Universidade Estadual de Londrina

GABRIEL ESTEVES MESSAS

ESTUDO SOBRE A SEGURANÇA DE DISPOSITIVOS DOMÉSTICOS CONECTADOS À INTERNET DAS COISAS

LONDRINA - PR

2022

GABRIEL ESTEVES MESSAS

ESTUDO SOBRE A SEGURANÇA DE DISPOSITIVOS DOMÉSTICOS CONECTADOS À INTERNET DAS COISAS

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Departamento de Computação da Universidade Estadual de Londrina, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Bruno Bogaz Zarpelão

LONDRINA - PR

2022

GABRIEL ESTEVES MESSAS

ESTUDO SOBRE A SEGURANÇA DE DISPOSITIVOS DOMÉSTICOS CONECTADOS À INTERNET DAS COISAS

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Departamento de Computação da Universidade Estadual de Londrina, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Prof. Dr. Bruno Bogaz Zarpelão
Universidade Estadual de Londrina

Prof. Dr. Guilherme Pina Cardim
Universidade Estadual de Londrina

Rildo Antônio de Souza
Universidade Estadual de Londrina

Londrina-PR, 3 de junho de 2022.

Este trabalho é dedicado à cura – ainda por vir – da minha ansiedade, depressão, dor de cabeça excruciante, distúrbios do sistema digestório e de todas as outras condições incapacitantes que me cercam há anos.

“Uma paixão forte por qualquer objeto assegurará o sucesso, porque o desejo pelo objetivo mostrará os meios.”

(William Hazlitt)

MESSAS, G. E. **Estudo sobre a segurança de dispositivos domésticos conectados à Internet das Coisas**. 66p. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina-PR, 2022.

RESUMO

Com a crescente globalização e a popularização da Internet, o acesso a dispositivos com tal capacidade se tornou mais fácil. Estes, cada vez mais conectados, agora marcam presença no ambiente doméstico, impulsionados pelo conceito da Internet das Coisas (*Internet of Things - IoT*). Tal proximidade ao usuário final torna relevantes discussões sobre a confiabilidade, em termos de segurança e privacidade, dos equipamentos em questão. Este trabalho, portanto, apresenta uma análise de determinados aparelhos deste tipo – um receptor de canais via satélite e uma câmera sem fio – no que tange à segurança, à proteção do usuário, de sua privacidade e de seus dados. Deste modo, o estudo de possíveis vulnerabilidades desses equipamentos expõe características perigosas, que culminam em falhas de segurança comprometedoras e vazamento de informações. Neste cenário, também se estabelece um protocolo de testagem a ser aplicado aos dispositivos, a fim de popularizar aferições de qualidade nesta categoria, o que contribui em ampla escala aos grupos envolvidos: indústria e usuários.

Palavras-chave: Internet das coisas. Teste de invasão. *Hacking* ético. Segurança da informação. Vulnerabilidade.

MESSAS, G. E. **Study on the security of domestic devices connected to the Internet of Things**. 66p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina-PR, 2021.

ABSTRACT

With the growing globalization and popularization of the Internet, the access to devices with this capability has become easier. Increasingly more connected, these devices are now added to the domestic environment, propelled by the concept of the Internet of Things (IoT). Such proximity to the end user brings to the light discussions on the trustworthiness of the equipment in question. This proposal, therefore, aims to thoroughly analyze selected gadgets of this kind – a satellite TV receiver and a wireless camera – in terms of security, user protection, and data privacy. In this sense, the study of possible vulnerabilities of these devices exposes dangerous characteristics, which culminate in compromising security failures and information leakage. In this scenario, it is also established a testing protocol to be applied to the devices, with the aim of popularizing quality measurements in this category, which can contribute widely to all involved groups: industry and users.

Keywords: Internet of things. Penetration testing. Ethical hacking. Information security. Vulnerability.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IoT	*Internet of Things*
IKS	*Internet Key Sharing*
SKS	*Satellite Key Sharing*
CS	*Card Sharing*
IPTV	*Internet Protocol Television*
TTL	*Time to Live*
RTSP	*Real Time Streaming Protocol*

SUMÁRIO

1 INTRODUÇÃO

O início do século XXI marcou uma sólida aceleração no desenvolvimento de tecnologias eletrônicas, especialmente na área da computação. Esta, agora, inclui dispositivos cada vez mais poderosos e inteligentes, os quais, aliados à crescente abrangência e evolução da Internet, contribuem para tornar a vida humana progressivamente mais fácil [1].

Tais fatos contribuíram fortemente para a democratização do acesso a aparelhos eletrônicos, de maneira que, atualmente, estes se fazem presentes até mesmo no ambiente doméstico [2], impulsionados pelo conceito da Internet das Coisas. Esta proximidade ao usuário final certamente acende um alerta no que tange à real confiabilidade dos equipamentos em questão, haja vista que possuem conexão direta à rede local e, consequentemente, acesso às informações intercambiadas [3].

Neste cenário, mesmo que as suspeitas sobre más intenções por parte do próprio dispositivo sejam descartadas, há, ainda assim, a possibilidade de que uma vulnerabilidade seja explorada por um terceiro envolvido. Logo, rotular um dispositivo como seguro é, ao mesmo tempo, garantir que seu código-fonte não é malicioso por si só e que este não é frágil e/ou abre brechas a ataques externos.

A boa notícia é que há maneiras de colocar tais máquinas à prova a fim de garantir os pontos mencionados [4]. Aquelas se resumem, em sua maioria, a um processo que tem como cerne um objetivo contraintuitivo a priori: invadir o próprio equipamento testado. Desta forma, o resultado se dá pelo sucesso - ou não - do propósito estabelecido, revelando um dispositivo inseguro ou confiável, respectivamente.

Em vista do exposto, o presente trabalho tem como intuito primário testar aparelhos selecionados, tais como receptores de IPTV, TV *Boxes* e câmeras Wi-Fi - tomados como exemplo -, realizando rotinas de invasão através da rede [5]. Os resultados, por suas vezes, mostrarão o nível de confiabilidade dos dispositivos, que servirão como representantes da classe de produto à qual estão integrados. Por fim, almeja-se estabelecer um protocolo de verificação que possa ser adotado para avaliação de segurança em dispositivos *IoT*.

O restante deste trabalho está organizado da seguinte maneira: o capítulo 2 descreve conceitos fundamentais para a realização do estudo, tais como, *Internet of Things*, segurança, vulnerabilidades e testes de invasão. O capítulo 3 explica a metodologia e o modo de operação utilizados nos testes. Por sua vez, o capítulo 4 apresenta os dispositivos analisados e descreve os testes de invasão realizados nesses, respectivamente. Já o capítulo 5

contém o protocolo desenvolvido para verificação de segurança em dispositivos *IoT*. Por fim, o capítulo 6 apresenta as conclusões obtidas durante os processos.

2 FUNDAMENTAÇÃO TEÓRICO-METODOLÓGICA

2.1 Internet das Coisas

2.1.1 Definição

Internet das Coisas, mais conhecida por seu termo em inglês: *Internet of Things* (*IoT*), é um conceito que se refere à interconexão digital de objetos cotidianos com a Internet. É uma extensão da Internet atual que possibilita que objetos do dia-a-dia, contanto que tenham capacidade computacional e de comunicação, se conectem à Internet [6].

Neste cenário, a conexão com a rede mundial de computadores possibilita, primeiramente, interagir remotamente com os objetos e, em segundo lugar, que esses próprios equipamentos sejam usados como provedores de serviços. Essas novas capacidades de aparelhos comuns abrem caminho a inúmeras possibilidades, principalmente no âmbito doméstico.

2.1.2 Tendência

Segundo a empresa de consultoria Gartner, em 2020, deveria haver, no mundo, aproximadamente 26 bilhões de dispositivos com um sistema de conexão à Internet das Coisas. Já a consultoria Abi Research previu que, no mesmo ano, existiriam 30 bilhões de dispositivos sem fio conectados a este segmento da Internet [7]. Estimava-se que cada ser humano esteja rodeado por 1 000 a 5 000 desses aparelhos, em média.

Atualmente, a IDC (*International Data Corporation*) prevê um aumento neste número para 41,6 bilhões de dispositivos *IoT* conectados até o ano de 2025, enquanto a Martech Advisor estima que tal valor possa atingir exorbitantes 125 bilhões em 2030. Até então, cada lar possuirá, estimadamente, 15 aparelhos conectados. Em 2020, para comparação, a média prevista foi de 10 aparelhos por utilizador [6].

O conceito de *smart home* (ou lar inteligente, em português), por exemplo, descreve um lugar equipado com aparelhos eletrônicos ligados a uma rede, Wi-Fi ou Bluetooth, constituído por um sistema integrado que permite controlar múltiplos dispositivos, como sistemas de iluminação e de temperatura, eletrodomésticos ou estações de

entretenimento, como receptores de canais via Internet ou gerenciadores de acesso a plataformas de streaming.

2.1.3 Arquiteturas e riscos de segurança

Os aparelhos desse tipo, geralmente, consistem em pequenos dispositivos de hardware, com recursos restritos e poder de processamento não muito surpreendente – bem longe de um *smartphone* atual, por exemplo. São bastante diversos, de acordo com a função a que servem: tamanhos, formas e modos de funcionamento variam largamente.

Em razão disso, seus sistemas operacionais – geralmente, "embarcados" – também seguem o padrão e são encontrados de maneiras bastante variadas, muitas vezes, desenvolvidos unicamente para o dispositivo em questão. As formas de comunicação destes, por suas vezes, também acompanham esta tendência e se apresentam em diversos tipos, tais como Wi-Fi, Bluetooth, ZigBee, redes celulares, NFC, RFID, etc.

Neste cenário, para tentar fugir das limitações físicas – baixo poder computacional, principalmente –, tais dispositivos, muitas vezes, combinam suas funções a sistemas hospedados em ambientes de Computação em Nuvem, haja vista que essa possui maior capacidade, ao menos, quando comparada ao exemplar *IoT*.

Todavia, essas características acarretam riscos e implicam grandes desafios técnicos. Tais dispositivos *IoT*, por serem inteligentes e possuírem conexão à rede, possuem um endereço IP, porta de entrada que pode permitir a um agente mal-intencionado se conectar e realizar ataques à segurança e à privacidade dos usuários do equipamento.

Adicionalmente, tendo em mente todas as instâncias do processo de funcionamento deste tipo de equipamento – dispositivo, comunicação, nuvem –, é possível entender a pluralidade de pontos onde uma falha pode acontecer.

Para servir como base, segundo a empresa de segurança virtual Kaspersky Lab, existem pelo menos 7 mil amostras de malwares em dispositivos *IoT*. Além disso, em outubro de 2016, os veículos de informação noticiaram o maior ataque de negação de serviço já registrado utilizando dispositivos da Internet das Coisas, tais como roteadores domésticos, câmeras de segurança e até babás-eletrônicas o que denota a necessidade de tratar o segmento com atenção.

Tal procedimento foi todo baseado em dispositivos *IoT*, os quais, infectados por um *malware*, foram "sequestrados" à distância e passaram a trabalhar de acordo com a vontade do agente malicioso, tal como detalha Greenstein [8].

Por serem máquinas pequenas, numerosas e diversas, desenvolver sistemas de defesa torna-se uma tarefa complexa. O fato de possuírem sistemas operacionais super leves e com arquitetura reduzida também não colabora, exatamente como reforçado por Rafferty et al [9]. Logo, garantir que a implementação do dispositivo em si não abre brecha a falhas é crucial.

2.2 Vulnerabilidades e Ameaças Virtuais

2.2.1 Definição

Uma vulnerabilidade é definida como uma condição que, quando explorada por um atacante, pode resultar em uma violação de segurança. Exemplos de vulnerabilidades são falhas no projeto, na implementação ou na configuração de programas, serviços ou equipamentos de rede [10].

De acordo com a ISO 27000 [11], conjunto de normas para Sistemas de Gestão de Segurança da Informação elaborada pela ISO (*International Organization for Standardization*), um ataque de exploração de vulnerabilidades ocorre quando um atacante, aproveitando-se de uma fraqueza do sistema, tenta executar ações maliciosas, como invadi-lo, acessar informações confidenciais ou tornar um serviço inacessível.

2.2.2 Tipos

As vulnerabilidades de segurança podem ser divididas em vários tipos com base em critérios diferentes – como onde a vulnerabilidade existe, o que a causou ou como ela pode ser usada. Algumas categorias amplas desses tipos de vulnerabilidade incluem [10]:

- Falhas humanas: o elo mais fraco em muitas arquiteturas de segurança cibernética é o elemento humano. Os erros do usuário podem facilmente expor dados confidenciais, criar pontos de acesso exploráveis para invasores ou interromper sistemas. Os próprios usuários internos às vezes executam arquivos maliciosos que facilitam a intrusão no sistema e a perda de dados e informações,

seja por falta de conhecimento, desatenção ou atividades maliciosas de colaboradores.

- Vulnerabilidades de rede: problemas com o *hardware* ou o *software* de uma rede que a expõem a uma possível invasão de terceiros. Os exemplos incluem pontos de acesso Wi-Fi inseguros, *firewalls* mal configurados e falhas de arquitetura.

- Vulnerabilidades de aplicações: fraquezas de softwares ou sistema operacional que expõem o dispositivo a riscos de segurança que possam ser utilizadas para ataques, como roubo de dados e sequestro de informações. Um exemplo comum são *softwares* desatualizados ou mal estruturados.

2.2.3 Vulnerabilidades e *IoT*

Uma preocupação atual, por exemplo, é o aumento no número de ataques a dispositivos englobados pela Internet das Coisas. Essa tecnologia é uma das bases da transformação digital e da própria Indústria 4.0, mas traz consigo um risco adicional para a infraestrutura digital dos lares e das empresas — algo com o qual os peritos em segurança devem saber lidar.

A simplicidade de tais aparelhos, à primeira vista, pode até afastá-los de quaisquer suspeitas, no que tange a uma possível falha de segurança e/ou comprometimento de privacidade. No entanto, é esta exata característica que torna possível que uma brecha seja explorada maliciosamente sem causar alarde e despertar a atenção do usuário diretamente relacionado ao produto. Eis a necessidade de tratá-los de maneira especial.

2.2.4 Ataques a Dispositivos *IoT*

A configuração estrutural dos aparelhos *IoT* – explicada na seção 2.1.3 – habilita diferentes formas de ataques. Para detalhá-las, a fundação OWASP (*Open Web Application Security Project*), entidade importante na área de segurança da Internet, analisou, em um estudo [12], dados de diversos dispositivos desse tipo e gerou uma lista com as dez causas de ameaça mais populares, apelidada de "*top* 10", a seguir:

- Senhas fracas, adivinháveis ou fixas no código: uso de senhas passíveis de serem descobertas facilmente, universalmente conhecidas ou gravadas no sistema do aparelho.

- Serviços de rede inseguros: serviços desnecessários ou com baixa segurança expostos por meio de portas de rede, principalmente quando abertas à Internet.

- Interfaces inseguras no ecossistema: interfaces *Web*, API, *cloud* ou *mobile* inseguras no ecossistema fora do dispositivo IoT em si, que permitem o comprometimento deste ou de seus componentes relacionados.

- Falta de mecanismo seguro de atualização: validação de firmware deficitária no dispositivo, falta de segurança no tráfego dos dados (não encriptados), falta de travas *anti-rollback*, falta de notificações de mudanças de segurança.

- Uso de componentes inseguros ou desatualizados: uso de componentes de software ou bibliotecas descontinuadas ou inseguras, que pode incluir customização insegura do sistema operacional e o uso de programas ou componentes de hardware de terceiros que possam ser comprometidos.

- Proteção à privacidade insuficiente: informações pessoais do usuário armazenadas no dispositivo ou no ecossistema utilizadas de maneira insegura, imprópria ou sem permissão.

- Armazenamento e transmissão de dados insegura: falta de encriptação ou controle de acesso a dados sensíveis em qualquer instância do ecossistema.

- Falta de gerenciamento do dispositivo: ausência de suporte de segurança nos aparelhos, como gerência de atualizações e plataforma de monitoramento do sistema.

- Configurações padrão inseguras: ajustes de fábrica padronizados sem segurança e/ou impossibilidade de alterar os parâmetros, tais como senhas de acesso.

- Falta de proteção física: ausência de segurança no acesso físico ao aparelho, o que possibilita atacantes em potencial a obter informação privilegiada para um futuro ataque.

2.3 Ethical Hacking

2.3.1 Terminologia

Em âmbito popular, o termo *hacker* é associado ao indivíduo que se dedica a burlar os limites de segurança de dispositivos, sistemas e redes de computadores, cuja intenção é sempre maliciosa [13]. No entanto, esta associação é feita de maneira errônea, uma vez que tal predicado pertence ao termo *cracker* – o verdadeiro vilão mal-intencionado.

Ethical Hacking (do inglês, *hacking* ético), por sua vez, é justamente a alcunha – desta vez, correta – utilizada para se referir ao trabalho desempenhado pelo *hacker* – agente sem intenções maliciosas. É o conceito que representa, de forma legítima, as atividades desempenhadas por aquele que trabalha nesta área, também chamado de *ethical hacker* (*hacker* ético).

2.3.2 Definição

Logo, o *hacker* ético é um profissional de tecnologia da informação especializado na área de segurança, cuja função é encontrar vulnerabilidades que um *hacker* malicioso poderia explorar. Para tal, este utiliza, dentre outros métodos, de uma rotina de testes que acabou conhecida como *pentest* (contração oriunda da expressão inglesa

penetration testing; em português, teste de invasão*).* Além disso, o objetivo primário daquele é gerar um relatório do sistema sob análise, a fim de resolver problemas e implementar melhorias.

Para realizar este serviço, no entanto, o *hacker* deve necessariamente ter permissão para investigar. Caso receba sinal positivo, ele tem ainda outra responsabilidade: respeitar a privacidade e acesso aos arquivos, já que uma série de informações sensíveis podem estar expostas [14]. Portanto, conclui-se que o profissional desta área deve ter conhecimentos iguais ou superiores a um *cracker*, e código de ética invejável.

2.4 Testes de Invasão

2.4.1 Definição

Penetration Testing (ou *pentest*, do inglês, Teste de Invasão) é o procedimento que engloba as atividades que buscam submeter redes, sistemas ou programas a uma série de testes de invasão [5]. Estas são realizadas através de técnicas e ferramentas variadas, testando diversos cenários que possibilitem a identificação de vulnerabilidades de segurança. Em suma, o teste de invasão simula um ataque malicioso, justamente para que sejam analisadas formas de evitá-lo.

Falhas desconhecidas em *hardwares* ou em *softwares*, junto a deficiências no sistema operacional, são em geral as categorias mais comuns de vulnerabilidades encontradas por testes desta natureza.

2.4.2 Classificação

O *pentest* – como é popularmente chamado – é comumente contratado por organizações para análise de segurança e pode ser subdividido em diversas categorias, sendo a mais relevante a que o classifica de acordo com o nível de conhecimento do perito sobre o sistema alvo:

- Análise *black-box*: é o tipo de teste em que o executor (*pentester*) não possui informação técnica alguma sobre a infraestrutura de rede,

especificações dos sistemas ou modo de funcionamento dos dispositivos sobre os quais o teste de invasão está sendo realizado.

- Análise *white-box*: nesta modalidade, o *pentester* conhece previamente toda a infraestrutura que será analisada, incluindo, dentre outros, o mapeamento de rede, a amplitude dos endereços IP, os firewalls e equipamentos de roteamento local existentes, além de informações específicas sobre as máquinas e softwares nelas em execução.

Testes do tipo *black-box* simulam um ataque de alguém que não esteja familiarizado com o sistema, enquanto um teste *white-box* simula o que pode acontecer após um vazamento de informações, em que o invasor tenha acesso ao código fonte, esquemas de rede e, possivelmente, até mesmo a algumas senhas. Há ainda o *grey-box*, onde o executor possui conhecimento parcial sobre o alvo.

2.4.3 Fases

O processo completo deve possuir metodologia bem definida e o protocolo, geralmente, se divide em:

- Fase de planejamento: momento anterior ao início do teste em si, onde se define o modelo da análise (interna ou externa, direitos e privilégios habilitados), as metas, os dados de origem e o escopo de trabalho; se desenvolve e adapta a metodologia a ser usada.

- Fase de teste: período de execução dos procedimentos técnicos de fato (testes de invasão), que engloba, geralmente, em sequência: a identificação da rede e dos dispositivos alvos; enumeração das ferramentas de intrusão; detecção e exploração de vulnerabilidades nas máquinas, utilização de sistemas comprometidos como um trampolim para novas intrusões; eliminação de falsos positivos.

- Fase de consolidação: onde se analisa os resultados obtidos durante as fases anteriores e, principalmente, constrói-se relatórios com os dados coletados, a fim de servirem como recomendações aos responsáveis pelos equipamentos testados para redução de riscos.

Logo, um teste de invasão é melhor aproveitado quando estruturado com base neste modelo de segmentação por etapas. Em suma, o método e a ordem apropriados favorecem com que a pessoa que realiza os processos – em uma dada fase – disponha das informações necessárias para atingir os respectivos objetivos.

3 METODOLOGIA

3.1 Objetivos

O presente trabalho tem como objetivo estudar o modo de funcionamento de testes de invasão e aplicá-los a dispositivos eletrônicos variados – essencialmente conectados à Internet, primariamente utilizados no ambiente doméstico (logo, associados à Internet das Coisas) – a fim de avaliar seus níveis de segurança.

Como resultado de tal, serão redigidos relatórios individuais de cada teste, a servirem como registro da qualidade e proteção dos sistemas sob prova. Aqueles apresentarão detalhes técnicos dos procedimentos realizados, bem como a descrição das respectivas falhas exploradas – quando estas existirem.

Para cada dispositivo analisado, o documento a ser gerado almeja atrair a atenção e incitar os responsáveis a corrigir as brechas de confiabilidade do equipamento. Mais do que isso, o relato em questão visa levantar questionamentos sobre a situação de aparelhos semelhantes e estimular a checagem destes mais eficazmente a fim de evitar defeitos.

Por fim, após todas as etapas e informações geradas, será certamente possível, como objetivo principal, estabelecer um protocolo de testagem genérico o suficiente passível de ser aplicado a equipamentos similares aos em evidência. Este tornará os procedimentos mais simples e poderá ser adotado por outros profissionais e pesquisadores para verificação de vulnerabilidades em ambientes conectados.

3.2 Modo de Operação

O processo se inicia com a determinação e escolha do dispositivo a ser testado. Este, por sua vez, deve atender a alguns requisitos para ser considerado candidato a alvo da análise, dentre os quais, obrigatoriamente, se incluem:

- Possuir endereço MAC e capacidade de receber um endereço IP;
- Capacidade de comunicação via protocolos de rede;
- Capacidade de transmissão de dados (via cabo de rede ou via Wi-Fi);
- Tendência a ser classificado como dispositivo *IoT*, dado o ambiente de utilização.

A seguir, com o dispositivo definido, deve-se garantir que este está instalado, configurado e é acessível através da rede. À medida que este se encontra em estado de funcionamento normal – tal como é utilizado para suas funções originais –, é possível partir à fase de execução propriamente dita, onde se iniciarão os testes.

Esta, por sua vez, realizada principalmente em um computador executando o sistema operacional Kali (Linux) na mesma rede local do aparelho, consiste, resumidamente, em fases que têm como cerne os seguintes objetivos, em ordem cronológica:

- Planejamento – Descrição do aparelho e considerações iniciais;
- *Fingerprinting* – Esta etapa tem como foco obter o máximo de informações possível sobre o ambiente em execução no aparelho, principalmente dados sobre o sistema operacional. Para tanto, existem variadas aplicações especificamente voltadas a essa função, cada qual com um mecanismo de ação diferente, mais adequado a determinada situação;
- Enumeração – Fase que busca elencar as possíveis formas de comunicação com o aparelho (portas e serviços). De maneira análoga à etapa anterior, também conta com auxílio de softwares descritores pré-existentes. Com os dados obtidos até este momento, é também possível realizar verificações manuais de tráfego de rede;
- Exploração – Consiste em estudar possíveis vulnerabilidades sobre os serviços descobertos até a fase em questão e explorá-las na prática;
- Documentação – Documentar o progresso parcial e os resultados.

4 EXECUÇÃO DOS TESTES DE INVASÃO

Neste capítulo, todas as informações dos alvos e as rotinas do processo de teste são documentadas, separadas por dispositivo. A subetapa de nome *descrição* engloba o estudo das características e propósitos do aparelho, uma fase crucial, que pode ser considerada o início da aquisição de conhecimento do teste de invasão (análoga ao planejamento).

4.1 DuoSat Troy HD Generation

4.1.1 Descrição

O primeiro dispositivo selecionado se encaixa, primariamente, na categoria de decodificador de canais via satélite, com algumas funções adicionais. Este fora escolhido pois figura em um segmento de crescente popularidade, cujos produtos ganharam espaço no mercado devido à boa relação custo-benefício. A Figura 1 apresenta uma foto do dispositivo, enquanto a Tabela 1 apresenta as suas especificações técnicas.

Figura 1 – DuoSat Troy HD Generation.
Fonte: <https://www.lojapremio.com.br/troy-generation>.

Tabela 1 - Especificações técnicas do DuoSat Troy HD Generation

Chipset	**Memória Flash**	**Memória RAM**
600MHz - 1755DMIPS	16MB	256MB DDR3

Este aparelho pode funcionar de diversos modos, no que tange à obtenção dos canais:

- *IKS* (*Internet Key Sharing*): uma antena para recepção do satélite de canais e conexão à Internet para obtenção das chaves de decodificação dos canais a partir do servidor oficial da marca;
- *SKS* (*Satellite Key Sharing*): uma antena para recepção do satélite de canais e outra para obtenção das chaves de decodificação dos canais;
- *CS* (*Card Sharing*): uma antena para recepção do satélite de canais e conexão à Internet para obtenção das chaves de decodificação dos canais advindas de um servidor privado contratado;
- *IPTV* (*Internet Protocol Television*): conexão à Internet para obtenção dos canais já em formato multimídia, tal como serviços de streaming, advindos de um servidor privado contratado.

Além dos canais, o dispositivo também oferece acesso a um repositório de filmes e séries – em um servidor proprietário da marca –, também acessível através da Internet. Adicionalmente, é possível controlar as funções do aparelho através de um aplicativo no celular, contanto que ambos estejam conectados à mesma rede.

A combinação dessas várias funções, que revelam o uso de diversos protocolos, em um dispositivo relativamente simples, certamente torna provável a existência de brechas de segurança, a serem verificadas na sequência.

4.1.2 Teste de Invasão

Para os testes, assume-se que o aparelho, a ser chamado por Troy ao longo do processo, está devidamente funcional e acessível na rede local. Este, conectado via Wi-Fi ao roteador central, recebeu o endereço IP fixo 192.168.1.100, para facilitar a identificação.

4.1.2.1 Fingerprinting

Como não é possível encontrar nenhuma informação relacionada ao sistema operacional embarcado no dispositivo por meio de suas especificações ou pesquisas, o objetivo principal da primeira fase se torna extrair essas informações. Para tanto, inicia-se o arsenal de recursos com o programa de nome ***p0f***. Este possui caráter passivo e consegue revelar informações de dispositivos que realizam conexão com a máquina na qual ele é executado.

Ao iniciá-lo e deixá-lo em execução por diversos minutos, não se obteve nenhuma tentativa de conexão por parte do Troy, o que já era, de certa forma, esperado, afinal, o aparelho não possui motivos claros para se relacionar com um outro dispositivo aleatório na rede, especialmente se não comandado ativamente pelo usuário. É forçada, então, uma conexão via comando ***telnet*** ao Troy, o que, imediatamente, produz resultados, conforme a Figura 2.

```
.-[ 192.168.1.110/40120 -> 192.168.1.100/80 (syn) ]-
|
| client   = 192.168.1.110/40120
| os       = Linux 2.2.x-3.x
| dist     = 0
| params   = generic tos:0x04
| raw_sig  = 4:64+0:0:1460:mss*44,7:mss,sok,ts,nop,ws:df,id+:0
|
`----
```

Figura 2 – Retorno do comando ***p0f*** com o Troy

É possível observar, portanto, que o item “os” informado pelo programa se refere à versão do *kernel* em execução no equipamento, neste caso, um *kernel* Linux, em versão entre 2.2.x e 3.x. Considerando-se as datas de lançamento de cada versão, o sistema do Troy pode executar uma versão criada tanto em 1999 quanto em 2011, respectivamente. No pior dos casos, é um sistema antigo.

A próxima ferramenta capaz de revelar informações é o comando ***ping***. Usa-se de sua habilidade de mostrar o *TTL* (*Time to Live*) do pacote enviado a partir do dispositivo para pré-determinar seu sistema operacional. Considerando uma conexão sem saltos, os seguintes valores de *TTL* indicam, resumidamente [15]:

- 64 – Unix/Linux (principais distribuições mais robustas)
- 128 – Windows
- 255 – Unix (outros sistemas mais simplificados)

O uso do comando com alvo no Troy revela um *TTL* de 255 – mostrado na Figura 3 –, o que indica o uso de um sistema baseado em Unix – certamente, com funcionalidades reduzidas.

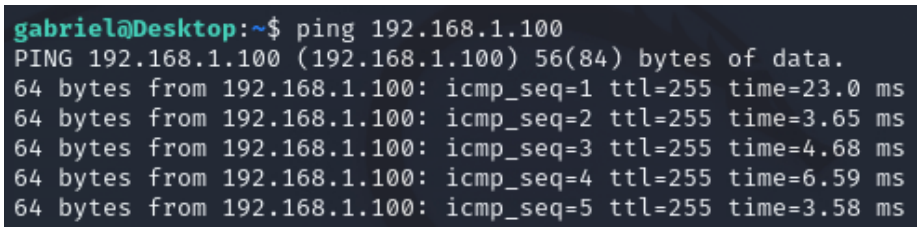

Figura 3 – Comando ***ping*** no Troy

No entanto, é possível – e recomendado – refinar ainda mais a especificação do sistema. Parte-se, então, para o também bastante conhecido comando ***nmap***. Este possui diversas funções – a serem usadas em próximos passos –, inclusive a detecção do sistema operacional do dispositivo alvo, que será usada agora.

O uso do comando (com argumento *-O*, nível normal) tendo o Troy como alvo não retorna nenhuma correspondência; com argumento *–osscan-guess*, nível agressivo, obtém-se uma lista de plausíveis sistemas, vide Figura 4. Todos os nomes listados se referem a sistemas embarcados em dispositivos *IoT*, o que confirma o caráter do Troy já debatido anteriormente.

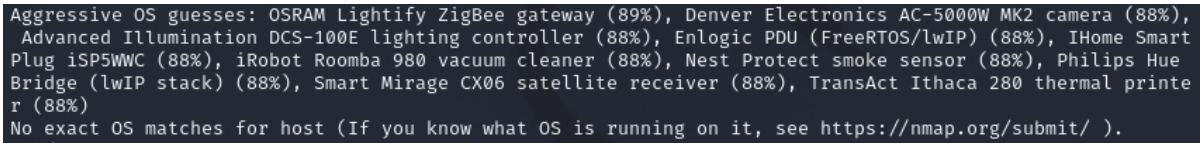

Figura 4 – Comando ***nmap*** para detecção de sistemas no Troy

Este resultado, entretanto, ainda é vago e não acrescenta muito ao dossiê de informações do Troy. Como um último recurso, pode-se usar o programa de nome ***xprobe2***, que analisa os pacotes trocados pelo dispositivo para determinar características de seu sistema. O uso do comando no Troy retorna resultados promissores, com a definição de um sistema operacional compatível: o *NetBSD*, em versões antigas (possivelmente vulneráveis), de acordo com a Figura 5.

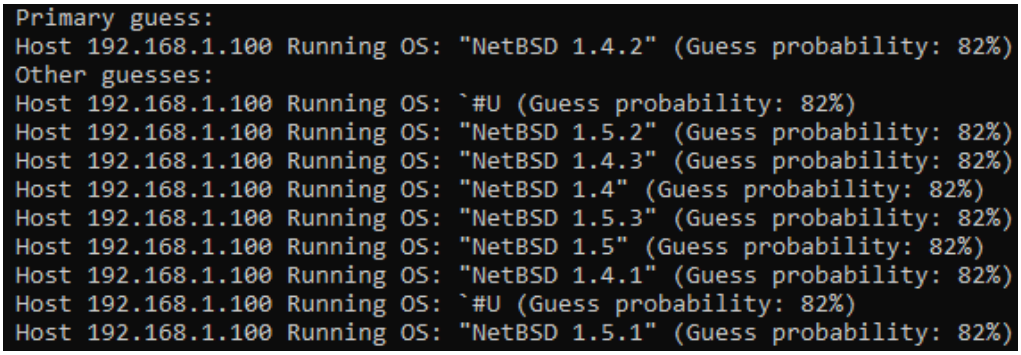

Figura 5 – Comando ***xprobe2*** no Troy

Com o sistema filtrado de maneira mais satisfatória, é possível prosseguir à fase seguinte. Esta informação obtida será crucial para definir os métodos utilizados no decorrer do processo.

4.1.2.2 Enumeração

Esta fase tem como objetivo descobrir e listar as formas de conexão possíveis com o aparelho testado: tal como o nome diz, enumerar – neste caso, portas e seus respectivos serviços associados. Com a especificação dos serviços disponíveis, o processo de pesquisa de vulnerabilidades pode ser refinado e filtrado, otimizando as operações do teste.

Para tal, a primeira subetapa conta, quase que exclusivamente, com o comando ***nmap*** – já citado e utilizado em etapas anteriores. Este agora volta e será utilizado novamente, desta vez em sua função principal: o mapeamento de portas.

O uso do comando (com argumento -sV, para detecção de serviços TCP) em toda a extensão de portas (65535 valores) retorna, após 13 horas de execução, todas as portas – e respectivos serviços, quando disponível – nas quais pode ser possível se conectar ao Troy, conforme a Figura 6.

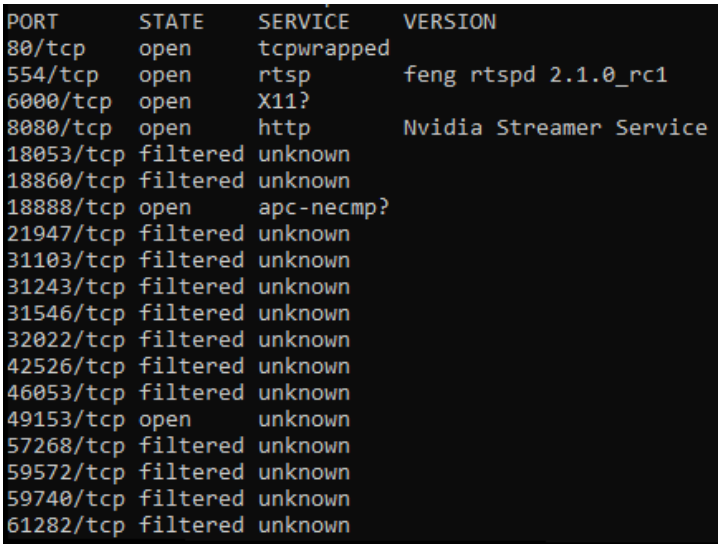

Figura 6 – Comando ***nmap*** para detecção de portas no Troy

As portas em estado *open* aceitam conexões impreterivelmente, já as em estado *filtered* são possivelmente bloqueadas por um firewall, filtro ou regra do próprio aparelho – mas, ainda assim, é possível tentar explorá-las.

Um detalhe importante notado durante a execução do comando é que os limites de *timeout* e quantidade de tentativas de conexão tiveram de ser frequentemente aumentados pelo ***nmap***, pois o Troy passava a demorar a responder tanto mais quanto mais requisições eram feitas. Isso, consequentemente, causava a interrupção dos serviços no aparelho, que demoravam a voltar após o encerramento dos testes.

Ou seja, descobre-se a primeira falha do Troy: susceptibilidade a ataques do tipo *flooding* (“enchente” de requisições), mais conhecidos como ataques de negação de serviço (do inglês, *Denial of Service* - *DoS*).

O mesmo comando – desta vez, com argumento -sU ao invés de -sV, para foco em portas UDP – não retornou resultados, ou seja, não há serviços para este protocolo em escuta no Troy.

Com a enumeração finalizada, é possível prosseguir à próxima fase, onde as informações descobertas serão, de fato, exploradas.

4.1.2.3 Exploração

Neste cenário, é sensato seguir a lista porta a porta, individualmente, investigando seu funcionamento e explorando suas possíveis vulnerabilidades.

A começar pela porta 80, o serviço indicado é o "*tcpwrapped*", o que, na verdade, se refere ao TCP *Wrapper*, um gerenciador de acesso e conexões presente em sistemas baseados no Unix (caso do Troy). Este sistema não é, de fato, o serviço em execução na porta, mas sim apenas um interceptador da conexão realizada.

Esta porta é característica de servidores Web, por convenção, logo, é justo simplesmente tentar acessar o IP do Troy por um navegador de Internet (requisição HTTP GET) e analisar o resultado. Por sua vez, o retorno desta requisição é uma página Web simples com um campo de *input* de senha para login, tal como mostra a Figura 7.

Figura 7 – Página Web inicial na porta 80 do Troy

Para surpresa, simplesmente avançar – com o campo vazio – efetua a autenticação e uma nova página é mostrada, conforme a Figura 8. Ou seja, o usuário do Troy certamente não faz ideia da existência desta página de configuração – e sua função – e, muito menos, da necessidade de cadastrar uma senha de acesso para ela – ação que, se existe, não é de fácil acesso.

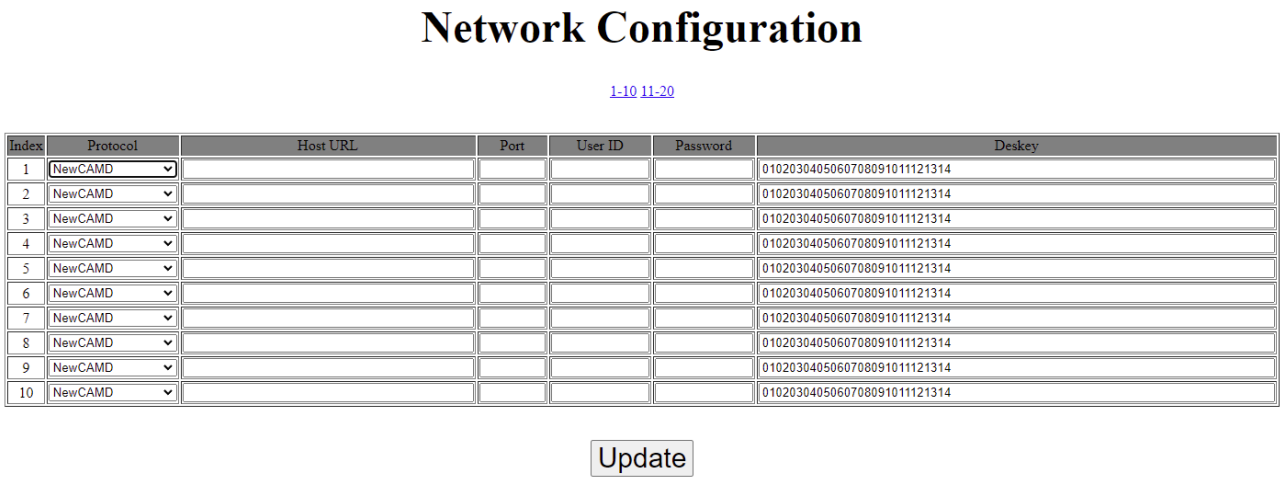

Figura 8 – Página Web exibida após autenticação no Troy

O conteúdo se refere à configuração de servidores de terceiros e credenciais de acesso a estes para o sistema de obtenção de chaves de decriptação de canais do Thor (*Card Sharing*, mencionado anteriormente). É claramente um servidor Web simples, com páginas estáticas em HTML e pouquíssimo código em JavaScript.

Para se ter uma noção melhor do escopo deste servidor, é possível executar uma varredura nos diretórios utilizando os comandos ***dirb***, ***dirbuster***, ***gobuster*** e afins. No entanto, todos os programas falharam, a princípio, retornando, simplesmente, falha na resposta do servidor (Troy). Isto poderia indicar uma espécie de defesa do servidor *Web* do Troy, que recusaria requisições de características suspeitas.

O que acontece, porém, é causado exatamente pelo oposto: o servidor em questão (e o dispositivo) é tão rudimentar que não suporta a velocidade das requisições e acaba colapsando, logo a resposta negativa. Além disso, o equipamento entrou em um *loop* e chegou a reiniciar inúmeras vezes durante o processo, mostrando que tal tipo de enumeração baseada em força-bruta parece não ser eficaz contra o Troy.

Felizmente, é possível obter grande parte das informações apenas analisando os códigos das páginas Web, as quais não trazem nenhuma informação atrativa. Ao analisar as requisições HTTP, observa-se o cabeçalho de resposta "*Server: Test Http Server*", o que indica um servidor Web totalmente genérico, fato que dificulta a obtenção de informações. Também se nota o envio da senha de autenticação como texto plano, sem encriptação, facilmente interceptável por um *sniffer* de rede.

Neste cenário, um agente mal intencionado com acesso à rede poderia, facilmente, obter acesso às credenciais do(s) servidor(es) cadastrados na página Web e usufruir dos serviços de *Card Sharing* às custas do usuário do Troy.

É válido também cadastrar as informações do computador que realiza a testagem (IP e porta) em uma linha da tabela exibida na página Web do Troy, na tentativa de que o aparelho estabeleça conexão reversa. No entanto, durante os testes, escutando com o comando ***netcat***, a conexão, assim que firmada, era imediatamente encerrada pelo Troy, destruindo a possibilidade de um *reverse shell* – situação em que o aparelho alvo forneceria ao atacante acesso a uma sessão de terminal.

Como a baixa capacidade do aparelho torna o processo de testagem bem incapacitante – e levando em conta que dados sensíveis já foram descobertos -, é plausível classificar a insistência na operação como inviável.

Segue-se, então, à próxima porta reportada pelo *scan* do ***nmap***: a 554, correspondente ao serviço RTSP (*Real Time Streaming Protocol*). Inicialmente, a enumeração

de diretórios nesta porta através da execução de scripts pré-existentes do ***nmap*** retorna uma infinidade de possíveis caminhos, vide Figura 9.

```
PORT    STATE SERVICE VERSION
554/tcp open  rtsp    feng rtspd 2.1.0_rc1
|_rtsp-methods: OPTIONS,DESCRIBE,SETUP,PLAY,PAUSE,TEARDOWN
| rtsp-url-brute:
|   discovered:
|     rtsp://192.168.1.100/0
|     rtsp://192.168.1.100/1
|     rtsp://192.168.1.100/11
|     rtsp://192.168.1.100/0/video1
|     rtsp://192.168.1.100/1.AMP
|     rtsp://192.168.1.100/1/1:1/main
|     rtsp://192.168.1.100/1/cif
|     rtsp://192.168.1.100/1/stream1
|     rtsp://192.168.1.100/12
|     rtsp://192.168.1.100/4
|     rtsp://192.168.1.100/CAM_ID.password.mp2
```

Figura 9 – Script de enumeração RTSP do nmap (resumido)

Em seguida, deve-se partir ao teste de acesso efetivo ao conteúdo desses *endpoints*, o que pode ser realizado através de requisições RTSP diretas (semelhantes às HTTP) ou com o auxílio de programas especializados. No entanto, novamente, os testes sequenciais de acesso a essa porta são relativamente lentos, uma vez que o dispositivo frequentemente colapsa e exige longos minutos para restabelecer os serviços.

O interessante, porém, é que, de fato, absolutamente todas as *streams* listadas retornam código de resposta 200 (OK), o que significa que são perfeitamente acessíveis e completamente sem autenticação: mais uma falha grave. Logo, basta que o usuário do Troy possua uma destas configurada e funcional para que um *hacker* obtenha acesso ao conteúdo transmitido.

Avança-se à próxima porta descoberta: a 6000, convencionalmente associada ao serviço apelidado de X11 (*X Window System*), um framework para aplicações que provê interface gráfica remota e entrada de comandos para sistemas baseados em Unix. De início, a execução do script pré-contido no pacote ***nmap*** – de nome ***x11-access*** – já falha, o que indica que o computador que executa os testes não possui acesso automático ao servidor remoto (*X* Server), isto é, não consta entre os *hosts* permitidos.

A falha, no entanto, é do tipo *timeout* excedido, e não uma rejeição. Adicionalmente, a presença deste serviço é um tanto quanto duvidosa, uma vez que o aparelho não possui nenhuma funcionalidade de acesso remoto ou algum tipo de mecanismo que justifique a conexão a um servidor X11. Diversas outras tentativas com outros comandos

também falham, logo, deve ser considerada a possiblidade de um erro de implementação ou ausência de serviço escutando na porta, o que é plausível.

Parte-se, então, à próxima porta a ser testada: a 8080, caracterizada pelo ***nmap*** como o serviço de nome *Nvidia Streamer Service*. É improvável que este seja realmente o serviço em execução escutando na porta, uma vez que o Troy não possui relação alguma com os nomes em questão. Além do mais, esta porta é conhecida por ser uma alternativa/um complemento à 80, executando também um servidor *Web* HTTP (caso mais provável ao Troy).

Portanto, deve-se seguir um caminho semelhante ao utilizado no teste da porta 80. Primeiramente, realiza-se o ato de simplesmente acessar a URL por um navegador (na forma IP:8080), o que retorna código de erro 404 (não encontrado), mostrado na Figura 10. Apesar da resposta aparentemente negativa, ao menos tem-se a certeza de que há um servidor Web funcionando nesta porta.

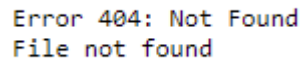

Figura 10 – Servidor Web do Troy na porta 8080

É razoável, então, tentar enumerar os diretórios e arquivos deste servidor, a fim de conseguir acesso a algum conteúdo possivelmente desprotegido. Para tanto, usa-se novamente programas do estilo ***gobuster***. A execução dos comandos, dessa vez, é bem mais normal e, consequentemente, bem mais rápida; o servidor não colapsou nenhuma vez durante os testes. No entanto, não foi descoberto um único caminho passível de ser acessado pelo navegador; todas as tentativas recaíram sob a mesma resposta.

Este resultado, aliado ao conhecimento sobre o aparelho, quase que certamente indica, novamente, um erro de implementação do equipamento. Tal servidor Web é totalmente desnecessário, uma vez que não há absolutamente nenhuma outra funcionalidade no Troy que possa utilizar de mais uma interface Web. Este é, portanto, um serviço deixado ativo por descuido, apenas mais uma porta aberta sem necessidade: um perigo em potencial para o usuário do aparelho.

Mesmo assim, caso houvesse algum conteúdo relevante no servidor, este poderia ser facilmente manipulado por um agente mal intencionado, uma vez que a autenticação utilizada para os métodos HTTP *PUT* e *DELETE*, por exemplo, é do tipo *digest*.

Esta, por sua vez, é considerada fraca, e, aliada ao fato de a conexão não ser segura (HTTPS), permite que a *hash* seja obtida pela simples interceptação do cabeçalho dos pacotes com um *sniffer*¸ a qual pode ser decifrada com auxílio de diversas ferramentas.

Avança-se, portanto, à porta na sequência: 18053, uma das que aparecem rotuladas como *filtered* pela busca do ***nmap***. Tal rótulo indica que o comando não foi capaz de efetuar conexão e trocar dados com um serviço escutando por esta porta de maneira padrão. A princípio, em situações normais, este fato se dá por uma regra de firewall ou restrição do próprio sistema para aumentar a segurança do recurso, no entanto, levando-se em consideração todo o conhecimento sobre o Troy, é possível que se trate, de fato, de uma porta fechada (um falso-positivo).

Para resolver tal ambiguidade, pode-se executar a busca do ***nmap*** novamente, desta vez focando apenas nas portas com estado *filtered*. O resultado é, portanto, o esperado: a porta aparece como completamente fechada, o que esclarece que houve, de fato, uma inconsistência no Troy durante o *scan*, que gerou resultados equivocados.

Presume-se, então, que todas a outras portas com a mesma característica sejam também fruto de uma falha momentânea do equipamento. Testa-se, pois, cada uma destas, individualmente. Novamente, a resposta obtida revela falha total na conexão para todas as portas da lista, o que indica, com altíssima chance, que tais serviços realmente não existem no aparelho.

Retorna-se, por conseguinte, à próxima porta cujo status reportado era *open*: a 18888. Esta, rotulada como o serviço de nome *apc-necmp* pelo ***nmap***, possui informações quase inexistentes por entre a internet, o que torna o processo de investigação complicado.

Inicialmente, realiza-se a conexão pelo comando ***telnet***, a fim de estabelecer comunicação de baixo nível com o dispositivo e analisar o comportamento da porta. Esta, no entanto, age de maneira estranha, retornando – intermitentemente – o mesmo conteúdo – com menção ao termo *test* – para diferentes requisições, como visto na Figura 11.

Figura 11 – Retorno da porta 18888 do Troy

Adicionalmente, simplesmente acessar a URL – via HTTP, especificando a porta 18888 – pelo navegador desencadeia o *download* de um arquivo. Este, sem nome atribuído, possui cerca de 61KB e, ao ser visualizado como um texto, revela o conteúdo

“abcde12345”, seguido de 61914 espaços em branco, evidenciando, claramente, o caráter de um servidor de teste.

As tentativas de explorar vulnerabilidades – e a própria existência destas – de um servidor HTTP, se tornam, entretanto, muito restritas, uma vez que não há informações sobre o servidor, não há interface Web e todos os diretórios de URL possíveis acarretam o mesmo comportamento: o download do arquivo teste. Além do mais, não há conteúdos sensíveis a serem obtidos, mesmo após potencial invasão; é, puramente, um “resquício evolutivo”, um erro deixado para trás pelo desenvolvedor do *firmware* do Troy.

Por fim, chega-se à última porta da lista: a 49153, cujo serviço atrelado é também desconhecido, segundo o resultado da busca com o ***nmap***. Esta, por meio de buscas na internet, não revela nenhuma aplicação ou protocolo convencionalmente atrelado, comportamento típico de portas cujo número é mais alto, condição em que as informações se tornam cada vez mais esparsas.

Por meio de análise com o *sniffer* de rede, nota-se que não há tráfego espontâneo na porta (bem como em nenhuma outra). Mesmo após conexão (bem-sucedida) por meio do comando ***telnet***, o aparelho não responde a nenhuma requisição enviada.

O processo de testagem, portanto, se torna restrito, uma vez que não há maneiras de interagir com o recurso sob prova (se é que este existe e funciona, de fato). Sem maiores informações privilegiadas, esse é mais longe que se pode (e qualquer outro agente mal intencionado poderia) chegar.

Com a exaustão de todas as portas providas pela busca inicial com o comando ***nmap***, esgota-se também as tarefas da fase de exploração, haja vista que todas as possibilidades de interação com o dispositivo alvo foram analisadas. Cede-se espaço, portanto, à fase de documentação.

4.1.2.4 Documentação

Nesta etapa, por sua vez, os processos de testagem realizados com o Troy são descritos mais sucintamente, a fim de registrar os pontos mais importantes do comportamento e dos resultados obtidos, juntamente a comentários pontuais e considerações finais sobre a operação no dispositivo.

Considera-se a análise realizada como do tipo *black-box*, uma vez que não houve liberdade de acesso ao equipamento e/ou informação privilegiada sobre seu

funcionamento. Toda a interação, portanto, foi realizada via conexão à distância, sem acessos administrativos elevados ou qualquer outra vantagem.

Inicialmente, o processo começou com a obtenção de informações sobre o dispositivo, seguida pela etapa apelidada de *fingerprinting.* A próxima, de nome enumeração, teve como foco a utilização da ferramenta ***nmap***. Nesta busca, as seis portas abertas encontradas foram, em ordem crescente de seus números: 80, 554, 6000, 8080, 18888, 49153. Um detalhe importante analisado durante a execução do comando é que o *scan* demorou bem mais que o normal – tanto mais para cada porta quanto mais tempo havia passado –, pois o Troy passava a demorar a responder quanto mais requisições eram feitas. Isso, consequentemente, causava a interrupção dos serviços no aparelho, que demoravam a voltar após o encerramento dos testes.

Ou seja, descobre-se a primeira falha do Troy: susceptibilidade a ataques do tipo *flooding* ("enchente" de requisições), mais conhecidos como ataques de negação de serviço (do inglês, *Denial of Service* - *DoS*).

Na próxima fase, chamada de exploração, as portas descobertas no passo anterior foram submetidas, uma a uma, ao processo de análise de vulnerabilidades.

A primeira delas (80) revelou um servidor *Web* com página de *login* simples com apenas um campo de inserção de senha. Neste cenário, simplesmente avançar – sem inserir um caractere no campo de senha – deu certo e o processo de login foi "concluído", expondo a próxima página.

Recomenda-se, fortemente, forçar o usuário a cadastrar uma senha ao iniciar o uso do aparelho, ou então, ao menos, fornecer uma senha aleatória pré-cadastrada, a fim de aumentar a segurança desta autenticação. Adicionalmente, se o login estiver, incondicionalmente, funcionando com uma senha vazia, deve-se, obviamente, corrigir tal comportamento vulnerável.

De qualquer forma, um agente mal intencionado poderia facilmente interceptar o tráfego da requisição de login de um usuário do Troy utilizando um *sniffer* de rede. Isto se dá pelo fato de que o conteúdo desta é inteiramente transferido em texto plano (inclusive a senha). O ideal seria, portanto, encriptar o tráfego HTTP entre a página Web e o servidor do Troy.

Além disso, a página *Web* revelada após autenticação, por sua vez, mostra credenciais confidenciais inseridas pelo usuário, o que configura, portanto, uma grave falha de segurança. Vale mencionar também que, durante os testes, qualquer tentativa de enumerar diretórios nesta porta – algo que utiliza alta frequência de requisições – derrubava o servidor

HTTP e causava colapso no aparelho, que chegava até a desligar/reiniciar algumas vezes, o que reforça a susceptibilidade a ataques do tipo *flood/DoS.*

Avançando à segunda porta (554), a enumeração de possíveis diretórios RTSP por força-bruta revelou uma enorme quantidade de *endpoints* existentes. Em sequência, todos retornaram sucesso na conexão, com código de status 200 (OK), o que significa que são completamente abertos e não requerem qualquer tipo de autenticação.

Portanto, basta que o usuário do Troy possua um destes caminhos configurado e funcional para que um agente mal intencionado tenha acesso ao conteúdo transmitido: uma falha de segurança grave. Recomenda-se o uso de autenticação para estes extremos.

A próxima porta da lista (6000), geralmente, associada ao *framework* de nome X11, acarretou testes de acesso sempre terminados em falha. No entanto, os retornos foram do tipo “limite de tempo excedido” e não do tipo “falha na autenticação”, o que indicou um problema no estabelecimento da comunicação com tal porta no aparelho.

O fato de o Troy ser um simples decodificador de canais e não possuir qualquer tipo de funcionalidade de acesso remoto, aliado às consistentes falhas na conexão à esta porta, em adição à total nulidade de tráfego neste recurso (observado com *sniffer* de pacotes), fizeram com que o veredito fosse de que o equipamento não possui, de fato, um serviço funcional escutando por trás dessa porta; pode se tratar de um erro de implementação ou serviço desconhecido não descoberto durante os testes.

Seguindo à próxima, de número 8080, outro servidor Web foi descoberto, cujas requisições eram completadas normalmente. Estas, no entanto, retornavam com código de status 404 (não encontrado), o que indica a não existência de conteúdo para o diretório base – posteriormente, constatou-se que não havia conteúdo para nenhum caminho.

Novamente, o veredito se baseia na hipótese de que tal recurso foi implementado erroneamente e/ou esquecido no aparelho, devido a uma série de razões, entre as quais se incluem a desnecessidade de outra interface *Web* no Troy, a ausência de tráfego na porta, a inexistência de diretórios acessíveis via protocolo HTTP GET e o caráter do equipamento.

A próxima porta testada, 18888, foi, certamente, uma das mais diferentes tratadas durante o processo, devido à falta de informações. Neste cenário, um teste de acesso à porta pelo navegador carregou a requisição e imediatamente iniciou o *download* de um arquivo. Este, sem nome especial, possuía cerca de 61KB, e, ao ser aberto como um arquivo de texto, revelou os dizeres “abcde12345” seguidos de 61914 espaços vazios.

Em virtude disso, a porta foi classificada como uma óbvia falha de implementação – à medida que tal teste esquecido no sistema apenas expõe mais um ponto de acesso, e, pior ainda, totalmente sem utilidade – que pode comprometer a segurança do dispositivo (ou, ao menos, seu funcionamento, caso seja, por exemplo, alvo de ataques do tipo *DoS*).

A última porta, por sua vez, de número 49153, deu origem à etapa mais curta de todo o processo de testagem. Aliada à completa falta de informações sobre a porta, a total inexistência de fluxo de pacotes nesta direção e a incapacidade de resposta do dispositivo a comandos do tipo ***telnet*** e ***netcat*** fizeram com que a continuação da análise se tornasse praticamente impossível. Esta, portanto, teve de ser encerrada, com veredito semelhante aos anteriores, com o agravante de que, desta vez, não houve resposta substancial alguma da porta.

Por fim, pode-se concluir que o Troy é um aparelho multimídia com sistema de recursos de Internet incipientes. Seria possível argumentar, entretanto, que a complexidade dos serviços é proporcional à complexidade do próprio dispositivo e suas funções – as quais são baixas –, contudo, quando se trata de aplicações que lidam diretamente com dados pessoais sensíveis – tal como credenciais de acesso –, tal analogia deve ser terminantemente refutada.

Neste cenário, uma série de recomendações poderiam ser geradas e um conjunto de melhorias atribuídas ao sistema do aparelho. Primeiramente, quanto às várias portas cujos serviços são "resquícios de desenvolvimento" – essencialmente, testes –, estas devem ser obviamente fechadas, por uma conjunção de motivos, não somente relacionados à segurança.

Isto é, dar-se ao luxo de manter uma porta não essencial aberta acarreta apenas em mais um extremo para entrada de requisições. Haja vista que o Troy possui *hardware* relativamente "fraco", lidar com mais tráfego impacta severamente no desempenho do aparelho, culminando, possivelmente, em um efeito de *DoS*, como observado durante os testes. Adicionalmente, uma inconsistência no trato destas requisições pode abrir brechas para pacotes com conteúdo malicioso ou comportamentos indesejados, o que apresenta perigo ao usuário do aparelho.

Em sequência, quanto às portas funcionais, estas devem ter seus mecanismos de segurança melhorados. Entretanto, de certa forma, o uso da maioria das portas do Troy é injustificável, de tal forma que manter um serviço escutando nelas – abertas, incondicionalmente – é uma exposição a riscos desnecessária. Ainda assim, não seria

necessário simplesmente fechá-las, mas deixar suas habilitações a critério do usuário, o qual as ativaria localmente pelo aparelho e as usaria apenas quando – e se – desejado.

Finalmente, conclui-se que o Troy é um aparelho vulnerável a invasões de agentes mal intencionados, do ponto de vista da privacidade do usuário, principalmente, uma vez que dados sensíveis de seu utilizador podem ser acessados facilmente, considerando situação e circunstâncias normais.

4.2 A9 Mini IP Camera

4.2.1 Descrição

O segundo dispositivo escolhido se encaixa, primariamente, na categoria de câmera de vigilância sem fio via *Wi-Fi*, com a possibilidade de gravar localmente em cartão de memória. Este fora escolhido por seu baixo custo de obtenção e – consequentemente – alto volume de vendas em plataformas na Internet, o que implicaria na presença em grande número de residências.

Neste equipamento, os modos de conexão existentes são: o Ad Hoc, onde a câmera se conecta diretamente ao dispositivo (celular) através de uma rede Wi-Fi gerada por ela; e o AP (*Access Point*), onde a câmera se conecta ao dispositivo através de um ponto de acesso Wi-Fi (o que permite conexão à distância). A Figura 12 apresenta uma foto do dispositivo, enquanto a Tabela 2 apresenta as suas especificações técnicas.

Figura 12 – A9 Mini IP Camera.
Fonte: <https://www.clickfull.com.br/a9-mini-camera-seguranca-espia-c-bateria-a9>.

Tabela 2 – Especificações técnicas da A9 Mini IP Camera

Resolução	1920x1080
Taxa de quadros por segundo	30
Formato de vídeo	AVI
Formato de compressão	H.264

Curiosamente, todo o conteúdo disponível na Internet – inclusive no próprio manual de utilização do produto – não faz menção alguma ao uso do aparelho no computador, por exemplo; tudo o que há é um tutorial para conexão da câmera a um aplicativo para celulares. Tal fato vai contra a hipótese de que a câmera é realmente um dispositivo de imagem via protocolo IP, uma vez que isto implicaria em poder ser acessada através de qualquer plataforma.

A característica em questão pode representar, de fato, o uso de um diferente protocolo de comunicação – que não se preocupou em abranger outras plataformas –, advindo de uma construção mais econômica do aparelho, a julgar por sua faixa de preço mais baixa. De qualquer modo, se a conexão ao celular é realizada, certamente há um modo análogo para espelhar tal ação no computador, a ser testado a seguir, juntamente à sua segurança.

4.2.2 Teste de Invasão

Para os testes, assume-se que o aparelho, a ser chamado por A9 ao longo do processo, está devidamente funcional e acessível na rede local. Este, conectado via Wi-Fi ao roteador central, recebeu o endereço IP fixo 192.168.1.102, para facilitar a identificação.

4.2.2.1 Fingerprinting

Novamente, como não é possível encontrar nenhuma informação relacionada ao sistema operacional embarcado no dispositivo por meio de suas especificações ou pesquisas, o objetivo principal da primeira fase se torna extrair essas informações. Para tanto, inicia-se o arsenal de recursos com o programa ***p0f***.

Assim como no teste anterior – com o Troy –, ao tentar utilizar tal programa, não se obteve nenhuma tentativa de conexão por parte da A9, uma vez que a câmera não realiza conexões espontâneas a equipamentos disponíveis na rede local –

comportamento esperado e seguro. Portanto, é forçada uma conexão via comando ***telnet*** à A9, o que produz resultados, como mostra a Figura 13.

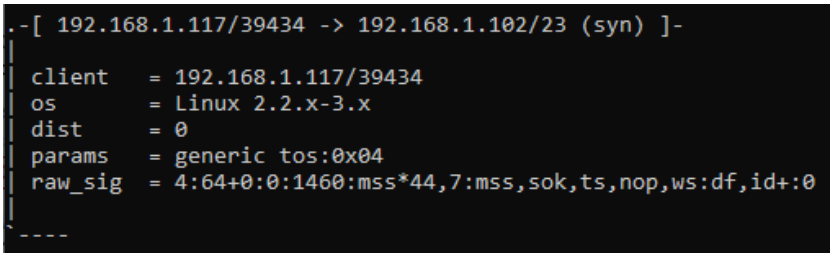

Figura 13 – Retorno do comando ***p0f*** com a A9

O retorno do comando, desta vez, é também idêntico àquele observado no teste do Troy: há a indicação do possível sistema operacional como um baseado no *kernel* do Linux, em versão entre 2.2.x e 3.x. Esta é uma amplitude ainda muito grande, o que requer que as análises continuem, a fim de estreitar mais tal informação.

A próxima ferramenta utilizada nesta fase é o comando ***ping***. O uso deste, com alvo na A9, revela um *TTL* de 255 – vide Figura 14 –, o que indica o uso de um sistema baseado em Unix – certamente, com funcionalidades reduzidas, corroborando o resultado do utilitário anterior.

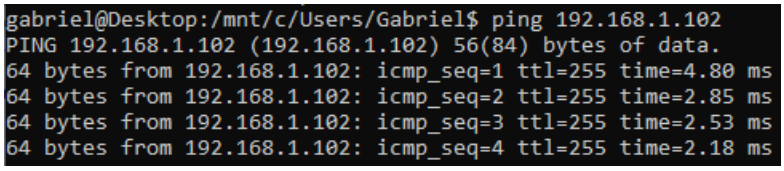

Figura 14 – Comando ***ping*** na A9

No entanto, é aconselhável refinar ainda mais a especificação do sistema. Parte-se, então, para o comando ***nmap***, cujo uso – da mesma maneira do teste anterior – retorna uma lista de sistemas compatíveis, mostrada na Figura 15. Todos os nomes listados se referem a sistemas embarcados em dispositivos *IoT*, o que confirma o caráter da A9 já debatido anteriormente.

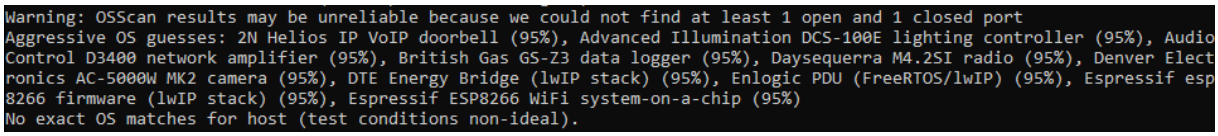

Figura 15 - Comando ***nmap*** para detecção de sistemas na A9

Este resultado, entretanto, ainda é vago e não acrescenta muito ao dossiê de informações da A9. Como um último recurso, usa-se o programa de nome ***xprobe2***, o qual apresenta resultados mais específicos, com a definição de um sistema operacional detalhado: o *NetBSD*, em versões antigas (possivelmente vulneráveis), de forma muito semelhante ao Troy, tal como representado na Figura 16.

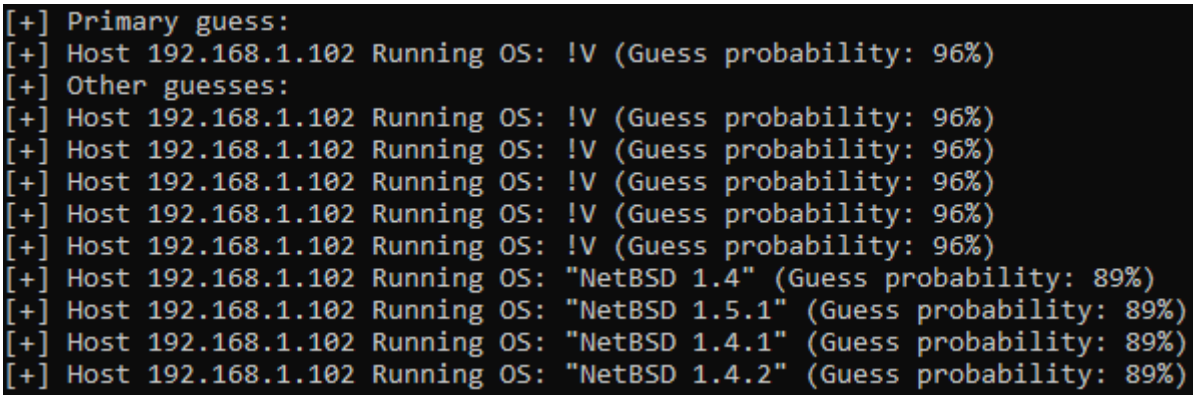

Figura 16 – Comando ***xprobe2*** na A9

Contudo, durante os testes, um dado chamou a atenção: o próprio *hostname* da A9 (associado a seu endereço MAC – emitido para o roteador) apareceu como "*rtthread*", o que – constatado após pesquisas pelo nome – significa literalmente o nome de seu sistema operacional, o RT-Thread OS [16], um sistema voltado exatamente a dispositivos embarcados, com versões para ambientes com recursos restritos e passível de ser customizado sob demanda para equipamentos *IoT* semelhantes.

Pode-se concluir, portanto, que os recursos utilizados para detecção do sistema operacional chegaram perto do sucesso, mas não atingiram a exata identificação do nome. Neste caso, por exemplo, um dado trivial acabou revelando a informação desejada, fato que merece destaque, a fim de conhecimento. Com a informação do sistema obtida de maneira mais satisfatória, é possível prosseguir à fase seguinte.

4.2.2.2 Enumeração

Avança-se à enumeração, a fim de descobrir e listar as formas de conexão possíveis com o aparelho testado, neste caso, portas e seus respectivos serviços associados. Para tal, usa-se o comando ***nmap*** – já citado e utilizado em etapas anteriores – desta vez em sua função principal: o mapeamento de portas.

O uso do comando (com argumento -sV, para detecção de serviços) em toda a extensão de portas (65535 valores) retorna, após poucos segundos de execução – menos de um minuto –, dados incomuns: absolutamente nenhuma porta respondeu às requisições, ou seja, não há um único serviço do tipo TCP acessível na A9, o que, certamente, restringe muito as possibilidades de sucesso do teste.

No entanto, ainda há outras chances, para as mesmas 65535 portas, porém, utilizando o UDP como arquitetura de pacotes. O teste, portanto, – substituindo o argumento -sV pelo -sU – apresenta, após poucos segundos de execução, três possíveis portas da A9 – demonstradas pela Figura 17 – às quais uma conexão possa ser realizada.

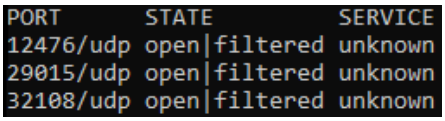

Figura 17 - Comando ***nmap*** para detecção de portas na A9

Todas estas, entretanto, são classificadas pelo programa como no estado "*open*|*filtered*", o que acontece para tipos de *scan* onde as portas abertas não dão nenhuma resposta. Tais portas não são uma garantia de sucesso, contudo, há grandes chances de êxito, uma vez que este retorno é o melhor possível para o teste no modo UDP.

Com a enumeração finalizada, é possível prosseguir à próxima fase, onde as informações descobertas serão, de fato, exploradas.

4.2.2.3 Exploração

Neste cenário, segue-se a lista porta a porta, individualmente, investigando seu funcionamento e explorando suas possíveis vulnerabilidades, de forma semelhante à maneira como seria realizada caso fossem portas TCP. A diferença possivelmente se encontrará na dificuldade – um pouco maior neste caso –, haja vista que portas UDP são levemente menos populares e têm menos informações disponíveis.

Outro fator crítico é a diferença de comportamento de portas UDP, em relação às TCP: as primeiras simplesmente não respondem a uma solicitação de conexão para indicar "sucesso", logo, sem acesso ao dispositivo alvo para configurar tal rotina, até mesmo saber se a porta realmente recebeu a informação se torna impossível.

Inicialmente, a começar pela porta 12476, a pesquisa por informações relacionadas na Internet termina malsucedida. Isso significa, portanto, que não há padrão ou

convenção de uso estabelecida para essa porta; qualquer tipo de serviço pode estar sendo executado por detrás dela na A9.

O mesmo se repete para as outras duas portas da lista, o que restringe o modo de ação normal do teste e requer a troca para uma abordagem diferente. Adicionalmente, a pesquisa por câmeras semelhantes na Internet traz alguns resultados úteis: há, até mesmo, microprojetos de análise e *hacking* destes dispositivos, contendo informações úteis.

No entanto, a grande maioria destes traz câmeras – aparentemente – mais expostas, com várias portas TCP disponíveis, incluindo a 23 (do comando ***telnet***), algo que transforma – e facilita – o processo. Isto significa que, para o caso desta A9, o procedimento será mais complicado e requererá outra estratégia.

De acordo com dados obtidos das pesquisas, quando funcionando no modo AP (que permite conexão à distância), a câmera faz uso de um servidor externo (quase sempre um domínio chinês) para se comunicar com o dispositivo que a requisita (celular) e firmar a conexão. Tal fluxo de conexão abre brechas para problemas, uma vez que dados privados do usuário podem estar sendo expostos em algum ponto do caminho.

De fato, durante os testes, ao realizar a configuração inicial da A9 no modo AP e tentar acessá-la pelo celular – mesmo com ambos os dispositivos na mesma rede local –, a câmera realizou uma requisição de DNS aos domínios “cn.ntp.org.cn”, “ntp.ntsc.ac.cn”, “us.jxldz.icu” e “eu.jxldz.icu”, de origem chinesa. Ao iniciar a aplicação no celular, este tenta encontrar a câmera realizando uma emissão em *broadcast* para uma das portas UDP descobertas pela busca do ***nmap*** na A9.

Assim que estes servidores externos sincronizam as informações para ambos os dispositivos, a comunicação entre a câmera e o celular passa a ser P2P (*Peer-to-Peer*) por meio de um túnel UDP e uma conexão à Internet externa não é mais requerida. Em vista disso, seria possível concluir que a A9 não – simplesmente – compartilharia a transmissão do vídeo com um servidor externo, apenas configurações de conexão.

Essencialmente, esta maneira de conexão significa que, uma vez configurada, a A9 pode ser acessada de qualquer lugar do mundo, contanto que se saiba o código UID da câmera (fixo, igual ao SSID da rede Wi-Fi emitida por ela), nome de usuário e senha (cadastrados pelo utilizador) – dados requeridos na aplicação do celular, conforme a Figura 18.

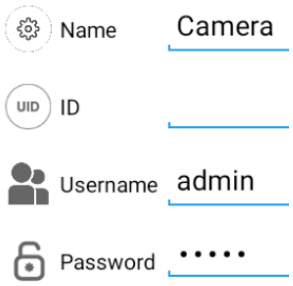

Figura 18 – Informações necessárias no aplicativo da A9 para celular

Mesmo após compreender os mecanismos e ciclos do túnel UDP – através de observação de pacotes na rede – e tentar replicá-los à mesma porta por meio de outro aparelho de origem – ao estilo "*man-in-the-middle*" –, a A9 permaneceu fiel ao dispositivo original e não alterou sua conexão, ignorando todas as tentativas de sequestrar o tráfego.

Entretanto, ao executar o *scan* pelo ***nmap*** novamente, constatou-se que, periodicamente – a cada sessão –, uma nova porta UDP se tornava disponível na câmera. Neste cenário, portanto, ao tentar simular toda a conexão desde o começo por meio de outro aparelho de origem nesta nova porta, a A9 respondeu positivamente e o túnel UDP foi firmado com sucesso. Isto é, ao copiar o fluxo de dados intercambiados entre os aparelhos – em hexadecimais, não encriptados, obtidos através de um *sniffer* de rede – e reproduzi-los, a câmera simplesmente aceitou todos os comandos.

Tal procedimento foi realizado por meio da criação de um algoritmo na linguagem *Python* implementando a biblioteca *socket*. Após analisar o comportamento do dispositivo e entender a lógica da ordem dos dados enviados e recebidos, foi possível instruir o programa a reproduzi-los. Este, portanto, passou a simular, em sequência, as requisições UDP capturadas pela análise de pacotes realizada anteriormente, até que a conexão fosse firmada com sucesso.

Este resultado, por sua vez, significa que é possível ignorar o dispositivo oficialmente autenticado com a A9 e tomar posse do equipamento através de outro aparelho (o de um possível agente mal intencionado). Configura-se, portanto, uma grave falha de segurança, uma vez que qualquer autenticação se torna irrelevante, à medida que "apenas" copiar o fluxo de dados dos pacotes – já "autenticados" – se mostra eficaz.

Com a exaustão de todas as portas providas pela busca inicial com o comando ***nmap*** e com a função principal – e única – da A9 violada, esgota-se também as

tarefas da fase de exploração, haja vista que todas as possibilidades de interação com o dispositivo alvo foram analisadas. Cede-se espaço, portanto, à fase de documentação.

4.2.2.4 Documentação

Considera-se novamente a análise realizada como do tipo *black-box*, mantendo as mesmas condições da anterior. O processo, inclusive, começou de maneira semelhante, com a obtenção de informações sobre o dispositivo. Os dados retornados, por suas vezes, indicaram um poder computacional modesto na A9, aliado a um *chip* voltado à programação embarcada/*IoT*, o que sugeriu a provável presença de um sistema operacional compatível: um baseado em Unix.

Além disso, o próprio *hostname* da A9 – "rtthread", revelado durante as buscas – indicava exatamente seu sistema operacional: o RT-Thread OS, um sistema voltado à programação embarcada em dispositivos *IoT*. Em sequência, a mesma abordagem com o comando ***nmap*** foi aplicada, o que revelou – após múltiplas checagens – a inexistência total de portas TCP disponíveis no aparelho.

Alterou-se a busca, então, para focar em portas UDP, o que revelou a presença de três portas, posteriormente refinadas e classificadas em: duas portas de controle, cujo número é fixo, e uma porta de acesso efetivo, cujo número varia a cada sessão (início/reinício) do aparelho.

Ao observar o fluxo de dados em um *sniffer* de rede por um dado período de tempo, foi possível extrair um padrão: diversas trocas de mensagem – com conteúdo pequeno e seguindo padrões levemente variáveis – entre a câmera e a aplicação, alternadas, seguidas de novas mensagens, no mesmo padrão, mas desta vez com conteúdo maior, compatível a uma transmissão multimídia.

Tais mensagens, por seus turnos, carregavam informações não criptografadas, o que fez com que fosse possível coletar seus conteúdos e organizar uma coleção de dados atrelada à ordem em que eram enviadas nos pacotes. Após algumas tentativas com um algoritmo customizado de envio e recebimento de pacotes via *socket* UDP, finalmente, os dados e suas sequências de envio foram acertados, e a câmera firmou a conexão, estabelecendo o túnel UDP permanente com o programa.

A partir de então, com o procedimento de *handshake* finalizado, bastava enviar a mensagem equivalente à requisição do conteúdo multimídia. A câmera, por sua vez, iniciava a transmissão dos pacotes direcionados à aplicação desenvolvida. Deste modo, sem

acesso a nenhuma credencial de acesso ou senha, apenas copiando o fluxo de pacotes da rede, foi possível autenticar-se na A9 como um usuário legítimo e simular precisamente a ação do aplicativo para celular.

Configura-se, portanto, uma grave falha de segurança, uma vez que todas as credenciais de autenticação estão sendo literalmente ignoradas e o conteúdo da câmera exposto a qualquer agente mal intencionado com acesso à rede. Finalmente, como remediação, sugere-se o uso de tráfego encriptado entre as instâncias, de tal forma que um usuário externo observando os pacotes não consiga obter acesso ao conteúdo verdadeiro carregado por estes.

5 PROTOCOLO DE TESTAGEM

Em virtude das análises realizadas anteriormente, levando-se em conta tanto os procedimentos que as compuseram quanto os resultados finais, torna-se evidente a importância do processo de testagem de segurança de dispositivos, comprovada pelas inúmeras inconsistências encontradas nos equipamentos durante as verificações.

Adicionalmente, nota-se também, a necessidade da existência de uma padronização dos métodos das avaliações, uma vez que uma infinidade de aparelhos *IoT* – provavelmente por seu caráter e criação recentes – não possuem informações relevantes disponíveis, o que dificulta o processo – que seria realizado mais às cegas que o esperado – e o torna ainda mais suscetível a falhas.

Neste cenário, desenvolver e compartilhar um protocolo para testes de segurança em dispositivos *IoT* se torna valioso, à medida que tornar aqueles mais acessíveis e assertivos também significa democratizar e popularizar sua realização. Logo, mais testes – abrangendo mais dispositivos – atingem maior poder de divulgação e expressão, alcançando os respectivos desenvolvedores e/ou responsáveis, bem como a comunidade utilizadora – maior beneficiada com as informações.

5.1 Manual de Procedimentos Técnicos

Considerando todo o exposto até o momento sobre os dispositivos da categoria em questão, reitera-se sua esperada e comum simplicidade, passível de ser explorada com as técnicas de análise e reação empregadas acima, as quais serão reforçadas abaixo, de maneira ainda mais prática.

A começar, assume-se que o dispositivo *IoT* a ser testado está devidamente ligado e funcional, se encontra conectado à Internet e é acessível, pela rede, a partir do dispositivo de origem utilizado na análise.

Neste cenário, admite-se, também, que o dispositivo alvo da avaliação é unicamente determinado e conhecido, tornando desnecessário qualquer tipo de varredura pela rede em busca de dispositivos. Por sua vez, isto implica que se tem acesso a informações, tais como o endereço IP e MAC do aparelho, para facilitar o processo – uma vez que o objetivo é provar a segurança do equipamento em si, não de outros fatores, como a rede – mas não se extrapolam as vantagens a ponto de caracterizar um teste do tipo *white-box*.

5.1.1 *Fingerprinting*

O fato de conhecer o dispositivo alvo não significa, necessariamente, que se possua conhecimento sobre seus detalhes técnicos ou especificações, logo, a fase de *fingerprinting* pode ainda ser necessária. Os procedimentos abaixo servem, majoritariamente, para revelar dados sobre a construção do equipamento, tais como o sistema operacional utilizado, conhecimento de grande valia para o executor do teste.

Inicialmente, pode-se fazer uso do programa ***p0f***, seguindo o molde:

```
1.  sudo p0f
```

Por ser um rastreador passivo, caso o dispositivo alvo não exerça contato com o computador de origem, é possível forçar a conexão realizando qualquer comando que troque dados com uma porta de rede do alvo, tal como:

```
1.  telnet IP_DO_DISPOSITIVO
```

Mesmo que o aparelho não possua tal serviço e/ou sua porta equivalente esteja fechada, haverá uma resposta – neste caso, negativa –, o que é suficiente para que o ***p0f*** detecte os pacotes, realize a análise e mostre as informações coletadas.

A seguir, pode-se usar o próprio comando ***ping***, nativo em muitos sistemas, seguindo o padrão:

```
1.  ping IP_DO_DISPOSITIVO
```

Caso o dispositivo não bloqueie pacotes do protocolo ICMP – utilizado pelo ***ping*** –, o processo será concluído e trará as estatísticas padrão já conhecidas. Dentre estas, a única importante para o momento é o TTL (*Time to Live*), uma marca cujo valor é singular de acordo com o sistema executado pelo aparelho, seguindo o padrão:

- 64 – Unix/Linux (principais distribuições mais robustas)
- 128 – Windows
- 255 – Unix (outros sistemas mais simplificados)

Logo, o valor mais próximo ao resultado obtido na execução representa uma dica importante acerca do ambiente em execução no equipamento. Para os dispositivos *IoT* em questão, entretanto, o resultado deve ser, frequentemente, 255, uma vez que são movidos por sistemas baseados no *kernel* do Unix em versões reduzidas, para sistemas embarcados.

Outro bom programa para efetuar esta análise é o ***xprobe2***, um algoritmo que analisa as características dos pacotes de rede do equipamento para determinar seu sistema operacional. De maneira simplificada, seu uso deve ser da forma:

```
1.  sudo xprobe2 [-p PROTOCOLO:PORTA:ESTADO] IP_DO_DISPOSITIVO
```

Tal que PROTOCOLO seja “tcp” ou “udp”, PORTA seja o número da respectiva porta direcionada e ESTADO seja “open” ou “closed”, de acordo com a situação. É aconselhável fornecer tal informação no argumento “-p” para aumentar a acurácia do programa, contudo, caso não se possua o dado, o argumento pode ser omitido.

Entre os resultados obtidos, estarão listados possíveis sistemas operacionais em execução no aparelho alvo, acompanhados de suas probabilidades, estimadas pelo programa.

Por fim, o último recurso recomendado para este fim é o ***nmap***, programa com muitos recursos para diversos assuntos relacionados ao atual. Neste caso, seu uso deve seguir a forma:

```
1.  sudo nmap IP_DO_DISPOSITIVO -O -v [--osscan-guess]
```

Este comando, por sua vez, explora diversos traços da comunicação com o dispositivo e estima o ambiente em execução no aparelho.

Após todas estas execuções, as especificações do sistema operacional do equipamento certamente estarão melhor definidas e a próxima fase do processo de testagem se tornará mais prática.

5.1.2 Enumeração

Para esta etapa, o protocolo roga a utilização do *scan* do programa ***nmap*** como recurso inicial e principal. Este pode ser dividido em duas etapas, de acordo com a característica do dispositivo:

Caso haja comunicação via protocolo TCP disponível, pode-se executar a análise através do seguinte comando:

```
1.  sudo nmap IP_DO_DISPOSITIVO -sV -v -p -
```

Este realiza uma varredura em todas as 65535 portas do dispositivo alvo, utilizando o protocolo TCP, enviando pacotes e examinando seu comportamento, a fim de detectar o estado das respectivas portas. Os resultados, por suas vezes, representam as portas classificadas como “abertas” ou, ao menos, “filtradas”, acompanhadas de seus respectivos serviços convencionados.

Tal lista significa, na prática, uma pré-seleção muito eficaz de caminhos nos quais o teste de exploração pode se estabelecer. Contudo, caso absolutamente nenhuma porta seja reportada pela busca, o dispositivo certamente emprega diretrizes de segurança mais refinadas e não expõe suas portas incondicionalmente, ou simplesmente não possui, de fato, portas TCP acessíveis.

Caso este seja o caso, deve-se partir para a análise com o protocolo UDP, a ser realizada de acordo com o molde:

```
1.  sudo nmap IP_DO_DISPOSITIVO -sU -v -p -
```

Esta funciona de maneira análoga à anterior de diferente protocolo. A diferença, no entanto, recai sobre a natureza de portas desse tipo, as quais simplesmente não respondem afirmativamente a uma tentativa de conexão indicando sucesso. Tal característica torna o processo um pouco mais dúbio, contudo, ainda assim, os resultados são de grande utilidade.

Com tais listas geradas, os pontos de partida do processo de exploração se tornam bem definidos e este pode, de fato, ser iniciado.

5.1.3 Exploração

Esta fase, por sua vez, itera sobre as entradas da recém gerada lista de portas possivelmente vulneráveis, uma a uma, e aplica processos de exploração adaptados e compatíveis à porta em questão.

Deve-se notar que, para cada uma das 65535 portas, para ambos os protocolos – TCP e UDP –, haverá um serviço atrelado. Este, idealmente, será diferente para cada uma delas, logo, é praticamente inviável citar recomendações para todas; o protocolo, portanto, abordará os mais comuns, presentes na maioria dos dispositivos, a fim de servir como um guia primário. Para casos não listados, pode-se seguir a lógica descrita nesta seção e modificar o plano, de acordo com a situação.

5.1.3.1 SSH

Inicialmente, tratando-se de portas TCP, há a possibilidade de que o equipamento testado tenha revelado a existência da porta de número 22 aberta e aceitando conexões. Esta, por sua vez, é tipicamente atrelada ao serviço SSH (*Secure Shell*), um protocolo frequentemente usado para acesso a sessões de terminal remotas, passível de existir em um dispositivo *IoT* devido à necessidade de interação com o sistema do aparelho aliada à inexistência de periféricos de entrada e saída.

Neste cenário, pode-se começar a avaliação adquirindo mais informações sobre o serviço no equipamento alvo, através do algoritmo ***ssh-audit***, a ser usado da forma a seguir:

```
1.  python3 ssh-audit.py IP_DO_DISPOSITIVO
```

Este, por sua vez, recolhe uma série de características e parâmetros de configuração da aplicação servidora do SSH e os exibe ao final da execução. Deve-se atentar para programas e versões comprovadamente vulneráveis, além das formas de autenticação habilitadas.

Em sequência, para provar a segurança do serviço em questão – considerando que a configuração deste no aparelho requeira credenciais no estilo usuário e senha –, pode-se utilizar o programa de nome ***hydra***, da seguinte maneira:

```
1.  hydra -L CAMINHO_PARA_O_ARQUIVO_DE_USUÁRIOS -P CAMINHO_PARA_O_ARQUIVO_DE_SENHAS
    IP_DO_DISPOSITIVO -e nsr ssh
```

Tal que os arquivos mencionados sejam dicionários de nomes de usuário e senhas, respectivamente, a serem obtidos e, opcionalmente, customizados, pelo utilizador. O comando mencionado testa a autenticação no serviço utilizando as inúmeras credenciais fornecidas (força bruta) e informa assim que alguma combinação seja aceita. Isso é útil, portanto, caso o aparelho utilize palavras-passe triviais ou padrão – inalteradas –, o que configura uma falha de segurança.

5.1.3.2 Telnet

Segundo, ainda para o mesmo protocolo de rede (TCP), caso, durante as buscas, o dispositivo alvo faça menção à porta de número 23 – principal, mas não exclusivamente – escutando abertamente no alvo ou qualquer termo que envolva o nome

telnet, certamente trata-se do protocolo de mesmo nome. Este, por sua vez, é muito semelhante ao mencionado no subcapítulo anterior, servindo, na maioria das vezes, para estabelecer uma conexão remota entre cliente e servidor e prover uma sessão de terminal deste àquele.

No entanto, entres esses dois, há uma grande diferença no que tange à segurança: o *telnet* simplesmente não possui mecanismos de autenticação – apenas requer as credenciais da própria conta do usuário no sistema operacional – ou proteção de dados, expondo o conteúdo intercambiado na rede sem encriptação adicional. Logo, para testar a conexão com o aparelho alvo, basta realizar o seguinte comando:

```
1.  telnet IP_DO_DISPOSITIVO
```

Caso o dispositivo responda de maneira afirmativa e o sistema alvo não requeira credenciais de acesso para a conta do usuário, o executor recebe acesso a uma sessão de terminal. Por outro lado, caso o sistema exija o *login*, é possível testar a segurança das palavras-passe através de testes de força-bruta, no entanto, é ainda mais fácil analisar o fluxo de pacotes do *telnet* na rede através de um *sniffer* e obter as credenciais assim que o usuário legítimo entra.

Portanto, caso o serviço em questão tenha sido esquecido pelos desenvolvedores e esteja em execução no dispositivo *IoT* testado, depara-se com uma ameaça à segurança de seu utilizador, com potencial para revelar dados pessoais e comprometer sua privacidade.

5.1.3.3 Servidores *Web* HTTP

Adicionalmente, também para o protocolo TCP, é possível que o dispositivo tenha reportado a presença da porta 80 – ou também 443, 8080, 8443, mais comumente – com o status aberta. Tal entrada é, geralmente, atrelada a servidores *Web* HTTP, fato que, combinado ao comum baixo poder de processamento desses aparelhos, frequentemente implica em aplicações simples e fracas, no que tange à segurança.

Para explorar vulnerabilidades em serviços desse tipo, deve-se, primeiramente, acessar a respectiva URL e analisar diretamente os conteúdos obtidos, haja vista que, muitas vezes, um erro pode ser explícito ou de fácil percepção. Deve-se atentar, também, para a possibilidade de um campo de entrada de dados ser suscetível a ataques do

tipo *SQL Injection* ou *Command Injection*. Contudo, independentemente dos resultados da inspeção, pode-se iniciar uma análise abrangente com o comando ***httprint***, sob a forma:

```
1.  httprint -h URL_DO_SERVIDOR_WEB -s CAMINHO_PARA_O_ARQUIVO_DE_ASSINATURAS
```

De tal forma que o arquivo mencionado seja o contentor das assinaturas identificadoras dos servidores *Web*, que pode ser obtido oficialmente. Este comando, por sua vez, analisa o tráfego obtido na comunicação com o serviço HTTP e o compara às entradas do arquivo de assinaturas, realizando a correlação e indicando, quando possível, detalhes da aplicação servidora, tais como seu nome e versão – dados úteis para a busca por vulnerabilidades.

Neste cenário, também é justo mencionar a utilização do comando ***nikto***, a ser feita da maneira:

```
1.  nikto -host URL_DO_SERVIDOR_WEB
```

Este é um comando que realiza uma análise automática de arquivos, conteúdos e configurações do servidor em busca de recursos ou características potencialmente vulneráveis. Os resultados, portanto, demonstram as seções do servidor e/ou das páginas *Web* que possuem artefatos possivelmente inseguros, passíveis de serem explorados.

Adicionalmente, outra boa opção a ser mencionada é o programa ***uniscan***, cuja execução se dá da seguinte forma:

```
1.  sudo uniscan -u URL_DO_SERVIDOR_WEB -qwedsgj
```

Tal comando é um *scanner* de vulnerabilidades que realiza uma análise do serviço HTTP escutando na URL fornecida e retorna os resultados obtidos. Os dados apresentados, por sua vez, são analogamente semelhantes à uma união entre os dois comandos citados anteriormente, no entanto, cada um destes apresenta uma vantagem e dados diferentes: eis a necessidade de citá-los.

Após este processo, espera-se possuir uma quantidade de informações sobre a aplicação servidora suficiente para que se busque vulnerabilidades conhecidas do modelo e versão em questão.

Um outro aspecto importante da análise de servidores *Web* é a enumeração de caminhos e diretórios, ou seja, as subseções acessíveis pela URL. Esta atividade pode ser realizada, por exemplo, através do programa ***gobuster***, pela seguinte maneira:

```
1.  gobuster dir -u URL_DO_SERVIDOR_WEB -w CAMINHO_PARA_O_ARQUIVO_DE_PALAVRAS
```

Tal que o último argumento aponte para o arquivo de texto contendo o dicionário de possíveis caminhos, que pode ser obtido oficialmente junto ao programa. Este, por sua vez, realiza inúmeras tentativas de requisições HTTP (força bruta) aos diretórios fornecidos e apresenta os resultados (códigos de status) de cada uma. Deve-se atentar para descobertas promissoras de caminhos sem segurança de acesso, que podem revelar conteúdos sensíveis.

Para uma abordagem diferente, mais focada na análise do conteúdo das requisições HTTP, pode-se utilizar tanto um *sniffer* de rede normal – como o ***Wireshark*** – ou, mais especificamente, o programa ***Burp Suite*** (ambos, com interface gráfica), que apresenta um tratamento diferenciado para este tipo de dado. Nele, é possível examinar cuidadosamente os pacotes com o protocolo HTTP e observar suas características, além de manipular as requisições realizadas.

Neste cenário, é válido focar no modo como as informações são transportadas, especialmente na encriptação do conteúdo intercambiado nas requisições. Caso não haja proteção deste – ou seja, é transportado em texto plano –, já há uma grande ameaça à segurança, que pode ocasionar vazamento de dados sensíveis.

Em sequência, uma boa estratégia é tentar quebrar a autenticação do *website* (caso haja). Para tal, quando se trata de segurança do tipo *basic*, *digest* ou *ntlm*, o programa ***patator*** é uma ferramenta útil que testa o servidor por meio de ataques de força bruta, a ser usado como:

```
1.  patator http_fuzz url=URL_DO_SERVIDOR_WEB auth_type=[basic | digest | ntlm]
    user_pass=FILE0:FILE1 0=CAMINHO_PARA_O_ARQUIVO_DE_USERNAMES
    1=CAMINHO_PARA_O_ARQUIVO_DE_SENHAS method=[GET|POST|HEAD|...]
```

Tal que os arquivos mencionados sejam dicionários de nomes de usuário e senhas, a serem obtidos e, opcionalmente, customizados, pelo utilizador. Caso a execução seja bem sucedida, o programa retornará êxito e fornecerá as credenciais de autenticação no servidor, que podem ser usadas para acessar seções restritas.

Por fim, conta-se, também, com a possibilidade de usar os scripts agrupados no programa ***nmap***, sob a forma:

```
1.  nmap -sV --script “http*” -p PORTAS_DO_SERVIDOR_WEB IP_DO_DISPOSITIVO
```

Este comando aplica todos os scripts pré-existentes no repositório do ***nmap*** relacionados a servidores HTTP à respectiva porta do equipamento. Estes, por suas vezes, executam varreduras automáticas acerca de diversas vulnerabilidades conhecidas desse tipo de serviço e retornam sucesso caso o dispositivo seja afetado por elas.

A saber das especificações e fraquezas do servidor *Web* em questão, o agente executor do teste pode personalizar os ataques para focar em pontos mais promissores, a fim de provar que as vulnerabilidades encontradas são, de fato, exploráveis.

5.1.3.4 Protocolo RTSP

Ainda se tratando de portas TCP, é possível que a enumeração das portas do dispositivo retorne a presença da de número 554 – ou também 8554, não exclusivamente. Estas, geralmente, se referem a serviços atrelados ao protocolo RTSP (*Real Time Streaming Protocol*), passíveis de serem encontradas em aparelhos *IoT* que lidam com transmissão de mídia.

Para analisar tais extremos, portanto, é possível utilizar programas de reprodução multimídia, como o ***VLC Media Player***, caso se saiba exatamente a *URL* do recurso, tal como mostrado na Figura 19. Desta forma, é possível, por exemplo, testar as credenciais de autenticação do *endpoint* – ou, até mesmo, se são realmente necessárias, uma vez que a não obrigatoriedade destas configuraria uma ameaça à segurança do dispositivo.

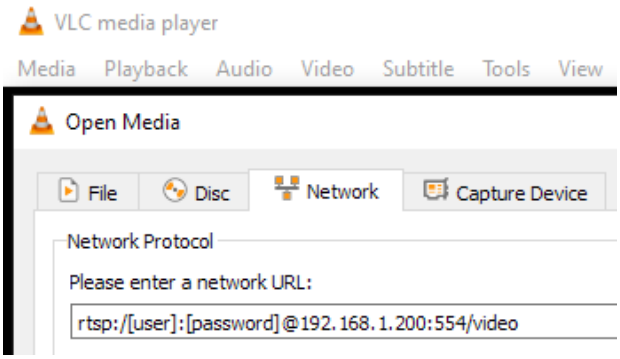

Figura 19 – Acesso a *streams* RTSP no VLC

Para um teste mais automatizado – ou quando não se tenha informação exata sobre os caminhos –, pode-se contar novamente com os *scripts* do repositório do ***nmap***, a serem aplicados da seguinte forma:

```
2.  nmap -sV --script "rtsp*" -p PORTAS_RTSP IP_DO_DISPOSITIVO
```

Este comando funciona de maneira análoga ao semelhante da seção anterior, aplicando algoritmos de análise de informação e exploração de vulnerabilidades, os quais são úteis para expor falhas de segurança ou de implementação mais simples.

Adicionalmente, pode-se partir para uma varredura dos possíveis caminhos válidos de conteúdo do servidor RTSP, para se compreender o escopo do recurso em questão. Esta ação pode ser realizada através do programa ***cameradar***, sob a seguinte conformação:

```
1.  cameradar -t IP_DO_DISPOSITIVO
```

Assim como o ***gobuster*** – mencionado no subcapítulo anterior – está para servidores *Web*, o ***cameradar*** está para serviços baseados no RTSP. Isto é, ele busca enumerar todos os *endpoints* acessíveis pela porta-alvo através de inúmeras requisições geradas por um dicionário de caminhos (força bruta). Os resultados, por sua vez, informam ao executor as URLs nas quais pode haver emissão de conteúdo, juntamente à indicação da presença ou não de autenticação.

Neste cenário, caso se obtenha sucesso na enumeração, mas se depare com extremos protegidos por autenticação, é recomendado testar a força das credenciais – a fim de descartar a possibilidade de haverem sido usadas senhas padrão inalteradas ou palavras-passe triviais. Esta atividade, por sua vez, pode ser exercida pelo script de nome ***rtsp-killer*** – um algoritmo criado na linguagem *Python* –, sob a forma:

```
1.  python3 rtsp-killer.py -t IP_DO_DISPOSITIVO
```

Este, por seu lado, realiza inúmeras tentativas de conexão aos caminhos disponíveis utilizando credenciais já conhecidas e/ou comuns a esse tipo de serviço. Caso o resultado seja positivo – e o conteúdo possa ser exibido –, pode-se afirmar que o dispositivo testado possui baixa segurança neste tipo de *streaming* de mídia, o que configura uma grande ameaça à privacidade de seu usuário.

5.1.3.5 Protocolo X11

Em sequência, também sobre portas TCP, é possível que se note a presença da de número 6000 – por convenção, mas não necessariamente. Esta, geralmente, se refere a serviços atrelados ao protocolo X11, também chamado de X-Window ou simplesmente X, um *framework* que fornece uma base para interfaces gráficas de usuário e interação remota de dispositivos de entrada e saída. Tal recurso é passível de ser encontrado em aparelhos *IoT*,

caso necessitem de uma interface de controle, uma vez que normalmente não possuem tela ou periféricos para interação direta.

Para executar provas de segurança a esta porta, pode-se aplicar o script ***x11-access***, presente no repositório do ***nmap***, através do seguinte comando:

```
1.  nmap -sV --script "x11-access" -p PORTA_DO_X11 IP_DO_DISPOSITIVO
```

Este algoritmo testa se o computador possui acesso liberado ao X11 no dispositivo alvo, o que pode acontecer caso este esteja mal configurado ou seja intencionalmente desprotegido.

Adicionalmente, caso não se obtenha sucesso, deve-se partir para o uso de um módulo do ***Metasploit Framework***, de nome ***open_x11***, acessível da maneira a seguir:

```
1.  msfconsole
2.  use auxiliary/scanner/x11/open_x11
```

A partir daí, deve-se definir as variáveis necessárias – mostradas pelo comando "*show options"* – e finalmente executar o *script* com o comando "*run"*.

Em caso afirmativo para qualquer um dos dois comandos acima, depara-se com uma grave ameaça à segurança do aparelho, uma vez que este pode ser acessado remotamente por um agente na rede sem qualquer necessidade de autenticação, expondo totalmente o ambiente do equipamento, o que traz claro potencial para ferir a privacidade do utilizador principal.

Se este for o caso, para obter tal acesso na prática, basta inserir os seguintes comandos no computador de origem:

```
1.  export DISPLAY=IP_DO_DISPOSITIVO:0
2.  xterm
```

Este último, por sua vez, inicia uma sessão de terminal do aparelho alvo remotamente e possibilita o agente conectado a, por exemplo, ter acesso sobre os conteúdos armazenados no sistema de arquivos do equipamento. Sendo o acesso ao terminal uma das vias de entrada para *hacking* mais poderosas, neste cenário, deveria ser reportada uma falha de segurança grave.

5.1.3.6 Protocolo UDP

Caso o dispositivo testado reporte a presença de portas sob o protocolo de rede UDP, o processo de análise pode tomar um rumo diferente. Devido ao fato de que portas UDP, algumas vezes, não possuem serviços convencionais atrelados – e também por sua própria natureza de funcionamento –, geralmente não é possível inferir o tipo da aplicação em execução.

Este fato torna, por exemplo, a busca de *fingerprinting* do programa ***nmap*** não tão eficaz, à medida que, usualmente, nada além dos meros números das portas são retornados, sem dados adicionais. Além disso, pesquisar informações relacionadas às respectivas portas listadas também perde eficácia, uma vez que encontrar correlação se torna mais difícil.

Neste cenário, ainda assim, pode haver informação sobre o serviço associado à porta, a qual deve ser usada como chave de busca por mais detalhes acerca da aplicação em questão. A partir daí, caso seja esta a situação, o processo de exploração se assemelha ao do TCP, análogo aos descritos nos cinco subcapítulos imediatamente acima: foca-se em um serviço ou aplicação definida, enumera-se suas vulnerabilidades – pré-existentes ou descobertas –, e explora-se uma a uma.

Por outro lado, caso os dados obtidos sejam tão rasos quanto o simples número da porta e não se consiga fatos adicionais, deve-se partir a uma exploração mais prática. Esta, por sua vez, consiste, primeiramente, na observação do comportamento e do fluxo de dados intercambiados pela porta, a qual pode ser alcançada através de um programa *sniffer* de rede, tal como o ***Wireshark***.

Por este meio, pode-se compreender as características do pacote e, consequentemente, o tipo de conteúdo que trafega. Neste processo, deve-se atentar para parâmetros de segurança, tais como a presença de encriptação dos dados – bem como seu tipo e eficácia, caso exista – e a exposição de informações sensíveis, na forma de, por exemplo, credenciais e/ou senhas de autenticação.

Adicionalmente, é possível se conectar à porta em questão através, por exemplo, do comando ***netcat*** e inserir comandos, a fim de observar sua reação – por outro canal, primariamente o *sniffer*, haja vista que pode não haver resposta da porta UDP no mesmo caminho.

Logo, à medida que se entende o funcionamento do tráfego de dados nesta extremidade do aparelho, pode-se aplicar processos de análise e exploração já conhecidos, adaptados à conformidade do protocolo UDP.

6 CONCLUSÃO

Diante de todo o exposto e com base nas pesquisas realizadas durante o trabalho, pode-se afirmar que dispositivos do ramo *IoT* estão, de fato, alcançando fatias de mercado cada vez maiores. Seja pelo baixo preço final, alta facilidade de criação ou ambos combinados, tais equipamentos estão sendo desenvolvidos e ofertados aos montes ao usuário final, fomentando números e previsões de futuro surpreendentes.

Esta massificação da produção contribuiu para o estabelecimento de projetos cada vez mais efêmeros, que dão lugar a inúmeros sucessores em períodos de tempo curtos. Neste cenário, em consonância ao reduzido custo de fabricação, uma grande porção destes aparelhos acaba sendo lançada portando *firmwares* e sistemas operacionais de qualidade não tão alta, gerados às pressas, apenas para satisfazer o mercado.

Na prática, instâncias de tal fato puderam ser observadas no decorrer do capítulo 4 deste trabalho, onde exemplares de dispositivos desta categoria foram testados e revelaram falhas programáticas. Entre estas, desconsiderando as causadas por descuido – numerosas, porém irrelevantes para a análise, haja vista que não são exclusivas aos equipamentos em questão –, nota-se uma significativa porcentagem ocasionada – mesmo que, indiretamente – pela baixa capacidade de processamento do aparelho.

Isto é, por vezes, pôde-se notar a presença de uma aplicação não tão robusta nas unidades testadas, seja pela escolha de um programa acessório mais simplista ou pela opção por uma biblioteca de versão mais antiga, em busca de maior leveza na execução. Esta prática, no entanto, não é a melhor forma de lidar com as eventuais restrições de *hardware* de equipamentos *IoT*: em um ambiente que emprega manipulação de rede e Internet – tópicos de rápido e constante desenvolvimento –, não utilizar os recursos mais atualizados é assumir uma exposição a potenciais riscos.

No caso destes dispositivos, o emprego de componentes de software obsoletos, de fato, desempenha um papel considerável na geração de vulnerabilidades, o que, consequentemente, torna-os mais suscetíveis a sucumbir a ataques. Estes, por suas vezes, abrem brechas para possíveis comprometimentos de dados pessoais do usuário: falhas graves, exatamente como as evidenciadas em ambos os testes de invasão no decorrer deste trabalho.

Adicionalmente, pode-se delegar outra significativa parte da culpa pelas falhas à não encriptação dos dados intercambiados pela rede. Frequentemente, uma simples análise de pacotes utilizando um programa do tipo *sniffer* com acesso à rede do equipamento

alvo revelou, durante os testes realizados, dados sensíveis expostos em texto plano. De posse de informações desse tipo – tais como credenciais de acesso –, um agente mal intencionado consegue acesso ainda mais facilmente, o que foi corroborado durante as análises do capítulo quatro.

Esta ausência de codificação se deve, primeiramente, a falhas de projeto – que simplesmente optam por não considerar este aspecto da segurança – e, mais adiante, à falsa premissa – como clarificado por Martin [16] – de que a adição de tratativas para elevar a proteção (tais como o acréscimo de SSL/TLS ao tráfego de um servidor) impactaria a performance do equipamento, o que seria crítico e o tornaria inviável.

Contudo, as situações citadas no parágrafo anterior configuram equívocos de conceito. Um dispositivo não está fadado a ser inseguro apenas porque possui baixa capacidade de processamento: caso esta fosse a realidade, qualquer equipamento eletrônico sem múltiplos processadores ou alta quantidade de memória RAM seria uma ameaça. Logo, a questão é como se adaptam aplicações para estes casos a fim de que funcionem de maneira análoga no ambiente com recursos reduzidos.

Certamente, há aparelhos *IoT* com níveis de segurança largamente satisfatórios no extremo oposto deste cenário. Estes, todavia, são ofuscados pela grande quantidade de exemplares de baixa qualidade produzidos desde a recente popularização da área. Espera-se, entretanto, que, à medida em que a tecnologia em questão ganhe espaço, questionamentos sobre sua segurança – como o presente trabalho – se tornem mais frequentes e, consequentemente, impulsionem o mercado a seguir um padrão de qualidade mais adequado.

Neste cenário, faz sentido dar continuidade à realização dos processos mencionados logo acima. Para futuros trabalhos, por exemplo, englobar outros dispositivos – além dos selecionados nesta análise –, tais como os que empregam sistemas mais robustos (como o Android), certamente há de gerar testes com caminhos diferentes. Estes, por suas vezes, produziriam conteúdos igualmente valiosos, desta vez sobre novos aspectos do segmento estudado.

Por fim, entende-se que a democratização do entendimento sobre testes de invasão e *hacking* ético, como proporcionado por este trabalho, contribui para a realização destes procedimentos por parte de um maior público. Fato este que também colabora para a adequação do padrão de qualidade dos aparelhos em questão: objetivo final desejável para o segmento.

REFERÊNCIAS

[1] SHUKLA, H. Reasons How Internet Has Made Our Life Easier. *Woman's Era*, 16 Agosto 2021. Disponivel em: <https://womansera.com/reasons-how-internet-has-made-our-life-easier/>. Acesso em: 1 Abril 2022.

[2] HABIB, L. Computers and the Family: A Study of Technology in the Domestic Sphere, 2000.

[3] HOOIJDONK, R. V. The hidden dangers of IoT devices, 2019. Disponivel em: <https://internetofthingsagenda.techtarget.com/blog/IoT-Agenda/The-hidden-dangers-of-IoT-devices>. Acesso em: 16 agosto 2021.

[4] LAMPE, J. How to Test the Security of IoT Smart Devices, 2014. Disponivel em: <https://resources.infosecinstitute.com/topic/test-security-iot-smart-devices/>. Acesso em: 19 agosto 2021.

[5] PENETRATION testing (pentest): o que é e como funciona o teste de intrusão? *Backup Garantido*, 2020. Disponivel em: <https://backupgarantido.com.br/blog/penetration-testing/>. Acesso em: 28 Fevereiro 2022.

[6] INTERNET of Things. *Intermixit*, 2022. Disponivel em: <https://intermixit.com/internet-of-things/>. Acesso em: 25 Fevereiro 2022.

[7] MORE Than 30 Billion Devices Will Wirelessly Connect to the Internet of Everything in 2020. *Abi Research*, 2013. Disponivel em: <https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne/>. Acesso em: 8 setembro 2021.

[8] GREENSTEIN, S. The Aftermath of the Dyn DDOS Attack. *IEEE Micro*, v. 39, p. 66-68, Julho 2019. ISSN 10.1109/MM.2019.2919886.

[9] RAFFERTY, L. et al. Intelligent Multi-Agent Collaboration Model for Smart Home IoT Security. In: _____ *2018 IEEE International Congress on Internet of Things (ICIOT)*. [S.l.]: [s.n.], 2018. p. 65-71.

[10] VULNERABILIDADE na área de TI. *SManager*, 2021. Disponivel em: <https://smanager.com.br/vulnerabilidade-ti/>. Acesso em: 25 Fevereiro 2022.

[11] MESQUITA, M. O que é a ISO-27001 e o que ela agrega para sua empresa? *Triplait*, 2020. Disponivel em: <https://triplait.com/o-que-e-a-iso-27001/>. Acesso em: 26 agosto 2021.

[12] OWASP Internet of Things Project. *OWASP*, 2019. Disponivel em: <https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=Main>. Acesso em: 25 Fevereiro 2022.

[13] ETHICAL Hacking. *Portal GSTI*, 2021. Disponivel em: <https://www.portalgsti.com.br/ethical-hacking/sobre/>. Acesso em: 26 agosto 2021.

[14] ETHICAL hacking: entenda o conceito por trás do hacker do bem. *Blog Unyleya*, 2020. Disponivel em: <https://blog.unyleya.edu.br/bitbyte/ethical-hacking/>. Acesso em: 26 agosto 2021.

[15] DEFAULT TTL (Time To Live) Values of Different OS. *Subin's Blog*, 2014. Disponivel em: <https://subinsb.com/default-device-ttl-values/>. Acesso em: 01 Março 2022.

[16] MARTIN, L. TechBeacon. *Encryption won't affect app performance—if you do it right*. Disponivel em: <https://techbeacon.com/security/relax-good-encryption-practices-wont-affect-app-performance>. Acesso em: 11 Abril 2022.

[17] MORENO, D. *Introdução ao Pentest*. [S.l.]: Novatec, 2015.