tenham acesso a bancos de dados com informações importantes, sendo comumente descritos como uma das maiores ameaças para aplicações *Web* [47] [48]. O primeiro ataque desse tipo foi reportado por Jeff Forristal em um artigo para a Phrack Magazine no ano de 1998 [49] e desde então diversos ataques a grandes bases de dados aconteceram, com estudos mostrando uma média de até 71 tentativas de Injeção SQL por hora em determinados períodos [50].

Esse tipo de ataque é realizado através da tentativa de inserção de requisições SQL para serem executadas. Zhendong Su e Gary Wassermann [51] definem como o foco desses ataques, aplicações *web* com as seguintes características: recebimento de cadeias de caracteres como entrada; geração de uma sequência de caracteres através da combinação de constantes e valores digitados em campos de entrada; requisições sendo gerada sem o respeito à gramática SQL; a requisição gerada não provê informação a respeito da fonte de seus caracteres e subcadeias. A ferramenta trata a forma mais comum desse ataque, na qual o atacante se utiliza de interfaces onde há entradas a serem inseridas pelo usuário e que são transmitadas em submissões de formulários *Web* via HTTP GET ou POST, aproveitando-se do fato de eles executarem requisições em banco para autenticar os dados fornecidos na entrada ou para inserir informações na página.

Injeções SQL podem ser evitadas verificando o tipo de entrada inserida em campos da página para evitar que os caracteres sejam tratados como algum tipo de *token* da gramática de banco, codificando as cadeias de caracteres para que símbolos especiais sejam transformados em caracteres normais e analisados dessa forma pela aplicação, utilizando ferramentas de casamento de parâmetros para verificar quais entradas são consideradas como "boas" [52] e, por último, devem-se identificar todas as entradas da aplicação e qual a influência delas no fluxo de acesso, para reduzir assim a quantidade de vetores de ataque.

Como forma de verificar se há alguma vulnerabilidade a essas ameaças, são utilizas pela ferramenta cadeias padronizadas que têm o objetivo de gerar efeitos adversos na aplicação ao serem inseridas em uma requisição. As cadeias utilizadas representam tentativas de se obter erros em situações específicas, sendo possível a inserção de mais cadeias no arquivo de teste para que elas também sejam executadas. A LCE obtida anteriormente é utilizada aqui como a base de caminhos de autenticação que serão testados, sendo também oferecida na ferramenta a opção de incluir novos caminhos ao teste.

```
==============================================================
[4] SQL Injection
Deseja incluir algum caminho para teste?:
1 - Sim
2 - Nao
2
www.aaa.com.br
Testando / - Campos: author viewport description keywords msa
pplication-TileColor msapplication-TileImage msapplication-co
nfig theme-color google-site-verification
Testando /admin/ - Campos: robots username password next
Testando /login/ - Campos: author viewport description msappl
ication-TileColor msapplication-TileImage msapplication-confi
g theme-color google-site-verification username password
Resposta: Website is not classic SQL injection vulnerable
Tratamento de SQL Injection OK
==============================================================
```

Figura 4.4: Teste de Injeção SQL Indicando Campos e Caminhos Testados.

Para cada caminho da LCE, a ferramenta, num primeiro passo, verifica a disponibilidade de conexão através de um *Request* e analisa o código-fonte da página para encontrar os campos de formulário existentes. Uma lista de campos é então criada e testes de Injeção SQL são executados para cada um desses campos. Os resultados são obtidos através de envios de formulário que são simulados com novos pedidos de *Request*, nos quais os compo-

nentes identificados nos formulários são preenchidos com as cadeias de caracteres definidas para testes. Um exemplo pode ser visto na Figura 4.4, no qual são identificados e testados cada elemento de formulário da interface. As respostas obtidas são analisadas para encontrar sinal de algum gerenciador de banco de dados, do inglês *Database-Management System (DBMS)*, indicando assim que a cadeia de caracteres inserida gerou uma SQLi.

Vulnerabilidade a XSS

Esse tipo de ataque é considerado em muitos casos como sendo persistente, ou seja, a entrada fornecida pelo atacante é mantida na página. São utilizados campos de comentários e caixas de mensagem que permitem escrever informações que serão enviadas para o *site*. Após um comentário ter sido submetido, ele é mostrado normalmente como parte do conteúdo de uma página *web*. Porém, se o texto informado não for devidamente analisado, é possível que comandos possam ser inseridos com o objetivo de mesclá-los ao código da interface, gerando efeitos futuros no momento que algum usuário tente acesso àquela página comprometida. Algumas formas desse ataque se utilizam somente de URLs modificadas da página alvo para enganar o usuário, não armazenando assim nenhuma informação permanente na base de dados.

```
==============================================================
[5] XSS - Cross-site Scripting
Escolha um metodo:
1 - Inserir os caminhos manualmente
2 - Buscar os caminhos recursivamente (pode demorar)
2
Testando /static/docs/
Testando /perfil/registre-se/
Testando /sobre/projeto/
Testando /termos-de-uso/
Testando /apps/
Testando /login/
Testando /
Testando /contato/
Testando /perfil/minha-conta/
Resposta:
Tratamento de XSS OK
==============================================================
```

Figura 4.5: Resultado de um Teste de XSS Realizado na Interface, com o Detalhamento dos Caminhos.

A ferramenta dá a possibilidade de inserção manual de caminhos nos quais deseja testar a vulnerabilidade ou que seja realizada uma busca recursiva na interface informada. E, para cada um desses caminhos, tentativas de inserção de *scripts* funcionais são realizadas através de *Requests*, utilizando um arquivo base com várias codificações de um mesmo *script*, sendo possível a inserção de outros. Cada resultado é analisado e uma verificação é realizada para identificar se comandos insertos no conteúdo do *site* estão sendo tratados ou inseridos diretamente. Um exemplo de teste pode ser visto na Figura 4.5. Após todas as verificações, é informado o resultado e se foi identificado algum efeito permanente na página; por meio dessa análise, pode-se saber se a interface está comprometida

Uso de *Web Application Firewall*

Web Application Firewall é uma solução que realiza a proteção de aplicações sem depender do modo de desenvolvimento utilizado por elas. É uma primeira linha de defesa contra alguns ataques por conseguir filtrar tentativas de CSRF, Injeção SQL e outros antes que eeles cheguem à interface [53]. Há várias soluções no mercado e isso dificulta uma detecção que funcione de forma padronizada. A solução encontrada foi utilizar os padrões de flags de comunicação do protocolo TCP (acrônimo para o inglês *Transmission Control Protocol*) e analisar mensagens enviadas em uma sequência errada para analisar a resposta do servidor a esses pacotes.

A ferramenta verifica a existência de algum tipo de filtro mais avançado na conexão com a interface, o que indica a existência de algum mecanismo de proteção intermediário gerenciando requisições. Para realizar essa verificação, um pacote de pilha de protocolos TCP/IP é montado e enviado com a *flag* ACK configurada, confirmando erroneamente algum tipo de recebimento [54]. E se houver um filtro básico de pacotes ou filtro nenhum, um pacote TCP com a *flag* RST configurada será obtido como resposta, indicando que o servidor recebeu e processou o primeiro pacote. O que mostra que não há um WAF intermediando a conexão e analisando o que é recebido.

Caso o pacote recebido como resposta seja um ICMP (sigla para o inglês *Internet Control Message Protocol*) do tipo 3 com código 1, 2, 3 [55], 9, 10 [56] ou 13 [57], significa que um WAF bloqueou o pedido por não haver uma comunicação prévia entre as partes conforme definido no padrão de mensagens do protocolo TCP.

4.2.3 Ubba - Módulo de Serviço de Rede

Neste módulo, a ferramenta verifica os serviços disponibilizados pelo servidor, sendo necessária somente a inserção do endereço ou IP de acesso ao *host* em que se deseja executar os testes.

O primeiro passo para realização dos testes é a definição dos serviços que devem ser analisados, identificando as portas abertas e filtradas. Na ferramenta, isso pode ser feito através de uma inserção manual ou busca automática das portas de acesso. Caso opte-se pela segunda, um pacote de pilha de protocolos TCP/IP é montado e enviado com a *flag* SYN configurada, indicando um pedido para iniciar uma conexão TCP [54], e a resposta recebida é então analisada. Caso a porta esteja aberta, o retorno será um pacote TCP com as *flags* SYN e ACK configuradas, indicando aceitação. A ferramenta então envia um pacote com a *flag* RST para encerrar a conexão antes de ser completamente configurada, pois já há confirmação de que a porta está operando. Caso a porta esteja fechada, o retorno recebido será um pacote TCP com a *flag* RST configurada. Por último, no caso de a porta estar aberta com uma conexão filtrada impedindo o uso, o pacote recebido será um pacote ICMP do tipo 3 com código 1, 2, 3, 9, 10 ou 13. Após essa análise, é gerada uma tabela indicando as portas abertas, os serviços que estão rodando nelas e o estado da porta em questão, sendo feita inicialmente uma tentativa de análise das portas 80 de 443 para indicar o servidor que está sendo executado. Um exemplo dos resultados fornecidos é indicado na Figura 4.6.

É importante que portas não necessárias estejam desabilitadas. E um problema recorrente é que portas previamente habilitadas para serviços de teste continuam abertas após a finalização destes, permitindo acessos indevidos à infraestrutura e aos dados armazenados pelo dispositivo. Para sanar esse problema, deve-se fazer uma verificação automática de serviços disponíveis e analisar os resultados para verificar quais estão sendo disponibilizados erroneamente.

```
=============================================================
Escaneando portas...
Porta           Servico                 Estado
22              ssh                     Aberta
80              http                    Aberta
Tempo de scan:  0:05:50.777000
=============================================================
[1] Analise do servidor pelas portas [Banner Grabbing]
Obtendo informacoes dos servicos disponiveis
[+] Port 22 [ssh]:                      Open
[+] Port 80 [http]:                     Open
 -  Servidor: Apache/2.4.7 (Ubuntu)
=============================================================
```

Figura 4.6: Resultado da Verificação de Portas e da Análise dos Servidor.

Teste de Resistência a Ataques de Negação de Serviço (DoS)

Ataques de negação de serviço (conhecidos como DoS) são caracterizados pela tentiva de se negar o uso legítimo de um serviço. Um ataque de negação de serviço distribuído (DDoS) utiliza de várias entidades atacantes para realizar esse objetivo. Há ainda métodos que se utilizam de redirecionamentos intermediários para potencializar e amplificar a capacidade do ataque, inclusive com possibilidade de abuso da infraestrutura fornecida pela IoT [58].

Uma das formas mais comuns de se realizar um ataque DDoS é enviando um fluxo grande de pacotes para a vítima de forma a inabilitar o acesso de clientes legítimos ao provedor. Outra forma comum de se executar esse ataque é enviar alguns pacotes deformados de forma a confundir a aplicação ou o protocolo, causando uma parada ou forçando o sistema a reiniciar. O importante em um ataque desses é que ele sobrecarrega o servidor e limita o acesso de cliente reais aos serviços que são providos.

Aqui é testada a resiliência do serviço em questão e a quantidade de pedidos escusos que são respondidos. Para isso, dois tipos de ataque de negação de serviço são executados em todas as portas habilitadas. O primeiro utiliza uma técnica de HTTP Flood com encadeamento de execução (*Multithreading*). HTTP Flood é um tipo de DDoS que gera pacotes quase válidos de requisições GET ou POST para produzir volume de tráfego ou qualidade de ataque, é considerado um ataque de camada 7 pois atua direto na camada de aplicação com o protocolo HTTP [59].

Antes de realizar o ataque, a ferramenta agrega uma lista de serviços de validação de páginas HTML. Esses serviços verificam a formatação realizando diversas requisições aos sites e analisando a qualidade de código da página. Como a ferramenta opera em rede local utilizando uma só máquina, não há a possibilidade de gerar uma rede de *bots*, por isso os serviços de validação servirão para esse intuito, gerando diversas requisições e consumindo acesso do servidor. Após a agregação dos serviços de validação como *bots*, a ferramenta gera diversas *threads* e inicia uma lista com requisições GET a serem enviadas. Com isso, durante um tempo pré-definido, tarefas são incluídas nessa lista e a ferramenta distribui aleatoriamente os *bots* nessas tarefas. Após o fim do tempo definido, a lista é mantida até que todas as requisições sejam atendidas. Por fim, é mostrada uma comparação da quantidade de tentativas com quantas foram atendidas, para indicar a qualidade do filtro utilizado pelo servidor.

O segundo ataque, chamado de SYN Flood, atua na camada de transporte e faz uso do protocolo TCP. Pelo padrão do protocolo, para se realizar uma conexão, é necessário que o cliente envie um pacote SYN solicitando uma conexão. O servidor então responde com um SYN-ACK indicando que permite a comunicação e aguarda a outra mensagem com a resposta do cliente confirmando a conexão. Porém, caso o SYN-ACK não seja respondido, o servidor continua processando o pedido e aguardando a confirmação, enviando

periodicamente novas confirmações até atingir um limite. O SYN Flood se aproveita desse detalhe e envia pacotes com dados de remetente falsos para que o servidor fique processando solicitações e enviando respostas a falsos endereços. Com isso, a ferramenta gera pacotes SYN com dados falsos e durante um tempo predetermiando, envia esses pacotes para o servidor. No final, o tempo de resposta antes e depois do ataque é verificado e mostrado em tela para que a resiliência do serviço provido possa ser analisado. Esses testes são feitos para cada porta e os resultados são mostrados conforme a Figura 4.7.

```
=============================================================
[2] Teste de resistencia a DDOS
Iniciando ataque

[+] Porta 00
[|] Tempo de resposta inicial: 0:00:00.206000
[|] HTTP Flood: Realizando DDoS... 0 tentativas, 0(0%) com sucesso.
[|] - Tempo de resposta apos o flood: 0:00:00.203000
[|] Syn Flood: Realizando DDoS... 0 pacotes/segmentos enviados
[|] - Tempo de resposta apos o flood: timed out
=============================================================
```

Figura 4.7: Testes de Resistência a DDoS Utilizando HTTP Flood e SYN Flood.

Teste de Fuzzing

Fuzzing é uma forma de se descobrir problemas nos serviços fornecidos através de entradas aleatórias inseridas em campos ou métodos da aplicação [60]. O objetivo é gerar algum erro crítico, uma parada no serviço ou exceções no fluxo. É um método de se descobrir problemas cruciais em aplicações e testar a robustez de sua segurança. Tem a vantagem de produzir bons resultados com pouco esforço envolvido.

O método de *Fuzzing* aplicado na ferramenta envolve utilizar comandos comuns dos protocolos conhecidos e inserir *strings* aleatórias junto a eles em várias sequências de caracteres. Assim, um *socket* é criado e uma conexão entre o dispositivo e a ferramenta é estabelecida. Essas sequências são então enviadas e ampliadas em tamanho com caracteres aleatórios após cada tentativa para gerar ainda mais estresse no servidor. É verificado se o alvo continua respondendo às solicitações normalmente ou se a conexão foi perdida. Nesse teste é importante que o sistema a ser testado esteja acessível e visível para que qualquer tipo de erro ou exceção possa ser identificado no momento em que ocorrer. As informações geradas no final dos testes são informadas conforme pode ser visto na Figura 4.8.

```
==============================================================
[3] Teste de fuzz com strings montadas e strings aleatorias

[+] Porta 80:
[|] Insira o comando a ser testado: GET
Porta 80: 3 erros consecutivos com string de tamanho 5
Excecao: timed out

Uma das portas nao respondeu aos requests
==============================================================
```

Figura 4.8: Exemplo de teste de *Fuzzing*.

4.2.4 Hvitserk - Módulo de Encriptação

Neste módulo, o objetivo é a análise dos métodos de proteção utilizados para garantir que a comunicação entre os dispositivos está sendo feita de forma encriptada, que boas práticas de proteção estão sendo utilizadas e que a informação está segura no momento da transmissão. Aqui, são analisados os protocolos permitidos na comunicação segura, qual é o padrão utilizado pelo serviço em uma conexão e o tamanho da cifragem utilizada. Além disso são feitos testes para verificar se a transmissão faz uso do Perfect Foward Secrecy (PFS) para evitar que vazamentos gerem problemas de segurança em transmissões anteriores.

Os únicos dados necesários para os testes são o endereço do servidor ao qual se deseja direcionar as análises e a porta do serviço que deverá ser testado. Sendo que, antes da realização dos testes, é feita uma verificação para atestar a disponibilidade do sistema através do envio de um pacote TCP/IP para a tupla (*host*,porta) com a *flag* SYN configurada.

Análise dos Protocolos Utilizados e da Priorização Dada pelo Servidor

O primeiro teste realizado é a verificação do uso de protocolos de segurança. Sendo analisados tanto o uso de protocolos Secure Sockets Layer (SSL) quanto de seu sucessor, *Transport Layer Security (TLS)*. E devido às vulnerabilidades encontradas nos protocolos SSLv2 [61] e SSLv3 [62], o uso dos mesmos é considerado uma falha de segurança grave e será tratado como problemático na análise.

A ferramenta inicialmente checa o uso dos protocolos vulneráveis (SSL) e, devido às inconsistências de segurança desses, indica que há um problema caso algum dos dois seja utilizado. Após isso, os protocolos TLS1, TLS1.1 e TLS1.2 são verificados e uma tabela de análise dos protocolos disponíveis é criada. Indicando quais protocolos devem ser evitados e quais devem ser utilizados.

Após a exibição dessa tabela, uma nova conexão é criada, desssa vez deixando a cargo do servidor escolher o protocolo para identificar quais métodos serão priorizados. Por fim, é mostrado o protocolo em que a comunicação foi negociada, junto da cifra e do tamanho em bits. Esses dados são então armazenados em uma Lista de Protocolos Habilitados, chamada aqui de LPH para referência mais à frente. Um exemplo dos resultados fornecidos pode ser visto na Figura 4.9.

```
=============================================================
[1] Testando Protocolos
Protocolo        Status          Tipo
- SSLv2          Nao oferecido   NAO SEGURO
- SSLv3          Nao oferecido   NAO SEGURO
- TLSv1          Oferecido       -
- TLSv1_1        Oferecido       -
- TLSv1_2        Oferecido       RECOMENDADO
=============================================================
[2] Protocolo padrao do servidor
Preferido: TLSv1.2
Cifra: ECDHE-ECDSA-AES128-GCM-SHA256
Tamanho em bits: 128
=============================================================
```

Figura 4.9: Lista de Protocolos Utilizados e Análise do Protocolo Padrão do Servidor.

Uso de *Perfect Foward Secrecy (PFS)*

Perfect Foward Secrecy (em português,"Sigilo Encaminhado Perfeito") é um propriedade de protocolos de comunicações seguros que protege senhas utilizadas anteriormente de serem afetadas pelo vazamento de novas senhas geradas [63]. Essa característica pode ser obtida quando um protocolo de segurança realiza a produção de chaves de autenticação sem a utilização de um algoritmo determinístico em sua suite de cifras.

Uma suite de cifras é um conjunto de algoritmos utilizados para garantir a segurança em uma comunicação. Sendo geralmente composta por: um protocolo de comunicação (TLS ou SSL); uma cifra para troca de chaves; uma cifra de fluxo, uma cifra de autenticação; e uma função de *Hash*. Em TLS, há a possibilidade de uso de implementações fundamentadas em troca de chaves Diffie-Hellman (DHE-RSA, DHE-DSA) e em Diffie–Hellman baseados em curva elíptica (ECDHE-RSA, ECDHE-ECDSA) para se ter características de PFS [64] em uma suite de cifras.

A ferramenta testa a utilização de métodos de troca de chaves que respeitem a utilização de PFS através de uma lista de suites de cifras com os respectivos protocolos utilizados na conexão. Para cada suite de cifras, uma tentativa de conexão é realizada através da tupla (suite, protocolo). A cada teste, uma verificação é feita para analisar se a conexão foi feita de forma correta e se o servidor aceitou que a encriptação fosse feita utilizando aquela suite. Após a execução dos testes, uma lista com as cifras aceitas é gerada para permitir uma análise de se o serviço utiliza pelo menos algum tipo de *Perfect Forward Secrecy* conforme mostrado na Figura 4.10.

```
=================================================================
[3] Uso the Perfect Forward Secrecy
PFS habilitado com as cifras: ECDHE-RSA-AES128-GCM-SHA256,
ECDHE-RSA-AES256-GCM-SHA384
=================================================================
```

Figura 4.10: Resultado de uma Análise do uso de PFS.

Teste de Cifragem Fraca

É necessário assegurar que a suite de cifras utilizada garanta autenticação e encriptação de 128 bits ou maior, pois cifras fracas (entre 40 e 56 bits) utilizam um tipo de criptografia facilmente quebrável [65]. E para verificar a força das cifras utilizadas, a ferramenta faz uso da LPH gerada na análise inicial e, a cada item da lista, realiza uma tentiva de conexão na tupla (*host*, porta) utilizando o protocolo informado e avaliando tanto a suite de cifra utilizada quanto o tamanho da autenticação e da encriptação permitidas pela suite. Após essa análise, é gerada uma lista com os resultados obtidos e se os parâmetros analisados cumprem com os requisitos mínimos de segurança estabelecidos. Essa representação pode ser visualizada na Figura 4.11.

```
=====================================================================
[4] Testando Cifragem Fraca  (minimo 128bits)
Protocolo        Tamanho da Cifra        Cifra
TLSv1            128 bits(OK)            ECDHE-ECDSA-AES128-SHA
TLSv1.1          128 bits(OK)            ECDHE-ECDSA-AES128-SHA
TLSv1.2          128 bits(OK)            ECDHE-ECDSA-AES128-GCM-SHA256
```

Figura 4.11: Resultado do Teste de Cifragem Fraca.

4.2.5 Ivar - Módulo de *Firmware*

Firmware é definido pela IEEE *Standard Glossary of Software Engineering Terminology* como a combinação de dispositivo de hardware com instruções computacionais e dados que residem como um sistema apenas de leitura no dispositivo [66]. Por geralmente ser composto do conjunto de instruções mais básicas que operam com sistemas da Internet das Coisas, é importante analisar os erros comumente encontrados. Por essa categoria necessitar de uma verificação mais técnica dos dados, a ferramenta faz as extrações mais básicas em um arquivo de *firmware* e gera os resultados em disco para possível análise futura, testando também a possibilidade de extração do sistema completo de arquivos.

É necessário somente informar à ferramenta o arquivo binário com o *firmware* e incluir na mesma pasta que este o arquivo de soma de verificacação (*checksum*), que é utilizado para confirmar a integridade dos dados, sendo geralmente fornecido pelo fabricante do dispostivo ou pelo desenvolvedor do sistema. De qualquer forma, a primeira análise irá gerar a assinatura do MD5 junto com o tipo de arquivo identificado para caso seja necessária uma verificação manual. Os resultados são então apresentados conforme mostrado na Figura 4.12.

```
=============================================================
Arquivo 'abc.bin'
Tipo: application/octet-stream
MD5 signature: dc605cda27d7226bc43cad5e5cd0d34f
=============================================================
[1] Verificacao do MD5
Procurando arquivo 'abc.bin.md5'
Arquivo encontrado!
MD5:dc605cda27d7226bc43cad5e5cd0d34f Valido!
=============================================================
```

Figura 4.12: Exemplo da Primeira Análise Mostrando o Tipo do Arquivo e a Validação da Assinatura MD5.

Análise de Palavras Legíveis e Extração em Hexadecimal

O primeiro tipo de extração é feito através da análise de sequências de caracteres legíveis do arquivo, gerando um resultado em disco para caso seja útil em algum tipo de verificação do conteúdo. Para esse teste, considera-se como legível qualquer caracter que possa ser impresso em tela.

O processo é executado lendo-se todo o conteúdo do arquivo "A", catalogando as sequências válidas que possuem tamanho mínimo de 10 e gravando todas em um arquivo de saída com nome identificado como "strings-A.txt". O arquivo é então disponibilizado na pasta da ferramenta para análise.

O segundo tipo de extração é feito com a obtenção do conteúdo do arquivo e uma conversão dos dados obtidos para hexadecimal, utilizando um procedimento conhecido como *hexdump*. Os dados são então organizados em linhas, com cada linha contendo o endereço do dado extraído, a conversão das informações em hexadecimal e a representação desse conteúdo em caracteres do padrão ANSI, tornando possível a leitura do resultado em algumas partes. O conteúdo é então disponibilizado em um novo arquivo "hex-A"salvo em disco. Um exemplo do conteúdo gerado pode ser visto na Figura 4.13.

```
00000000:  01 00 00 00 4f 70 65 6e  57 72 74 00 00 00 00 00  |....OpenWrt.....|
00000010:  00 00 00 00 00 00 00 00  00 00 00 00 72 33 39 33  |............r393|
00000020:  36 35 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |65..............|
00000030:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000040:  07 03 01 01 00 00 00 01  00 00 00 00 a1 ef c9 a9  |................|
00000050:  a7 fd 37 2f 11 84 74 4d  99 5d b9 93 00 00 00 00  |..7/..tM.]......|
```

Figura 4.13: Conteúdo Gerado em um *Hexdump* nos Testes.

Extração do Sistema de Arquivos do *Firmware*

Aqui é feito um esforço para identificar o sistema de arquivos como sendo utilizando conversões conhecidas de tipo como uefi, pfs, me e flash. Primeiro, o conteúdo é extraído diretamente do arquivo, então é analisado e comparado com sistemas de arquivos conhecidos, verificando sua composição e, caso seja encontrado o tipo, o mesmo é reportado na análise. Após essa verificação, é feita uma tentativa de extração com base no que foi identificado. A extração é então disponibilizada e o conteúdo que foi obtido é mostrado para análise em caso de sucesso na extração.

4.3 Considerações Finais

Neste capítulo foram apresentados os detalhes da ferramenta Ragnar, desenvolvida para realizar testes de intrusão em dispositivos da Internet das Coisas. Inicialmente, foi feito um mapeamento das funcionalidades com uma apresentação da estruturação da ferramenta e a organização de seus módulos. Em seguida, foi descrita a linguagem e os pacotes utilizados para a sua implementação. Por último, foram detalhados o objetivo e o funcionamento de seus módulos.

No próximo capítulo serão apresentados os testes realizados para a verificação do funcionamento da ferramenta.

Capítulo 5

Resultados Obtidos

Neste capítulo serão apresentados os testes realizados para verificar o nível de detecção da ferramenta proposta neste trabalho em dispositivos da Internet das Coisas, bem como testes de comparação com outras ferramentas que realizam análises semelhantes. Para isso, a Seção 5.1 descreve os objetivos que são esperados na realização destes testes. A Seção 5.2 detalha qual o ambiente utilização para realização dos testes. Por fim, a Seção 5.3 apresenta os procedimentos utilizados e os resultados dos testes de detecção.

5.1 Objetivos dos Testes

O objetivo dos testes de deteção, apresentados na Seção 5.3, é verificar o funcionamento da ferramenta na verificação de problemas reais de segurança em dispostivos IoT. Para isso, foram realizados testes específicos para cada módulo através da execução completa de uma rotina de teste. Os testes são feitos utilizando inicialmente ambientes sem a devida segurança para pode avaliar a precisão da ferramenta, seguidos de testes com interfaces protegidas.

5.2 Ambiente de Testes

Nesta seção serão apresentados os detalhes referentes à infraestrutura utilizada para a execução dos testes presentes neste capítulo.

5.2.1 Configuração do Ambiente

Para a realização de todos os testes apresentados neste capítulo foi utilizada uma máquina-alvo baseado em um computador de placa única (abreviado como SBC, do inglês *single-board computer*) em dois cenários de configuração. Esse recurso foi escolhido por ser

bastante utilizado em soluções na Internet das Coisas [67]. Além disso, em todos os testes realizados, foi utilizada uma mesma máquina local para executar as rotinas da ferramenta. A Tabela 5.1 apresenta as configurações presentes nesta máquina e na SBC utilizada.

Tabela 5.1: Configuração dos Computadores Utilizados nos Testes

Infraestrutura	Item de Configuração	Configuração
Máquina de Origem	Processador	Intel Core i5 3.0 GHz
	Memória	8 GB de Memória RAM
	Armazenamento	1 TB de HD
	Sistema Operacional	Kali Linux
Máquina-Alvo (Cenário 1)	Placa Única	ODROID-C2
	Processador	ARM Cortex A53 1.5Ghz
	Memória	2 GB de Memória RAM
	Armazenamento	64 GB Micro-SD
	Sistema Operacional	Kali ODROID-C2
	Servidor *Web*	BaseHTTP 0.3 Python 2.7.13
	SSH	Basic SSH
	Telnet	Habilitado
Máquina-Alvo (Cenário 2)	Placa Única	ODROID-C2
	Processador	ARM Cortex A53 1.5Ghz
	Memória	2 GB de Memória RAM
	Armazenamento	64 GB Micro-SD
	Sistema Operacional	Ubuntu Mate (ODROID-C2)
	Servidor Web	Apache 2 + SQL + PHP7.0
	SSH	Secure pySSH
	Telnet	Filtrado

De forma a atestar a qualidade das análises e confirmar a usabilidade da ferramenta, no Cenário 1 foram executados testes utilizando um ambiente propositalmente configurado para que as vulnerabilidades estejam presentes. Assim, o Google Gruyere[1] foi utilizado para instanciar um servidor *Web* na Máquina-Alvo. Essa configuração permitiu um ambiente de teste amplos para módulo de Interface *Web* por possuir os seguintes problemas (entre outros):

- Vulnerabilidade a XSS
- Vulnerabilidade a SQLi
- Ausência de um *Web Application Firewall*
- Nenhuma proteção contra *Clickjacking*
- Fácil enumeração de páginas

[1] https://google-gruyere.appspot.com/

Não sendo inserida nenhuma proteção no primeiro momento para que o módulo de Encriptação possa ser executado neste serviço. Junto a isso, um servidor básico de SSH e outro de Telnet foram habilitados para possibilitar as análises do módulo de Serviços *Web*.

Já no Cenário 2, foram feitos testes utilizando configurações devidamente protegidas para avaliar os falsos-positivos que poderiam ser gerados. Um servidor Apache versão 2 foi configurado com opção de acesso seguro através também da porta 443. A X-Frame-Options foi configurada no código para permitir somente execuções a partir do próprio domínio e não foram incluídos caminhos comuns de autenticação. Além disso, as devidas precauções foram tomadas para sanitizar os problemas de Injeção SQL e execução de XSS. Um WAF simples foi configurado para barrar as requisições mais simples que pudessem ser verificadas como problemáticas ou mal intencionadas.

Por último, um arquivo de *firmware* foi utilizado em um único cenário, sendo escolhido o *plugin* Kankun Small K devido a seu extenso uso como teste em ferramentas de análise de arquivos binário.

5.3 Teste de Detecção

O objetivo dos testes apresentados nesta seção é testar as funcionalidades implementadas na ferramenta proposta por este trabalho. Estes testes foram realizados por meio de execuções dos módulos descritos, utilizando dois cenários previstos para execuções de testes de vulnerabilidade. Estas execuções englobam todas as principais funcionalidades implementadas.

Para os os cenários descritos na seção anterior, cada rotina padrão dos módulos da ferramenta foi executada seguindo os procedimentos descritos abaixo:

- Execução do módulo na interface instanciada
- Análise das requisiçoes recebidas pelo servidor na Máquina-Alvo
- Análise dos resultado obtidos na Máquina de Origem
- Comparação dos resultados

Para fins de comparação, a Máquina de Origem utilizará também as seguintes ferramentas nos testes que elas forem úteis para comparação:

- **Nmap**: para análise das portas abertas e filtradas e também dos serviços oferecidos.
- **TestSSL**: Verificação dos protocolos, preferências do servidor e cifras utilizadas.
- **Binwalk**: Análise de *firmware*

5.3.1 Interface *Web*

Todas as execuções foram realizadas na mesma interface da Máquina-Alvo, cuja configuração pode ser vista na Tabela 5.1, com a ferramenta sendo executada sempre em uma máquina origem única, também especificada na Tabela 5.1. Os resultados esperados das execuções utilizando o Cenário 1 eram:

- Teste do Uso de HTTPS: A ferramenta deve acusar a inexistência de uma interface HTTPS por não haver protocolo definido.
- Proteção contra *Clickjacking*: Deve-se emitir um aviso de que a X-Frame-Options não está devidamente configurada.
- Enumeração de Páginas de Autenticação: O caminho comum '/login' deve ser acusado.
- Injeção SQL: Um aviso de que há texto de DBMS na mensagem recebida quanto se executa um *login* deve ser emitido.
- Execução de XSS: Uma lista contendo pelo menos a pagina principal da interface como vulnerável deve ser apresentada.
- Teste de WAF: Deve-se acusar que não foi detectado um *Web Application Firewall*

De maneira semelhante, os resultados esperados das execuções utilizando o Cenário 2 eram:

- Teste do Uso de HTTPS: A ferramenta deve acusar a utilização de um protocolo quando a porta especificada for a 443.
- Proteção contra *Clickjacking*: Deve-se mostrar que X-Frame-Options está devidamente configurada.
- Enumeração de Páginas de Autenticação: Nenhum caminho deve ser acusado.
- Injeção SQL: Nenhum texto de DBMS deve ser encontrado na mensagem recebida.
- Execução de XSS: Devido ao tratamento das informações inseridas, nenhum caminho deve ser mostrado como vulnerável a XSS.
- Teste de WAF: Deve-se mostrar que a interface está filtrada por um *Web Application Firewall*

5.3.2 Serviços de Rede

Todas as execuções foram realizadas com três serviços diferentes habilitados na Máquina-Alvo: telnet na porta , cuja configuração pode ser vista na Tabela 5.1, com a ferramenta sendo executada sempre em uma máquina origem única, também especificada na Tabela 5.1. Os resultados esperados das execuções utilizando o Cenário 1 eram:

- Verificação de Portas e Análise dos Serviços: A ferramenta deve identificar a presença de serviços nas portas 22, 23 e 8008. Bem como o fato de a porta 23 estar sem a devida autenticação.
- Teste de resistência a DDoS: A ferramenta deve indicar o tempo de resposta antes e depois do envio dos pacotes, é esperado que pelo menos um ataque gere exceção de tempo de resposta estourado.
- Teste de *Fuzzing*: Deve ser mostrado que pelo menos a porta 8008 não respondeu bem aos testes com os comandos inseridos.

De maneira semelhante, os resultados esperados das execuções utilizando o Cenário 2 eram:

- Verificação de Portas e Análise dos Serviços: A ferramenta deve identificar a presença de serviços nas portas 22, 80 e 443. A porta 23 só deve ser mostrada caso seja solicitado a inclusão de portas filtradas.
- Teste de resistência a DDoS: A ferramenta deve indicar o tempo de resposta antes e depois do envio dos pacotes, é esperado que os filtros incluídos não permitam a queda dos serviços oferecidos. Diminuição no tempo de resposta é normal nesse caso.
- Teste de Fuzzing: Deve ser mostrado que pelo menos a porta 22 respondeu normalmente aos comandos incluídos e não gerou estouro de requisição.

5.3.3 Encriptação

Aqui todas requisições foram feitas utilizando o mesmo serviço na Máquina-Alvo, considerando somente no Cenário 1 a interface *web* oferecida. Já no segundo, por haver um serviço de HTTPS habilitado, será considerada como o foco dos testes a interface habilitada na porta 443. Os resultados esperados das execuções utilizando o Cenário 1 eram:

- Teste de Protocolos: Deve ser gerada uma lista onde é indicado que não há uso de nenhum protocolo de proteção no servidor. Devido a isso, também deve ser mostrado que não dá protocolo padrão.

- Uso de Perfect: Deve ser indicado que não há uso desse aparato.
- Teste Cifragem Fraca: Não será gerada uma lista de senhas pois não há um sistema de proteção.

De maneira similar, os resultados esperados das execuções utilizando o Cenário 2 eram:

- Teste de Protocolos: Deve ser gerada uma lista onde é mostrado que há uso dos protocolos TLSv1, TLSv1.1 e TLSv1.2, reportando este último como sendo o padrão utilizado .
- Uso de Perfect: Deve ser indicado que há uso desse aparato, mostrando o uso das suites de cifras ECDHE-RSA-AES-GCM-SHA256 e ECDHE-RSA-AES-GCM-SHA384.
- Teste Cifragem Fraca: Deve-se indicar que os protocolos TLSv1, TLSv1.1 e TLSv1.2 utilizam cifras para 128 bits. Com as suites sendo ECDHE-ECDSA-AES128-SHA nos dois primeiros e ECDHE-ECDSA-AES128-GCM-SHA256 no último.

5.3.4 Análise de *Firmware*

Aqui todas as análises utilizaram o *plugin* Kankun Small K junto com um arquivo de MD5 fornecido pela desenvolvedora do software. Os resultados esperados são descritos como segue:

- Identificação do Tipo de Arquivo e Análise do MD5: O arquivo deve ser corretamente descrito como 'application/octet-stream' e sua assinatura MD5 deve ser igual à fornecida pelo fabricante.
- Extração de palavras legíveis e extração em Hexadecimal: Os dados devem ser corretamente analisados e extraídos. Sendo gerados como resultado dois arquivos. O primeiro deve possuir sequências de caracteres separadas por linhas. O segundo deve possuir a codificação do conteúdo em hexadecimal, com a indicação do endereço de extração e também do conteúdo original para comparação.
- Extração do Sistema de Arquivos: A ferramenta deve gerar uma composição do sistema original de arquivo utilizado, apresentado a hierarquia e quais os dados que elas armazenam.

5.3.5 Resultados

Todos os testes de execução apresentaram os resultados esperados conforme pode ser visto na Tabela 5.2 e na Tabela 5.3. Isto comprovou que todas as funcionalidades da ferramenta operam de maneira correta.

Tabela 5.2: Resultados dos Testes Executados em cada Cenário

Resultados dos Testes em cada Cenário		
Módulo	Cenário 1	Cenário 2
Björn Interface *Web*	HTTPS não utilizado X-FRAME-OPTIONS ausente Caminho '/login' encontrado Texto de DBMS encontrado Caminhos vulneráveis a XSS WAF não encontrado	HTTPS habilitado na porta 443 X-FRAME-OPTIONS válida Nenhum caminho encontrado SQLi não gerou texto de DBMS Nenhum caminho vulnerável Possível WAF encontrado
Ubba Serviços de Rede	Portas 22, 23 e 8008 detectadas Porta 23 sem autenticação *Timeout* nos testes de DDoS *Fuzzing* gerou *timeout* na 8008	Portas 22, 80 e 443 detectadas Serviços protegidos DDoS não gerou *timeout* Fuzzing bem tratado
Hvitserk Encriptação	Protocolos seguros indisponíveis Sem protocolo padrão Sem PFS Lista de senhas indisponível	TLS v1, v1.1 e v1.2 habilitados Protocolo padrão é o TLSv1.2 Cifras de PFS habilitadas Proteção de 128 bits

Tabela 5.3: Resultados dos Testes de *Firmware*

Resultados dos Testes de *Firmware*	
Módulo	Resultados
Ivar *Firmare*	Tipo de arquivo encontrado: 'application/octet-stream' MD5 confere com o fornecido Arquivo de sequências de caracteres legíveis gerado Arquivo com conteúdo em hexadecimal gerado corretamente Hierarquia do sistema de arquivos original identificada

5.4 Considerações Finais

Neste capítulo foram apresentados os testes realizados para comprovar o funcionamento da ferramenta, comparando o resultado à saída de outras ferramentas que cumprem escopos semelhantes, mesmo que parcialmente.

Assim, com os resultados dos testes presentes neste capítulo, comprovou-se que os objetivos propostos foram alcançados, sendo estes uma melhoria na análise de dispositivos IoT através da centralização das funcionalidades em uma só ferramenta e do desenvolvimento sendo feito com a Internet das Coisas em mente.

Capítulo 6

Conclusão e Trabalhos Futuros

Neste trabalho foi proposta uma ferramenta de análise de vulnerabilidades para dispositivos da Internet das Coisas, a qual recebeu o nome de Ragnar. O objetivo era centralizar os testes e dar praticidade aos desenvolvedores. Utilizando-se da lista da OWASP, uma análise foi feita e os métodos viáveis foram aplicados para fornecer um produto utilizável e prático.

A finalidade no desenvolvimento da ferramenta foi integrar as diversas análises necessárias em uma só interface, tendo como base projetos já estabelecidos e confiáveis, bem como projetos que permitiram uma forma rápida de se realizar testes em produtos e sistemas. Com isso, a ferramenta garante tanto uma análise básica de dispostivos seguros quanto uma visão mais geral dos problemas e de como tratá-los.

Para isso, foi realizado um estudo detalhado acerca dos maiores problemas relatados pelos grandes projetos de segurança, sendo feita uma análise de seus pontos negativos e positivos, para que fosse escolhido o projeto utilizado para servir de base para essa implementação. Além disso, foram realizados testes com a ferramenta, nos quais foram verificados com sucesso as análises e os resultados obtidos, garantindo a centralização de informações em um ambiente de testes mais preciso e coeso, já que agoora os testes podem ser orientados para a Internet das Coisas. Assim, foram realizados testes para medir o desempenho da ferramenta, que apresentaram resultados indicando que a precisão do serviço prestado é muito semelhante à de ferramentas parecidas.

Como trabalhos futuros, sugere-se o desenvolvimento de novas análises em cada módulo, como testes de vulnerabilidades dos protocolos a ataques mais específicos e o aprimoramento do módulo de *firmware* para suportar mais tipos de arquivos. Por fim, sugere-se que sejam integrados na ferramenta resultados de testes de outros serviços de Pentest já disponibilizados, com comparação dos resultados obtidos.

Referências

[1] Luigi Atzori, Antonio Iera e Giacomo Morabito: *The internet of things: A survey.* Comput. Netw., 54(15):2787–2805, outubro 2010, ISSN 1389-1286. http://dx.doi.org/10.1016/j.comnet.2010.05.010. x, 6, 7, 8

[2] Manos Antonakakis, Tim April, Michael Bailey Matt Bernhard Elie Bursztein Jaime Cochran Zakir Durumeric J. Alex Halderman Luca Invernizzi Michalis Kallitsis Deepak Kumar Chaz Lever Zane Ma Joshua Mason Damian Menscher Chad Seaman Nick Sullivan Kurt Thomas e Yi Zhou: *Understanding the mirai botnet.* Em *Proceedings of the 26th USENIX Security Symposium*, 2017. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis. x, 12

[3] Kevin Ashton: *That 'internet of things' thing.* The RFID Journal, 2009. 5

[4] Roberto Minerva, Abiy Biru: *Towards a definition of the internet of things.* maio 2015. 6

[5] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini e Imrich Chlamtac: *Internet of things: Vision, applications and research challenges.* Ad Hoc Networks, 10(7):1497–1516, 2012, ISSN 15708705. http://dx.doi.org/10.1016/j.adhoc.2012.02.016. 6

[6] Ioan Toma, Elena Simperl e Graham Hench: *A joint roadmap for semantic technologies and the internet of things.* Proceedings of the Third STI Roadmapping Workshop, janeiro 2009. 7

[7] Graham Klyne, Jeremy J. Carroll: *Resource description framework (rdf): Concepts and abstract syntax.* http://www.w3.org/TR/rdf-concepts/, acesso em 25/05/2017. 7

[8] M. Botterman, for the European Commission Information Society e Networked Enterprise RFID Unit – D4 Media Directorate General: *Internet of things: An early reality of the future internet.* Report of the Internet of Things Workshop, maio 2009. 8

[9] M. Presser, A. Gluhak: *The internet of things: Connecting the real world with the digital world.* EURESCOM mess@ge – The Magazine for Telecom Insiders, vol. 2, 2009. http://www.eurescom.eu/message. 8

[10] A. Dunkels, J.P. Vasseur: *Ip for smart objects.* Internet Protocol for Smart Objects (IPSO) Alliance, White Paper 1, 2008. http://www.ipso-alliance.org. 8

[11] N. Gershenfeld, R. Krikorian, D. Cohen: *The internet of things.* Scientific American, 291(4):76–81, 2004. 8

[12] Caron, Xavier, Rachelle Bosua, Sean Maynard e Atif Ahmad: *The internet of things (iot) and its impact on individual privacy: An australian perspective.* 32, dezembro 2015. 10

[13] Ross Anderson: *Security engineering.* John Wiley & Sons, 2008. 10

[14] Wired: *Friday's east coast internet outage is a major ddos attack.* https://www.wired.com/2016/10/internet-outage-ddos-dns-dyn/, acesso em 12/05/2017. 11, 18

[15] OTA: *Online trust alliance.* https://otalliance.org/, acesso em 02/06/2017. 14

[16] OTA: *Iot trustworthy working group (itwg).* https://otalliance.org/initiatives/internet-things, acesso em 02/06/2017. 14

[17] OTA: *Iot security privacy trust framework v2.0.* https://otalliance.org/system/files/files/initiative/documents/iot_trust_framework2.1.pdf, acesso em 02/06/2017. 14

[18] Joshua Corman, Stacey Higginbotham e: *Podcast - episode 2: Is it too late to secure the internet of things?*, 2015. Acesso em 02/06/2017. 15

[19] Joshua Corman: *Background and history | i am the cavalry.* https://www.iamthecavalry.org/about/history/, Acesso em 02/06/2017. 15

[20] S. E. Shladover, J. Petit e: *Potential cyberattacks on automated vehicles.* IEEE Transactions on Intelligent Transportation Systems, 16(2):546–556, abril 2015, ISSN 1524-9050. 15

[21] J. Lewis: *A smart bomb in every garage? driverless cars and the future of terrorist attacks.* https://www.start.umd.edu/news/smart-bomb-everygarage-driverless-cars-and-future-terrorist-attacks, acesso em 21/03/2017. 15

[22] M. Dohler, I. Vilajosana, X. Vilajosana e J. Llosa: *Smart cities: An action plan.* Em *Proc. Barcelona Smart Cities Congress*, páginas 1–6, dezembro 2011. Barcelona, Espanha. 15

[23] David Niewolny: *How the internet of things is revolutionizing healthcare.* Freescale Semiconductors, outubro 2013. 16

[24] S. D. T. Kelly, N. K. Suryadevara e S. C. Mukhopadhyay: *Towards the implementation of iot for environmental condition monitoring in homes.* IEEE Sensors Journal, 13(10):3846–3853, outubro 2013, ISSN 1530-437X. 16

[25] OWASP: *2014 owasp project handbook.* https://www.owasp.org/images/d/d8/PROJECT_LEADER-HANDBOOK_2014.pdf, acesso em 21/03/2017. 16

[26] OWASP: *Owasp internet of things project.* https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project, acesso em 21/03/2017. 16

[27] OWASP: *Top iot vulnerabilities.* https://www.owasp.org/index.php/Top_IoT_Vulnerabilities, acesso em 21/03/2017. 16, 22

[28] OWASP: *Tester iot security guidance.* https://www.owasp.org/index.php/IoT_Testing_Guides, acesso em 21/03/2017. 16, 22

[29] OWASP: *Official owasp top 10 document repository.* https://github.com/OWASP/Top10, acesso em 21/03/2017. 17

[30] Cenzic: *Application vulnerability trends report: 2014.* http://www.cenzic.com/, acesso em 22/08/2017. 17

[31] S. Esnaashari, I. Welch e P. Komisarczuk: *Determining home users' vulnerability to universal plug and play (upnp) attacks.* Em *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, páginas 725–729, March 2013. 18

[32] SQ Software: *Iot white paper.* http://sq-software.com/wp-content/uploads/2017/01/IoT-WhitePaper.pdf, acesso em 12/10/2017. 24

[33] X. Lin, P. Zavarsky, R. Ruhl e D. Lindskog: *Threat modeling for csrf attacks.* Em *2009 International Conference on Computational Science and Engineering*, volume 3, páginas 486–491, Aug 2009. 25

[34] Mrunali P. Pathak, Nida Kausar Khan e Tejashree C. Tantak: *Novel approach to detect and prevent web attacks.* 2016. 25

[35] E. Lawrence: *Ie8 security part vii: Clickjacking defenses - ieblog - msdn blogs.* https://blogs.msdn.microsoft.com/ie/2009/01/27/ie8-security-part-vii-clickjacking-defenses/, acesso em 05/03/2017. 25

[36] OWASP: *Enumerate infrastructure and application admin interfaces (otg-config-005).* https://www.owasp.org/index.php/Enumerate_Infrastructure_and_Application_Admin_Interfaces_(OTG-CONFIG-005), acesso em 12/10/2017. 25, 36

[37] Barry Peddycord III, Peng Ning e Sushil Jajodia: *On the accurate identification of network service dependencies in distributed systems.* 26

[38] OWASP: *Buffer overflows.* https://www.owasp.org/index.php/Buffer_Overflows, acesso em 12/10/2017. 27

[39] Daniel Miessler - IoT Attack Surface Mapping: *Iot attack surfaces - defcon-2015.* https://www.iotvillage.org/slides_DC23/IoT-Attack-Surfaces-DEFCON-2015-2.pdf, acesso em 12/10/2017. 29

[40] Dennis Turpitka: *Linux vs. windows device driver model: architecture, apis and build environment comparison.* `http://xmodulo.com/linux-vs-windows-device-driver-model.html`, acesso em 13/10/2017. 32

[41] T. Berners-Lee, R. Fielding e H. Frystyck: *Hypertext transfer protocol-http=1.0 group*, 1996. `https://tools.ietf.org/html/rfc1945`, acesso em 03/02/2017, RFC 1945, HTTP Working. 34

[42] Paul J, T. Berners-Lee, Jeffrey C. Mogul Lassy Masinter R. Fielding e James Gettys: *Hypertext transfer protocol–http/1.1*, 1999. `https://tools.ietf.org/html/rfc2616`, acesso em 03/02/2017, The Internet Society. 34

[43] S. Lawrence, R. Khare e: *Upgrading to tls within http/1.1. internet*, 2000. `https://tools.ietf.org/html/rfc2817`, acesso em 01/04/2017, RFC 2817. 35

[44] M. Zalewski: *Browser security handbook*, 2000. `http://code.google.com/p/browsersec/wiki/Part2#Arbitrary_page_mashups_(UI_redressing)`, acesso em 02/03/2017, RFC 2817. 35

[45] Lin-Shung Huang, Alexander Moshchuk, Helen J Wang Stuart Schecter e CollinJackson: *Clickjacking: Attacks and defenses.* Em *USENIX security symposium*, páginas 413–428, 2012. 35

[46] OWASP: *Clickjacking defense cheat sheet.* `https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet`, acesso em 05/03/2017. 35

[47] P. Bisht, P. Madhusudan e V. N. Venkatakrishnan.: *Candid: Dynamic candidate evaluations for automatic prevention of sql injection attacks.* ACM Transactions on Information and System Security (TISSEC), 13(2):14, 2010. 37

[48] Laurie Williams, Stephen Thomas e: *Using automated fix generation to secure sql statements.* Em *Proceedings of the Third International Workshop on Software Engineering for Secure Systems*, página 9. IEEE Computer Society, 2007. 37

[49] Jeff Forristal: *Nt web technology vulnerabilities.* Phrack Magazine, 8(54), dec 1998. `http://phrack.org/issues/54/8.html#article`, acesso em 05/03/2017. 37

[50] Imperva: *Hacker intelligence summary report – an anatomy of a sql injection attack.* `https://www.imperva.com/docs/HII_An_Anatomy_of_a_SQL_Injection_Attack_SQLi.pdf`, acesso em 06/04/2017. 37

[51] Zhendong Su, Gary Wassermann: *The essence of command injection attacks in web applications.* Em *ACM SIGPLAN Notices*, volume 41, páginas 372–382. ACM, 2006. 37

[52] William GJ Halfond, Alessandro Orso e Panagiotis Manolios: *Using positive tainting and syntax-aware evaluation to counter sql injection attacks.* Em *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, páginas 175–185. ACM, 2006. 38

[53] Ryan C Barnett, Jeremiah Grossman: *Battling Hackers and Protecting Users, Web Application Defender's Cookbook.* John Wiley & Sons, Inc., New York, NY, USA, 2013, ISBN 9781118362181. 40

[54] Jon Postel: *Rfc 793 - transmission control protocol.* 1981. 40, 41

[55] Jon Postel: *Rfc 792 - internet control message protocol.* 1981. 40

[56] Robert Braden: *Requirements for internet hosts - communication layers.* 1989. 40

[57] Fred Baker: *Requirements for ip version 4 routers.* 1995. 40

[58] João José Costa Gondim, Robson de Oliveira Albuquerque, Anderson Clayton Alves Nascimento Luis Javier García Villalba e Tai-Hoon Kim: *A methodological approach for assessing amplified reflection distributed denial of service on the internet of things.* Sensors, 16(11):1855, 2016. 42

[59] Deepak Kshirsagar, Sandeep Kumar: *Http flood attack detection using ontology.* Em *Proceedings of the International Conference on Advances in Information Communication Technology & Computing*, página 15. ACM, 2016. 42

[60] Michael Sutton, Adam Greene e Pedram Amini: *Fuzzing: brute force vulnerability discovery.* Pearson Education, 2007. 43

[61] Sean Turner, Tim Polk: *Prohibiting secure sockets layer (ssl) version 2.0.* 2011. 44

[62] Adam Langley, Alfredo Pironti, Richard Barnes e Martin Thomson: *Deprecating secure sockets layer version 3.0.* 2015. 44

[63] Hung-Min Sun, Her-Tyan Yeh: *Password-based authentication and key distribution protocols with perfect forward secrecy.* Journal of Computer and System Sciences, 72(6):1002 – 1011, 2006, ISSN 0022-0000. `http://www.sciencedirect.com/science/article/pii/S0022000006000481`. 45

[64] Emilia Käsper: *Fast elliptic curve cryptography in openssl.* Em *International Conference on Financial Cryptography and Data Security*, páginas 27–39. Springer, 2011. 45

[65] Ivan Ristić: *Ssl/tls deployment best practices.* 2012. 46

[66] Jane Radatz, Anne Geraci e Freny Katki: *Ieee standard glossary of software engineering terminology.* IEEE Std, 610121990(121990):3, 1990. 47

[67] Melanie Swan: *Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0.* Journal of Sensor and Actuator Networks, 1(3):217–253, 2012. 51

Apêndice A

Modelos de Redes de Computadores

Há dois modelos importantes que são utilizados na arquitetura de redes de computadores. O primeiro é o Modelo OSI, bastante utilizado para aplicações teóricas pois elucida de forma mais detalhada as funções de cada camada de internet. O segundo é o modelo TCP/IP, que é bastante difundido e aplicado nas redes mundo afora. É importante saber como eles funcionam

A.1 Modelo OSI

O modelo OSI é dividido em 7 camadas principais: Física, Enlace, Redes, Transporte, Sessão, Apresentação e Aplicação. Cada camada é responsável por um tipo de abstração diferente para torná-la uma camada com fuções bem definidas. Elas são feitas com base nos padrões internacionais para minimizar a comunicação para só quando for necessário. O número de camadas foi definido de um forma que seja grande o suficiente para poder realizar funções grandes, mas pequeno o suficiente para ser algo prático.

A.2 Funções das Camadas no Modelo OSI

A.2.1 Física

Transmitir *bits* num canal, fazendo uma boa distinção entre os estados 0 e 1 de forma e evitar que problemas mecânicos, elétricos e de *timing* interfiram na conexão.

A.2.2 Enlace

Envio de *frames* de dados recebidos da camada física para usuários de uma mesma rede, pode ser feito com ou sem controle de confirmações (ACKs) dependendo do serviço requisitado. Deve ser projetada de forma a controlar os envios para evitar problemas de

afogamento do destinatário (geralmente enviando informações sobre o *buffer* disponível). Em redes de *Broadcast* é responsável pelo controle de acessos (usando uma subcamada chamada *Medium Access*).

A.2.3 Redes

Envio de pacotes da fonte ao destinatário em redes diferentes utilizando tabelas de roteamento estáticas ou dinâmicas com as informações dos integrantes da rede e possíveis caminhos. Deve lidar com problemas de engarrafamento (gerado por tráfego alto de processos), endereçamento diferenciado de um mesmo *host* em subredes diferentes e com a diferença de protocolos utilizados entre duas redes.

A.2.4 Transporte

Faz o transporte de dados recebidos da Camada de Sessão, segmentando e enviando para a Camada de Redes. Já no destinatário faz-se o procedimento inverso juntando todos os segmentos recebidos da Camada de Redes. No momento da conexão, determina qual o tipo de serviço prestado (sem erro, *point-to-point*, etc) com base no tipo de dado a ser enviado. Também é responsável pelo controle de fluxo e decisão do modo de operação, podendo esse último ser orientado ou não à conexão.

A.2.5 Sessão

Realiza a comunicação de aplicações entre *hosts* diferentes, possibilitando uma identificação do ponto atual de conexão para evitar que um erro na conexão impe;a que os usuários retornem no mesmo lugar onde a comunicação parou.

A.2.6 Apresentação

Faz serviços de natureza sintática e semântica. Realizando a conversão de dados recebidos da Camada de Aplicação para um padrão que será utilizado na conexão, bem como outros serviços de compressão de arquivos ou de criptografia.

A.2.7 Aplicação

Responsável pelos protocolos comuns utilizados pelos usuários (HTTP, SMTP, BitTorrent, DNS, ICMP, etc), faz a ligação entre o usuário da aplicação. É a camada de interesse do estudo.

A.3 Modelo TCP/IP

O modelo TCP/IP foi criado para ser resistente a ataques e sobreviver em caso de perda de *hardware*, somando-se a isso uma arquitetura flexível o suficiente para operar com diferentes tipos de aplicações. Ele é dividido em 4 camadas principais: Enlace, *Internet*, Transporte e Aplicação. Contrastando com o OSI, no qual o modelo em si é a parte mais importante, a força do TCP/IP está em seus protocolos amplamente utilizados.

A.4 Funções das Camadas no Modelo TCP/IP

A.4.1 Enlace

No modelo TCP/IP esta não é propriamente uma camada e sim uma interface entre *links* de transmissões e hospedeiros. Ela é projetada de forma a permitir uma camada de *internet* não-orientada à conexão.

A.4.2 *Internet*

Ao contrário do que o nome sugere, essa camada não é a *internet* em si, mas uma camada presente na rede mundial de computadores. Ela é o centro das operações no modelo TCP/IP. É nela que estão definidos alguns dos principais protocolos como o IP e o ICMP, sendo assim responsável pelo endereçamento de pacotes entre diferentes redes.

A.4.3 Transporte

Possui a mesma função da camda de Transporte no modelo OSI, possibilitando conversas entre diferentes hosts. Nela estão dois protocolos ponta-a-ponta que definem a forma como a comunicação será estabelecida. O primeiro protocolo, chamado de TCP, é orientado à conexão e permite o envio de informações de uma ponta à outra sem erros. Ele é utilizado amplamente na *internet* para serviços que exigem confirmação de envio e algum tipo de integridade no conteúdo. O segundo protocolo é chamado de UDP e é um protocolo não-orientado à conexão. Ele subtitui o TCP em aplicações que não exigem controle de fluxo ou garantia de entrega. É utilizado em situações em que é necessária uma conexão mais rápida. Os dois protocolos são igualmente importantes nessa camada.

A.4.4 Aplicação

Possui todos os protocolos de alto nível, como o ICMP, FTP, SMTP, entre outros. Eles têm funções muito importantes na comunicação entre a aplicação e a rede. O modelo

utilizado neste estudo será o TCP/IP com foco na camada de aplicação, por isso ela será mais detalhada ao longo do artigo.