Universidade Federal de Pernambuco

Josival dos Santos Silva

UP-Home: Uma Solução Adaptativa para Segurança de Casas Inteligentes com Suporte ao *Human-In-the-Loop*

Recife

2025

Josival dos Santos Silva

UP-Home: Uma Solução Adaptativa para Segurança de Casas Inteligentes com Suporte ao *Human-In-the-Loop*

Trabalho de Tese de Doutorado apresentado como requisito para obtenção do grau de Doutor em Ciência da Computação, área de concentração em Sistemas Distribuídos, do Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco.

Área de Concentração: Sistemas Distribuídos

Orientador: Nelson Souto Rosa

Coorientador: Fernando Antonio Aires Lins

Recife

2025

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Silva, Josival Dos Santos.
UP-Home: uma solução adaptativa para segurança de casas inteligentes com suporte ao Human-In-the-Loop / Josival Dos Santos Silva. - Recife, 2025.
133 f.: il.

Tese (Doutorado) - Universidade Federal de Pernambuco, Centro de Informática, Programa de Pós-Graduação em Ciências da Computação, 2025.
Orientação: Nelson Souto Rosa.
Coorientação: Fernando Antonio Aires Lins.
Inclui referências e apêndice.

1. IoT; 2. Smart home; 3. Segurança IoT; 4. Autoadaptação; 5. Atualização OTA; 6. Human-In-the-Loop. I. Rosa, Nelson Souto. II. Lins, Fernando Antonio Aires. III. Título.

UFPE-Biblioteca Central

Josival dos Santos Silva

“UP-Home: Uma Solução Adaptativa para Segurança de Casas Inteligentes com Suporte ao Human-In-the-Loop”

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Redes de Computadores e Sistemas Distribuídos.

Aprovada em: 27/02/2025.

_____

Orientador: Prof. Dr. Nelson Souto Rosa

BANCA EXAMINADORA

_____

Prof. Dr. Paulo Roberto Freire Cunha
Centro de Informática / UFPE

_____

Prof. Dr. Paulo Romero Martins Maciel
Centro de Informática / UFPE

_____

Prof. Dr. Ricardo Massa Ferreira Lima
Centro de Informática / UFPE

_____

Prof. Dr. Admilson de Ribamar Lima Ribeiro
Departamento de Computação / UFS

_____

Prof. Dr. Obionor de Oliveira Nóbrega
Departamento de Computação / UFRPE

Dedico esse trabalho ao Rei da Glória, Jesus Cristo. Maranata!

AGRADECIMENTOS

A Deus, Criador do Universo, por me conceder o dom da vida, iluminar meu caminho e permitir a concretização desta grande conquista.

À minha esposa, Mauriceia, pela paciência, pelo incentivo e por caminhar ao meu lado em cada etapa. Aos meus filhos, Jonathas, Esther e Davih, por serem minha maior inspiração e a razão mais profunda para a construção de um legado.

Aos meus pais, por toda a educação, e aos meus familiares, pelo apoio.

Ao meu orientador, Nelson Souto Rosa, pela orientação exemplar, pela paciência constante, pelos valiosos conselhos e, sobretudo, pelo incentivo decisivo que possibilitou a realização de um trabalho à altura de um doutorado.

Ao meu coorientador, Fernando Antonio Aires Lins, pela excelente coorientação, pela constante disposição em ajudar, pela atenção dedicada e pelo apoio fundamental ao longo desta jornada.

A todos os professores do CIn que contribuíram significativamente para esta etapa da minha vida, oferecendo constante incentivo à pesquisa e ao meu desenvolvimento acadêmico.

À Universidade Federal de Pernambuco, pela infraestrutura para a realização do trabalho.

Aos colegas que me acompanham nesta jornada.

E a todos que contribuíram comigo, direta ou indiretamente.

"O Senhor dá a sabedoria; da sua boca é que vem o conhecimento e o entendimento."

Provérbios 2.6

RESUMO

Os dispositivos das *smart homes* são propensos a ataques que podem comprometer a segurança dos seus usuários. Ao mesmo tempo, a não padronização dos meios de atualização destes dispositivos faz da *smart home* um ambiente potencialmente vulnerável. Não menos importante, há o fato de que muitos fabricantes de IoT lançam seus produtos e, em seguida, os abandonam, deixando de oferecer suporte a atualizações que poderiam evitar ou mitigar novos ataques. Além disso, as atualizações geralmente dependem inteiramente de ações executadas pelos usuários. Existem várias propostas de soluções que promovem segurança em *smart homes*. No entanto, observa-se a ausência de trabalhos que envolvam o desenvolvimento de soluções de segurança adaptativas, ou seja, soluções que garantam a segurança das *smart homes* com pouca ou nenhuma intervenção dos usuários. Neste contexto, esta tese de doutorado apresenta UP-Home, uma ferramenta adaptativa que gerencia a segurança da pilha de software de dispositivos em *smart homes*. Ao mesmo tempo, em atualizações de segurança críticas, UP-Home auxilia o humano sobre ações a serem tomadas, o chamado *Human-In-the-Loop* (HIL). Assim, técnicas de computação adaptativa são utilizadas para garantir a autoproteção dos dispositivos domésticos com possibilidade de assistência humana. A combinação da abordagem *Human-In-the-Loop* e computação adaptativa fortalece a segurança da *smart home*, oferecendo uma defesa que se adapta a vulnerabilidades em constante evolução. Ao mesmo tempo, a ferramenta proposta busca garantir que os dispositivos da *smart home* atendam a requisitos de segurança estabelecidos por padrões utilizados pela indústria. Os resultados dos experimentos em um ambiente real mostram a capacidade da UP-Home em mitigar vulnerabilidades em diferentes níveis de criticidade. Desta forma, a ferramenta mostrou-se efetiva em melhorar a autoproteção de dispositivos, de seus usuários e, por consequência, de toda a *smart home*.

Palavras-chaves: IoT. *Smart home*. Segurança IoT. Autoadaptação. Atualização OTA. *Human-In-the-Loop*.

ABSTRACT

Home devices are prone to attacks that can compromise the security of their users. At the same time, the lack of standardization in updating these devices makes the home a potentially vulnerable environment. No less important is the fact that many IoT manufacturers launch their products and then abandon them, failing to provide support for updates that could prevent or mitigate new attacks. In addition, updates usually depend entirely on actions performed by users. There are several proposals for solutions that promote security in homes. However, there is a lack of work involving the development of adaptive security solutions, that is, solutions that can guarantee the security of homes with little or no user intervention. In this context, this doctoral thesis presents UP-Home, an adaptive solution that manages the security of the software stack of home devices. At the same time, in critical security updates, UP-Home assists the human with the actions to be taken, the so-called *Human-In-the-Loop* (HIL). Thus, adaptive computing techniques ensure the self-protection of human-assisted home devices. The combination of the adaptive computing approach strengthens the home's security by offering an adaptive defence against constantly evolving vulnerabilities. At the same time, the proposed solution seeks to ensure that the home devices meet the security requirements established by industry standards. The results of the experiments in a natural environment demonstrate the ability of the home to mitigate vulnerabilities at different levels of criticality. Thus, the solution improved the self-protection of devices, their users and, consequently, the entire home.

Keywords: IoT. *Smart home*. IoT Security. Self-Adaptation. OTA Update. Human-In-the-Loop.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADB	*Android Debug Bridge*
API	*Application Programming Interface*
AVD	*Android Virtual Device*
BNF	*Backus-Naur Form*
CIA	*Confidentiality, Integrity and Availability*
CLI	*Command-Line Interface*
COAP	*Constrained Application Protocol*
CPS	*Cyber-Physical Systems*
CVE	*Common Vulnerabilities and Exposures*
CVSS	*Common Vulnerability Scoring System*
DDoS	*Distributed Denial-of-Service*
DTLS	*Datagram Transport Layer Security*
ECU	*Electronic Control Unit*
ETSI	*European Telecommunications Standards Institute*
FGSM	*Fast Gradient Sign Method*
FTP	*File Transfer Protocol*
GMP	*Greenbone Management Protocol*
GVM	*Greenbone Vulnerability Management*
HA	*Home-Assistant*
HIL	*Human-In-the-Loop*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IDS	*Intrusion Detection System*
IETF	*Internet Engineering Task Force*

IoT	*Internet of Things*
IoTSF	*IoT Security Foundation*
IRS	*Intrusion Response System*
LWM2M	*Lightweight Machine to Machine*
LWMesh	*Lightweight Mesh*
MAPE-K	*Monitor, Analyze, Plan, Execute - Knowledge*
MDP	*Markov Decision Process*
MLPLW	*Multi-layer Perceptron with Limited Weights*
MQTT	*Message Queuing Telemetry Transport*
NAT	*Network Address Translation*
NVD	*National Vulnerability Database*
OEMs	*Original Equipment Manufacturers*
OTA	*Over The Air*
OWASP	*Open Web Application Security Project*
P2P	*Peer-to-Peer*
PGD	*Projected Gradient Descent*
PROM	*Programmable Read-Only Memory*
ROM	*Read-Only Memory*
SARM	*Security Adaptation Reference Monitor*
SO	*Sistema Operacional*
SSH	*Secure Shell*
SUIT	*Software Updates for Internet of Things*
Telnet	*Teletype Network*
TI	*Tecnologia da Informação*
TIC	*Tecnologia da Informação e Comunicação*
VPN	*Virtual Private Network*
XML	*Extensible Markup Language*

XSS *Cross-Site Scripting*

YAML *YAML Ain't Markup Language*

ZASH *Zero-Aware Smart Home*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta uma visão geral desta tese de doutorado. O capítulo apresenta inicialmente o contexto, a motivação e os argumentos que justificam o desenvolvimento do trabalho. Em seguida, discute o problema de pesquisa, lista os objetivos, contribuição e limitações. Por fim, é apresentada a estruturação do restante do documento.

1.1 CONTEXTO

O termo *smart* é um conceito aplicado a tudo aquilo que utiliza tecnologia com alguma inteligência. Desse modo, as residências têm recebido de forma abrangente o termo *smart* (MARIKYAN; PAPAGIANNIDIS; ALAMANOS, 2019). Uma *smart home* é uma residência com dispositivos de *Internet of Things* (IoT) conectados a diferentes nuvens e aplicações de automação residencial (STOLOJESCU-CRISAN; CRISAN; BUTUNOI, 2021) (SIVARAMAN et al., 2015). Essa automação acontece em diferentes tarefas e é feita por meio das mais diversas aplicações de IoT, capacitando objetos inanimados a atuarem com pouca ou nenhuma intervenção humana (HASSIJA et al., 2019).

Controlar objetos eletrônicos (*coisas*) de forma remota é uma tarefa bem conhecida desde o início dos anos 1990. O termo *coisas* pode ser entendido como tudo ao nosso redor, desde um pequeno relógio a uma grande espaçonave. IoT, entre outras coisas, possibilitou que os usuários gerenciem seus equipamentos eletrônicos utilizando a Internet. Dessa forma, as *smart homes* contam com dezenas desses dispositivos, proporcionando acessibilidade e comodidade. Tais benefícios acontecem de forma automatizada, como nunca visto antes em residências, resultando no aumento do interesse da indústria e da academia nos últimos anos. Inclusive, a expansão do conhecimento sobre a aceitação e adoção de tecnologias da *smart home* tem motivado um grande número de pesquisas. Esses estudos investigam as percepções e atitudes dos usuários em relação a essa tecnologia, fornecendo uma visão avançada sobre como as pessoas estão adotando e aceitando a IoT em suas *smart homes* (FERREIRA; OLIVEIRA; NEVES, 2023) (HUSSIN; ABDOLLAH; AHMAD, 2023).

As *smart homes* modernas têm integrado uma variedade de dispositivos inteligentes disponíveis no mercado, os quais são capazes de gerar e trocar dados de forma contínua

e com mínima intervenção humana. Esse processo transforma a residência, que antes era apenas um local de moradia, em um ecossistema de dispositivos inteligentes, monitorando e controlando diferentes aspectos da vida cotidiana. Entre as funcionalidades oferecidas, destaca-se o acompanhamento remoto das atividades dos filhos, a irrigação automática do jardim, a alimentação de animais domésticos e até mesmo a vigilância da casa.

Com essa evolução, uma *smart home* pode incorporar tecnologias de automação avançadas, proporcionando aos seus habitantes um monitoramento e controle aprimorados das funções do ambiente (SCHIEFER, 2015).

1.2 MOTIVAÇÃO

Com a evolução no desenvolvimento de dispositivos inteligentes em ambientes domésticos, a casa que antes possuía pouca ou nenhuma atração tecnológica, passou a ter dispositivos inteligentes. Uma *smart home* pode ter dezenas de dispositivos IoT, que se conectam à Internet, utilizando diferentes nuvens e aplicações de automação residencial. Isto proporciona acessibilidade e conveniência nunca vistas antes nas residências.

A *smart home* é um domínio de aplicação IoT significativamente promissor, pois nela estão presentes os mais variados tipos de eletrodomésticos, câmeras de vigilância, dispositivos de sensores médicos e termostatos. Na *smart home* existem várias possibilidades de coleta de dados, e os usuários podem controlar remotamente os dispositivos que proporcionam aos residentes mais comodidade e conforto (LIU; ZHANG; FANG, 2018).

No entanto, diversos riscos de segurança surgem no ambiente de *smart homes* (YAR et al., 2021). Embora as *smart homes* ofereçam benefícios significativos, os riscos associados à segurança da informação podem, em alguns casos, superar as vantagens da conectividade. Primeiramente, o ambiente doméstico apresenta vulnerabilidades de segurança que podem resultar no vazamento de dados confidenciais para agentes maliciosos (HASSIJA et al., 2019) (ZHOU et al., 2019). Em segundo lugar, a falta de um gerenciamento das atualizações de software dos dispositivos (*e.g.,* a manutenção de pilhas de software desatualizadas e a obsolescência de aplicativos) acarreta vulnerabilidades críticas de segurança (GU et al., 2020). Essas condições podem permitir que informações trafeguem por redes desprotegidas, expondo os sistemas a ameaças específicas, como ataques de interceptação, exploração de falhas conhecidas e invasões remotas (LIN, 2019). Por fim, o abandono de dispositivos por parte dos fabricantes, sem o desenvolvimento de atualizações contínuas, pode deixar com-

ponentes vulneráveis sem correção, comprometendo não apenas os aparelhos individuais, mas toda a rede, além de expor os usuários a riscos de segurança significativos.

Abordagens convencionais, desenvolvidas manualmente e aplicadas de forma estática, têm se mostrado inadequadas para acompanhar a evolução constante das vulnerabilidades em segurança cibernética. Essa limitação impulsiona a criação de soluções de autoproteção capazes de identificar vulnerabilidades e mitigá-las por meio de técnicas de adaptação, visando o tratamento e a gestão de problemas ou exceções do software em tempo de execução (YUAN et al., 2013) (DEHRAJ; SHARMA, 2021).

Embora a interação humana não seja uma prática comum em sistemas autônomos, ela pode se tornar essencial para garantir que os requisitos de segurança de *smart homes* sejam atendidos e para apoiar a tomada de decisões em situações críticas (HARA; OKADA; OTA, 2023) (DOMASZEWICZ et al., 2016).

Tem sido discutida a importância da integração entre automação e intervenção humana em sistemas de *smart homes*, com ênfase na segurança e adaptabilidade. Domaszewicz et al. (2016) propõe o conceito de *soft actuation*, que emite sugestões sutis para o usuário realizar ações manuais, superando as limitações de decisões automatizadas e garantindo maior alinhamento com as preferências do usuário. Essa ideia se complementa com a abordagem de Pasquale et al. (2023), que utiliza o *Monitor, Analyze, Plan, Execute - Knowledge* (MAPE-K) para promover uma segurança adaptativa sustentável, combinando automação com intervenção humana para enfrentar novas e inesperadas vulnerabilidades, destacando a importância da participação humana em processos críticos de análise e adaptação. Essa participação humana é aprofundada em Arcaini, Mirandola e Riccobene (2020), que apresenta uma plataforma de automação com controle descentralizado baseada no MAPE-K, permitindo ajustes em tempo real e decisões colaborativas. Juntos, esses estudos destacam a necessidade de soluções híbridas, onde a automação seja complementada pela intervenção humana, garantindo sistemas mais seguros, responsivos e alinhados às necessidades dos usuários.

Dado que o ambiente de IoT é dinâmico e em constante expansão, os componentes de software que operam nesses dispositivos enfrentam mudanças frequentes, como atualizações e correções de *bugs*. Assim, é necessária a implementação de adaptações autônomas, com monitoramento e controle contínuos. Além disso, a falta de atualizações de software, que representa uma importante vulnerabilidade de segurança, deve ser tratada de forma dinâmica, ou seja, em tempo de execução e sem a interrupção total do funcionamento dos

dispositivos.

1.3 JUSTIFICATIVA

Smart homes geralmente carecem de um gerenciamento de segurança adequado, o que pode fazer com que os riscos associados ao uso de dispositivos IoT superem seus benefícios. Conforme apontado por Gu et al. (2020), as ameaças críticas de segurança frequentemente se originam nas plataformas de gerenciamento IoT e em seus aplicativos associados.

Com o crescimento exponencial dos dispositivos IoT, surge a necessidade de implementar um modelo de segurança que inclua uma abordagem de supervisão humana. O conceito conhecido como *Human-In-the-Loop* (HIL) reconhece a importância da interação humana para garantir decisões informadas e ações adaptativas em momentos críticos (NUNES; ZHANG; SILVA, 2015). A introdução do HIL permite que usuários e administradores interfiram diretamente nos dispositivos, monitorando e ajustando componentes e parâmetros, especialmente em cenários que exigem uma avaliação mais contextual e flexível. Essa abordagem adiciona uma camada vital de segurança, pois as decisões não se baseiam apenas em processos automatizados, mas também contam com a supervisão humana, crucial em ambientes onde as ameaças podem surgir de forma imprevisível.

As soluções para atualização da pilha de software dos dispositivos IoT adotam como base o método *Over The Air* (OTA) (BAUWENS et al., 2020), pois viabilizam atualizações através da rede. Estas soluções permitem a atualização do *firmware* (CHANDRA et al., 2016), podem utilizar *blockchain* para melhorar o processo de atualização (LEE; LEE, 2017), usar criptografia (FRISCH; REISSMANN; PAPE, 2017) ou ainda focar na integridade das atualizações (ZANDBERG et al., 2019). Por fim, as atualizações podem até ser programadas para ocorrer em horários pré-estabelecidos (LIN; BERGMANN, 2016).

Atualmente, as atualizações de software são feitas geralmente através de redes de comunicação. Um exemplo dessas atualizações ocorre com os *smartphones*, cujos sistemas operacionais e aplicações são atualizados várias vezes por ano. Neste caso, os usuários não precisam de ajuda externa, *e.g.,* ajuda de assistência técnica. As atualizações IoT possibilitam que seus desenvolvedores implementem novos *patches*, software ou *firmware* para os dispositivos em sua rede sem que seja necessário interromper a execução desses dispositivos. Quando as vulnerabilidades de uma máquina são descobertas, os fornecedores de software emitem um *patch*, que os usuários finais devem obter e instalar (TIINSIDE,

2021). As atualizações podem ser feitas, quando possível, em momentos em que os dispositivos não estejam em uso. As atualizações OTA podem ajudar seus desenvolvedores a manterem seguras suas aplicações em todo o ciclo de vida do dispositivo (DORNERWORKS, 2020). No entanto, um dos problemas de atualizações OTA é quando o cronograma de atualizações é longo o suficiente para causar obsolescência.

Kolehmainen (2018) destaca a falta de soluções de segurança adequadas para lidar com o aumento exponencial de dispositivos IoT, muitos dos quais são vulneráveis e carecem de proteção eficiente. Embora existam iniciativas para estabelecer padrões de segurança para IoT, como o *Software Updates for Internet of Things* (SUIT)[1] do *Internet Engineering Task Force* (IETF) e *IoT Security Foundation* (IoTSF)[2], muitos fabricantes negligenciam a manutenção de seus dispositivos e deixam de fornecer atualizações de segurança ao longo do ciclo de vida do produto, como evidenciado por estudos de (MORAN et al., 2019) e (KAFLE; FUKUSHIMA; HARAI, 2016).

Ao mesmo tempo que se observa a existência de problemas de segurança em *smart homes*, a capacidade de autoproteção é considerada uma das características essenciais dos sistemas computacionais modernos (ABDULLA et al., 2020). Os sistemas autoadaptativos, amplamente explorados na literatura, baseiam-se em quatro propriedades principais: autoconfiguração, autocura, auto-otimização e autoproteção (SALEHIE; TAHVILDARI, 2009). Dentre essas, a autoproteção é especialmente relevante para ambientes de IoT, ao permitir que os sistemas detectem violações de segurança e se recuperem automaticamente de seus efeitos (KEPHART; CHESS, 2003), promovendo maior resiliência em cenários onde os dispositivos estão continuamente expostos a ameaças.

Embora trabalhos recentes tenham avançado na segurança de sistemas IoT, desafios significativos permanecem (ALMUSAED; YITMEN; ALMSSAD, 2023), (PHILIP; LUU; CARTE, 2023). Algumas soluções, como as baseadas em *blockchain* para validação de atualizações, utilizam intensivamente os recursos computacionais, (*e.g.,* processador, armazenamento, energia) tornando sua aplicação inviável em dispositivos IoT com capacidade limitada (LEE; LEE, 2017) (SEDGEWICK; LEMOS, 2018). Outras, como as propostas para microcontroladores, são restritas a dispositivos específicos (*e.g.,* ESP8266) e não oferecem escalabilidade para ambientes mais complexos (FRISCH; REISSMANN; PAPE, 2017). Além disso, abordagens como a de Lin e Bergmann (2016) utilizam agendamentos periódicos de atu-

[1] https://www.ietf.org/blog/ietf105-iot-wrapup/
[2] https://iotsecurityfoundation.org/

alização, podendo não ser suficientes para lidar com vulnerabilidades críticas que surgem inesperadamente. Em um contexto onde as ameaças evoluem constantemente, a resposta rápida e contínua a essas vulnerabilidades é essencial para garantir a segurança da *smart home*.

Diante do que foi apresentado, o foco desta tese é no domínio de *smart homes* dentro da IoT. O desafio de ter os dispositivos IoT com segurança mesmo sob as constantes mudanças no ambiente da *smart home* motivou a escolha deste domínio de aplicação. A propriedade de autoproteção foi escolhida por conta da necessidade de executar ações de segurança no dispositivo IoT de forma dinâmica. Desta maneira, busca-se garantir que a segurança do dispositivo esteja em conformidade com padrões existentes no momento das atualizações. As medidas de segurança serão aplicadas através de ações de adaptação para buscar melhorar o nível de segurança da pilha de software presente nos dispositivos das *smart homes*.

1.4 PROBLEMA E QUESTÕES DE PESQUISA

O problema principal tratado nesta tese é **como manter a segurança de dispositivos IoT em *smart homes*, garantindo que atualizações contínuas na pilha de software atendam aos padrões de segurança da indústria e, ao mesmo tempo, incorporando a supervisão e intervenção humana para aumentar a confiança nas atualizações?**

- **Hipótese:** Considerando a dinâmica da *smart home*, a falta de gerenciamento de atualizações de segurança e a importância da interação humana, faz-se necessário a implementação de um sistema de segurança autônomo com suporte ao HIL que possa aplicar autoproteção. Esta medida pode reduzir significativamente os riscos de segurança associados aos dispositivos IoT em *smart homes*. Assim, com a diminuição de riscos associados à vulnerabilidades, falhas de gerenciamento de atualizações e potenciais ameaças cibernéticas, e com a inclusão da supervisão humana, a segurança da *smart home* pode ser melhorada.

Depois de estabelecer essa hipótese, duas questões de pesquisa serão respondidas no decorrer do trabalho:

- **Questão 1:** Como a estratégia de autoproteção, associada à atualização da pilha de software dos dispositivos IoT e suportada por intervenções humanas (*Human-In-the-Loop*), pode melhorar a segurança da *smart home*?
- **Questão 2:** Detectar e mitigar vulnerabilidades da casa com a ajuda de um sistema que integra o HIL pode promover estratégias proativas de melhorias da segurança?

1.5 OBJETIVOS

O principal objetivo desta tese de doutorado é conceber, projetar e implementar uma ferramenta de segurança para *smart homes* que utiliza conceitos de autoproteção (*self-protecting*) da computação autonômica (KEPHART; CHESS, 2003) (YUAN; ESFAHANI; MALEK, 2014) para garantir atualização contínua dos dispositivos. Além disso, será implementado o conceito de HIL para permitir supervisão e intervenção humanas em situações de alta criticidade, garantindo maior adaptabilidade e resiliência no sistema.

Para alcançar este objetivo principal, os seguintes objetivos específicos precisam ser atingidos:

1. Estabelecer um conjunto de *metas* (*goals*) de alto nível que precisam ser atendidas pelo dispositivo para melhorar a sua segurança. Estas metas precisarão também ser expressas e pré-configuradas na ferramenta proposta de forma que possam ser processadas computacionalmente;
2. Definir estratégias de *monitoramento* de parâmetros de segurança que forneçam uma visão atualizada sobre a segurança dos dispositivos presentes na *smart home*;
3. Projetar um mecanismo de *análise* que permita identificar eventuais violações das metas de segurança. Este mecanismo poderá tratar de violações que já aconteceram (*reativa)* ou que poderão acontecer (*proativa*);
4. Identificar um conjunto de ações que precisam ser executadas quando as violações forem identificadas. Estas ações farão parte de um *plano* de ação e precisarão ser *executadas* de forma coordenada em todos os dispositivos que tenham violado as metas de segurança;

5. Projetar, implementar e avaliar um protótipo computacional que possa ser implantado em uma *smart home*. O protótipo deve realizar as etapas da computação autonômica definidas nos itens anteriores, mais especificamente o monitoramento, análise com suporte ao HIL, planejamento e execução de ações de autoproteção.

1.6 CONTRIBUIÇÕES

Esta tese apresenta a concepção, projeto e implementação de uma ferramenta adaptativa para atualização de componentes e parâmetros dos dispositivos da *smart home*. A ferramenta proposta promove a atualização dinâmica dos dispositivos com suporte ao HIL para mitigar as vulnerabilidades de segurança das *smart homes*. UP-Home foi projetada seguindo o MAPE-K (IBM, 2005). Na prática, o UP-Home monitora continuamente a pilha de software dos dispositivos (e.g., aplicativos, sistema operacional e *firmware*) e atualiza essa pilha para que atenda aos padrões de segurança definidos pela indústria.

Para viabilizar essa abordagem, foram desenvolvidas e integradas diversas contribuições, incluindo:

- Identificação de parâmetros e metas de segurança importantes no contexto de *smart homes*. Enquanto parâmetros de segurança atuais são definidos de forma genérica para IoT, as necessidades de segurança de uma *smart home* exigem um refinamento destes parâmetros. Ao mesmo tempo, as metas de alto nível precisam ser expressas de forma que possam ser processadas computacionalmente pela ferramenta proposta.

- O projeto e implementação de estratégias de monitoramento de segurança que possam ser implantados em ambientes de *smart homes*. Neste ponto, estas estratégias devem considerar os parâmetros de segurança que podem ser efetivamente monitorados e transformá-los em métricas que possam ser usadas para identificar violações de segurança.

- Definição de uma estratégia de análise que permita definir, com precisão, a ocorrência de violações de segurança em dispositivos da *smart home*. Este é um componente importante da ferramenta proposta, pois ele definirá se intervenções precisam ser feitas ou não nos dispositivos da *smart home*.

- Identificação de um conjunto de ações relacionadas à segurança que precisam ser executadas nos dispositivos da *smart home*. Essas ações podem variar conforme o dispositivo e a aplicação, uma vez que a latência de resposta pode diferir de usuário para usuário em um dispositivo ou conjunto deles. Para isso, será necessário definir uma estratégia de coordenação das ações.

- A criação de uma notação (ou linguagem) para expressar os planos de atualização dos dispositivos.

- Identificação e composição de mecanismos de execução de atualizações de dispositivos de *smart homes*. Por exemplo, estes mecanismos precisarão garantir que as atualizações sejam realizadas em todos os dispositivos que estejam com violações de segurança.

- Realização de uma avaliação prática da ferramenta proposta em uma *smart home*. Essa avaliação visa analisar a eficácia da ferramenta proposta. Os resultados obtidos a partir do monitoramento contínuo da *smart home* serão utilizados para adaptar dinamicamente a pilha de software presente nos dispositivos, proporcionando um ambiente mais seguro. Nesse contexto, será verificado o aprimoramento do nível de segurança, medido pela redução das vulnerabilidades identificadas ao longo do tempo.

1.7 ESCOPO NEGATIVO

Está fora do escopo desta tese investigar aspectos de segurança associados a dispositivos da *smart home* que não permitam o monitoramento de parâmetros de segurança, como, *e.g.,* a versão do *firmware*. Essa limitação ocorre nos casos em que os dispositivos não oferecem suporte para o monitoramento de parâmetros ou não possibilitam a inicialização da atualização do *firmware* por meio do respectivo aplicativo.

Durante a avaliação, o foco não será direcionado ao tempo de execução no qual os dispositivos iniciam e concluem a atualização, mas sim ao tempo de detecção e ação para mitigar vulnerabilidades, especialmente em dispositivos desatualizados.

Com relação às propriedades de autoadaptação, como autocura, auto-otimização, autoconfiguração, é importante destacar que esses aspectos não serão abordados nesta tese

de doutorado. A pesquisa concentra-se na segurança proporcionada pela UP-Home por meio de adaptações dinâmicas em tempo de execução, com ênfase nas adaptações que ocorrem de forma reativa, e que se antecipam (proativa) na mitigação de vulnerabilidades em dispositivos da *smart home*.

1.8 ORGANIZAÇÃO E ESTRUTURA DA TESE

Os capítulos restantes deste documento estão organizados da seguinte forma:

- **Capítulo 2** apresenta o referencial teórico da tese, com as definições de IoT, segurança em IoT, sistemas autoadaptativos e atualizações OTA.
- **Capítulo 3** apresenta a revisão bibliográfica relacionada à área de pesquisa sobre sistemas autoadaptativos, mostrando as principais abordagens e técnicas utilizadas.
- **Capítulo 4** descreve a ferramenta proposta, levando em consideração os princípios de autoproteção, tipos de adaptação, 5W-1H, MAPE-K, uso de padrões de segurança e HIL.
- **Capítulo 5** apresenta diversos experimentos usados para realizar a avaliação experimental da ferramenta proposta nesta tese.
- **Capítulo 6** sumariza as contribuições, limitações e trabalhos futuros desta tese.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo introduz os conceitos necessários ao entendimento desta tese. Primeiramente, são introduzidos os princípios relacionados à IoT (Seção 2.1). Em seguida, são destacados os aspectos gerais de segurança IoT (Seção 2.2). As criticidades das vulnerabilidades IoT são apresentadas na Seção 2.3, enquanto a Seção 2.4 explora os principais padrões de segurança nesse domínio. Posteriormente, são apresentados os conceitos essenciais sobre sistemas autoadaptativos (Seção 2.5), seguidos de uma discussão sobre as atualizações remotas (Seção 2.6). Este capítulo conclui com uma análise sobre a integração do ser humano no processo de adaptação (Seção 2.7).

2.1 IOT

IoT é uma rede aberta de objetos inteligentes que podem se auto-organizar, compartilhar informações, dados e recursos, que respondem e se adaptam às mudanças ambientais (MADAKAM et al., 2015). A IoT consiste na onipresença de vários objetos ou coisas, incluindo tecnologias de sensores, atuadores e dispositivos móveis. Esses dispositivos interagem entre si para atingir objetivos compartilhados usando redes com e sem fio (TAN; WANG, 2010). Com base nisto, é possível alcançar uma maior integração com o mundo real através da IoT, eliminando a necessidade de intervenção humana. A IoT ajuda a transformar objetos tradicionais, como luzes, fechaduras e televisões, em objetos inteligentes, por estarem conectados à Internet.

O crescimento significativo do uso de objetos físicos conectados à Internet está transformando o cenário tecnológico de forma sem precedentes, consolidando a visão da IoT. A IoT possibilita que objetos físicos adquiram a capacidade de perceber o ambiente à sua volta, assimilando informações de maneira que imita a percepção humana, permitindo-lhes *ver*, *ouvir* e *pensar* sobre os eventos que ocorrem no entorno onde estão inseridos. Além disso, esses objetos têm a capacidade de estabelecer comunicações entre si, compartilhando informações para coordenar e executar tarefas de maneira eficiente e autônoma, revolucionando o modo como interagem com o ambiente e entre si (ALMUSAED; YITMEN; ALMSSAD, 2023).

2.1.1 Pilha de Software

Dispositivos de IoT frequentemente operam sob severas restrições de recursos computacionais, incluindo consumo de energia, capacidade de processamento da CPU e disponibilidade de memória (ZIKRIA et al., 2018). Essas limitações intrínsecas exigem o desenvolvimento de abordagens cuidadosamente planejadas para a implementação de softwares voltados à IoT. Nesse contexto, cada dispositivo precisa otimizar o uso de seus recursos limitados, garantindo desempenho e funcionalidade compatíveis com as demandas do ambiente em que está inserido.

A Figura 2.1 apresenta a pilha de software em que diferentes camadas são utilizadas para atender às necessidades de diversos dispositivos IoT.

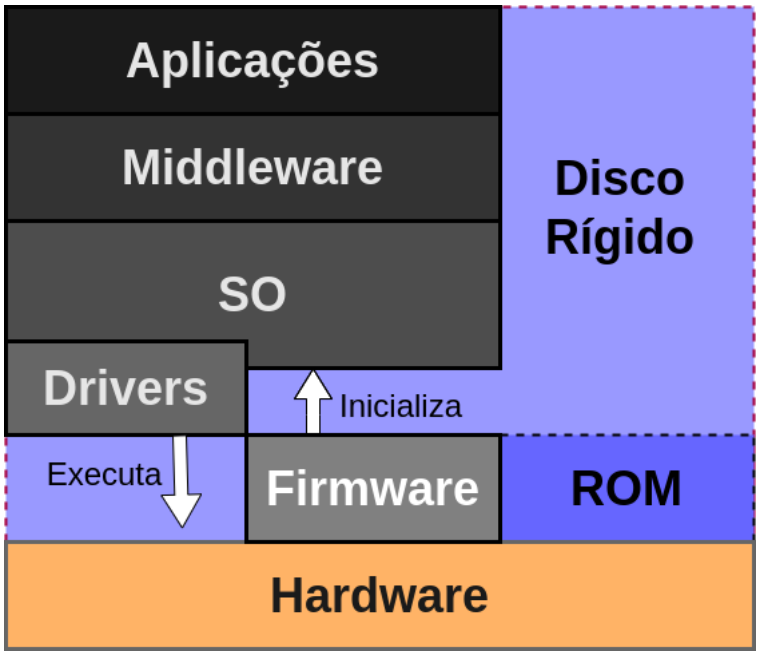

Figura 2.1 – Pilha de software

Fonte: PPLWARE (2020)

Os componentes desta pilha são descritos da seguinte forma:

- **Aplicações**. As aplicações operam sobre o *Sistema Operacional* (SO) e compõem a interface gráfica, deixando transparente para o usuário o envio e recebimento de comandos para os dispositivos. As aplicações podem ser desinstaladas e novamente instaladas, caso necessário.

- **Sistema Operacional (SO)**. O SO atua como intermediário entre o usuário e o hardware, simplificando a interação com os recursos computacionais e possibilitando

a execução de uma ou mais aplicações simultaneamente. Dessa forma, o SO gerencia e aloca os recursos de hardware para as aplicações, controlando sua execução e a operação dos dispositivos (SILBERSCHATZ; GALVIN; GAGNE, 2006). Além disso, este gerenciamento do SO é ainda mais importante em sistemas IoT, dada a restrição de recursos computacionais dos dispositivos IoT.

- ***Firmware***. O *firmware* é o software que geralmente reside em memórias não voláteis, como a *Read-Only Memory* (ROM). Quando instalado em uma ROM, o *firmware* não pode ser reprogramado, diferentemente de versões instaladas em memórias reprogramáveis, como a *Programmable Read-Only Memory* (PROM) (TAN et al., 2017). O *firmware*, que é um arquivo binário, pode conter o software integral de um dispositivo ou apenas uma parte dele. Em dispositivos com múltiplos microcontroladores, a imagem do *firmware* pode ser composta por várias subimagens, por exemplo, se o dispositivo possuir mais de um microcontrolador (MORAN; MERIAC; TSCHOFENIG, 2018).

Na base dessa pilha de software, encontram-se dispositivos com recursos de armazenamento extremamente limitados, nos quais a prioridade é a eficiência energética e a minimização do uso de memória. Conforme avança-se na pilha, as camadas de software se ajustam para suportar dispositivos com maior capacidade de processamento e memória, aproveitando o armazenamento local ou em nuvem, conforme necessário. Essa abordagem flexível permite que dispositivos IoT alcancem um equilíbrio entre a funcionalidade desejada e as restrições de recursos.

2.1.2 Smart Home

O conceito de casa inteligente, do inglês *smart home*, refere-se às residências que se tornaram realidade nas últimas décadas com o crescimento da IoT em diversos domínios. Darby (2018) apresenta duas definições principais de smart home (DARBY, 2018). A primeira definição foca na perspectiva do usuário e da própria casa, concebendo a *smart home* como uma residência automatizada, onde eletrodomésticos e sistemas internos são integrados para proporcionar uma melhor experiência tecnológica. Essa abordagem destaca valores como modernidade, conveniência e eficiência na gestão das tarefas cotidianas da vida doméstica. Nesse contexto, a casa se transforma em um espaço onde a tecnologia

está profundamente entrelaçada com o ambiente residencial, facilitando atividades diárias, melhorando a segurança e aumentando o conforto dos moradores.

A segunda definição adota uma perspectiva mais abrangente, centrada nos sistemas e na eficiência energética dos edifícios. Nessa visão, *Tecnologia da Informação e Comunicação* (TIC) e sistemas inteligentes de monitoramento e controle são implementados para gerenciar de forma eficiente aspectos como iluminação, aquecimento e refrigeração, além de permitir a geração de energia a partir de fontes renováveis. Essa abordagem visa otimizar o uso de recursos e promover práticas sustentáveis na operação das residências.

Em resumo, o conceito de *smart home* abrange tanto a perspectiva do usuário, focada em modernidade e conveniência, quanto uma visão sistêmica, voltada para a eficiência energética e sustentabilidade. O uso de sistemas baseados em IoT possibilita transformar residências em ambientes altamente eficientes, conectados e adaptáveis às necessidades de seus habitantes.

2.1.3 Classes de dispositivos

A IoTSF classifica os dispositivos IoT em diferentes classes, com base no nível de controle e conectividade em relação a um concentrador de integrações ou conexões (IoTSF, 2018). Essa categorização ajuda a definir as capacidades de gerenciamento e as limitações de segurança associadas a cada tipo de dispositivo:

- **Classe 1**. Dispositivos totalmente controlados e conectados. Nessa classe, o concentrador possui interfaces que permitem o controle completo do dispositivo IoT, bem como a coleta e o gerenciamento de dados. Isso inclui a capacidade de enviar comandos, monitorar o status do dispositivo em tempo real e realizar atualizações de software de forma remota.

- **Classe 2**. Dispositivos parcialmente controlados e/ou conectados. O concentrador pode realizar algumas tarefas de gerenciamento, como enviar atualizações de software e monitorar o tráfego de dados, mas existem limitações em termos de controle total das funcionalidades do dispositivo. Esses dispositivos têm uma conectividade intermediária, oferecendo controle parcial e opções limitadas de gestão de dados.

- **Classe 3**. Dispositivos com o nível mais básico de integração. Nesta classe, o con-

centrador não possui controle nem capacidade de gerenciar funções essenciais do dispositivo IoT, como atualizações de software ou coleta de dados. Esses dispositivos operam de forma independente e têm uma conectividade mínima, o que pode limitar as opções de monitoramento e controle centralizados.

Essa classificação é importante para compreender e gerenciar a diversidade de dispositivos IoT em ambientes residenciais. Com isso, desenvolvedores e profissionais de segurança podem considerar as características e limitações de cada categoria ao planejar e implementar sistemas IoT, promovendo um nível adequado de controle e segurança para os diferentes tipos de dispositivos.

2.2 SEGURANÇA EM IOT

Segundo Nieles, Dempsey e Pillitteri (2017), a segurança da informação é responsável por proteger tanto as informações quanto os sistemas que as processam. Essa proteção busca evitar o acesso não autorizado, o uso indevido, a divulgação indevida, a interrupção, a modificação ou a destruição das informações, garantindo assim os princípios de confidencialidade, integridade e disponibilidade.

Este conceito torna-se ainda mais importante em um ambiente IoT, enquanto novos problemas de segurança surgem diariamente. Um dos principais motivos que aumentam a complexidade da melhoria da segurança dos sistemas IoT é a heterogeneidade (BOOIJ et al., 2021). A heterogeneidade é uma característica marcante desse ecossistema, uma vez que dispositivos podem operar com diferentes sistemas operacionais, protocolos de comunicação e níveis de capacidade de processamento.

Em particular, é essencial que as *smart homes* observem rigorosos requisitos de segurança, pois a infraestrutura residencial conectada à Internet oferece uma variedade de serviços por meio de redes sem fio, que muitas vezes estão sujeitas a vulnerabilidades. Essas ameaças podem ter origem em falhas no desenvolvimento de software, na configuração incorreta dos dispositivos ou no comportamento do usuário, como a adoção de senhas fracas ou a falta de atualizações regulares.

Kang, Moon e Park (2017) apresentam requisitos de segurança para *smart homes*, essenciais para enfrentar os desafios inerentes à complexidade e diversidade dos dispositivos IoT:

- **Integridade**. Dispositivos IoT frequentemente utilizam redes sem fio, exigindo a implementação de medidas de segurança robustas. Um invasor pode introduzir um software malicioso que comprometa a integridade dos dados, alterando sua finalidade por meio de código infectado. Sem a integridade garantida, todo o sistema de uma *smart home* pode ser comprometido, resultando na indisponibilidade de serviços essenciais (JACOBSSON; BOLDT; CARLSSON, 2016). Além disso, garantir a integridade pode envolver o uso de técnicas como assinaturas digitais e verificações de integridade de software.

- **Disponibilidade**. Dispositivos inteligentes enviam e recebem dados de forma contínua, tanto entre si quanto de/para servidores em nuvem, utilizando redes sem fio. Um ataque, como o envio de requisições maliciosas para sobrecarregar o servidor, popularmente conhecido por ataque de negação de serviço, pode causar a indisponibilidade de serviços ou até mesmo do dispositivo em si. Para mitigar isso, é necessário implementar controle de acesso rigoroso e estratégias de resiliência que garantam que os dispositivos possam continuar operando seguramente, mesmo em cenários de falhas ou sobrecarga. Nesse contexto, a disponibilidade pode ser compreendida como um dos atributos fundamentais da dependabilidade de sistemas, conforme definido por Laprie (1992), que a caracteriza como a capacidade de um sistema em fornecer serviço correto e contínuo, mesmo diante de falhas ou ataques.

- **Autenticação robusta**. A segurança de dispositivos IoT é muitas vezes negligenciada, permitindo que invasores obtenham acesso não autorizado, introduzam códigos maliciosos e comprometam os serviços de uma *smart home*. A implementação de métodos de autenticação multifator e certificados digitais é essencial para impedir acessos indevidos e manter a integridade dos sistemas.

- **Criptografia de dados**. A proteção dos dados transmitidos entre dispositivos por meio de protocolos criptográficos é importante para evitar interceptações e modificações não autorizadas. Isso ajuda a assegurar que informações sensíveis permaneçam confidenciais e protegidas contra ataques.

- **Gerenciamento de acesso**. Dispositivos e serviços em uma *smart home* devem permitir controles de acesso detalhados, possibilitando que os usuários definam quem

pode acessar ou modificar funcionalidades específicas. Esse controle é importante para limitar a exposição a possíveis falhas de segurança.

- **Monitoramento e detecção de ameaças**. A implementação de sistemas de monitoramento contínuo é necessária para identificar atividades suspeitas ou comportamentos anômalos na rede. Tais sistemas podem alertar os usuários para possíveis ataques, permitindo uma resposta rápida para mitigar danos.

- **Atualizações automáticas e seguras**. Manter os dispositivos IoT atualizados com *patches* de segurança reduz a exploração de vulnerabilidades conhecidas. Processos de atualização automáticos e seguros são fundamentais para garantir que os dispositivos continuem protegidos contra novas ameaças.

- **Resiliência a falhas**. É importante que os sistemas sejam capazes de se adaptar, se recuperar e restaurar suas funções quando uma ou mais falhas ocorrem. Para isso, mecanismos de tolerância a falhas e adaptação podem reduzir os impactos de falhas futuras.

Em síntese, a segurança da informação é uma preocupação central em qualquer contexto, mas assume uma criticidade ainda maior nas *smart homes*, devido à diversidade, escala e complexidade dos dispositivos interconectados. Enfrentar esses desafios requer um compromisso contínuo com práticas de segurança avançadas, a colaboração entre desenvolvedores e usuários e uma abordagem proativa para a identificação e mitigação de ameaças emergentes.

2.3 CRITICIDADE DAS VULNERABILIDADES

Uma vulnerabilidade pode ser definida como uma fraqueza ou falha em um sistema de software, hardware, ou em um processo organizacional que pode ser explorada para comprometer a segurança do sistema. Essas vulnerabilidades, se não forem corrigidas, podem ser utilizadas por invasores para obter acesso não autorizado, interromper serviços ou manipular dados confidenciais, dentre outras ações maliciosas. Assim, uma vulnerabilidade pode se fazer presente por meio de uma segurança física deficiente, autenticação inadequada, criptografia imprópria, portas abertas desnecessárias, controle de acesso in-

suficiente, capacidades de gerenciamento de *patches* impróprias, práticas de programação fracas e mecanismos de auditoria insuficientes Hammi et al. (2022).

Programas como o *Common Vulnerabilities and Exposures* (CVE)[1] e o *National Vulnerability Database* (NVD)[2] desempenham um papel fundamental na identificação, definição e catalogação de vulnerabilidades de segurança cibernética publicamente divulgadas. Esses bancos de dados funcionam como referência para profissionais de segurança, auxiliando na avaliação e resposta a riscos em sistemas de TI. Ferramentas de teste de penetração, como o OpenVAS[3], frequentemente utilizam essas bases de dados para detecção e análise de vulnerabilidades.

A gravidade ou criticidade de uma vulnerabilidade é uma medida do risco potencial associado a ela e é avaliada com base em uma combinação de fatores, incluindo:

- **Existência da vulnerabilidade**. Se a falha foi identificada e documentada.
- **Facilidade de exploração**. O quão simples ou complexo é para um invasor explorar a falha.
- **Impacto potencial**. O nível de dano que a exploração pode causar, como perda de dados, acesso a informações confidenciais ou interrupção de serviços.

No OpenVas, as vulnerabilidades são classificadas em níveis como alto, médio, baixo, log e falso positivo, de acordo com sua gravidade (AKSU; ALTUNCU; BICAKCI, 2019). Uma vulnerabilidade de criticidade alta, por exemplo, pode permitir a execução remota de código sem autenticação e, assim, comprometer totalmente um sistema, podendo resultar em perda de controle e dados críticos. Já uma vulnerabilidade de baixa criticidade, como um log de depuração exposto, pode fornecer apenas informações básicas sobre o sistema, representando um risco limitado e de menor urgência para correção.

Para melhor ilustrar esses conceitos, será introduzido um exemplo prático. Uma vulnerabilidade de criticidade alta pode ser ilustrada por falhas como a CVE-2021-44228, conhecida como *Log4Shell*, que permite a execução remota de código em sistemas que utilizavam o *Log4j*. Essa vulnerabilidade teve impacto global e exigiu uma resposta imediata devido ao seu potencial de exploração em larga escala. Por outro lado, uma vulnerabilidade de baixa criticidade pode ser um formulário de *login* que expõe mensagens de

[1] https://cve.mitre.org/
[2] https://nvd.nist.gov/
[3] https://openvas.org/

erro detalhadas, fornecendo informações mínimas que poderiam, em raras circunstâncias, ajudar um invasor a entender a estrutura do sistema.

Entender a criticidade das vulnerabilidades permite que as equipes de segurança priorizem suas ações de correção de maneira eficaz, concentrando esforços em problemas de maior risco e mitigando ameaças antes que possam ser exploradas por agentes mal-intencionados.

2.4 PADRÕES DE SEGURANÇA EM IOT

Atualmente existem diferentes padrões de segurança que podem ser usados como *checklist* para verificar o nível de segurança de dispositivos de IoT:

- *Open Web Application Security Project* (OWASP). O projeto OWASP[4] destaca as dez principais vulnerabilidades em dispositivos e ecossistemas IoT, servindo como guia para fabricantes, desenvolvedores e usuários finais na mitigação de riscos de segurança.

    1. **Senhas fracas, previsíveis ou codificadas**. Muitos dispositivos utilizam credenciais padrão que não são alteradas, facilitando ataques de força bruta.
    2. **Serviços de rede inseguros**. Portas abertas ou serviços desnecessários podem expor dispositivos a ataques como *Distributed Denial-of-Service* (DDoS) ou acessos não autorizados.
    3. **Interfaces de ecossistema inseguras**. Falhas em interfaces Web, móveis ou em nuvem, como autenticação inadequada ou ausência de criptografia, são comuns.
    4. **Mecanismos de atualização inseguros**. Atualizações sem verificação de integridade ou sem criptografia aumentam o risco de ataques de *malware*.
    5. **Uso de componentes inseguros ou obsoletos**. Dependências desatualizadas introduzem vulnerabilidades no sistema.
    6. **Proteção de dados insuficiente**. Dados sensíveis são armazenados ou transmitidos sem criptografia, expondo informações à interceptações.

[4] https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project

7. **Falta de gerenciamento de dispositivos**. A ausência de práticas para adicionar, atualizar ou remover dispositivos compromete a segurança do ecossistema.
8. **Configurações padrão inseguras**. Dispositivos vêm frequentemente com configurações padrão que não são otimizadas para segurança.
9. **Ausência de endurecimento físico**. Dispositivos sem proteção física estão suscetíveis a manipulações e acessos não autorizados.
10. **Transmissão e armazenamento de dados inseguros**. A transferência ou armazenamento de dados sem proteção adequada expõe informações sensíveis a ataques.

- **OTA**. OTA[5] (*Online Trust Alliance*) fornece orientações para tornar as transações *online* mais seguras e proteger melhor os dados dos usuários (PRIYA et al., 2018). A estrutura OTA, conhecida como *IoT Trust by Design*, apresenta oito princípios de melhores práticas de segurança para dispositivos IoT, com um programa de certificação. Além disso, pode ser adotada por fabricantes, distribuidores, formuladores de políticas e consumidores para aumentar a confiança e reduzir os riscos no uso de dispositivos IoT (Internet Society, 2018). Esses princípios incluem:

  1. **Autenticação.** Exige autenticação forte para dispositivos e usuários, com suporte a senhas únicas, autenticação multifatorial e medidas contra ataques de força bruta.
  2. **Criptografia.** Requer criptografia robusta de dados em trânsito e armazenamento para proteger informações sensíveis.
  3. **Segurança.** Implementação de práticas seguras em todo o ecossistema (dispositivos, aplicativos e serviços de *backend*), incluindo atualizações regulares e testes contínuos para minimizar vulnerabilidades.
  4. **Privacidade.** Fornecimento de políticas claras sobre coleta, uso e retenção de dados, garantindo consentimento ativo do consumidor para finalidades adicionais.
  5. **Atualizações.** Os dispositivos devem oferecer atualizações seguras e automatizadas, com notificações claras ao usuário.

[5] https://www.internetsociety.org/ota/

6. **Controle do Usuário.** Oferece aos consumidores opções para gerenciar preferências de privacidade e apagar dados antes de transferir ou desativar dispositivos.
7. **Transparência.** Exige divulgações completas sobre funcionalidade, duração do suporte, impactos de falhas de conectividade e políticas de retenção de dados.
8. **Comunicações.** Garante a segurança de comunicações com consumidores, prevenindo ataques de engenharia social e *phishing*.

- **IoTSF**. IoTSF propôs uma lista abrangente de requisitos de segurança que pode ser usada para avaliar o nível de proteção de dispositivos e sistemas IoT. Esses requisitos são organizados em diversas categorias, incluindo controle de acesso e autenticação, proteção de dados, gestão de vulnerabilidades, resiliência operacional e atualizações seguras. Com base nessa avaliação, é possível posicionar um produto IoT em uma das cinco classes de segurança (Classe 0 a Classe 4), permitindo classificar seu nível de conformidade com as boas práticas de segurança (BADRAN, 2019).

- **ETSI**. O *European Telecommunications Standards Institute* (ETSI)[6] oferece suporte ao desenvolvimento, ratificação e testes de padrões aplicáveis globalmente para sistemas e aplicativos (ETSI, 2019b). Para isto, estabelece diretrizes rigorosas sobre atualizações de software seguras para dispositivos IoT, visando proteger os dispositivos contra vulnerabilidades e garantir a segurança contínua após sua implementação. Esses requisitos são fundamentais para prevenir a exploração de falhas de segurança e garantir que os dispositivos IoT permaneçam protegidos mesmo após sua venda e instalação no mercado, minimizando os riscos de ataques cibernéticos e outros problemas de segurança.

  1. **Atualizações Automáticas e Seguras**. O padrão exige que os dispositivos IoT tenham um mecanismo para garantir que as atualizações de software sejam feitas de forma automática, sempre que possível. Essas atualizações devem ser realizadas de maneira segura, sem que o processo possa ser manipulado por agentes mal-intencionados. Isso inclui garantir que as atualizações não possam ser forjadas ou adulteradas durante o processo de transmissão, ou instalação.

[6] https://www.etsi.org/

2. **Verificação de Integridade**. Antes de instalar qualquer atualização, o dispositivo deve verificar a integridade dos pacotes de atualização para garantir que não houve alterações no conteúdo ou qualquer tipo de corrupção durante o *download*. Essa verificação deve ser feita utilizando mecanismos de criptografia, como assinaturas digitais.
3. **Transparência e Notificação ao Usuário**. O padrão também exige que os consumidores sejam notificados claramente sobre a disponibilidade de atualizações e sua natureza. As notificações devem informar aos usuários sobre os riscos envolvidos, como a correção de vulnerabilidades, e os benefícios da atualização.
4. **Suporte Continuado**. Dispositivos IoT devem ter um período de suporte ativo durante o qual as atualizações de segurança continuarão a ser fornecidas. Isso ajuda a manter o dispositivo seguro ao longo de sua vida útil, mesmo após o lançamento do produto no mercado.
5. **Reversibilidade das Atualizações**. Caso uma atualização cause falhas ou problemas, o dispositivo pode reverter para uma versão anterior segura do software, permitindo que o sistema continue a funcionar sem exposição à novas vulnerabilidades.

O ETSI, assim como OWASP e IoTSF, desempenha um papel crucial na definição de padrões de segurança em *Tecnologia da Informação* (TI). Isto inclui padrões relacionados à IoT, oferecendo orientações e diretrizes para os envolvidos no desenvolvimento e fabricação de produtos IoT voltados para o consumidor.

2.5 SISTEMAS ADAPTATIVOS

Um software adaptativo avalia seu comportamento e o altera quando a avaliação indica que este software não está fazendo o que deveria fazer ou quando pode melhorar a funcionalidade, desempenho ou segurança (LEHMAN, 1996). Salehie e Tahvildari (2009) afirmam que o ponto chave do software adaptativo é que seu ciclo de vida não deve ser interrompido após seu desenvolvimento e configuração inicial. Weyns (2017) define que um sistema adaptativo deve lidar de forma autônoma com as mudanças e incertezas ambientais, impactando o sistema e seus objetivos. Autonomia significa ajustar o sistema com pouca ou nenhuma intervenção humana, quando isto é possível de ser aplicado. En-

tretanto, por existir cenários nos quais o ser humano compreende melhor o contexto, a participação humana é aplicada para auxiliar na decisão. Segundo Kephart e Chess (2003), o autogerenciamento é uma característica dos sistemas computacionais autônomos. Isso visa diminuir a necessidade dos administradores de sistema realizarem ações de operação e manutenção. Assim, o autogerenciamento proporciona aos usuários uma máquina funcionando com desempenho máximo sem interromper sua execução. A computação autônoma tem quatro propriedades autoadaptativas (SALEHIE; TAHVILDARI, 2009):

- **Autoconfiguração (*Self-Configuring*)** é a capacidade de se configurar automaticamente de acordo com políticas de alto nível, representando objetivos do nível de negócios.

- **Autocura (*Self-Healing*)** é a capacidade de detectar, obter diagnósticos e reparar problemas resultantes de *bugs* ou falhas de software e hardware. Além disso, é possível se antecipar a problemas futuros e, portanto, tomar medidas para evitar uma falha.

- **Auto-otimização (*Self-Optimization*)** é a capacidade de buscar continuamente maneiras de melhorar sua operação, identificando e aproveitando oportunidades para se tornar mais eficiente em desempenho ou custo. Tempo de resposta, rendimento, utilização e carga de trabalho são exemplos de preocupações importantes relacionadas a essa propriedade.

- **Autoproteção (*Self-Protecting*)** é a capacidade de um sistema se defender contra problemas (*e.g.,* intrusão, exploração de vulnerabilidades e falhas em cascata) que se não forem devidamente tratados, podem acarretar sérias consequências para o sistema. Para enfrentar esses desafios, sistemas de autoproteção podem adotar duas abordagens complementares: (i) **Reativa**, que atua na defesa contra ataques ou falhas já ocorridos, aplicando medidas corretivas para mitigar os danos e restaurar o funcionamento do sistema; e (ii) **Proativa**, que visa identificar potenciais problemas antes que eles se manifestem, tomando medidas preventivas para evitar ou minimizar seus impactos. A integração dessas abordagens é essencial para proteger o sistema em um ambiente dinâmico e suscetível a constantes ameaças, promovendo uma maior resiliência e segurança.

Weyns (2017) apresenta um modelo conceitual para sistemas autoadaptativos. Um modelo conceitual define um conjunto de elementos básicos de um sistema adaptativo e a relação entre eles. Um ponto importante a ser considerado é que modelos conceituais de sistemas autoadaptativos não se preocupam em apresentar como é implantada a distribuição do software no hardware. A distribuição dos sistemas autoadaptativos se dá com a implantação de vários componentes de software que se comunicam por meio de uma rede. A Figura 2.2 apresenta o modelo conceitual de um sistema autoadaptativo, como definido por Weyns (2017).

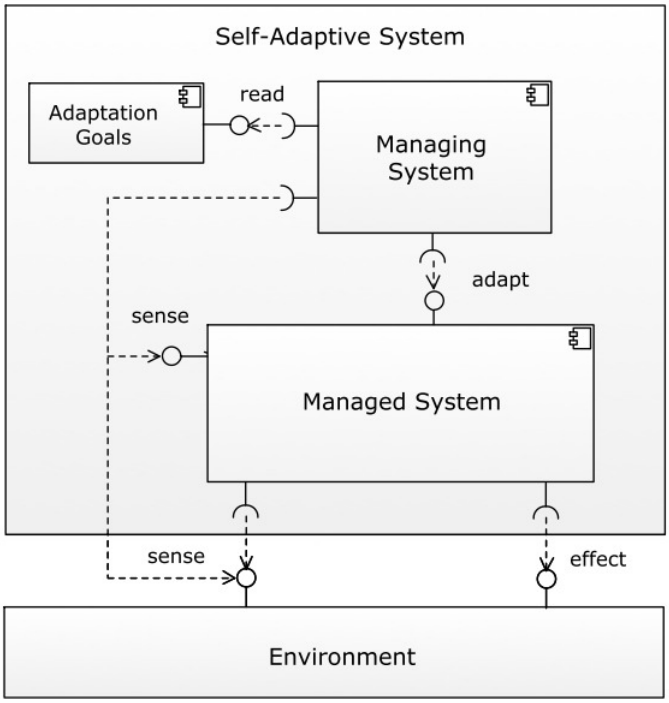

Figura 2.2 – Modelo conceitual de sistema autoadaptativo

Fonte: Weyns (2017)

O **Sistema Gerenciador (*Managing System*)** compreende a lógica de adaptação que trata de uma ou mais metas de adaptação. Para realizar as metas de adaptação, o *Sistema Gerenciador* monitora o ambiente e o *Sistema Gerenciado* e adapta este último quando necessário, fazendo assim com que o *Sistema Gerenciado* cumpra seu papel (WEYNS, 2017). O *Sistema Gerenciador* é tipicamente estruturado com o uso do MAPE-K (KEPHART; CHESS, 2003), que é um modelo de referência para soluções autoadaptativas. O MAPE-K estrutura a lógica de adaptação em quatro etapas e com o uso de uma base de conhecimento: *Monitor*, *Analyze*, *Plan*, *Executor* e *Knowledge*. A Figura 2.3 apresenta os componentes do MAPE-K.

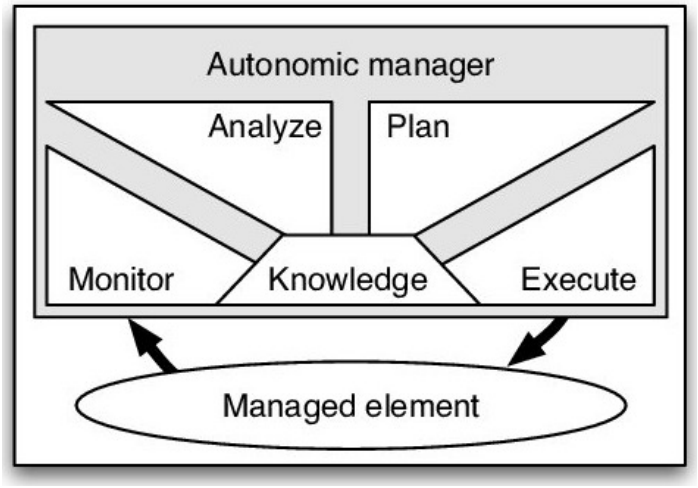

Figura 2.3 – MAPE-K

Fonte: Kephart e Chess (2003)

O **Monitor** coleta (por meio de sensor) os dados do *Sistema Gerenciado* e do ambiente. Em seguida, ele pré-processa esses dados para atualizar o conteúdo do componente *Knowledge*. O **Analyze**, por sua vez, decide se há necessidade de adaptação do *Sistema Gerenciado*. O componente *Analyze* usa as representações dos objetivos de adaptação, disponíveis no componente *Knowledge*. O terceiro componente, o **Plan**, planeja as ações de adaptação que serão executadas. O componente seguinte, o **Execute**, executa o plano de adaptação que ajusta o *Sistema Gerenciado* conforme necessário. Por fim, o **Knowledge** fornece uma abstração de aspectos relevantes da gestão do sistema (*e.g.,* dados, modelos e diretrizes), seu ambiente e as metas de autoadaptação. A cada interação do *loop* de controle MAPE-K, o componente *Knowledge* é atualizado, aprimorando o conhecimento sobre o *Sistema Gerenciado*.

O **Sistema Gerenciado (*Managed System*)** consiste no código que implementa a funcionalidade do sistema autoadaptativo (lógica do negócio). Este código funciona *sentindo* o ambiente e realizando ações que têm efeito sobre este mesmo ambiente. Por exemplo, no caso de uma *smart home*, o sistema gerenciado é a pilha de software que executa nos dispositivos de IoT. Para que as adaptações do sistema gerenciado aconteçam, ele precisa ser equipado com sensores de monitoramento e atuadores que executem as adaptações.

As **Metas de adaptação (*Adaptation Goals*)** definem propriedades de qualidade (invariantes) do sistema gerenciado. Estas metas precisam ser satisfeitas enquanto o sistema executa e violações das metas disparam o processo de adaptação. Por exemplo, uma

meta para o dispositivo de IoT pode ser *os componentes da pilha de software da coisa precisam estar sempre atualizados*. Quando uma nova versão de um componente surge e a pilha não é atualizada, há uma violação da meta. O *Sistema Gerenciador* mantém o *Sistema Gerenciado* sob as *Metas de Adaptação*.

O **Ambiente (*Environment*)** é tudo aquilo que não faz parte do sistema autoadaptativo, porém há interação entre estes dois elementos. No ambiente estão entidades físicas e virtuais, mas o sistema autoadaptativo não tem controle sobre o ambiente.

2.5.1 Adaptação Paramétrica e Composicional

Na literatura sobre sistemas adaptativos, destacam-se dois tipos principais de mecanismos de adaptação: paramétrica e composicional (MCKINLEY et al., 2004). A adaptação paramétrica é caracterizada pela capacidade de modificar o comportamento do sistema mediante ajustes em um ou mais parâmetros. Isso permite que o sistema se ajuste em tempo real, conforme as condições vigentes ou os requisitos preestabelecidos. Esses ajustes podem envolver diversas variáveis, desde configurações de segurança, de desempenho, ou até políticas de tomada de decisão. Por exemplo, um sistema pode bloquear ou liberar o acesso do dispositivo na rede local, ajustar a taxa de transmissão de dados para otimizar a largura de banda disponível ou alterar os níveis de energia para prolongar a vida útil da bateria em dispositivos IoT. A adaptação paramétrica, portanto, permite que o sistema seja flexível e sensível às variações nas condições operacionais, mantendo um nível de segurança desejado sem necessitar de grandes mudanças estruturais.

Em contraste, a adaptação composicional envolve a modificação do comportamento do sistema através da adição, substituição, remoção ou reconfiguração de componentes. Isso significa que o sistema pode ser modificado estruturalmente para se adequar a novos contextos ou requisitos. Os componentes podem representar módulos de software, como *firmware*, hardware físico ou até mesmo unidades funcionais inteiras. Por exemplo, em um ambiente IoT, um sistema pode substituir um sensor por outro mais eficiente ou adicionar novos módulos de segurança para responder a ameaças que possam explorar vulnerabilidades existentes. A adaptação composicional é particularmente valiosa quando são necessárias transformações mais profundas do sistema para acomodar mudanças significativas em sua função ou capacidade. Essa forma de adaptação permite que o sistema evolua ao longo do tempo, integrando novas tecnologias e capacidades sem a necessidade

de uma reformulação completa.

2.6 ATUALIZAÇÕES OTA

Segundo Guissouma et al. (2018), a frequência de atualizações aumentará significativamente nos próximos anos, mesmo com a possibilidade de atualizações mensais de software. OTA atualiza um dispositivo através da rede e permite que um dispositivo sem fio atualize software de forma mais rápida e mais segura do que intervir fisicamente no dispositivo (TIINSIDE, 2021) (DORNERWORKS, 2020).

Com esse crescente interesse em atualizações por meio da rede, as atualizações OTA aumentam a necessidade de medidas de segurança que possam mitigar as vulnerabilidades em meio às atualizações. Assim, a pilha de software (e.g., aplicações, sistema operacional e *firmware*) precisa de medidas de segurança para ser atualizada por meio da rede. Especificamente sobre *firmware*, Kolehmainen (2018) apresenta três abordagens possíveis para o desenvolvimento de mecanismos de segurança relacionados à atualização:

- **SUIT**. A arquitetura SUIT trata da padronização de entregas de *firmware* (MORAN; MERIAC; TSCHOFENIG, 2018). Na especificação SUIT, as imagens do *firmware* são acompanhadas por um manifesto descrito como uma estrutura de dados que inclui informações como a localização da imagem do *firmware* para entrega, dependências, informações criptográficas e dados do dispositivo.

- ***Lightweight Machine to Machine* (LWM2M)**. LWM2M é um protocolo baseado na arquitetura cliente-servidor para gerenciamento de dispositivos. As comunicações no LWM2M acontecem usando o *Constrained Application Protocol* (COAP) e suportam criptografia via *Datagram Transport Layer Security* (DTLS) (POLK; MCKAY; CHOKHANI, 2014).

- ***Blockchain***. A tecnologia *blockchain* pode ser usada para melhorar a segurança de um sistema IoT, inclusive para atualização de *firmware* de dispositivos IoT. Lee e Lee (2017) aplicaram *blockchain* para garantir a integridade do processo de atualização de *firmware*, permitindo a verificação da versão instalada no dispositivo e a validação da autenticidade do *firmware* antes do download e da instalação da

versão mais recente. Steger et al. (2018) usaram *blockchain* em sistemas automotivos, em um mecanismo de atualização dos softwares dos veículos.

2.7 *HUMAN-IN-THE-LOOP*

O conceito de HIL (CáMARA; MORENO; GARLAN, 2015) refere-se a sistemas e processos que integram a interação humana ativa em ciclos de operação ou decisão, especialmente em ambientes automatizados ou baseados em inteligência artificial. Essa abordagem permite ajustar algoritmos, melhorar a precisão e auxiliar em decisões éticas em um contexto específico. Em sistemas ciberfísicos e aprendizagem de máquina, o HIL permite que humanos supervisionem, corrijam ou orientem modelos, em busca de robustez e alinhamento aos objetivos reais.

Esta abordagem permite a configuração do recebimento das notificações e bloqueio de dispositivos com vulnerabilidades críticas sem impactar os serviços. Assim, o contexto em que as vulnerabilidades ocorrem recebe suporte humano para que a atualização de componentes ou parâmetros do dispositivo não impacte o funcionamento da rede da *smart home*.

Estudos recentes destacam várias aplicações de HIL, desde aprendizado de máquina (TAHERISADR; FARUQUE; ELMALAKI, 2023), no desempenho humano-robô em ambientes de pesquisa (SLADE et al., 2024), até segurança para IoT (TAHERISADR; STAVROULAKIS; ELMALAKI, 2023) (FERNANDES et al., 2024).

2.8 CONSIDERAÇÕES FINAIS

No início deste capítulo foram apresentadas as definições de IoT, *smart home*, da pilha de software e classes de dispositivos presentes na *smart home* (Seção 2.1). Na Seção 2.2 foram tratadas as questões de segurança em IoT, com destaque para *smart homes*. A criticidade de vulnerabilidades foi apresentada na Seção 2.3. Alguns dos principais padrões de segurança para IoT existentes foram detalhados na Seção 2.4. A melhoria da segurança provida por sistemas autoadaptativos foi destacada na Seção 2.5. Ainda na mesma seção, foram expostos os componentes do *loop* de controle, necessários para apoiar a implantação deste conceito. Na Seção 2.6, destacaram-se os conceitos de atualizações OTA. Por fim, na última seção foram apresentados conceitos de HIL.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta as principais propostas associadas ao tema central desta tese. Na Seção 3.1, são apresentadas as pesquisas relacionadas a sistemas de segurança em *smart homes*. A segunda Seção (3.2) discute técnicas autoadaptativas aplicadas à segurança de *smart homes*, com ênfase em autoproteção. Na terceira Seção (3.3), trabalha-se com aprimoramentos na segurança por meio do método OTA. A quarta Seção 3.4 foca em estudos que mesclam *Human-In-the-Loop* com sistemas autoadaptativos. Por último, a quinta Seção (3.5) compara UP-Home com as soluções existentes.

3.1 SEGURANÇA EM *SMART HOMES*

Os trabalhos relacionados à segurança de *smart homes*, incluem o uso de algoritmos genéticos para detecção de malware (NAEEM et al., 2024), detecção de vulnerabilidades de firmware (KAUSHIK; BHARDWAJ; DAHIYA, 2025), controle de acesso às ações do usuário (SILVA; SANTOS, 2024) e modelo para segurança baseado na confiabilidade (HELLAOUI; BOUABDALLAH; KOUDIL, 2016).

O estudo de Naeem et al. (2024) aborda ataques adversários em sistemas de detecção de *malware*. Ela introduz uma metodologia de aprendizado profundo de vários estágios que converte binários de *malware* em imagens para análise aprimorada. Algoritmos genéticos otimizam modelos de classificação (*e.g.,* Xception, VGG19) para melhorar sua resistência a ataques adversariais gerados por métodos *e.g, Fast Gradient Sign Method* (FGSM) e *Projected Gradient Descent* (PGD). Os modelos otimizados são então combinados para aumentar a precisão da detecção e classificar famílias de *malware* eficazmente. A combinação de modelos de aprendizado e abordagens clássicas equilibra precisão, tempo de processamento e uso de recursos. Testes experimentais em bancos de dados reais de *malware* mostraram melhorias na taxa de detecção, resistência a ataques e eficiência energética. Os autores discutem também a escalabilidade do sistema para dispositivos IoT, como câmeras de segurança e *smart* TV, que possuem recursos computacionais limitados.

Kaushik, Bhardwaj e Dahiya (2025) investigam as vulnerabilidades de segurança inerentes aos dispositivos de *smart homes*. Os autores apresentam uma solução para analisar e fazer engenharia reversa de *firmware*, o software de baixo nível que controla esses dis-

positivos, para identificar possíveis fraquezas. O emprego de ferramentas avançadas para extrair informações do *firmware* (*e.g.*, sistemas de arquivos, bibliotecas e binários) possibilitou realizar uma análise detalhada para identificar padrões de vulnerabilidades, como configurações inapropriadas, dependências inseguras e algoritmos criptográficos desatualizados. O trabalho destaca a presença cada vez mais comum de vulnerabilidades críticas baseadas em rede que podem permitir que invasores comprometam sistemas de *smart home*, frequentemente agravadas pelo uso de práticas de codificação inseguras e pela falta de recursos essenciais de segurança. Os resultados experimentais apresentados no estudo mostraram que o *framework* identificou falhas críticas em dispositivos IoT, muitas vezes exploráveis por invasores sem a necessidade de acesso físico. Os autores concluem que, ao oferecer melhores práticas para fabricantes e usuários, é possível aprimorar a segurança do firmware dos dispositivos IoT da *smart home*, mitigando riscos e aumentando a confiabilidade e segurança dos dispositivos conectados.

Silva e Santos (2024) apresentam o *Zero-Aware Smart Home* (ZASH), um sistema projetado para aumentar a segurança e a privacidade em *smart homes* controlando as ações do usuário em dispositivos IoT. O ZASH utiliza uma abordagem *Zero Trust* (ZT) com autenticação contínua, percepção de contexto e análise do comportamento do usuário para mitigar ataques de personificação e impor o controle de acesso. Os autores detalham a arquitetura do sistema, incluindo seus protocolos de comunicação e máquinas de estado, e avaliam seu desempenho em um ambiente simulado de *smart home*. O modelo especifica que nenhuma entidade, interna ou externa, deve receber confiança automática; todas as ações e solicitações são submetidas a verificação e autenticação, incluindo uma análise contínua de riscos para identificar comportamentos anômalos ou maliciosos. Isso possibilita a avaliação da legitimidade das solicitações e inclui um componente de resposta adaptativa que ajusta permissões conforme necessário. Por meio de várias simulações, eles demonstram que o ZASH protege efetivamente a privacidade do usuário, responde rapidamente, se adapta a mudanças de dispositivo, resiste a ataques de personificação, isola dispositivos e impõe o controle de acesso, oferecendo uma solução promissora para abordar preocupações de segurança em sistemas de *smart home*.

Hellaoui, Bouabdallah e Koudil (2016) apresentam um modelo para segurança adaptativa com base na confiabilidade dos dispositivos IoT. Este modelo serve para promover ações de segurança (*e.g.,* medidas criptográficas) para economizar energia enquanto o ambiente IoT permanece seguro. O objetivo principal da solução é mitigar a sobrecarga de

autenticação, permitindo que apenas pacotes necessários sejam autenticados, reduzindo custos computacionais e de comunicação. Assim, cada nó calcula e decide periodicamente se deve autenticar a mensagem recebida ou não, dependendo do nível de confiança associado ao remetente da mensagem. O remetente, nesse caso, é o nó vizinho que emitiu ou retransmitiu a mensagem. O nível de confiança deste nó receptor está ligado ao nó remetente para determinar a aplicação da autenticação. A autenticação da mensagem recebida é aplicada de maneira dinâmica (em tempo de execução). O método de avaliação deste trabalho foi composto por três componentes complementares: experiências anteriores, observações diretas e recomendações de terceiros.

Embora esses trabalhos também se concentrem na segurança de *smart homes*, eles são estáticos no sentido de que as ações/medidas de segurança são estabelecidas e permanecem inalteradas, mesmo em cenários onde, por exemplo, surgem novas vulnerabilidades ou ataques.

3.2 SISTEMAS AUTOADAPTIVOS PARA SEGURANÇA DE *SMART HOMES*

El-Maliki e Seigne (2016) apresentaram o *Security Adaptation Reference Monitor* (SARM), uma solução IoT para ajustar dinamicamente a segurança dos ambientes de IoT. O ajuste considera o risco de segurança do ambiente atual e também otimiza seu consumo de energia. Conceitualmente, o SARM é dividido em uma unidade funcional (sistema gerenciado) e uma unidade de monitoramento (sistema gerenciador) conectadas por um *loop* baseado em Teoria de Controle. A cada interação, são ajustados os parâmetros de segurança considerando os riscos de ameaças no ambiente e o desempenho do sistema. Isto ocorre sob as políticas e as restrições de intervenção em tempo de execução, ou seja, de forma dinâmica. Isto faz o sistema se ajustar automaticamente à sua melhor configuração com base no monitoramento do contexto, evitando assim qualquer tomada de decisão estática.

A segurança e a privacidade dos dados do usuário no projeto de um sistema para *smart home* foram discutidas por Alhafidh e Allen (2016). Os autores propuseram um módulo de monitoramento de segurança que analisa cenários baseados no comportamento de indivíduos, identificando padrões de interação entre usuários e dispositivos IoT. O sistema constrói um modelo dinâmico que reflete as atividades e necessidades do ambiente doméstico, utilizando um *loop* de controle para monitorar dispositivos e usuários, equilibrando

as demandas de diferentes pessoas na *smart home*. Além disso, foi apresentado um modelo de visualização para redes de *smart homes*, auxiliando empresas como Amazon, Google e Microsoft a compreenderem os serviços mais relevantes para seus clientes. O sistema usa autoadaptação para otimizar o uso de dispositivos IoT, priorizando decisões automáticas sem intervenção direta do usuário, sem abordar explicitamente o conceito de HIL. Ele combina aprendizado e previsão para analisar e antecipar ações dos usuários, integrando dispositivos e serviços heterogêneos por meio de um *gateway* inteligente, superando problemas de interoperabilidade e oferecendo experiências personalizadas. O estudo destaca a importância da IoT no suporte a serviços locais e na integração de dispositivos distribuídos, enfatizando os benefícios da autoadaptação e das ações preditivas no ambiente de *smart homes*.

Alguns trabalhos propõem o uso do *loop* de controle MAPE-K como modelo de referência para a lógica de adaptação.

Um *Markov Decision Process* (MDP), que representa um modelo matemático para a tomada de decisão em sequência, pode ser usado para descrever formalmente um ambiente. Isto serve como base para o desenvolvimento de um *Intrusion Response System* (IRS), cujo principal elemento é o MDP, organizado por meio do MAPE-K, para planejar respostas de longo prazo a ataques de segurança por meio de ações de resposta atômica (IANNUCCI; ABDELWAHED, 2016). No trabalho de Iannucci e Abdelwahed (2016), foi projetado um IRS autônomo dentro de um *loop* de controle MAPE-K centralizado. Neste trabalho, usaram na fase de monitoramento (*monitor*) um conjunto de *Intrusion Detection System* (IDS) para analisar o tráfego de rede e os registros das cargas do sistema dos *hosts* do sistema gerenciado. Nas fases de *análise* e *planejamento*, os eventos gerados são correlacionados e mesclados em um único fluxo de dados, e calculados para executar ações conforme as políticas adotadas para conduzir o sistema gerenciado ao estado de destino. Em seguida, essas ações são implementadas em tempo de execução. O *Knowledge* contido no controlador, fundamentado na Teoria de Controle, permite que o sistema gerenciado reaja de forma dinâmica e adaptativa às condições que encontra. Usando um MDP, o controlador aplica conceitos probabilísticos para modelar a incerteza e otimizar a tomada de decisão. Assim, ele aprende com os resultados das ações passadas, ajustando continuamente sua estratégia para garantir que o sistema mantenha sua segurança, desempenho ou estabilidade desejada. O uso de técnicas probabilísticas no *analisador* é um diferencial, permitindo ao sistema priorizar ameaças com base em suas probabilidades de ocorrência

e impacto.

Ribeiro et al. (2016) abordam o desafio de criar sistemas autoadaptativos voltados a IoT, onde os dispositivos operam em ambientes dinâmicos e em tempo real. Os autores propõem um padrão arquitetural de gerenciamento baseado na modelagem de loops de controle para permitir que aplicativos de IoT tomem decisões autônomas. Os autores adotaram a abordagem descentralizada, combinando os padrões *mestre/escravo* e o *planejamento regional*. Na fase de monitoramento do MAPE-K (no escravo), foi usado o modelo de aprendizagem de máquina *Multi-layer Perceptron with Limited Weights* (MLPLW) (RIBEIRO et al., 2016). Enquanto isso, as fases de análise e planejamento são executadas no componente mestre. Assim como a fase de monitoramento, a fase de execução também opera no componente escravo. Assim, a arquitetura busca melhorar as soluções existentes ao focar em uma abordagem descentralizada para loops de controle, o que permite uma adaptação mais eficaz às condições locais em ambientes distribuídos. Os autores demonstram a aplicabilidade do padrão por meio de um exemplo de arquitetura de autoproteção e exploram os desafios e consequências de sua implementação em configurações de IoT com recursos limitados.

Usar *blockchain* pode facilitar a implementação de medidas de segurança em sistemas autoadaptativos para IoT. Segundo Sedgewick e Lemos (2018), a adaptação foi integrada a uma *blockchain* para proteger redes de dispositivos contra comportamentos maliciosos, utilizando um *loop* de controle MAPE-K e listas de controle de acesso para aprovar ou rejeitar solicitações, registrando todas as transações. O estudo adota uma abordagem descentralizada com estrutura *mestre/escravo*, protegendo a rede IoT de ataques a nós específicos. Embora o trabalho priorize a segurança, ele não aborda diretamente conceitos como HIL, mas destaca a viabilidade de funcionalidades adaptativas. A proposta explora *blockchains* como base para sistemas distribuídos e autoadaptativos, integrando um modelo descentralizado com *blockchain* permissionada para adaptar continuamente a lista de controle de acesso. Em casos de comportamento malicioso, é alcançado consenso para atualizar a lista e excluir nós comprometidos. Como exemplo, o estudo ilustra uma casa inteligente, demonstrando como *blockchains* permissionadas mantêm uma visão consistente da rede, mesmo na presença de nós maliciosos.

3.3 MELHORIAS DE SEGURANÇA COM ATUALIZAÇÕES OTA

Incluir um mecanismo de atualização em dispositivos IoT é uma medida de segurança necessária. No contexto atual, essa necessidade é inerente ao fato de que crimes cibernéticos estão em alta. Com base nisto, a presente seção apresenta trabalhos relacionados com as atualizações OTA. Lee e Lee (2017) se concentram em autenticação e atualização segura de *firmware*, que são um desafio de segurança para os dispositivos embarcados em um ambiente IoT. Os autores propõem um esquema de atualização de *firmware* que utiliza *blockchain* para verificar com segurança a versão do *firmware* e baixar o *firmware* mais recente para os dispositivos. A verificação de atualização de *firmware* é feita explorando um método de atualização distribuída usando técnicas de *blockchain*. A implementação de uma rede *Peer-to-Peer* (P2P) foi usada para o compartilhamento de *firmware* usando o *BitTorrent*. Assim, são aplicadas algumas condições para atualizar o *firmware* com mais segurança; por exemplo, a versão do *firmware* do dispositivo deve ter a integridade verificada por meio do *blockchain*. O esquema se concentrou em minimizar o tempo em que ataques podem ocorrer, permitindo que um dispositivo embarcado pudesse verificar rapidamente sua versão mais recente, se necessário. Além disso, o modelo garante a integridade das atualizações ao implementar mecanismos de *hash* e assinaturas digitais. Esses mecanismos permitem que dispositivos detectem modificações não autorizadas no *firmware* durante o processo de distribuição. A abordagem também é projetada para ser escalável, suportando quantidade significativa de dispositivos conectados em ambientes heterogêneos, como IoT.

Zandberg et al. (2019) investigaram padrões e bibliotecas de código aberto que possibilitam atualizações seguras de *firmware* para dispositivos IoT, utilizando a arquitetura *back-end* SUIT, um padrão da IETF projetado para melhorar a autenticação e a integridade das atualizações. O trabalho enfatiza a manutenção de atualizações OTA durante todo o ciclo de vida do dispositivo, abordando questões como proteção da integridade do *firmware*, métodos para iniciar atualizações (pelo dispositivo ou externamente) e procedimentos em casos de transferência de propriedade ou mudanças contratuais. Para avaliar o custo funcional do OTA, foi feita uma análise comparativa baseada nos requisitos de memória da configuração *Baseline*. Os autores também realizaram testes práticos em dispositivos IoT com limitações típicas de hardware, destacando como essas restrições impactam diretamente a implementação e a operação de atualizações seguras. Apesar da

eficácia em termos de segurança, desafios como consumo energético e tempo de instalação continuam existindo em dispositivos com recursos limitados. No entanto, o uso de padrões abertos favorece a interoperabilidade, permitindo uma adoção mais ampla em diversas plataformas IoT.

Frisch, Reißmann e Pape (2017) propuseram um sistema que promove segurança no desenvolvimento e publicação de atualizações OTA, com uso de criptografia para dispositivos embarcados baseados em microcontroladores ESP8266. No processo de atualização OTA foi utilizado o protocolo *Message Queuing Telemetry Transport* (MQTT)[1]. O mecanismo proposto foi integrado ao *gateway Home-Assistant* (HA) (*Home-assistant*) para verificar a existência de atualizações disponíveis. Neste trabalho, é apresentado um conjunto de requisitos para delinear o procedimento das atualizações, incluindo mecanismos para desenvolver as atualizações, validar, distribuir e instalá-las nos dispositivos de destino. Os autores destacam a compatibilidade do mecanismo com o protocolo *Hypertext Transfer Protocol* (HTTP) para a comunicação entre dispositivos e servidores, simplificando a integração com infraestruturas existentes. Além disso, o método proposto é projetado para ser escalável, suportando diversos dispositivos simultaneamente, o que é essencial para aplicações em larga escala, como cidades inteligentes ou redes industriais.

Tradicionalmente, para a atualização de dispositivos IoT, é necessária a intervenção manual ou automatizada por meio de atualizações OTA, que ocorre por meio da rede. Atualmente, a segunda opção é significativamente mais utilizada, por conta do aumento crescente de dispositivos IoT. Chandra et al. (2016) apresentam uma solução que viabiliza a atualização OTA para *firmware* de dispositivos IoT, baseando-se no protocolo de rede *Lightweight Mesh* (LWMesh). Esse protocolo fornece uma comunicação de baixo consumo de recursos computacionais entre os dispositivos conectados. A solução se baseia em um serviço de gerenciamento descentralizado, evitando um ponto único de falha. No entanto, sua abordagem depende totalmente da plataforma IoT usando LWMesh e não pode ser totalmente generalizada para todo e qualquer domínio. A solução visa enfrentar desafios de segurança ao implementar mecanismos que verificam a autenticidade e integridade do *firmware* antes de sua aplicação, prevenindo ataques como injeções de código malicioso e atualizações não autorizadas.

Para possibilitar um ambiente mais seguro, Lin e Bergmann (2016) propuseram uma arquitetura de *gateway* para atualizações de *firmware* dos dispositivos IoT. O *gateway*

[1]https://developer.ibm.com/br/technologies/iot/articles/iot-mqtt-why-good-for-iot/

gerencia o processo de atualização automaticamente, estabelecendo um agendamento para as atualizações em horários convenientes. Além disso, a solução pode reverter a atualização se a instalação da atualização resultar em uma perda inesperada de funcionalidade. Outro ponto importante é quando um novo dispositivo é identificado na rede; assim que isto acontece, o sistema oferece suporte a configurações mais seguras (protocolos e atualizações seguras) para o novo dispositivo. Este trabalho se destaca pelo uso de autenticação e controle de acesso baseado em políticas, que podem ser aplicados para limitar o alcance de possíveis invasores. Além disso, os autores enfatizam a importância de atualizações de *firmware* seguras, com o uso de OTA para corrigir vulnerabilidades de modo a manter os dispositivos mais protegidos contra novas ameaças.

3.4 SISTEMAS ADAPTATIVOS COM SUPORTE AO *HUMAN-IN-THE-LOOP*

Por fim, o último grupo de trabalhos considera o envolvimento humano na adaptação. Este grupo possui abordagens que abrangem aspectos desde o projeto de sistemas adaptativos com HIL até a definição de modelos de comportamento humano.

Gil et al. (2020) abordaram o desafio de projetar interações humanas (*Human-In-the-Loop*) no desenvolvimento e operação de *Cyber-Physical Systems* (CPS). Embora os CPS estejam se tornando cada vez mais autônomos, a contribuição humana continua decisiva para lidar com tarefas complexas e não estruturadas. No trabalho, os autores buscam fornecer técnicas e métodos de *design* para facilitar a criação de soluções HIL, enfatizando a importância de ferramentas de visualização e interfaces intuitivas que permitam ao ser humano compreender o estado do sistema. Para isto, foram discutidos objetivos a serem considerados no design de soluções *Human-In-the-Loop*, como: promover a participação humana, mesmo com recursos limitados, tornar a colaboração natural e compreensível, e evitar que a participação humana cause incômodo *(e.g.,* excesso de interação). Assim, é apresentado um método para identificar obstáculos tecnológicos e especificar estratégias de colaboração entre humanos e sistemas. A abordagem enfatiza a necessidade de uma forte consideração do elemento humano ao integrar humanos dentro do CPS. Nos experimentos, avaliou-se o tempo de resposta do usuário ao assumir e repassar o controle do protótipo de carro autônomo, permitindo a análise da atenção e compreensão do cenário em emergências. Os autores também destacam desafios na engenharia dessas interações, incluindo a complexidade de integrar decisões humanas em sistemas altamente

automatizados e dinâmicos.

Taherisadr, Faruque e Elmalaki (2024) apresentaram o sistema ERUDITE, uma solução para IoT centrada no ser humano (*Human-In-the-Loop*), projetado para personalizar e aprimorar experiências de aprendizagem. Foram utilizados dispositivos de neurotecnologia vestível, especificamente EEG, para decodificar sinais cerebrais e inferir o estado de aprendizagem de um aluno em tempo real. Ao entender quando a aprendizagem aumenta ou diminui, o ERUDITE fornece *feedback* adaptativo personalizado para o indivíduo, visando, em última análise, melhorar o desempenho da aprendizagem. Assim, o sistema ajusta dinamicamente conteúdos e métodos de ensino com base nas respostas e preferências do usuário, *e.g.*, entender quando a aprendizagem humana aumenta ou diminui. O sistema proposto combina sensores IoT (*e.g.,* eletroencefalograma (EEG), óculos de Realidade Virtual (RV)), algoritmos de aprendizado de máquina e modelos de comportamento humano para criar um ambiente de aprendizado adaptativo. O ERUDITE utiliza *feedback* do usuário e análises em tempo real para ajustar materiais de estudo, horários e estratégias pedagógicas, aumentando o engajamento e a eficácia. Avaliado em um ambiente de aprendizagem, o ERUDITE demonstrou que, ao usar sinais cerebrais como sensor para inferir estados de aprendizado e sonolência (classificadores), oferece adaptações personalizadas ao ambiente de ensino. O sistema é projetado para praticidade e escalabilidade, com foco na implantação de ponta e baixo consumo de recursos, tornando-o adequado para integração em futuras salas de aula inteligentes e outros ambientes de aprendizagem.

Schenkluhn et al. (2024) exploram como a Realidade Aumentada (RA) pode simplificar a configuração de dispositivos IoT na automação de configuração em *smart homes*, com foco em uma abordagem centrada no ser humano. Os autores analisam desafios técnicos como a configuração complexa das aplicações IoT, compatibilidade de dispositivos e personalização, propondo uma solução para a localização de dispositivos com sensores de RA e atuadores de eletrodomésticos conectados. O protótipo desenvolvido detecta lâmpadas inteligentes por meio de visão computacional com *OpenCV*, utilizando o *gateway* HA. Neste estudo, a interação humana com os dispositivos foi estabelecida de três formas: manual, semi-automática e automática, otimizando a integração de novos aparelhos por meio de ajustes baseados no uso contínuo. O sistema facilita a configuração inicial de dispositivos estáticos, economizando tempo e reduzindo frustrações. Resultados experimentais mostram melhorias na usabilidade, diminuição do tempo de configuração e maior satisfação dos usuários. A pesquisa destaca a importância de equilibrar automação

e controle humano, permitindo ajustes que garantam confiança e conforto. Por fim, os autores sugerem que uma abordagem centrada no ser humano, que considere a psicologia do usuário, é crucial para a adoção bem-sucedida de RA em casas inteligentes e defende uma solução adaptável que equilibre automação e controle do usuário durante a configuração.

Domaszewicz et al. (2016) introduzem o conceito de *Soft Actuation*, um modelo para ambientes inteligentes como escritórios e *smart homes*, que integra os usuários ao processo de controle e automação, caracterizando-se como uma solução com HIL. O modelo promove a colaboração entre sistemas inteligentes e usuários, permitindo que decisões automáticas sejam ajustadas ou complementadas por ações humanas. Neste sistema, é observado o comportamento dos usuários e as condições do ambiente para propor ajustes em dispositivos, como iluminação e temperatura. Diferentemente de mudanças automáticas imutáveis, as sugestões podem ser validadas, rejeitadas ou ajustadas pelos usuários, promovendo maior personalização e aceitação. A arquitetura inclui um loop de controle iterativo, no qual o sistema monitora condições e analisa dados para propor adaptações, com o usuário como camada final de decisão. Isso melhora a precisão das decisões e minimiza desconfortos ou conflitos que podem surgir em soluções totalmente automatizadas. A participação humana no ciclo resolve situações em que sensores ou algoritmos não capturam completamente o contexto, ou as preferências pessoais. Apesar de o foco ser a interação humano-sistema, o trabalho também aborda questões de privacidade e segurança, melhorando a transparência e controle pelos usuários.

3.5 AVALIAÇÃO COMPARATIVA DOS TRABALHOS RELACIONADOS COM UP-HOME

Ao analisar os trabalhos apresentados neste capítulo, é possível observar uma evolução na busca por soluções que tratam da segurança em ambientes de IoT, especialmente no contexto de *smart homes*. A crescente complexidade e sofisticação dos ataques cibernéticos tornam ainda mais necessário o desenvolvimento de estratégias autoadaptativas que possam responder de forma dinâmica e eficiente a novas ameaças. Fazendo uma síntese dos trabalhos deste capítulo, percebe-se que:

- Os ataques cibernéticos continuam crescendo em número e sofisticação, demandando soluções autônomas que adaptem continuamente a segurança dos dispositivos IoT. Naeem et al. (2024) destacam a detecção de *malware* com base em imagens oti-

mizadas, permitindo respostas rápidas a novas ameaças. Essa abordagem reforça a necessidade de sistemas autoadaptativos, como o *loop* de controle adotado pela UP-Home, que poderia integrar técnicas de análise baseada no nível de criticidade para aumentar sua robustez contra ataques mais severos.

- Estratégias de análise de *firmware* também têm sido fundamentais para mitigar vulnerabilidades em dispositivos IoT. Kaushik, Bhardwaj e Dahiya (2025) propõem um arcabouço para identificar e explorar falhas em *firmwares* de dispositivos inteligentes. Embora focado em análise forense, o trabalho ressalta a importância de mecanismos para monitorar e validar continuamente atualizações, complementando os sistemas de controle de versões já integrados a UP-Home.

- Sistemas de controle adaptativo, como o proposto por Silva e Santos (2024), introduzem o conceito de *Zero Trust*, em que cada ação do usuário é avaliada antes de ser autorizada. Essa abordagem pode ser integrada ao UP-Home para aumentar a segurança das interações humanas com dispositivos IoT, garantindo maior controle sobre o comportamento dos usuários.

- A confiabilidade na comunicação entre os nós de uma rede IoT é essencial para a segurança de IoT. Trabalhos como os de El-Maliki e Seigne (2016) e Hellaoui, Bouabdallah e Koudil (2016) exploram como a adaptação baseada na tríade *Confidentiality, Integrity and Availability* (CIA) pode melhorar a resiliência contra ameaças. UP-Home reforça essas práticas com a integração de *loops* MAPE-K, atualizações OTA e HIL.

- Diferente da solução de Alhafidh e Allen (2016), que prioriza decisões automáticas e aprendizado preditivo, a UP-Home foca na segurança adaptativa, garantindo que as decisões sobre proteção sejam ajustadas dinamicamente com base no contexto e nas necessidades do usuário, aumentando a flexibilidade e a resiliência do sistema.

- A adaptação em tempo real de parâmetros de segurança é amplamente explorada na literatura, sendo uma estratégia essencial para tornar os sistemas IoT mais responsivos e eficientes. Muitos trabalhos investigam a autoadaptação com base em loops de controle. Ribeiro et al. (2016) utilizam o loop MAPE-K para ajustar configurações conforme o ambiente e o comportamento dos dispositivos IoT, enquanto Iannucci e Abdelwahed (2016) empregam métodos probabilísticos para gerenciar a segurança

autonomamente. A combinação de arquiteturas e técnicas adaptativas também é um foco de pesquisa para aprimorar segurança e desempenho. Ribeiro et al. (2016) propõem a associação entre MAPE-K e Teoria de Controle, destacando como diferentes modelos de adaptação podem ser integrados para melhorar a resposta a ameaças e otimizar a operação dos sistemas. A ferramenta UP-Home inova ao integrar HIL com esses mecanismos de adaptação, permitindo maior flexibilidade e assertividade ao incluir a intervenção humana em momentos críticos. Essa abordagem não apenas fortalece a segurança do sistema, mas também amplia sua capacidade de responder de forma dinâmica a novas ameaças, garantindo um equilíbrio entre automação e supervisão humana.

- Na promoção de segurança em IoT, o uso de *blockchain* tem se destacado como uma ferramenta essencial para validar e autenticar atualizações de *firmware*, garantindo maior confiabilidade e resiliência. Trabalhos como Sedgewick e Lemos (2018) demonstram como redes descentralizadas podem fortalecer a segurança dos dispositivos, enquanto Lee e Lee (2017) validam a integridade das atualizações. Além disso, Zandberg et al. (2019) propõem o uso de padrões abertos para aprimorar a segurança no transporte e aplicação de *firmware* com atualizações OTA. No contexto da UP-Home, atualizações OTA podem ser empregadas para implementar segurança adaptativa em tempo real, detectando vulnerabilidades e assegurando que as atualizações sejam aplicadas de forma confiável.

- A atualização de dispositivos IoT ocorre manualmente ou via OTA, sendo esta última a mais adotada devido ao crescimento da IoT. Chandra et al. (2016) propõem uma solução baseada no protocolo LWMesh para atualizações OTA, garantindo comunicação eficiente e descentralizada. Já UP-Home permite a intervenção humana (HIL) para ajustes críticos, aumentando a resiliência contra ataques cibernéticos, enquanto a abordagem de Chandra et al. (2016) foca apenas na proteção automática baseada em protocolos de rede.

- Questões relacionadas ao ciclo de vida dos dispositivos IoT são discutidas por Lin e Bergmann (2016), que abordam desafios como privacidade. O uso de *gateways* como o *Home-assistant*, citado por Schenkluhn et al. (2024) destaca soluções para facilitar a localização de dispositivos com aplicação de visão computacional, enquanto Frisch,

Reißmann e Pape (2017) propõem métodos seguros para transporte de *firmware*. UP-Home já implementa validação das atualizações, mas poderia ampliar a gestão de versões com suporte mais avançado para reversão de atualizações problemáticas.

- O fator humano é cada vez mais relevante em sistemas inteligentes. Trabalhos como Gil et al. (2020) e Taherisadr, Faruque e Elmalaki (2024) enfatizam a personalização e o aprendizado adaptativo com o suporte de interações *human-in-the-loop* (HIL). Gil et al. (2020) destacam métodos para engenharia de interações, enquanto Taherisadr, Faruque e Elmalaki (2024) focam em sistemas personalizados para aprendizado. Já Schenkluhn et al. (2024) mostram como a personalização de sistemas em *smart homes* pode melhorar a experiência do usuário. Essas abordagens podem complementar a UP-Home ao integrar suporte ao usuário em tempo real, melhorando a interação homem-máquina em cenários dinâmicos.

- Schenkluhn et al. (2024) exploram a automação centrada no humano em *smart homes*, mostrando como a adaptação das configurações pode melhorar a experiência do usuário. Já Domaszewicz et al. (2016) introduzem o conceito de *soft actuation*, no qual as interações humanas ajustam dispositivos em tempo real. UP-Home pode avançar nessa direção ao permitir níveis mais granulares de personalização baseados nas preferências dos usuários.

A Tabela 3.1 apresenta uma avaliação comparativa entre os trabalhos apresentados neste capítulo. A segunda coluna da tabela indica os tipos de *loop* de controle, que podem ser MAPE-K, Teoria de Controle ou Não definido. Na terceira coluna, o grau de descentralização mostra quais trabalhos foram desenvolvidos para atuar *Centralizado* ou *Descentralizado*, ou *Não definido*. A quarta coluna mostra o aspecto de segurança tratados ou *Não definido*. As duas últimas colunas correspondem respectivamente ao uso de HIL, ou se aborda atualizações *Over The Air*. A marcação (✓) nas últimas duas colunas refere-se ao uso de HIL e *Over The Air*. Da mesma forma, a marcação (X) significa que o trabalho não tem a respectiva condição.

Observando-se a tabela, a lógica de adaptação pode ser distribuída (descentralizada) ou não. Essa questão é comum quando componentes do MAPE-K, como *Analyzer* e *Plan*, precisam utilizar mais recursos computacionais externamente, já que em IoT a escassez de recursos predomina (CPU, memória). As questões centrais de cada cenário ditam como

Tabela 3.1 – Trabalhos relacionados

Trabalho	**Loop de Controle**	**Grau de descentralização**	**Propriedade de Segurança**	***Human-In-the-Loop***	***Over The Air***
Naeem et al. (2024) (NAEEM et al., 2024)	Não definido	Não definido	Integridade	*X*	*X*
Kaushik, Bhardwaj e Dahiya (2025) (KAUSHIK; BHARDWAJ; DAHIYA, 2025)	Não definido	Não definido	Confidencialidade, Integridade	*X*	*X*
Silva e Santos (2024) (SILVA; SANTOS, 2024)	Não definido	Centralizado	Confidencialidade	*X*	*X*
Hellaoui, Bouabdallah e Koudil (2016) (HELLAOUI; BOUABDALLAH; KOUDIL, 2016)	Não definido	Não definido	Autenticação	*X*	*X*
El-Maliki e Seigne (2016) (EL-MALIKI; SEIGNE, 2016)	Teoria de Controle	Centralizado	Autenticação, Integridade	*X*	*X*
Alhafidh e Allen (2016) (ALHAFIDH; ALLEN, 2016)	Não definido	Centralizado	Não definido	*X*	*X*
Sedgewick e Lemos (2018) (SEDGEWICK; LEMOS, 2018)	MAPE-K	Descentralizado	Confidencialidade, Controle de Acesso	*X*	*X*
Ribeiro et al. (2016) (RIBEIRO et al., 2016)	MAPE-K	Descentralizado	Confidencialidade, Disponibilidade	*X*	*X*
Iannucci e Abdelwahed (2016) (IANNUCCI; ABDELWAHED, 2016)	Teoria de Controle, MAPE-K	Centralizado	Confidencialidade, Integridade	*X*	*X*
Lee e Lee (2017) (LEE; LEE, 2017)	Não definido	Não definido	Confidencialidade, Integridade	*X*	✓
Zandberg et al. (2019) (ZANDBERG et al., 2019)	Não definido	Não definido	Confidencialidade, Integridade	*X*	✓
Frisch, Reißmann e Pape (2017) (FRISCH; REISSMANN; PAPE, 2017)	Não definido	Não definido	Integridade, Disponibilidade	*X*	✓
Chandra et al. (2016) (CHANDRA et al., 2016)	Não definido	Não definido	Autenticação, Integridade	*X*	✓
Lin e Bergmann (2016) (LIN; BERGMANN, 2016)	Não definido	Centralizado	Autenticação, Controle de Acesso	✓	✓
Gil et al. (2020) (GIL et al., 2020)	Não definido	Não definido	Confidencialidade	✓	*X*
Taherisadr et al. (2024) (TAHERISADR; FARUQUE; ELMALAKI, 2024)	Não definido	Não definido	Confidencialidade, Integridade	✓	*X*
Schenkluhn et al. (2024) (SCHENKLUHN et al., 2024)	Não definido	Não definido	Confidencialidade	✓	*X*
Domaszewicz et al. (2016) (DOMASZEWICZ et al., 2016)	Não definido	Não definido	Confidencialidade, Disponibilidade	✓	*X*
UP-Home	**MAPE-K**	**Centralizado**	**Confidencialidade, Integridade, Disponibilidade**	✓	✓

a lógica de adaptação deve ser feita, se centralizada ou descentralizada, distribuindo a funcionalidade do MAPE-K (KRUPITZER et al., 2015). Com exceção dos trabalhos que não são voltados diretamente à autoadaptação ((CHANDRA et al., 2016), (LEE; LEE, 2017), (ZANDBERG et al., 2019), (FRISCH; REISSMANN; PAPE, 2017), (LIN; BERGMANN, 2016)), alguns trabalhos relacionados não definiram explicitamente o tipo de *loop* de controle

utilizado ((HELLAOUI; BOUABDALLAH; KOUDIL, 2016), (ALHAFIDH; ALLEN, 2016)), apesar de apresentarem soluções adaptativas.

Entre os trabalhos com abordagem clara, o *loop* de controle MAPE-K descentralizado foi utilizado por Sedgewick e Lemos (2018) e Ribeiro et al. (2016). Por outro lado, Iannucci e Abdelwahed (2016) (IANNUCCI; ABDELWAHED, 2016) adotaram o MAPE-K centralizado, enquanto El-Maliki e Seigne (2016) (EL-MALIKI; SEIGNE, 2016) aplicaram a Teoria de Controle como modelo de adaptação. Em alguns casos, como em Iannucci e Abdelwahed (2016), houve a integração entre os dois paradigmas, utilizando tanto a Teoria de Controle quanto o MAPE-K.

Além disso, dois trabalhos exploraram o uso de *blockchain*. O primeiro (SEDGEWICK; LEMOS, 2018) propõe um sistema autoadaptativo para gerenciar o controle de acesso em redes IoT, enquanto o segundo (LEE; LEE, 2017) utiliza *blockchain* para garantir a segurança em atualizações de *firmware*.

Diversos estudos abordam a segurança em casas inteligentes, explorando diferentes abordagens para mitigar riscos em dispositivos IoT. Naeem et al. (2024) utilizam algoritmos genéticos para fortalecer a detecção de malware contra ataques adversariais, enquanto Kaushik, Bhardwaj e Dahiya (2025) propõem uma estrutura para identificar vulnerabilidades no firmware de dispositivos domésticos. O sistema ZASH, de Silva e Santos (2024), adota o modelo *Zero Trust* para controle de acesso e autenticação contínua, protegendo contra ataques de personificação. Por fim, Hellaoui, Bouabdallah e Koudil (2016) desenvolvem um modelo adaptativo baseado na confiabilidade dos dispositivos, otimizando a autenticação e reduzindo o consumo de energia sem comprometer a segurança.

Entre os trabalhos avaliados, destaca-se também o uso de aprendizado de máquina (*machine learning*), como em Ribeiro et al. (2016), que propõe um padrão arquitetural para adaptação em ambientes IoT.

Ao analisar os domínios de aplicação, percebe-se que cinco trabalhos tratam diretamente de soluções adaptativas para *smart homes* ((SEDGEWICK; LEMOS, 2018), (CHANDRA et al., 2016), (FRISCH; REISSMANN; PAPE, 2017), (LIN; BERGMANN, 2016)), e outros focam em IoT de forma genérica, sem especificar um domínio específico. Soluções como IDS e IRS são comumente usadas para detectar e responder a ameaças ((RIBEIRO et al., 2016), (ALHAFIDH; ALLEN, 2016), (ZANDBERG et al., 2019), (FRISCH; REISSMANN; PAPE, 2017)).

A segurança da informação (integridade, autenticação e autorização) aparece como

um elemento central na maioria dos trabalhos, especialmente naqueles que tratam de atualizações OTA e comunicação entre dispositivos ((ZANDBERG et al., 2019), (FRISCH; REISSMANN; PAPE, 2017)). Adicionalmente, trabalhos destacam a importância do HIL, permitindo a intervenção humana em momentos críticos (GIL et al., 2020), (TAHERISADR; FARUQUE; ELMALAKI, 2024), (SCHENKLUHN et al., 2024), (DOMASZEWICZ et al., 2016).

Os trabalhos relacionados a esta tese apresentam diversas abordagens para lidar com as vulnerabilidades dos dispositivos e sistemas IoT, com destaque para o uso de *loops* de controle, HIL, atualizações OTA, e aspectos de segurança. Neste contexto, a ferramenta UP-Home se diferencia ao integrar esses conceitos de forma única, destacando-se por sua abordagem de adaptação dinâmica com a assistência humana nas decisões críticas.

Com isso, percebe-se a necessidade de construção de soluções autoadaptativas que abordem questões de segurança relacionadas às *smart homes*, como atualizações OTA, além do uso de modelos de referência como o MAPE-K. Adicionalmente, a integração de tecnologias como *blockchain* e a incorporação do HIL se mostram elementos-chave para o desenvolvimento de sistemas mais seguros e robustos em ambientes IoT.

3.6 CONSIDERAÇÕES FINAIS

A análise de medidas de segurança foi investigada sob diversas perspectivas neste capítulo. A Seção 3.1 apresentou estudos sobre sistemas de segurança em *smart homes*. Na Seção 3.2, foram discutidas técnicas autoadaptativas aplicadas à segurança do ambiente doméstico. Posteriormente, a Seção 3.3 tratou de aprimoramentos de segurança por meio de atualizações OTA. A Seção 3.4 abordou a integração de HIL. Por último, a Seção 3.5 apresentou uma análise de posicionamento da ferramenta proposta em relação às alternativas atuais, por meio de uma avaliação comparativa dos estudos.

4 UP-HOME: UMA SOLUÇÃO ADAPTATIVA PARA SEGURANÇA DE CASAS INTELIGENTES

Este capítulo apresenta a ferramenta UP-Home proposta nesta tese. Inicialmente, o capítulo discute os princípios adotados para o desenvolvimento da ferramenta (Seção 4.1). Na Seção 4.2, é introduzida a arquitetura da UP-Home. Em seguida, é apresentado como a ferramenta foi projetada (Seção 4.3). Por fim, a implementação da ferramenta é detalhada (Seção 4.4).

4.1 PRINCÍPIOS

UP-Home (*UPdated Home*) tem como objetivo dar suporte à atualização contínua da pilha de software dos dispositivos presentes em *smart homes*. Com isso, se espera que esses dispositivos atendam aos padrões de segurança estabelecidos pela indústria e sejam mitigadas vulnerabilidades da *smart home*. Além disso, UP-Home implementa um mecanismo *Human-In-the-Loop* (HIL) para tomada de decisão, especialmente focando em vulnerabilidades mais críticas. Dessa forma, com o HIL, a ferramenta aperfeiçoa a segurança sem causar interrupção nas aplicações em execução na *smart home*, mantendo a rede em pleno funcionamento.

UP-Home foi projetada segundo os princípios apresentados a seguir:

Princípio 1 (Autoproteção). Conforme mencionado na Seção 2.5, sistemas adaptativos evoluem de forma contínua e têm aplicabilidade em diversas áreas, desde a otimização de processos (KEPHART; CHESS, 2003) até a melhoria da eficiência da infraestrutura de uma rede (SALEHIE; TAHVILDARI, 2009). Assim, da mesma forma que os sistemas podem sofrer mudanças para melhorar seu desempenho, medidas no contexto da segurança podem ser aplicadas por meio de adaptação. O ambiente no qual os dispositivos da *smart home* estão inseridos é dinâmico e sujeito a mudanças imprevistas, como a introdução de novos dispositivos que habilitam novos serviços, como *File Transfer Protocol* (FTP), *Secure Shell* (SSH) e *Teletype Network* (Telnet) (BHARDWAJ et al., 2024). Além disso, muitos sistemas devem operar de forma contínua, mesmo diante de alterações no ambiente, na disponibilidade de recursos e nos requisitos dos usuários. Entretanto, a presença de vulnerabilidades não implica necessariamente na interrupção do sistema, mas sim na possível exposição a ameaças, como ataques maliciosos que podem comprometer sua funcionalidade.

Para atender essas necessidades, é preciso um sistema que gerencie a pilha de software, permitindo que ela se adapte e realize ajustes necessários para atender metas de segurança, *e.g.,* todas as vulnerabilidades detectadas devem ser mitigadas, os dispositivos da *smart home* devem estar atualizados. UP-Home foca na propriedade de autoproteção (ver Seção 2.5), visando detectar e mitigar possíveis ameaças à segurança em tempo de execução. A aplicação de estratégias de autoproteção por meio da UP-Home permite que a pilha de software dos dispositivos permaneça constantemente atualizada e apta a lidar com falhas de segurança de maneira reativa e proativa.

Princípio 2 (Adaptação paramétrica e composicional). Sistemas adaptativos podem implementar dois tipos básicos de adaptação: adaptação composicional e adaptação paramétrica (MCKINLEY et al., 2004) (ver Seção 2.5.1). Essas duas formas de adaptação desempenham papéis distintos na capacidade de um sistema se ajustar às mudanças e necessidades do ambiente em que opera. Esses dois tipos de adaptações podem ser combinados em sistemas adaptativos para fornecer a capacidade de se ajustar a diferentes cenários ou desafios. Assim, os sistemas adaptativos buscam garantir a resiliência e a eficiência operacional em ambientes de contínua mudança. A entrada de novos dispositivos, por exemplo, não apenas expande a gama de serviços disponíveis, mas também pode introduzir novas superfícies de ataque, ampliando os desafios de segurança. Isso ocorre porque os novos serviços habilitados podem expor vulnerabilidades antes inexistentes, exigindo mecanismos mais sofisticados de detecção e mitigação de ameaças.

Em última análise, a escolha entre adaptação paramétrica e composicional dependerá das características específicas do sistema, de seus objetivos e das mudanças que devem enfrentar ao longo de sua vida útil. Isso acontece com a modificação, substituição ou remoção de componentes obsoletos, como um *firmware* no caso da adaptação composicional. No caso da adaptação paramétrica, são alterados parâmetros para bloquear um serviço propenso à vulnerabilidade, ou até mesmo, bloquear um dispositivo na *smart home*. Neste caso, o bloqueio do dispositivo acontece apenas com vulnerabilidades mais críticas, sendo necessário o suporte humano na tomada de decisão, para evitar que serviços essenciais sejam interrompidos.

Princípio 3 (5W-1H). O conceito de soluções adaptativas incorpora a abordagem conhecida como *5W-1H*, que se refere a seis questões fundamentais que devem ser consideradas no projeto e na implementação dessas soluções (SALEHIE; TAHVILDARI, 2009): *Onde* adaptar?, *Quando* adaptar?, *O que* adaptar?, *Por que* adaptar?, *Quem* adapta? e *Como*

adaptar?. Cada uma dessas questões desempenha um papel importante na concepção de sistemas adaptativos, e a ferramenta UP-Home aborda explicitamente cada um desses pontos.

Princípio 4 (MAPE-K). O modelo MAPE-K (*Monitor*, *Analyze*, *Plan*, *Execute*, *and Knowledge*) (KEPHART; CHESS, 2003) foi utilizado para estruturar o UP-Home (ver Seção 2.5). Nesse contexto, serviços que podem ser vulneráveis (*e.g.,* FTP, SSH e Telnet) e os componentes da pilha de software do dispositivo (*e.g., firmware* e o sistema operacional) são monitorados, assim como as preferências dos usuários. Com base nesse monitoramento, é verificada a existência de vulnerabilidades que comprometam as metas de segurança. Finalmente, são definidas ações para ajustar parâmetros ou componentes da pilha de software conforme as violações de segurança acontecem.

Princípio 5 (*Human-In-the-Loop*). UP-Home permite que os usuários contribuam com informações que auxiliem na tomada de decisões sobre as questões de segurança da *smart home*. Esta abordagem possibilita configurar o recebimento de notificações e, caso o usuário(a) queira, bloquear dispositivos com vulnerabilidades críticas. Assim, UP-Home atua com o suporte humano nas decisões mais críticas, para que a atualização de componentes ou parâmetros do dispositivo ocorra sem impactar os serviços em execução. Por exemplo, na *smart home*, algumas vezes durante o dia, os dispositivos precisam executar ações diversas, como ligar ar-condicionado, alimentar peixes do aquário, irrigar plantas do jardim, entre outros. Assim, um sistema autônomo poderia bloquear o acesso do dispositivo quando detectar uma vulnerabilidade mais crítica. Embora isolar um dispositivo com vulnerabilidade crítica pareça coerente, uma tomada de decisão como esta necessitaria de apoio humano, para evitar a interrupção de serviços essenciais.

Princípio 6 (Uso de padrões de segurança). A busca por melhorias na segurança das *smart homes* é fundamental Komninos, Philippou e Pitsillides (2014). Isso se deve ao fato de que as *smart homes* podem estar conectadas a uma variedade de serviços por meio de redes com e sem fio, tornando-as potencialmente suscetíveis a diversas ameaças cibernéticas. UP-Home adota requisitos de segurança definidos pelos padrões da ETSI (ETSI, 2019a), ou seja, usa requisitos alinhados com os padrões recomendados pela indústria, tais como:

- O dispositivo deve verificar após a inicialização e, periodicamente, se há atualizações de segurança disponíveis (Requisito **5.3-5**);

- O dispositivo deve notificar o usuário quando a aplicação de uma atualização de software interromper o funcionamento básico do dispositivo (Requisito **5.3-12**);
- Para dispositivos restritos que não podem ter seu software atualizado, o produto deve ser isolável e o hardware substituível (Requisito **5.3-15**).

Baseando-se nos princípios adotados pela UP-Home, a Figura 4.1 apresenta uma visão geral do cenário de uso da ferramenta proposta.

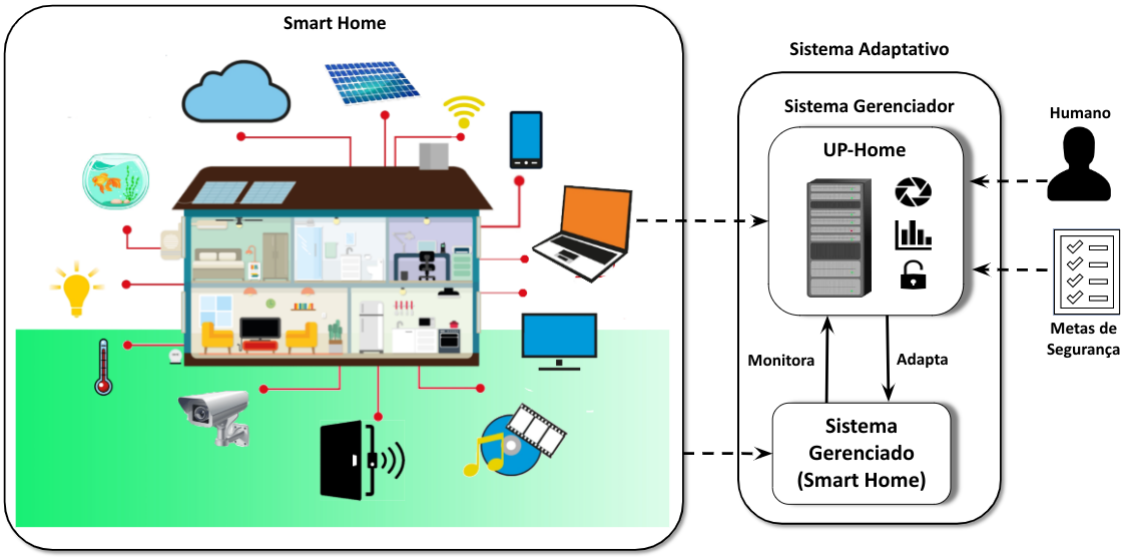

Figura 4.1 – Cenário de uso da UP-Home

Os dois principais componentes desta figura são o *Sistema Gerenciado* e o *Sistema Gerenciador*. O *Sistema Gerenciado* é constituído por entidades físicas e virtuais controladas pelo *Sistema Gerenciador*. O *Sistema Gerenciador* observa o *Sistema Gerenciado* e pode usar dados coletados para determinar se algum ajuste é necessário, *e.g.,* quando as metas de segurança definidas pelo *Sistema Adaptativo* são violadas. O elemento central do *Sistema Adaptativo* é o *Sistema Gerenciador*, estruturado segundo o MAPE-K. O *Sistema Gerenciador* (UP-Home) implementa a lógica da adaptação, ou seja, todas as atividades do processo de adaptação. Deste modo, ele monitora o *Sistema Gerenciado* (Smart Home), e quando ocorrem violações das metas de segurança, realiza adaptações no *Sistema Gerenciado*. Uma violação pode ser caracterizada simplesmente como uma alteração nos parâmetros monitorados, levando a um valor não permitido. Na Seção 4.2 serão apresentados mais detalhes do *Sistema Gerenciador*.

O *Sistema Gerenciado* inclui todos os dispositivos da *smart home* e recebe adaptações em seus parâmetros ou componentes para lidar com as mudanças que ocorrem no am-

biente. O suporte a essas adaptações no *Sistema Gerenciado* consiste em dois elementos principais: *Sensores* e *Atuadores*. Os *Sensores* capturam informações sobre os parâmetros monitorados em tempo de execução e os *Atuadores* realizam as ações necessárias no *Sistema Gerenciado*. A *smart home* pode ser composta por dispositivos variados e interconectados, que incluem desde televisores e sistemas de iluminação, controladores de irrigação de jardim até fechaduras de portas inteligentes e alimentadores de peixes no aquário. Esses dispositivos oferecem conveniência e acessibilidade, permitindo controle local e remoto das funcionalidades da casa. No entanto, essa conectividade também introduz um ambiente propenso a riscos por meio de vulnerabilidades. Por isso, é necessário estabelecer metas de segurança que definam invariantes de alto nível a serem observados durante toda a execução da ferramenta. Com um monitoramento que abrange dados tanto na nuvem quanto na própria *smart home*, gatilhos podem ser acionados para iniciar os processos de adaptação, buscando assim melhorar a segurança contínua da *smart home*.

UP-Home atua em tempo de execução, observando se existem novos requisitos de segurança ou removendo requisitos antigos. Isso exige atualizações contínuas no *Sistema Gerenciado*, garantindo que a ferramenta permaneça alinhada com as melhores práticas de segurança recomendadas pela indústria.

As seções seguintes detalham os elementos que compõem a UP-Home.

4.2 ARQUITETURA

A partir dos princípios apresentados na seção anterior, a arquitetura da UP-Home foi definida como mostrado na Figura 4.2.

No que diz respeito ao *Princípio 1*, referente à autoproteção (*Self-Protecting*), UP-Home é uma ferramenta capaz de melhorar a segurança da *smart home* quanto a problemas relacionados à desatualização de dispositivos e a vulnerabilidades pré-existentes, recém-surgidas, ou que possam ser exploradas. Tais problemas, comuns nas *smart homes*, têm o potencial de provocar falhas em cascata e ampliar-se em larga escala, particularmente devido a ataques maliciosos ou à difusão de falhas não adequadamente corrigidas. Dessa forma, UP-Home ajusta parâmetros e atualiza componentes, com base em políticas de segurança, para proteger os dispositivos da *smart home* e seus usuários. Ademais, sua flexibilidade adaptativa permite não só proteger usuários e dispositivos conectados contra ataques cibernéticos, mas também garantir a evolução do sistema para enfrentar novos de-

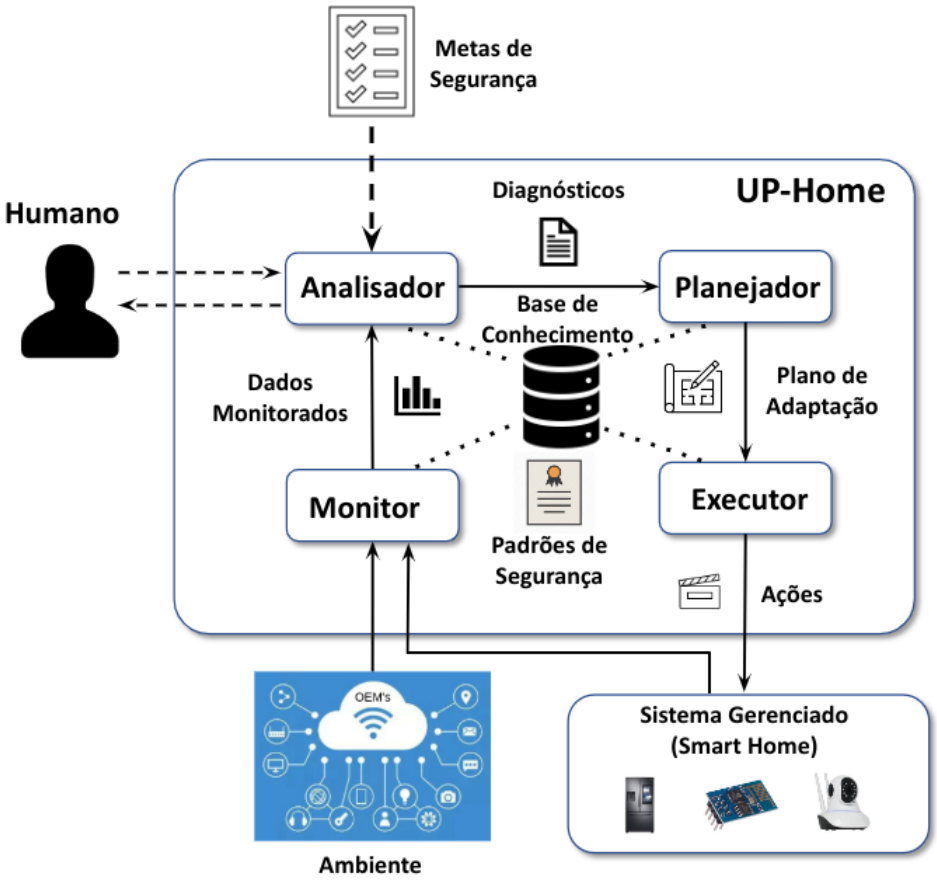

Figura 4.2 – Arquitetura da UP-Home

safios (*e.g.,* enfrentar novas ameaças), maximizando sua resiliência e eficácia. Finalmente, a integração das abordagens reativa e proativa é uma característica importante da UP-Home, permitindo proteger o sistema em um ambiente dinâmico e propenso a ameaças constantes, promovendo maior segurança.

Com relação ao *Princípio 2*, onde são abordados os tipos de adaptação (composicional ou paramétrica), UP-Home realiza essas adaptações por meio de OTA (sem intervenção física) considerando as diferentes classes de dispositivos presentes na *smart home*. É importante destacar que os dispositivos são agrupados em classes distintas, cada uma com suas características específicas de gerenciamento. Alguns dispositivos são classificados como totalmente gerenciados, enquanto outros são parcialmente gerenciados, operando com restrições na coleta de informações do dispositivo, e outros são totalmente restritos (IoTSF, 2018).

O tipo de gerenciamento de dispositivos pode impactar diretamente as ações de adaptação que podem ser executadas nos dispositivos. A Tabela 4.1 apresenta uma visão geral dessas ações de adaptação.

Classes	Integrar	Monitorar	Desenvolver	Atualizar	Reverter
C1	✓	✓	✓	✓	✓
C2	✓	✓	*X*	✓	*X*
C3	✓	*X*	*X*	*X*	*X*

Tabela 4.1 – Classes dos dispositivos e as respectivas possibilidades de intervenção sobre o *firmware*

Esta tabela apresenta como três diferentes classes de dispositivos se relacionam com diferentes intervenções feitas nos dispositivos. A coluna *Integrar* corresponde à possibilidade de comunicação e controle dos dispositivos, *e.g.,* ligar e desligar luzes, acessar imagens das câmeras. A coluna *Monitorar* se refere à possibilidade de coletar dados do dispositivo, *e.g.*, qual a versão do *firmware* em uso. A coluna *Desenvolver* corresponde à possibilidade de desenvolvimento de um *firmware* próprio, com pleno gerenciamento, *e.g.,* autenticação de usuário e armazenamento dos dados. A coluna *Atualizar* trata da possibilidade de executar medidas de segurança na atualização do *firmware*, *e.g.,* verificar a origem. Por fim, a coluna *Reverter* considera a possibilidade de se fazer o *roolback* do *firmware*.

Estas classes são descritas a seguir.

- **Classe C1**. Os dispositivos desta classe operam sem restrições para se implantar adaptação e, por serem de código aberto, qualquer pessoa pode desenvolver sua própria ferramenta, *e.g.,* novo *firmware*. Assim, nesta classe de dispositivos é possível monitorar, desenvolver, atualizar e reverter o *firmware* com medidas de segurança, *e.g.,* autenticação e validação da integridade do *firmware*. Os dispositivos mais comuns nesta classe são microcontroladores, tais como o Esp32 e Esp8266[1].

- **Classe C2**. Os dispositivos desta classe são parcialmente gerenciados, ou seja, possuem *firmware* com código proprietário e podem ser integrados ao *gateway* (*e.g.,* HA), mas possuem apenas permissões para monitorar seus parâmetros, *e.g.,* versão do *firmware*. Segundo a IoTSF (IoTSF, 2018), a maioria dos dispositivos da *smart home* são de classe C2. Os dispositivos de classe C2 não suportam a aplicação de medidas de segurança, *e.g.,* fazer checagem da fonte e da integridade do *firmware* nas atualizações. Apesar disso, é possível identificar a versão do *firmware* em execução nos dispositivos.

[1]https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

- **Classe C3**. Os dispositivos desta classe não contam com implementações de código aberto e operam sob maior restrição do que as Classes C1 e C2. Por esse motivo, dispositivos da Classe C3 não permitem monitoramento de parâmetros de segurança. Comumente, os dispositivos Classe C3 são câmeras IP, smart TVs que não possuem o sistema operacional Android e controladores utilizados em sistemas fotovoltaicos *Electronic Control Unit* (ECU), dentre outros. Apesar dessa diferença, alguns dispositivos podem mudar de uma classe para outra apenas mudando o *firmware*. Por exemplo, o Sonoff[2] é um dispositivo Classe C2 com algumas restrições de gerenciamento de parâmetros impostas pelo fabricante. No entanto, ao substituir o *firmware* original por Tasmota[3], esses dispositivos passam a ser totalmente gerenciáveis. Isso permite gerenciar parâmetros de segurança no desenvolvimento, monitoramento e controle de atualizações de *firmware*.

Apesar das restrições para adaptar componentes da pilha de software, UP-Home atua independentemente da classe do dispositivo. Desse modo, ela monitora a versão do *firmware* dos dispositivos e verifica se existe alguma versão mais recente no site dos fabricantes dos dispositivos. UP-Home aciona o módulo responsável por atualizações e envia os comandos para iniciar a atualização do *firmware* diretamente nos dispositivos reais. Isto minimiza riscos de falhas ou incompatibilidades, já que a atualização é automaticamente aplicada ao dispositivo afetado na rede da *smart home*.

Essa abordagem proporciona segurança aprimorada durante o procedimento de atualização, garantindo que os dispositivos IoT funcionem com o *firmware* mais atual e seguro, livre de vulnerabilidades conhecidas. Assim, o UP-Home altera, por meio de atualizações OTA, os componentes da pilha de software dos dispositivos e ajusta parâmetros no concentrador da rede, como parte de sua estratégia de adaptação, visando reduzir as vulnerabilidades.

Com relação ao *Princípio 3* (Seção 4.1) sobre *5W-1H* (KRUPITZER et al., 2015), as respostas às perguntas no UP-Home são as seguintes:

- ***Onde adaptar?***. Esta questão diz respeito à identificação dos locais ou contextos específicos onde a adaptação é necessária. No caso da UP-Home, a ferramenta determina onde, na infraestrutura da *smart home*, as mudanças ou ajustes são ne-

[2]https://sonoff.tech/
[3]https://tasmota.github.io/docs/

cessários para corrigir problemas de segurança. A adaptação ocorre nos dispositivos para manter seus parâmetros de segurança em conformidade com os padrões estabelecidos pela indústria.

- ***Quando adaptar?***. Refere-se ao momento para realizar adaptações de forma reativa ou proativa. A adaptação ocorre quando há detecção de vulnerabilidades, tais como: presença de dispositivo com *firmware* desatualizado ou se alguma vulnerabilidade for detectada no monitoramento. Para isto, dependendo do escore da vulnerabilidade, é aplicado o HIL (*Princípio 5*).

- ***O que adaptar?***. Implica determinar quais atributos ou elementos do sistema são passíveis de modificação por meio de ações de adaptação, além de definir o que requer alteração em cada contexto específico. Isso abrange tanto parâmetros quanto componentes e recursos do sistema. No caso de UP-Home, isso contempla o *firmware*, a aplicação e parâmetros (*e.g.,* IP, porta), que se relacionam com protocolos empregados na *smart home* (*e.g.,* FTP, SSH e Telnet).

- ***Por que adaptar?***. Esta questão está relacionada à motivação da adaptação. No caso da *smart home*, a adaptação é motivada pela necessidade de manter a segurança da *smart home*. Isto acontece na violação de alguma *Meta de Segurança*, *e.g., manter a pilha de software atualizada*, *manter vulnerabilidades mitigadas*.

- ***Quem adapta?***. Refere-se a quem é responsável em executar a adaptação. Assim, UP-Home opera de forma autônoma, especialmente no que diz respeito à atualização de *firmware* ou para lidar com vulnerabilidades de criticidade baixa à média. Seguindo o *Princípio 5* sobre a implantação do HIL, UP-Home permite a participação de humanos na decisão sobre como tratar vulnerabilidades críticas, *e.g.,* vulnerabilidade com escore alto. Ao permitir que o humano auxilie nas decisões, busca-se garantir que serviços essenciais não sejam atingidos com bloqueio do dispositivo com vulnerabilidade de criticidade alta.

- ***Como adaptar?***. UP-Home realiza a adaptação por meio da substituição de componentes ou do ajuste de parâmetros configuráveis nos dispositivos. Por exemplo, se um dispositivo apresentar um *firmware* desatualizado que o torne vulnerável a ataques de injeção de pacotes, o sistema automaticamente inicia o processo de atualização para a versão mais recente disponível. Além disso, caso um dispositivo da

rede esteja operando com uma configuração insegura (*e.g.,* serviço FTP, SSH, Telnet exposto na porta padrão), UP-Home pode modificar essa configuração, alterando portas, reforçando regras de *firewall* ou restringindo temporariamente o acesso à rede. Em situações críticas, como a detecção de um dispositivo comprometido tentando exfiltrar dados, o sistema pode bloquear completamente seu tráfego ou isolá-lo da rede da *smart home*. A adaptação ocorre sempre que uma *Meta de Segurança* é comprometida, com o *Sistema Gerenciador* (Figura 4.1) intervindo diretamente nos dispositivos afetados.

Em conformidade com o *Princípio 4*, referente à aplicação do MAPE-K, a estrutura da UP-Home inclui os elementos especificados na Seção 4.1.

O *Monitor* coleta dados do *Sistema Gerenciado*. Neste monitoramento, são observados parâmetros de serviços propensos às vulnerabilidades, tais como FTP, SSH e Telnet. Assim, as informações sobre o estado dos dispositivos (*e.g.,* se estão atualizados), informações sobre os dispositivos (*e.g.,* a entrada de um novo dispositivo) e suas respectivas integrações recém-habilitadas (e.g., sensores de temperatura e umidade, controladores de iluminação) são enviadas para o *Analisador*, e armazenadas na *Base de Conhecimento*.

O *Analisador* processa os dados do *Monitor* para determinar a necessidade de uma adaptação. Este componente compara as informações obtidas com as metas de segurança. Além disso, considera as preferências do usuário, como ser notificado sobre vulnerabilidades, aplicar medidas de precaução a outros dispositivos ou bloquear o dispositivo vulnerável, além de mitigar o problema. Com essa avaliação, o *Analisador* decide se é preciso ajustar o *Sistema Gerenciado*. A adaptação é exigida quando uma *Meta de Segurança* está em risco, como na identificação de um dispositivo com *firmware* desatualizado. Igualmente, a detecção de uma vulnerabilidade no monitoramento leva o *Analisador* a adaptar considerando sua importância e as preferências dos usuários. Os resultados do *Analisador* incluem possíveis ameaças como intrusões de rede, *malware*, e acessos físicos não permitidos. Quando ajustes são exigidos, o *Analisador* repassa essas informações ao *Planejador*, caso contrário, a decisão e as justificativas são registradas na *Base de Conhecimento*.

O *Planejador* é responsável por criar um plano de adaptação. Esse plano contém todas as ações necessárias para restaurar o *Sistema Gerenciado*, garantindo que ele satisfaça as metas de segurança.O plano de adaptação na UP-Home é estruturado conforme a

criticidade das vulnerabilidades detectadas, utilizando como base no escore CVSS[4]. Esse plano considera não apenas o escore da vulnerabilidade, mas também os possíveis impactos de ações corretivas na funcionalidade da rede e dos dispositivos da *smart home*.

Por fim, o *Executor* é o componente responsável por executar as ações de alto nível do *Plano de Adaptação*.

Como mencionado na Seção 4.1, a *Base de Conhecimento* armazena todas as informações necessárias à adaptação. Assim, são armazenados detalhes dos dispositivos (*e.g.,* IP, porta, protocolo, nome do dispositivo), parâmetros que precisaram ser adaptados (*e.g., accept/jump/block*), metas de segurança (requisitos), vulnerabilidades mitigadas (*e.g.,* CVE, CVSS, NVD) e informações sobre os componentes da pilha de software (*e.g.,* versão do *firmware*).

A arquitetura da UP-Home foi projetada para aderir a padrões de segurança (*Princípio 6*) amplamente reconhecidos (*e.g., ETSI TS 103 645*), que visam a proteção de dispositivos IoT. Esses padrões são incorporados à ferramenta para mitigar vulnerabilidades e promover a confiança na segurança do ambiente conectado. Exemplos de requisitos implementados incluem a verificação automática e periódica de atualizações de segurança, notificações ao usuário sobre interrupções causadas por atualizações e mecanismos para isolamento de dispositivos não atualizáveis. Essa abordagem assegura que a UP-Home ofereça resiliência e conformidade com as melhores práticas da indústria, integrando segurança diretamente na base de sua arquitetura.

4.3 PROJETO

UP-Home visa melhorar a segurança das *smart homes* por meio de adaptação. A Figura 4.3 apresenta a sequência de operações de alto nível de funcionamento da UP-Home. A ferramenta (*UP-Home*) integra informações do *Ambiente* (fabricantes), dispositivos da *smart home* (Smart Home) e preferências humanas (*Humano*) para tomar decisões dinâmicas sobre adaptações. Isso inclui buscar vulnerabilidades (*Identificar vulnerabilidades da smart home*) e avaliar versões de *firmware* (*Checar versões do firmware*) de tempos em tempos. Em seguida, a partir das preferências do humano (*Buscar preferências do humano*) e considerando as informações coletadas da *smart home*, UP-Home decide se uma adaptação é necessária ou não (*Adaptação necessária?*). Se for preciso realizar uma

[4]https://nvd.nist.gov/vuln-metrics/cvss

adaptação, UP-Home cria um plano de adaptação (*Criar plano de adaptação*) e executa-o (*Executar ações do plano de adaptação*). Caso contrário, as informações sobre a análise são apenas registradas na base de conhecimento da UP-Home.

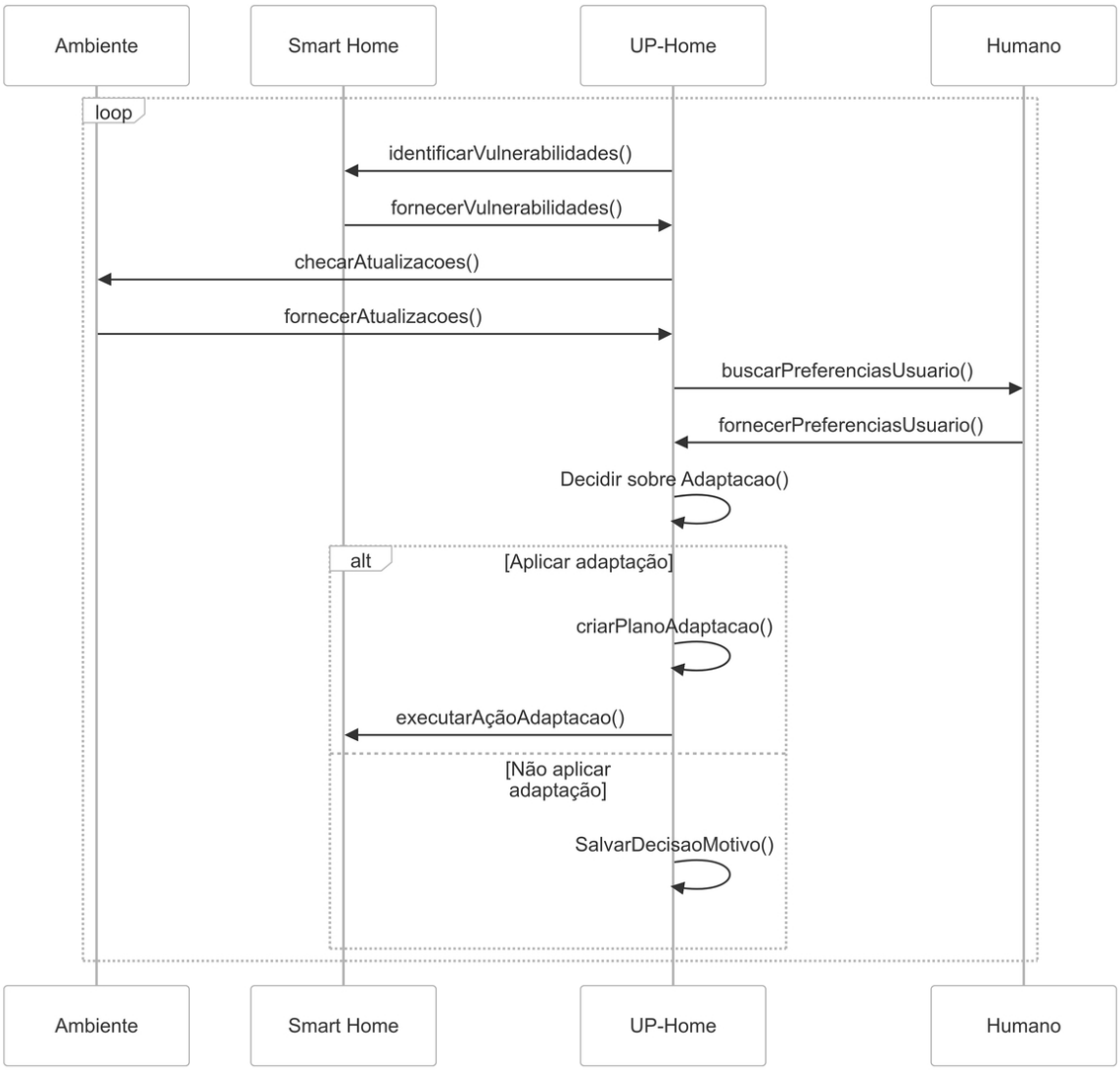

Figura 4.3 – Diagrama de sequência da UP-Home

As seções a seguir apresentam detalhes do projeto de cada um dos componentes da UP-Home.

4.3.1 Monitor

Informações detalhadas sobre a pilha de software dos dispositivos da *smart home* são coletadas pelo *Monitor*. Isso inclui dados sobre o *firmware*, sistema operacional e serviços

potencialmente vulneráveis, *e.g.,* FTP, SSH e Telnet. Assim, o monitoramento contínuo identifica vulnerabilidades que podem ser exploradas por ameaças, fornecendo uma visão do estado de segurança da *smart home*. O *Monitor* utiliza agentes de software para capturar eventos e estados do sistema, gerando métricas analisadas posteriormente. A coleta de dados é configurável para diferentes níveis de granularidade, permitindo uma análise mais detalhada do *Analisador*. Assim, o *Monitor* também detecta novos dispositivos conectados à rede, coleta (por meio de sensores) dados do Sistema Gerenciado, envia esses dados para o *Analisador*, e atualiza a *Base de Conhecimento*.

A Figura 4.4 apresenta o diagrama de sequência do componente *Monitor*, destacando suas interações e processos. A coleta de dados é feita a partir de três fontes principais. Inicialmente, são obtidos relatórios gerados por testes de penetração. Essas varreduras na rede da *smart home* produzem arquivos que listam as vulnerabilidades detectadas, contendo informações essenciais como IP, porta, protocolo, escore da vulnerabilidade, baseado em CVSS, e NVD, que é o identificador da base de vulnerabilidades. Outra fonte de dados resulta da integração com os dispositivos, que pode retornar a lista de dispositivos com informações cruciais, como modelo, fabricante e a versão do *firmware* dos dispositivos presentes na *smart home*. Por último, diferentemente das duas primeiras fontes de dados consultadas na *smart home*, a pesquisa nos sites dos fabricantes (*Original Equipment Manufacturers* (OEMs)) permite obter informações sobre as atualizações de *firmwares* dos dispositivos. Esses dados coletados são encaminhados ao *Analisador* e armazenados na *Base de Conhecimento* para processamento e tomada de decisão no *Analisador*.

A ferramenta UP-Home incorpora um mecanismo de monitoramento contínuo para aprimorar a segurança da *smart home*. A Figura 4.4 apresenta a sequência de operações desse processo. A ferramenta integra informações dos dispositivos e os relatórios de vulnerabilidades da *smart home* (*Smart Home*), além de dados obtidos nos sites dos fabricantes (*Ambiente*). O módulo de monitoramento (*Monitor*) coleta essas informações e as envia para o módulo de análise (*Analisador*), que, por sua vez, interage com a *Base de Conhecimento* para processar e armazenar os dados.

Inicialmente, o sistema busca relatórios de vulnerabilidades (*Identificar vulnerabilidades*), e da lista de dispositivos conectados (*Buscar dispositivos*), recebendo consequentemente da *smart home* essas informações (*Fornecer dados*). Em seguida, busca informações sobre versões de firmware nos sites dos fabricantes (*Ambiente*). Por fim, envia as informações coletadas ao *Analisador* (*Enviar dados*), e armazena os resultados desses

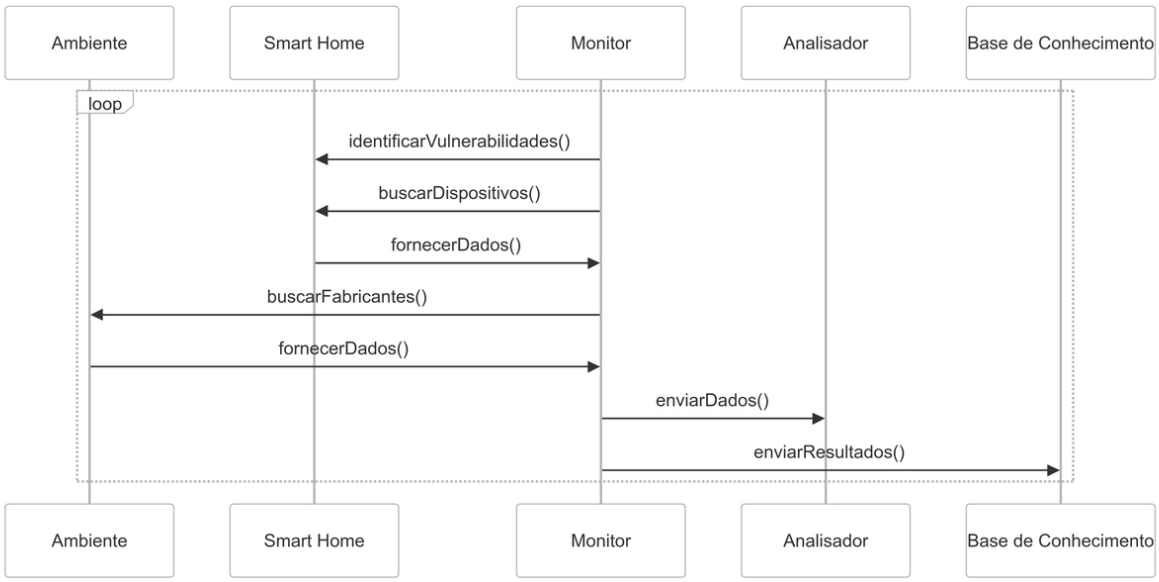

Figura 4.4 – Diagrama de sequência do *Monitor*

monitoramentos na *Base de Conhecimento* para aprimorar futuras decisões e otimizar o gerenciamento da segurança da *smart home*.

4.3.2 Analisador

O *Analisador* é responsável por processar os dados coletados pelo *Monitor*, avaliando a criticidade das vulnerabilidades identificadas e os riscos que estas representam para a segurança da *smart home*. A Figura 4.5 ilustra o diagrama de sequência detalhado do *Analisador* da UP-Home, destacando suas funções principais e as interações envolvidas no processo de tomada de decisão sobre a adaptação de segurança.

A Figura 4.5 ilustra a sequência de interações entre os componentes do sistema, com as preferências do usuário (*Humano*), o monitor de dados (*Monitor*), o módulo de análise (*Analisador*), o que recebe o resultado da análise (*Planejador*) e o que armazena essas informações (*Base de Conhecimento*).

O processo inicia-se com o envio de dados coletados pelo *Monitor* ao *Analisador* (*Enviar dados*) e recebe da *Base de Conhecimento* metas de segurança (*Fornecer metas*). Para aplicação do HIL (*Humano*), as preferências do usuário são consultadas (*Buscar preferências*) e repassadas ao *Analisador* (*Enviar preferências*) para garantir que as decisões sejam personalizadas.

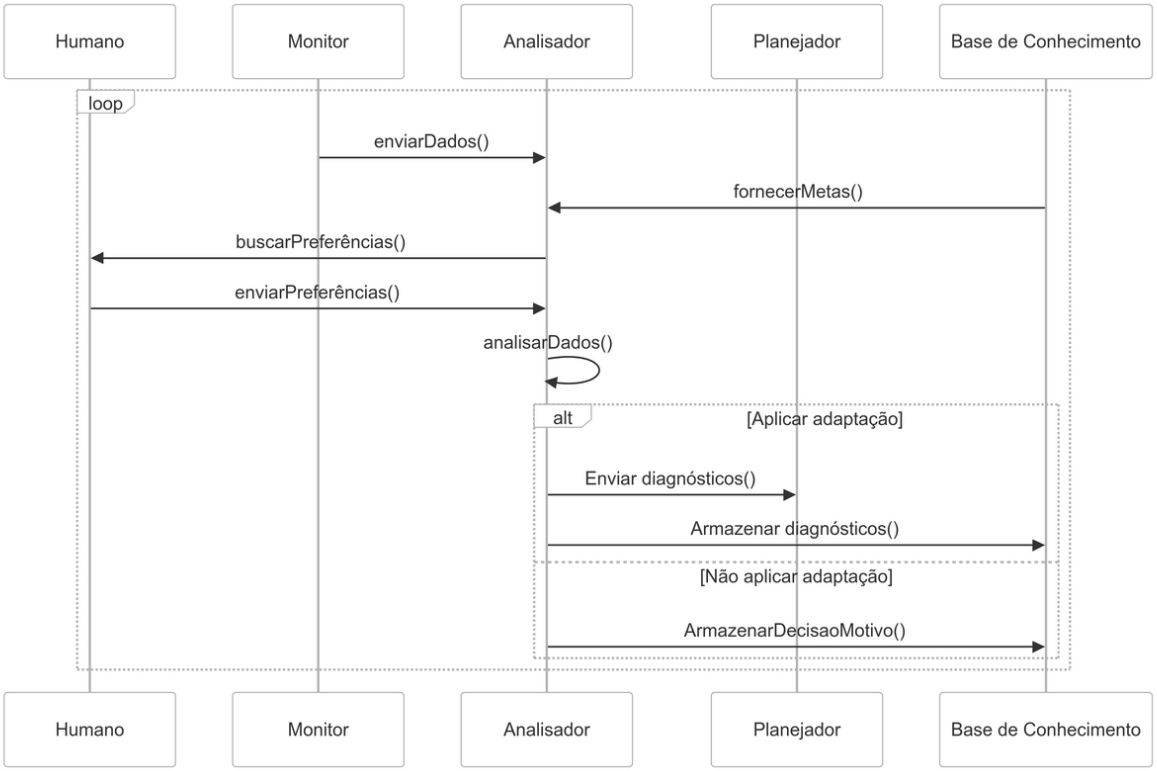

Figura 4.5 – Diagrama de sequência do *Analisador*

Ao receber os dados, o *Analisador* processa as informações (*Analisar dados*). Com base na análise realizada, o sistema pode seguir dois caminhos distintos.

A primeira possibilidade é aplicar adaptação, se a necessidade de ajuste for identificada, o *Analisador* envia diagnósticos ao *Planejador* (*Enviar diagnósticos*), que armazena essas informações na *Base de Conhecimento* (*Armazenar diagnósticos*). Nesta etapa, realiza-se uma análise detalhada para identificar violações das metas de segurança. A análise verifica se a versão do firmware dos dispositivos é inferior à última disponibilizada pelo fabricante, se vulnerabilidades foram detectadas e quais foram os níveis de criticidade. Da mesma forma, é verificado se usuários em suas preferências decidiram ser notificados sobre detecção de vulnerabilidades, se desejam aplicar medidas de segurança preventivamente ou bloquear dispositivos com vulnerabilidade crítica. Outra possibilidade desse processo de decisão é não aplicar a adaptação, quando não há necessidade de ajustes. E se esta for a decisão, esta e o motivo são armazenados (*Armazenar decisão motivo*) para referência futura.

Esse ciclo contínuo permite que o sistema utilize interações anteriores, aprimorando suas recomendações e garantindo que a *smart home* se adapte de forma dinâmica às

necessidades do usuário, melhorando a segurança e a eficiência operacional.

O *Analisador* conta com o suporte ao HIL para permitir que humanos participem de decisões críticas para a segurança da *smart home*. Essa abordagem combina análise automatizada e intervenção humana, com o propósito de, em situações onde as ações automáticas não sejam suficientes ou requeiram uma avaliação mais detalhada, o usuário possa tomar decisões. Essa interação permite que o usuário aplique uma medida preventivamente, ou bloqueie dispositivos com vulnerabilidades de alta criticidade até que sejam atualizados ou substituídos. O uso do HIL permite que usuários desempenhem um papel ativo no gerenciamento de dispositivos IoT. Para viabilizar o envio de notificações e a interação com a UP-Home, pode ser integrada uma plataforma de mensagens, Telegram[5] possibilitando que o usuário seja informado em tempo real sobre vulnerabilidades críticas, com detalhes sobre riscos e impactos. Com essas informações, o usuário pode analisar a situação e decidir se deseja aplicar medidas preventivas a outros dispositivos na *smart home* ou bloquear o dispositivo comprometido na rede.

Na prática, os usuários especificam suas preferências para receber notificações quando vulnerabilidades são detectadas. O usuário oferece informações à UP-Home respondendo inicialmente a três perguntas:

- Q1 - O usuário quer ser notificado sobre vulnerabilidades com escore CVSS Médio?
- Q2 - O usuário deseja que uma ação preventiva fosse adotada para outros dispositivos da *smart home* ao identificar uma vulnerabilidade?
- Q3 - O usuário deseja que o dispositivo seja completamente bloqueado quando uma vulnerabilidade com escore CVSS Alto for identificada?

Em relação a essas questões, alguns pontos precisam ser considerados:

- Se nenhuma vulnerabilidade for detectada, nenhuma ação será tomada pelo UP-Home;
- Toda vulnerabilidade detectada será mitigada, independentemente do seu escore CVSS;
- Toda vulnerabilidade com escore CVSS alto detectada será notificada ao usuário;

[5]https://web.telegram.org/k/

- O usuário pode escolher previamente (respondendo *Sim* à Q1) se deseja ser notificado quando uma vulnerabilidade de escore CVSS médio for detectada, para receber um resumo da vulnerabilidade *e.g.,* impacto que ela pode causar;

- Se uma vulnerabilidade de qualquer escore CVSS for detectada, o usuário pode autorizar (respondendo *Sim* à Q2) que uma ação preventiva seja aplicada nos demais dispositivos da *smart home*. Adicionalmente, o usuário pode autorizar o bloqueio do dispositivo infectado respondendo *Sim* à Q3.

A Tabela 4.2 apresenta todas as combinações de ações de adaptação possíveis devido à participação humana.

Tabela 4.2 – Decisão do *Analisador* com *Human-In-the-Loop*

Vulnerabilidade detectada?	**Q1**	**Q2**	**Q3**	**Decisão de adaptação**
Não	S / N	S/N	S / N	Nenhuma Adaptação
Sim / CVSS **Baixo**	S / N	S	S / N	Mitigar / Prevenir
Sim / CVSS **Baixo**	S / N	N	S / N	Mitigar / Não Prevenir
Sim / CVSS **Médio**	S	S	S / N	Mitigar / Notificar / Prevenir
Sim / CVSS **Médio**	S	N	S / N	Mitigar / Notificar / Não Prevenir
Sim / CVSS **Médio**	N	S	S / N	Mitigar / Não notificar / Prevenir
Sim / CVSS **Médio**	N	N	S / N	Mitigar / Não notificar / Não prevenir
Sim / CVSS **Alto**	S / N	S	S	Mitigar / Notificar / Prevenir / Bloquear
Sim / CVSS **Alto**	S / N	S	N	Mitigar / Notificar / Prevenir / Não bloquear
Sim / CVSS **Alto**	S / N	N	S	Mitigar / Notificar / Não prevenir / Bloquear
Sim / CVSS **Alto**	S / N	N	N	Mitigar / Notificar / Não prevenir / Não bloquear

A interação do usuário com a ferramenta pode facilitar uma decisão mais adequada para a *smart home*. Na prática, o usuário está familiarizado com os dispositivos e pode avaliar, *e.g.,* se um dispositivo é crucial para o funcionamento correto da *smart home*. Por outro lado, se o usuário não responder no tempo determinado, UP-Home pode executar uma ação pré-definida para minimizar os riscos. Esta abordagem combina a automação com a supervisão humana, fornecendo um controle mais sólido e flexível para a segurança da *smart home*, principalmente quando há vulnerabilidades de escore alto.

A decisão sobre adaptação (*Decidir se uma adaptação é necessária*), mostrada na Figura 4.5, é tomada de acordo com o Algoritmo 1. O algoritmo analisa as vulnerabilidades detectadas na rede da *smart home* e determina ações de mitigação, prevenção e notificação, além de verificar a necessidade de atualização de firmware dos dispositivos. A lógica implementada segue um fluxo estruturado, começando pela coleta e processamento

das vulnerabilidades e, posteriormente, avaliando as versões de firmware dos dispositivos cadastrados.

Algorithm 1 Adaptation Checking

```
1: Input: vulns, userPrefs, devices, manufacturers, k
2: Output: diag
3: vulns,ml,pl,bl,un,up ← GetKnowledge(k)
4: mlt ← {}
5: for v ∈ vulns do
6:     n ← GetName(v), i ← GetIP(v), p ← GetPort(v), pt ← GetProt(v)
7:     cvss ← GetCvss(v), nvd ← GetNvd(v)
8:     if p ∉ pl or i ∉ bl then
9:         mlt ← mlt ∪ {<i, p, pt>}
10:        if userPrefs == PREVENTALL then
11:            pl ← pl ∪ {p}
12:        end if
13:        if cvss == MEDIUM then
14:            if userPrefs == NOTIFYUSER then
15:                un ← un ∪ {<n, i, nvd>}
16:            end if
17:        end if
18:        if cvss == HIGH then
19:            un ← un ∪ {n, i, nvd}
20:            if userPrefs == BLOCKDEVICE then
21:                bl ← bl ∪ {i}
22:            end if
23:        end if
24:    end if
25: end for
26: for d ∈ devices do
27:    for f ∈ manufacturers do
28:        sw ← GetSwVersion(d), md ← GetModel(d), fd ← GetManufacturer(d)
29:        lu ← GetlastVersion(f), mf ← GetModel(f), ff ← GetManufacturer(f)
30:        if md == mf and fd == ff then
31:            if sw < lu then
32:                up ← up ∪ {<md, fd>}
33:                mlt ← mlt ∪ {<md, fd>}
34:            end if
35:        end if
36:    end for
37: end for
38: if mlt ∩ lm ≠ ∅ then
39:    knowledge ← knowledge ∪ {<lm, pl, un, bl, up>}
40:    diag ← True
41: else
42:    diag ← False
43: end if
44: return diag
```

O Algoritmo 1 analisa os dados dos relatórios de varreduras na *smart home* e de-

termina as ações para mitigação, prevenção, notificação de vulnerabilidades, bloqueio de dispositivos e atualização de *firmwares*. O fluxo do algoritmo é dividido em duas etapas principais: análise de vulnerabilidades e verificação de atualização de *firmware*.

Na primeira etapa do algoritmo (Linhas 3-25), as vulnerabilidades são obtidas a partir da base de conhecimento (GetKnowledge(k)). Em seguida, cada vulnerabilidade é processada, extraindo informações como nome (GetName), endereço IP (GetIP), porta (GetPort), protocolo (GetProt), escore CVSS (GetCvss) e identificador da base de vulnerabilidades (GetNvd). O algoritmo verifica se a porta afetada não está na lista de portas (pl) ou se o IP não está na lista de bloqueios (bl) (Linha 8). Caso a vulnerabilidade detectada anteriormente não tenha sido, como medida preventiva, inserida na lista de portas (pl) ou bloqueada por meio do IP, os dados da vulnerabilidade (i, p e nvd) são adicionados à lista temporária de mitigação (mlt, Linha 9). Se a preferência do usuário for aplicar medida preventiva (PREVENTALL, Linha 10), a porta da respectiva vulnerabilidade é adicionada à lista de mitigação (pl, Linha 11), e em execução posterior do Algoritmo 1, será usada na decisão. Na sequência, dois blocos baseiam-se no CVSS da vulnerabilidade para resolver sobre notificação de usuário e bloqueio de dispositivo, respectivamente. Em casos de CVSS Médio (cvss == MEDIUM), o usuário pode ser notificado sobre o risco, caso tenha essa opção ativada (Linhas 14-16). Para vulnerabilidades de CVSS Alto (cvss == HIGH), a notificação ao usuário é por padrão acionada (Linha 19), e se o usuário respondeu sim à opção BLOCKDEVICE, o dispositivo é adicionado à lista de bloqueios (bl, Linha 21).

Na segunda etapa do algoritmo (Linhas 26-37), é verificada a necessidade de atualização de *firmware* dos dispositivos da *smart home*. Cada dispositivo (d) é comparado com os fabricantes (f) para identificar a versão mais recente do *firmware* (GetlastVersion(f)). Caso a versão do fabricante (lu) seja superior à instalada no dispositivo (sw), o dispositivo é marcado para atualização na lista (up, Linha 32), e na lista (mlt, Linha 33). A comparação da versão do *firmware* do dispositivo com a versão mais recente do fabricante (Linha 31) viabiliza a decisão sobre a implementação de atualizações e contribui significativamente para atender a meta de segurança estabelecida (e.g., todos os dispositivos precisam estar atualizados).

Por fim, o algoritmo verifica se houve alguma nova mitigação identificada comparando mlt com lm (Linha 38). Se houver novas ações a serem registradas, a *Base de Conhecimento*(*knowledge*) é atualizada com as novas informações sobre mitigação (ml), prevenção

(pl), notificações (un), bloqueios (bl) e atualizações (up) (Linha 39). O algoritmo define *diag* como True, indicando a necessidade de adaptações, que serão utilizadas pelo *Planejador* para gerar um Plano de Adaptação. Se não forem necessárias novas ações, *diag* será definido como False (Linha 40-42).

Esse *design* garante que o sistema de segurança da *smart home* seja dinâmico e adaptável, permitindo a mitigação automática de ameaças enquanto considera as preferências do usuário (HIL).

4.3.3 Planejador

Uma vez que são identificadas possíveis violações das metas de segurança, o *Planejador* cria um plano de adaptação para ajustar a pilha de software dos dispositivos. A Figura 4.6 apresenta o diagrama de sequência do *Planejador* da UP-Home.

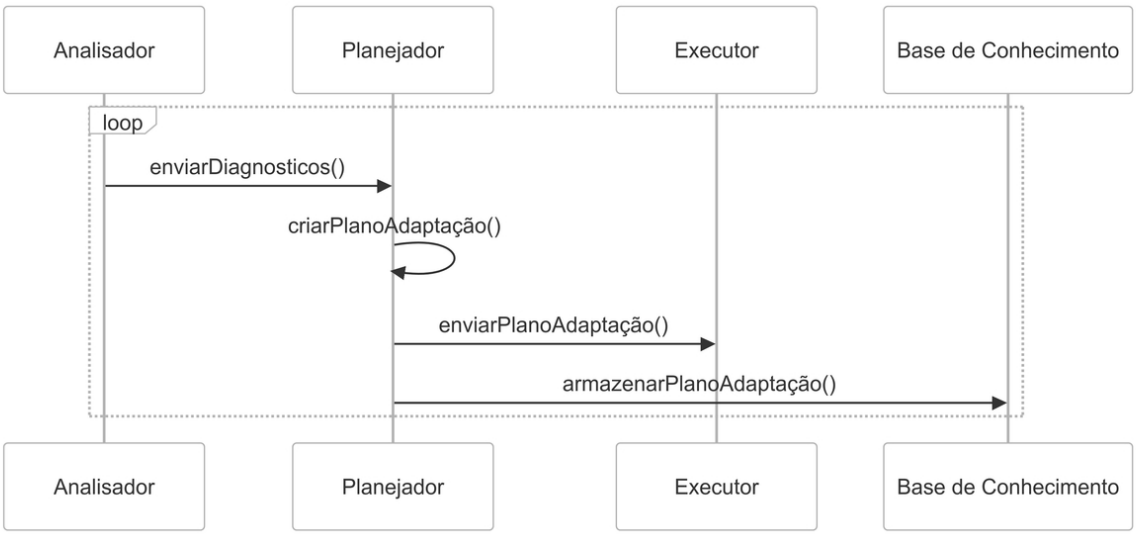

Figura 4.6 – Diagrama de sequência *Planejador*

O *Planejador* da UP-Home possui um mecanismo de adaptação dinâmico, permitindo que o sistema evolua com base nas análises realizadas. A Figura 4.6 ilustra as interações entre componentes que adaptam a *smart home*: *Analisador*, *Planejador*, *Executor* e *Base de Conhecimento*.

Quando o *Analisador* identifica a necessidade de adaptação, envia um diagnóstico detalhado ao *Planejador* por meio de (*Enviar diagnósticos*), iniciando assim o papel do *Planejador*. Ao receber essas informações, o *Planejador* cria um plano de adaptação para

a *smart home*(*Criar plano adaptação*). Esse plano pode envolver ajustes em dispositivos inteligentes, ajustes de parâmetros ou componentes para melhorar a segurança da *smart home*.

Após a criação do *Plano de Adaptação*, ele é enviado ao *Executor* (*Enviar plano adaptação*), responsável por implementar as mudanças necessárias na *smart home*. Paralelamente, o *Planejador* também armazena o *Plano de Adaptação* na *Base de Conhecimento* (*Armazenar plano adaptação*) para futuras referências e aprendizado contínuo do sistema.

Esse ciclo se repete continuamente, garantindo que a *smart home* se mantenha atualizada, segura e alinhada com as metas de segurança. O armazenamento dos planos permite que a UP-Home use essas decisões, otimizando as futuras adaptações de maneira personalizada.

O processo de criação do *Plano de Adaptação* (Criar plano adaptação), apresentado na Figura 4.6, foi desenvolvido conforme o Algoritmo 2. Esse algoritmo analisa os diagnósticos repassados pelo *Analisador*. Além disso, foi criada uma notação específica para o *Plano de Adaptação*, baseada em uma gramática formal, detalhada no Apêndice A.

Algorithm 2 Adaptation Plan Generation

```
1: Input: k
2: Output: adaptationPlan
3: cli ← {}
4: adb ← {}
5: ml, pl, bl, un, up ← GetLists(k)
6: pendings ← GetPendings(k)
7: for ml, pl, un, up ∈ pendings do
8:     cli ← cli ∪ {mitigate(ml.ip, ml.port, ml.prot)}
9:               ∪ {prevent(pl.port)}
10:              ∪ {block(bl.ip)}
11:              ∪ {notify(un.name, un.ip, un.nvd)}
12:    adb ← adb ∪ {update(up.model, up.manufacturer)}
13: end for
14: adaptationPlan ← cli ∪ adb
15: return adaptationPlan
```

O *Adaptation Plan Generation* (Algoritmo 2) é o algoritmo responsável por criar um *Plano de Adaptação* com base nas informações extraídas da *Base de Conhecimento* (k). O (Algoritmo 2 processa ações ainda não executadas (*pendings*) relacionadas à mitigação (ml), prevenção (pl), bloqueio (bl), notificação (un) e atualização (up) de *firmware*, organizando-as em dois tipos de adaptação (cli e adb).

Inicialmente, são definidas duas estruturas, *Command-Line Interface* (CLI)[6] para armazenar ações de adaptação paramétrica e *Android Debug Bridge* (ADB)[7] para ações de adaptação composicional (Linhas 3-4). Em seguida, as listas ml, pl, bl, un e up são extraídas do conhecimento por meio da função GetLists(k), reunindo informações sobre vulnerabilidades a serem mitigadas, portas a serem prevenidas, dispositivos a serem bloqueados, usuários que precisam ser notificados e dispositivos que devem ser atualizados (Linha 5).

Com as informações carregadas, o algoritmo obtém a lista de ações pendentes por meio da função GetPendings(k), garantindo que apenas medidas ainda não aplicadas sejam consideradas (Linha 6). A iteração sobre ações pendentes (pendings) permite que cada ação seja processada e adicionada à estrutura correspondente. Ações de mitigação, prevenção, bloqueio e notificação são adicionadas à estrutura CLI, enquanto atualizações de firmware são armazenadas em ADB (Linhas 7-13). No final da execução, as duas listas são combinadas para formar o *Plano de Adaptação* (*adaptationPlan*) (Linha 14), retornado como saída do algoritmo (Linha 15).

Esse processo possibilita que todas as ações necessárias para responder às vulnerabilidades detectadas sejam organizadas de forma estruturada, permitindo que a UP-Home aplique medidas de segurança da *smart home* execute as adaptações apropriadas conforme definido pelo *Planejador*.

Assim, O *Planejador* cria o plano de adaptação com base no diagnóstico fornecido pelo *Analisador*. Este plano inclui medidas específicas para mitigar as vulnerabilidades detectadas, como reconfiguração de parâmetros de segurança ou atualização de software. O *Planejador* assegura que essas medidas sejam implementadas, de modo que qualquer interrupção nas operações dos dispositivos seja conhecida do usuário. Isso pode considerar preferências do usuário em relação a adaptações preventivas de segurança, como ampliar as reconfigurações de parâmetros para outros dispositivos ou efetuar bloqueios completos do dispositivo da *smart home*. Desse modo, pode-se responder reativamente ao detectar uma vulnerabilidade, ou agir preventivamente, aplicando proativamente ações a outros dispositivos da *smart home*.

[6]https://learn.microsoft.com/pt-br/appcenter/cli/
[7]https://developer.android.com/studio /command-line/adb?hl=en

4.3.4 Executor

As ações definidas no *Plano de Adaptação* são executadas pelo *Executor*. No caso da UP-Home, isso significa que as correções e ajustes na pilha de software são aplicados de forma automática, respondendo continuamente às ameaças de segurança. Por exemplo, o *Executor* pode aplicar mudança de parâmetros ou atualizar o *firmware* ou sistema operacional dos dispositivos, seguindo o que foi estabelecido no plano de adaptação. A Figura 4.7 apresenta o diagrama de sequência UP-Home.

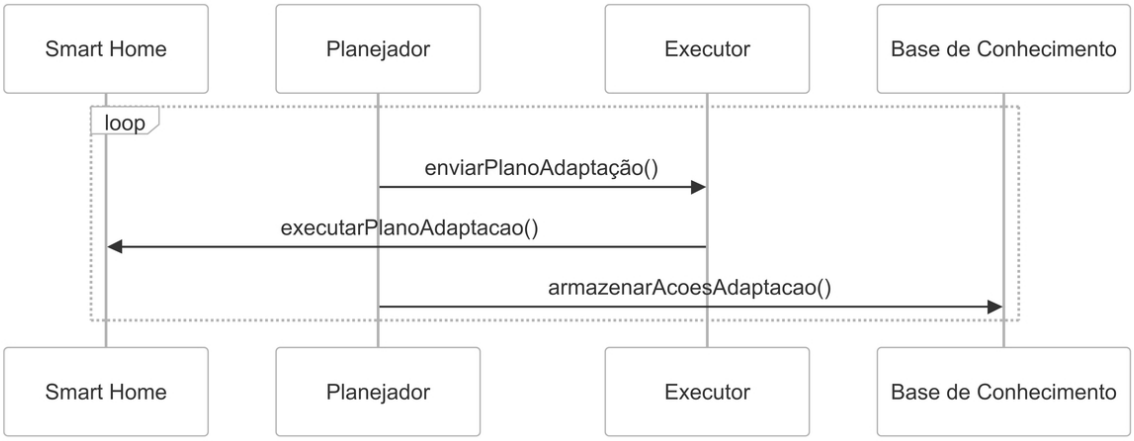

Figura 4.7 – Diagrama de sequência do *Executor*

A Figura 4.7 apresenta o diagrama de sequência do *Executor* da UP-Home , que descreve a interação entre os componentes principais do sistema durante a execução do *Plano de adaptação* na *smart home*: Smart Home, Planejador, Executor e Base de Conhecimento.

O processo é iniciado com o *Planejador*, que envia um Plano de adaptação ao Executor (Enviar plano adaptação). O Executor, ao receber o plano, inicia a aplicação das mudanças na *smart home* (Smart Home) Com a execução do *Plano de Adaptação* (Executar Plano Adaptação) são feitas diversas ações, como mitigação, notificação de vulnerabilidades, bloqueio de dispositivos ou atualização de dispositivos.

Paralelamente à execução, o Planejador também interage com a *Base de Conhecimento* (Armazenar ações Adaptações), para registrar as adaptações realizadas. Esse armazenamento é significativamente importante para manter um histórico das ações aplicadas, permitindo que futuras decisões sejam tomadas com base no conhecimento acumulado.

O processo descrito ocorre dentro de um *loop* contínuo, indicando que a adaptação é cíclica e pode ser repetida sempre que novas vulnerabilidades ou mudanças no ambi-

ente forem detectadas. Essa abordagem busca garantir que a segurança da *smart home* seja melhorada, de forma dinâmica e responsiva, adaptando-se constantemente às novas condições e mantendo um nível adequado de proteção.

4.3.5 Base de Conhecimento (*Knowledge*)

. A *Base de Conhecimento* mantém um registro contínuo dos dados monitorados, da lista de dispositivos, das ações realizadas e de outras informações relevantes durante todo o processo de monitoramento, análise, planejamento e execução. Esse repositório central permite o acúmulo de conhecimento ao longo do tempo, aprimorando a capacidade de adaptação do sistema, *e.g.,* armazenando a relação de IPs, portas e protocolos dos dispositivos que tiveram vulnerabilidades mitigadas. Além disso, a *Base de Conhecimento* armazena dados históricos sobre as vulnerabilidades detectadas, as ações tomadas e os resultados obtidos. No contexto de *smart homes*, ela pode incluir informações sobre padrões de ataque comuns, configurações aplicadas e respostas bem-sucedidas a incidentes anteriores, fortalecendo a resiliência e a eficácia das adaptações futuras.

4.4 IMPLEMENTAÇÃO

A implementação da UP-Home é composta por diversos componentes interconectados que colaboram na adaptação da pilha de software dos dispositivos. Esta seção apresenta detalhes da implementação e como as tecnologias utilizadas estão relacionadas com cada elemento da arquitetura e como foram implementadas na prática.

4.4.1 Implementação do MAPE-K

A implementação da UP-Home é estruturada com base nos componentes do modelo MAPE-K (Seção 4.1), e opera coordenadamente para responder às ameaças de segurança que surjam na *smart home*.

O componente *Monitor* da UP-Home executa três formas de monitoramento, sendo a primeira voltada para a geração de relatórios de penetração por meio da *Application Programming Interface* (API) GMP[8] (*Greenbone Management Protocol*) do *scanner* Open-

[8]https://docs.greenbone.net/API/GMP/gmp-21.4.html

VAS. A integração entre o OpenVAS e a UP-Home é realizada por *scripts* em Python, os quais automatizam as varreduras regulares e a extração dos dados dos relatórios. Esses *scripts* analisam os resultados das varreduras e os enviam ao componente *Analisador*, agilizando a detecção e resposta à novas ameaças. Além de oferecer uma visão aprofundada e atualizada da segurança da *smart home*, um monitoramento inicial cria artefatos fundamentais para realizar adaptações composicionais e paramétricas. O Código Fonte 1 é a implementação do *script* responsável pelo monitoramento. Assim, os relatórios do OpenVAS são entregues conforme solicitação da UP-Home, em formato XML, com *tags* (*e.g., reports, results, nvt, host, port, cvss_base*) das vulnerabilidades detectadas.

Código Fonte 1 – Monitoramento OpenVAS

```
1  from gvm.connections import UnixSocketConnection
2  from gvm.protocols.gmp import Gmp
3  from gvm.transforms import EtreeTransform
4  connection = UnixSocketConnection()
5  transform = EtreeTransform()
6  def process_reports(gmp):
7      response = gmp.get_reports()
8      for report in response.xpath('report'):
9          id_report = report.xpath('@id')[0]
10         process_results(id_report, gmp)
11 def process_results(id_report, gmp):
12     report = gmp.get_report(report_id=id_report)
13     for results in report[0][8].xpath('results'):
14         for result in results.xpath('result'):
15             nvt_id =result.xpath('nvt/@oid')[0]
16             for nvt in result.xpath('nvt'):
17                 name = nvt.xpath('name/text()')[0]
18                 ip=result.xpath('host/text()')[0]
19                 port = result.xpath('port/text()')[0]
20                 cvss_base = nvt.xpath('cvss_base/text()')[0]
21                 refs = nvt.xpath('refs/ref/@id')
22                 if float(cvss_base) > 0:
23             vulns.append({"id_report": id_report, "name": name, "ip": ip, "port":
                   ↪  port, "cvss_base": cvss_base, "refs": refs})
24 with Gmp(connection, transform=transform) as gmp:
25     gmp.authenticate("user", "password")
26     process_reports(gmp)
```

O Código Fonte 1, desenvolvido em Python, realiza o monitoramento de vulnerabilidades na *smart home* por meio do OpenVAS, utilizando a biblioteca *Greenbone Vulnerability Management* (GVM) para processar os relatórios de varreduras. A execução é iniciada no bloco principal (Linhas 24-25), onde é feita a autenticação no GVM (*gmp.authenticate*

("user", "password")), permitindo acesso aos relatórios da varredura. A conexão é estabelecida por meio da biblioteca *UnixSocketConnection()*, enquanto a *EtreeTransform()* converte as respostas do protocolo *Greenbone Management Protocol* (GMP) em uma estrutura XML manipulável (Linhas 1-5). Com a conexão estabelecida, é chamada a função responsável por coletar todos os relatórios disponíveis (Linha 26).

Para obter os relatórios de vulnerabilidades, foram criadas duas funções, uma para obter a lista de relatórios (*process_reports(gmp)*, Linhas 6-10) e outra para obter a lista de vulnerabilidades de cada relatório listado na função anterior (Linhas 11-23). Assim, para cada relatório encontrado, seu identificador (*id_report*, Linha 10) é extraído e passado para a função que processa os resultados (*process_results(id_report, gmp)*), responsável por processar os resultados detalhados de cada relatório de vulnerabilidades.

Assim, ao percorrer os relatórios da varredura, cada item é verificado, e informações como nome da vulnerabilidade (name), endereço IP (ip), porta de comunicação (port), escore CVSS (cvss_base) e identificador do NVD (*refs*), utilizando expressões XPath para navegar na estrutura do relatório XML (Linhas 13-21). Se o valor de cvss_base (Linha 22) for maior que zero, a vulnerabilidade é adicionada à lista *vulns* (Linha 23) como um dicionário contendo o *id_report*, nome da vulnerabilidade, IP, porta, escore CVSS e referências associadas. A verificação do escore da vulnerabilidade (Linha 22), aborda uma importante meta de segurança para a *smart home*. Qualquer vulnerabilidade com escore CVSS maior que zero pode causar impacto, mesmo que mínimo, portanto, deve ser mitigada.

O segundo monitoramento é realizado através da API do *gateway Home-assistant* HA[9], uma interface que permite a interação e o controle de dispositivos da *smart home* usando *scripts* Python. Esta API é um componente essencial para UP-Home acessar dados da *smart home* por meio do HA. Para isso, é feita uma requisição ao *gateway* HA, que retorna informações nas *tags* (*e.g., model, manufacturer, identifiers, sw_version*), que são catalogadas e enviadas para o *Analisador*. O Código Fonte 2 descreve a implementação do monitoramento com HA na UP-Home.

Código Fonte 2 – Monitoramento *Home-assistant*

```
1 import os, json, re
2 result = os.popen("hass-cli --server http://<IP>:8123 --output json --token <
    ↪ token> --columns=id,model,manufacturer,sw_version,identifiers[0].[0] device
    ↪  list").read()
```

[9]https://www.home-assistant.io/

```
3  device_list = json.loads(result)
4  for device in device_list:
5      sw_version = device.get('sw_version')
6      if not sw_version or not re.search(r'\d', str(sw_version)):
7          continue
8      devices.append({
9          "model": device.get('model'),
10         "manufacturer": device.get('manufacturer'),
11         "sw_version": device.get('sw_version')
12     })
```

O Código Fonte 2 realiza o monitoramento dos dispositivos presentes na casa que estão integrados ao HA. Para isto, é utilizada a API HASS-CLI[10] para coletar informações sobre os dispositivos cadastrados e formata os dados extraídos em uma estrutura manipulável.

A execução começa com a importação de três bibliotecas necessárias (Linha 1), uma para executar comandos no terminal (*os*), outra processa os dados retornados no formato JSON (*JavaScript Object Notation*) (*json*), e a terceira para manipulação de expressões regulares. Em seguida, é feita uma chamada ao HA CLI (hass-cli), para obter a lista de dispositivos (*os.popen(<COMANDO>)*). O comando enviado (Linha 2) especifica o IP do servidor e porta (http://<IP>:8123), o formato de saída (json), a autenticação por token (<token>) e define quais colunas devem ser recuperadas: modelo (*model*), fabricante (*manufacturer*), versão de software (*sw_version*) e identificador do dispositivo (*refs*). O resultado dessa execução é armazenado na variável *result* e convertido para uma estrutura JSON com (*json.loads(result)*, Linha 3).

Ao percorrer a lista de dispositivos retornada (device_list, Linha 4), é verificado para cada um se a versão de software (sw_version) está presente e contém ao menos um número (Linhas 5-6). Essa verificação impede que dispositivos sem versão definida ou com valores inválidos sejam adicionados. Caso os critérios sejam atendidos, um dicionário contendo modelo, fabricante e versão de software é criado e adicionado à lista devices (Linha 8-11).

Esse monitoramento permite que a *smart home* tenha um controle preciso dos dispositivos conectados ao HA, garantindo que informações sobre *firmware* sejam coletadas e posteriormente comparadas com os dados dos fabricantes. Essa abordagem facilita a detecção de dispositivos desatualizados e possibilita ações automáticas dentro do MAPE-K, como sugestões de atualização ou bloqueio de dispositivos vulneráveis.

Dentro deste contexto, o monitoramento busca verificar qual versão do *firmware* está

[10]https://developers.home-assistant.io/docs/api/rest/

em uso nos dispositivos compatíveis (ver Seção 4.2), garantindo que eles estejam atualizados e funcionando seguramente. O monitoramento por meio da API do HA foi crucial para implementar adaptação composicional, ou seja, adaptações que ocorrem devido às mudanças nos componentes da pilha de software. Isso permite que o sistema detecte automaticamente alterações e ajuste a operação da UP-Home para responder às atualizações de *firmware* ou outras alterações críticas nos dispositivos. Essa abordagem ajuda a manter a segurança e a funcionalidade da *smart home*, diminuindo o risco de vulnerabilidades relacionadas às versões desatualizadas de software.

O terceiro monitoramento da UP-Home utiliza um *script* em Python para verificar as páginas de *releases* de fabricantes (OEMs) dos dispositivos catalogados no segundo monitoramento (HA). Com isso, torna-se viável identificar a versão mais atual do *firmware* disponível, fornecendo informações cruciais na aplicação de adaptações composicionais. Para apesentar a implementação desse monitoramento de *releases*, será usado como exemplo a busca no site do fabricante Amazon, como mostrado no Código Fonte 3.

Código Fonte 3 – Monitoramento OEM

```
1  from urllib.request import urlopen, Request
2  from bs4 import BeautifulSoup
3  url = 'https://www.amazon.com/gp/help/customer/display.html?nodeId=
       ↪ GMB5FVUB6REAVTXY'
4  req = Request(url)
5  req.add_header('User-Agent', 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
       ↪ (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36')
6  r = urlopen(req)
7  bs = BeautifulSoup(r.read(), "lxml")
8  data_h = bs.find_all('h2')
9  data_p = bs.find_all('p')
10 for i in range(0-1, (len(data_h)-1)):
11     model = data_h[i].string
12     tag = data_p[i+3].span
13     if tag is not None:
14         extracted_text = tag.get_text(strip=True)
15         oem.append({'model': model, 'manufacturer': 'Amazon' ,'last_update':
               ↪ extracted_text})
```

O Código Fonte 3 realiza o monitoramento de dados de fabricantes (OEM) ao buscar informações sobre dispositivos integrados ao *gateway Home-assistant* HA. Para exemplificar como é feita a extração de informações de modelos de dispositivos, e suas últimas atualizações, foi acessado informações publicadas no site da fabricante Amazon.

A execução começa com a importação das bibliotecas necessárias: *urlopen* e *Request* da

urllib.request[11] para realizar requisições HTTP, e *BeautifulSoup* da *bs4*[12] para processar o *HyperText Markup Language* (HTML) da página (Linhas 1-2). Em seguida é definida a URL da página da Amazon (exemplo usado) contendo informações sobre os dispositivos (Linha 3). Após isto, é criado um objeto *Request*, adiciona um cabeçalho de *User-Agent* para simular uma requisição vinda de um navegador (Linhas 4-5) e faz a requisição HTTP para obter o conteúdo da página (Linha 6). Isso evita bloqueios que podem ser aplicados por servidores Web às requisições automatizadas.

Após obter os dados, a página é processada *BeautifulSoup(r.read(), "lxml")*, Linha 7), permitindo navegar na estrutura HTML. As informações relevantes são extraídas utilizando *find_all(h2)* (Linha 8) para os títulos (*data_h*), que representam os modelos dos dispositivos, e *find_all(p)* (Linha 9) para os parágrafos (*data_p*), onde possivelmente estão os detalhes sobre as atualizações.

O loop principal (Linhas 10-15) percorre os títulos e extrai os modelos de dispositivos. Em cada iteração, o código busca acessar um elemento *()* dentro do parágrafo associado (*data_p[i+3]*) para obter informações sobre a última atualização. Caso esse elemento exista (*tag is not None*, Linha 13), o texto extraído é armazenado na variável *extracted_text*. Por fim, os dados do dispositivo são adicionados à lista *oem* como um dicionário contendo o modelo (model), fabricante (Amazon) e a última atualização (last_update) (Linha 15).

Esse código permite que o sistema obtenha informações de *firmware* e atualizações diretamente da página do fabricante, possibilitando a verificação da obsolescência dos dispositivos integrados ao HA, melhorando assim a segurança da *smart home*.

Finalizada a obtenção de dados nos três monitoramentos, esse conjunto de dados é enviado ao componente *Analisador*. O *Analisador*, detalhado nas Seções 4.1, 4.2 e 4.3, é responsável por analisar e decidir se será feita alguma adaptação. As decisões são tomadas com base nos resultados dos monitoramentos, preferências do usuário e nas metas de segurança. Para isto, o Algoritmo 1 foi implementado em Python, como mostrado no Código Fonte 4.

Código Fonte 4 – Analisador

```
1 preferences = getPreferences()
2 for v in vulns:
3     cvss_base = float(v[1])
```

[11] https://docs.python.org/3/library/urllib.request.html
[12] https://pypi.org/project/beautifulsoup4/

```
4       if v[3] in mark_prevent_all or v[2] in mark_block_one:
5           continue
6       else:
7           mark_prevent_all.append(v[3])
8           mark_mitigate_one.append([v[2], v[3], v[4]], 'pending')
9           violatedGoals = True
10          if 3.9 < cvss_base <= 7:
11              if preferences[1]['notify']:
12                  notify_users(v[0], v[2], v[5])
13          elif 7 < cvss_base <= 10:
14              notify_users(v[0], v[2])
15              if preferences[2]['block']:
16                  mark_block_one.append([v[2]])
17  for d in devices:
18      if not isinstance(d, dict):
19          continue
20      for o in oem:
21          if not isinstance(o, dict):
22              print("Elemento inválido em oem:", o)
23              continue
24          if str(d['model']) == str(o['model']) and d['manufacturer'] == o['
                ↪ manufacturer']:
25              if d['sw_version'] < o['lastUpdate']:
26                  mark_update.append({'model': d['model'], 'manufacturer': d['
                        ↪ manufacturer'], 'status': 'pending'})
27                  violatedGoals = True
```

O Código Fonte 4 do *Analisador* da UP-Home, desenvolvido em Python, tem a função de avaliar vulnerabilidades e verificar atualizações de *firmware*, tomando decisões sobre mitigação, bloqueio e notificações, conforme as preferências do usuário. Para isto, processa duas categorias de informações: vulnerabilidades detectadas (vulns) e dispositivos conectados (devices).

Na primeira parte do código, as preferências do usuário são obtidas (*getPreferences()*) e armazenadas na variável *preferences* (Linha 1). O código percorre a lista de vulnerabilidades (*vulns*, Linhas 2-16), extraindo o escore *cvss_base* e convertendo-o para um valor numérico (*float(v[1])*, Linha 3). Para cada vulnerabilidade, verifica-se se a porta (v[3]) já está na lista de prevenção total (*mark_prevent_all*) ou se o IP do registrado na vulnerabilidade (*v[2]*) já está na lista de bloqueios (mark_block_one) (Linhas 4). Caso esteja, o código simplesmente ignora essa vulnerabilidade, pois já foi mitigada proativamente e segue para a próxima iteração (*continue*, Linha 5). Se a vulnerabilidade ainda não foi mitigada, a porta é adicionada à lista de medidas preventivas (*mark_prevent_all*, Linha 7),

e o IP, porta e protocolo são registrados na lista de mitigação em (*mark_mitigate_one*) com o *status* pendente (*pending*, Linha 8). Em seguida, a variável com *True* (*violatedGoals = True*, Linha 9), indicando que um objetivo de segurança foi violado.

Após essa etapa, o código decide com base no escore CVSS da vulnerabilidade. Se o *cvss_base* estiver entre 3.9 e 7 (Linha 10), verifica-se se as preferências do usuário indicam que ele deseja ser notificado (*preferences[1]['notify']*, Linha 11); caso afirmativo, a função de notificação (*notify_users(v[0], v[2], v[5])*, Linha 12) é chamada. Para vulnerabilidades mais severas (7 < *cvss_base <= 10*, Linha 13), a notificação ao usuário é enviada automaticamente (Linha 14), independentemente das preferências. Além disso, se o usuário tiver ativado a opção de bloqueio automático (preferences[2]['block'], Linha 15), o dispositivo vulnerável é adicionado à lista de bloqueio de dispositivo, que é feita por IP do dispositivo (*mark_block_one*, Linha 16).

Na segunda parte do código, o foco muda para a verificação de atualizações de *firmware*. O código percorre a lista de dispositivos (*devices*, Linhas 17-28), garantindo que cada item seja um dicionário (*isinstance(d, dict)*). Caso contrário, ignora o item (Linhas 18-19). Em seguida o código percorre a lista de fabricantes OEM (*oem*), que contém as informações mais recentes sobre *firmware* disponíveis (Linha 20). Para cada dispositivo encontrado, o código compara modelo e fabricante entre modelo do dipositivo (*devices*) e fabricante (*oem*) (Linha 24). Se houver uma correspondência e a versão do *firmware* do dispositivo (*d[sw_version]*) for inferior à última versão conhecida pelo fabricante (*o['lastUpdate']*) (Linha 25), o dispositivo é registrado na lista de atualização (mark_update) com o status pendente (*pending*), indicando que uma atualização de *firmware* é necessária (Linha 26). Como essa situação também representa uma violação dos objetivos de segurança, *violatedGoals* é definido como *True*.

o Código 4 desempenha um papel fundamental na UP-Home, garantindo que vulnerabilidades sejam identificadas e mitigadas e que os dispositivos da *smart home* permaneçam atualizados e protegidos contra ameaças conhecidas.

Ainda no *Analisador*, existe a participação humana (*Human-In-the-Loop*) para assegurar que serviços indispensáveis (*e.g.,* a abertura de fechaduras inteligentes e câmeras de segurança) não sejam interrompidos indevidamente. A implementação do HIL na UP-Home foi feita para viabilizar a interação humana ativamente das decisões de segurança. Para isso, a UP-Home integra-se com uma API de plataforma de mensagens Telegram, enviando notificações em tempo real ao usuário quando uma vulnerabilidade crítica é de-

tectada. As mensagens contêm informações das vulnerabilidades, seus riscos e possíveis impactos, permitindo que o usuário decida por estender preventivamente as adaptações aos demais dispositivos ou pela aplicação de bloqueio do dispositivo infectado. Caso não haja resposta em um período específico, o módulo de adaptação executa uma ação predefinida para garantir a segurança do ambiente. Por fim, finalizada a análise dos dados, o *Analisador* envia o diagnóstico da segurança da *smart home* ao *planejador*.

O *Planejador*, conforme detalhado na Seção 4.3, recebe dados do *Analisador* e cria um *Plano de Adaptação*. Como descrito no Algoritmo 2, o *Planejador* decide como realizar a adaptação com base nas informações recebidas e, dependendo do tipo de adaptação, executa os comandos CLI e ADB apropriados para o RouterOS ou *Android Virtual Device* (AVD)[13] respectivamente. Uma notação foi desenvolvida para estruturar o plano de adaptação (ver Apêndice A).

O Código Fonte 5 implementa a execução de ações específicas com base em listas de marcações prévias que categorizam dispositivos e vulnerabilidades em diferentes níveis de resposta.

Código Fonte 5 – Planejador

```
def planCreate():
    for i, one in mark_block_one:
        if mark_mitigate_one[i][3] == 'pending':
            executeMitigate(one['ip'], one['port'], one['prot'])
            mark_mitigate_one[i][3] = 'resolved'
    for up in mark_update:
        if mark_update[up]['status'] == 'pending':
            executeUpdate(up['model'], up['manufacturer'])
            mark_update[up]['status'] = 'resolved'
    if mark_prevent_all != []:
        for p in mark_prevent_all:
            executePrevent(p['port'])
    if mark_notify_users != []:
        for n in mark_notify_users:
            execute_notify_users(n['name', n['ip'], n['nvd']])
    if mark_block_one != []:
        for b in mark_block_one:
            executeMitigate(b['ip'])
```

O Código Fonte 5 é responsável por criar o *Plano de Adaptação* da UP-Home, coordenando as ações de mitigação, atualização, prevenção, notificação e bloqueio de dispositivos

[13]https://developer.android.com/studio/run/managing-avds?hl=pt-br

vulneráveis. Ele processa as listas de vulnerabilidades e dispositivos que precisam de intervenção e executa as medidas corretivas apropriadas.

A função de criação do plano de adaptação (*planCreate*, Linha 1) percorre inicialmente a lista de dispositivos a serem bloqueados (*mark_block_one*, Linha 2). Para cada item na lista, verifica se a vulnerabilidade correspondente na lista de mitigação (*mark_mitigate_one*) continua pendente (Linha 3). Se a mitigação ainda não foi realizada, executa a ação de mitigação chamando a função de mitigação (*executeMitigate*) com o endereço IP, porta e protocolo afetado (*one['ip'], one['port']*, Linha 4) e, em seguida, atualiza o *status* da mitigação para resolvida (*resolved*, Linha 5).

Na segunda parte do código, o foco está na atualização de dispositivos. O loop percorre a lista de atualizações pendentes (*mark_update*, Linha 6). Se o status da atualização for pendente (*pending*, Linha 7), a função de atualização (*executeUpdate*) é chamada, aplicando a atualização ao modelo e fabricante do dispositivo vulnerável (*up['model'], up['manufacturer']*, Linha 8). Após a execução da atualização, o status do dispositivo é alterado para (*resolved*, Linha 9), indicando que a ação foi concluída.

A terceira parte do código trata da prevenção de vulnerabilidades. Se a lista de ação preventiva (*mark_prevent_all*) não estiver vazia (Linha 10), o código percorre a lista e executa a ação preventiva (*executePrevent*) para cada porta incluída na política de prevenção (*p['port']*, Linha 12).

A quarta parte do código adiciona a funcionalidade de notificação ao usuário. Se a lista de notificações (*mark_notify_users*) não estiver vazia (Linha 13), o código percorre a lista e executa a função de notificação ao usuário (*execute_notify_users*), enviando informações sobre a vulnerabilidade detectada, incluindo nome, IP e identificador NVD (*n['name'], n['ip'], n['nvd']*, Linha 15).

Por fim, o código verifica se há dispositivos marcados para bloqueio (*mark_block_one*, Linha 16). Se houver, o loop percorre essa lista e executa a função de mitigação (*executeMitigate*), passando o endereço IP do dispositivo para ser bloqueado (*b['ip']*, Linha 18).

Esse *script* organiza a execução de medidas de prevenção, mitigação, bloqueio e atualização de forma sistemática, garantindo uma resposta personalizada para cada situação. Esse código desempenha um papel fundamental no *Planejador* do modelo MAPE-K, consolidando as ações necessárias para garantir a segurança da *smart home*com base nas informações processadas pelo *Analisador* e preparando a execução das adaptações no sistema (*Executor*).

Um exemplo de *Plano de Adaptação* é mostrado a seguir:

```
1 Begin
2 cliMitigate(192.190.30.33, 21, FTP)
3 action(mitigate_one(/ip/firewall/filter/add action=drop
chain=forward dst-address=192.190.30.33 dst-port=21 protocol=TCP))
4 End
```

Neste exemplo de *Plano de Adaptação*, que implementa adaptação paramétrica, ações são repassadas para o *Executor*. Os dados de IP, porta e protocolo da vulnerabilidade detectada são transmitidos à função que mitiga uma vulnerabilidade por vez (Linha 2)). O *Executor* interpreta essas informações e executa o envio do comando ao RouterOS via CLI (Linha 3). Ao receber o comando, o RouterOS cria um filtro no *firewall* para bloquear o acesso ao IP (192.190.30.33) na porta 21, correspondente ao protocolo FTP.

Um outro exemplo de *Plano de adaptação* mais complexo pode ser visto a seguir:

```
1 Begin
2 adbUpdate(Echo Dot (2nd Generation), Amazon)
3 adb_command(launch_app(adb shell monkey -p Amazon
-c android.intent.category.LAUNCHER 1))
4 adb_command(navigate_taps(adb shell input tap 286 2278))
5 adb_command(navigate_taps(adb shell input tap 286 2278))
6 adb_command(navigate_taps(adb shell input tap 558 2193))
7 adb_command(navigate_taps(adb shell input tap 558 615))
8 adb_command(force_stop(adb shell am force-stop Amazon))
9 End
```

Ao detectar vulnerabilidades relacionadas à desatualização do *firmware*, é necessária uma ação imediata por meio de comandos ADB. Essa vulnerabilidade indica que o dispositivo, como o *Echo Dot (2ª geração)* da Amazon, precisa ser atualizado. O modelo e fabricante são repassados ao *Executor*, que interpreta as instruções e inicia uma sequência de comandos ADB para atualizar o dispositivo pela aplicação correspondente. Com o AVD conectado, comandos ADB são usados para abrir a aplicação do fabricante (Linha 3) e acessar as configurações do dispositivo, para iniciar a atualização (Linhas 4-7). Por fim,

o aplicativo é fechado, completando o processo de atualização do componente da pilha de software (Linha 8).

O componente *Executor* é responsável por executar o plano de adaptação definido pelo *Planejador*, utilizando as interfaces de comando do RouterOS por meio de comandos CLI e do emulador AVD por comandos ADB. O Código Fonte 6 apresenta implementa o *Executor*.

Código Fonte 6 – Executor

```
1  from librouteros import connect
2  api = connect(username=<USER>, password=<PASSWORD>, host=<IP_ROUTEROS>, port
       ↪ =8728)
3  def start(exec):
4      for e in exec:
5          print(e)
6  def executeMitigate(ip, port, protocol):
7      exec = api(cmd="/ip/firewall/filter/add", **{
8          "action": "drop",
9          "chain": "forward",
10         "src-address": "0.0.0.0/0",
11         "dst-address": ip,
12         "dst-port": port,
13         "protocol": protocol,
14         "comment": "Bloqueio_Prevent"+ip+"_"+port+"_"+protocol
15         })
16     start(exec)
17 def executePrevent(port):
18     exec = api(cmd="/ip/firewall/filter/add", **{
19         "action": "drop",
20         "chain": "forward",
21         "src-address": "0.0.0.0/0",
22         "dst-address": "0.0.0.0/0",
23         "comment": "Bloqueio_Prevent"+port,
24         "dst-port": port
25     start(exec)
26 def executeBlock(ip):
27     exec = api(cmd="/ip/firewall/filter/add", **{
28         "action": "drop",
29         "chain": "forward",
30         "src-address": "0.0.0.0/0",
31         "dst-address": ip,
32         "comment": "Bloqueio_"+ip})
33     start(exec)
34 def executeUpdate(model, manufacturer):
35     from time import sleep
36     import os
```

```
37      package = 'com.amazon.dee.app'
38      if manufacturer in package:
39          os.system("adb shell monkey -p com.amazon.dee.app -c android.intent.
                ↪ category.LAUNCHER 1")
40          sleep(1)
41          os.system("adb shell input tap 909 669")
42          sleep(1)
43          os.system("adb shell input tap 504 2081")
44  def execute_notify_users(name, ip, nvd):
45      import requests
46      message = 'Vulnerability found: '+ str(name) + 'in '+ str(ip) + ':' + nvd
47      url = f'https://api.telegram.org/<TOKEN>/sendMessage?chat_id=<CHAT_ID>&text={
            ↪ message}'
48      response = requests.post(url)
49      url = f'https://api.telegram.org/<TOKEN>/getUpdates'
50      response = requests.post(url)
51      print(response.json())
```

O Código Fonte 6 implementa o *Executor* da UP-Home, sendo responsável pela aplicação das ações definidas no *Plano de Adaptação*, incluindo mitigação, prevenção, bloqueio, atualização e notificação de vulnerabilidades. O código estabelece uma conexão com um dispositivo RouterOS, utilizando a biblioteca *librouteros*[14] (Linha 1) para enviar comandos de *firewall*, além de executar ações diretamente em dispositivos via ADB e enviar notificações ao usuário via API do Telegram.

A conexão com o RouterOS é estabelecida na inicialização (connect, Linha 2), utilizando credenciais de acesso (*username, password*) e o endereço IP do roteador (*host*, Linha 2). A função de início da execução (*start*, linha 3-5) foi adicionada para simplificar a execução de comandos, percorrendo a resposta da execução e imprimindo os resultados (Linha 5). Essa função (*start*) é responsável por enviar os comandos ao RouterOS.

A função de execução de mitigação (*executeMitigate*, Linhas 6-10) adiciona uma regra de *firewall* para bloquear tráfego de qualquer IP de origem (*src-address*) a um IP específico (*dst-address*) e porta (*dst-port*) e protocolo (*protocol*), definindo um comentário identificando o bloqueio (*comment*). Após enviar o comando ao roteador, a função *start* é chamada para processar a resposta da execução (Linha 16).

A executar de prevenção (*executePrevent*, Linhas 17-24) adiciona uma regra de *firewall* para bloquear todo acesso a uma porta específica, impedindo tráfego de entrada e saída nessa porta (*dst-port*). Assim como na mitigação, a função *start* é utilizada para processar

[14]https://github.com/luqasz/librouteros

a execução do comando (Linha 25).

A execução de bloqueio (*executeBlock*, Linhas 26-33) bloqueia completamente um dispositivo com base no seu endereço IP (*dst-address*), criando uma regra no firewall para impedir qualquer tráfego associado ao IP informado e chamando *start* para processar a execução (Linha 33).

A execução de atualização (*executeUpdate*, Linhas 34-43) realiza a atualização de *firmware* em dispositivos específicos, *e.g.,* Amazon, utilizando o comando ADB (Android Debug Bridge). O código importa as bibliotecas *sleep* e os para manipulação de comandos e pausas durante a execução (Linhas 40, 42). Se o fabricante do dispositivo estiver dentro da lista de pacotes conhecidos (*package*, Linha 38), comandos ADB são enviados para iniciar a atualização de um aplicativo da Amazon (*adb shell monkey -p com.amazon.dee.app*, Linha 31). Toques na interface são simulados com coordenadas pré-determinadas (*adb shell input tap*, Linha 41-43), permitindo a execução do processo de atualização.

A execução de notificação (*execute_notify_users*, Linhas 44-50) envia notificações ao usuário informando sobre a detecção de vulnerabilidades na rede da casa. A notificação contém o nome da vulnerabilidade (*name*), o endereço IP do dispositivo afetado (*ip*) e a referência NVD associada (*nvd*, Linha 46). A mensagem é enviada utilizando a Telegram API, através de um HTTP *POST* contendo o *TOKEN* de autenticação e o ID do *chat* do usuário (*url*, Linhas 47-48). O código também obtém informações do status da API (*getUpdates*, Linha 50) e imprime a resposta recebida (*print*(*response.json()*, Linha 51).

O código 6 é essencial para a fase de execução do modelo MAPE-K, aplicando as medidas planejadas pelo *Planejador*, garantindo a proteção da casa contra ameaças detectadas, automatizando processos de mitigação e permitindo notificações ao usuário.

4.4.2 Tecnologias utilizadas

O desenvolvimento da UP-Home foi realizado com soluções de código aberto. A Tabela 4.3 relaciona as tecnologias adotadas na implementação.

OpenVAS. Com esta ferramenta, foi possível realizar varreduras de segurança, também conhecidas como testes de penetração, para identificar vulnerabilidades na rede da *smart home*. OpenVAS é uma ferramenta de código aberto que faz análise de vulnerabilidades e fornece informações detalhadas sobre o escore das vulnerabilidades com base no CVSS. Na UP-Home, o *gateway* OpenVAS foi utilizado no componente *Monitor*.

Tabela 4.3 – Tecnologias utilizadas

Tecnologia	**Função**
OpenVAS	*Scanner*
Home-assistant (HA)	*Gateway*
RouterOS	*Gateway*
Beautiful Soup	Biblioteca Python
AVD	Emulador
ADB	API
CLI	API
Telegram	API

A base de dados de vulnerabilidades utilizada pelo OpenVAS é atualizada diariamente para refletir as ameaças mais recentes e suas respectivas soluções. Com isso, as vulnerabilidades são categorizadas com base em seus níveis de criticidade, permitindo uma abordagem estratégica na tomada de decisões em relação à adaptação de segurança. OpenVAS executa diferentes tipos de testes, desde a detecção de portas abertas até a identificação de falhas específicas em software e *firmware* de dispositivos conectados. Desse modo, OpenVAS pode identificar vulnerabilidades conhecidas e fornecer recomendações para mitigação, ajudando a fortalecer a segurança da *smart home*. Os resultados dessas varreduras são fornecidos em relatórios detalhados, que incluem descrições, criticidade e possíveis impactos correspondentes a vulnerabilidades. Os relatórios gerados pelo OpenVAS podem ser utilizados para auditorias de segurança, fornecendo evidências de que a rede está sendo monitorada e protegida adequadamente. Esses relatórios são obtidos e analisados pela UP-Home, permitindo a detecção e priorização de ameaças. Com essas informações, medidas corretivas puderam ser implementadas, de modo a reduzir o risco de exploração de falhas de segurança. Com o monitoramento contínuo do OpenVAS, é possível identificar e responder a incidentes de segurança em tempo real.

Home-Assistant. HA foi utilizado para integrar os diversos dispositivos presentes na *smart home*. Assim, o HA oferece integração e controle centralizados para dispositivos IoT. Na UP-Home, foi possível usar esse *gateway* para monitorar informações de segurança, incluindo o monitoramento das versões de *firmware* dos dispositivos. O HA possui uma arquitetura modular que permite a adição e remoção de integrações com os dispositivos da *smart home*, e suporta a criação de automações complexas usando *scripts* YAML (*Yet Another Markup Language*) ou uma interface visual. O HA foi instanciado em um

contêiner Docker, executado no sistema operacional Raspberry Pi OS, criando um ambiente de teste acessível e de baixo custo. Assim, o HA gerencia usuários e permissões para garantir que apenas pessoas autorizadas possam controlar ou monitorar dispositivos, e utiliza criptografia para proteger dados. Como o HA é compatível com vários dispositivos, foi possível integrá-lo com os assistentes de voz Alexa (Amazon)[15] e *Google Assistant* (Google)[16], bem como outras plataformas de automação, como IFTTT (If This, Then That) (KUMER et al., 2021).

Beautiful Soup. Esta biblioteca em Python é projetada para extrair dados de arquivos HTML e *Extensible Markup Language* (XML). *Beautiful Soup* foi usada na UP-Home para obter informações dos *releases* de *firmware* disponibilizados na Internet pelos fabricantes dos dispositivos IoT. Esses dados foram transformados em árvore de objetos Python, permitindo a manipulação e busca de elementos específicos, como *tags*, atributos e texto. A biblioteca é amplamente usada para *Web scraping*, permitindo que desenvolvedores acessem o conteúdo de páginas da *Web* para coletar informações automaticamente.

RouterOS. Este *gateway* permitiu a gestão das conexões dos dispositivos da *smart home*, atuando no gerenciamento da rede da *smart home*, e consequentemente provendo a função de *firewall*. O RouterOS possibilita a gestão do acesso à rede, permitindo a mitigação de vulnerabilidades por meio do bloqueio de dispositivos comprometidos, com a criação de filtros, de modo a bloquear IP, portas e protocolos específicos. Dessa forma, é possível usá-lo no *Executor* para mitigar ataques direcionados a serviços sensíveis, como FTP, SSH e Telnet. Em situações críticas, o RouterOS pode aplicar o bloqueio total de dispositivos comprometidos na rede da *smart home*.

ADB e AVD. ADB (*Android Debug Bridge*) é uma ferramenta usada para comunicação com dispositivos Android, permitindo a execução de operações, como instalação e atualização de aplicativos e transferência de arquivos. Na UP-Home, os comandos ADB (*e.g., adb shell input tap 452 323*) foram utilizados para atualizar componentes em dispositivos da *smart home*. Em conjunto com o ADB, foi utilizado o AVD (*Android Virtual Device*) para emular dispositivos Android em uma máquina virtual. O uso de AVD permitiu criar ambientes de testes isolados, onde novas configurações e atualizações puderam ser experimentadas sem riscos para os dispositivos reais. Desse modo, as aplicações que controlam os dispositivos presentes na *smart home* puderam ser acessadas para executar a atualização

[15]https://www.alexa.com/
[16]https://assistant.google.com/

de *firmware*. A execução de comandos diretamente em *smartphones* ou outros dispositivos Android permitiu que fossem executadas atualizações em cenários reais, assegurando que os dispositivos IoT operassem corretamente após a implementação das mudanças.

CLI. A CLI é uma interface de linha de comando utilizada para configurar e gerenciar dispositivos por meio da rede, *e.g.,* RouterOS. Na UP-Home, a CLI foi empregada pelo componente *Executor* para se comunicar com o *gateway* RouterOS. Essa interface é acessível por diversos métodos, como terminal local, SSH, Telnet ou console serial, permitindo que administradores interajam diretamente com o dispositivo de rede. Dessa forma, a CLI do RouterOS proporcionou acesso completo às funcionalidades do *gateway*, incluindo configuração de roteamento, *firewall*, *Network Address Translation* (NAT), controle de banda, *Virtual Private Network* (VPN), monitoramento de tráfego e gerenciamento de interfaces de rede.

Ferramentas e Scripts Utilizados. A UP-Home utiliza a API de acesso ao *Telegram* em conjunto com *scripts* Python, comandos CLI e automações *YAML Ain't Markup Language* (YAML) no HA para coordenar as funções dos diversos componentes. A API do *Telegram* é empregada para enviar notificações ao usuário sobre vulnerabilidades detectadas e ações realizadas pelo sistema. Esses *scripts* são responsáveis por executar varreduras de segurança, aplicar adaptações e gerenciar a comunicação com o usuário, possibilitando um ambiente seguro e responsivo.

4.5 CONSIDERAÇÕES FINAIS

Neste capítulo, detalhou-se a concepção e o desenvolvimento da UP-Home, abordando desde seus princípios fundamentais até os aspectos específicos de sua arquitetura, projeto e implementação. Inicialmente, na Seção 4.1 foram discutidos os conceitos básicos da UP-Home. Em seguida, na Seção 4.2 foi apresentada a arquitetura da ferramenta, que integra componentes essenciais como o *Monitor*, *Analisador*, *Planejador* e *Executor*, bem como a *Base de Conhecimento*. A Seção 4.3 apresentou o projeto da UP-Home , as interações realizadas entre os componentes por meio de diagramas de sequência e os algoritmos desenvolvidos. A Seção (4.4) apresentou a implementação prática desses conceitos, com a combinação de automação e intervenção humana (*Human-In-the-Loop*). Por fim, a Seção 4.4.2 destacou as tecnologias utilizadas na UP-Home.

5 AVALIAÇÃO

Este capítulo apresenta uma avaliação do funcionamento da UP-Home na prática, destacando seu impacto no monitoramento da segurança de uma *smart home*. A estrutura deste capítulo está organizada da seguinte maneira: inicialmente, a Seção 5.1 apresenta os objetivos da avaliação. Em seguida, a Seção 5.2 aborda a avaliação dos mecanismos de adaptação, analisando as adaptações paramétricas e composicionais. Na Seção 5.3, é discutida a atuação da UP-Home na mitigação e redução do número de vulnerabilidades detectadas em *smart homes*, destacando seu impacto na melhoria da segurança. Posteriormente, a Seção 5.4 detalha os experimentos que medem o tempo para execução de adaptações pela UP-Home. A Seção 5.5 apresenta a avaliação da interação da UP-Home com o conceito de *Human-In-the-Loop*, explorando como a participação humana pode auxiliar a ferramenta proposta em cenários complexos.

5.1 OBJETIVOS

A avaliação da UP-Home tem três objetivos principais: avaliar a capacidade da ferramenta em reduzir as vulnerabilidades de segurança de uma *smart home*; avaliar o tempo gasto no processo de adaptação; e entender o impacto da participação humana no processo de adaptação.

O primeiro objetivo foca na capacidade da UP-Home reduzir efetivamente a quantidade de vulnerabilidades de uma *smart home*. Essa análise é importante, pois permite quantificar a melhoria de segurança proporcionada pela UP-Home. Os experimentos relacionados a este objetivo envolvem a identificação e mitigação de ameaças, permitindo avaliar como UP-Home lida com diferentes tipos de vulnerabilidades.

O segundo objetivo consiste em medir o tempo de adaptação (paramétrica) da UP-Home, em resposta ao aumento de vulnerabilidades injetadas. Esta avaliação permite analisar a atuação da UP-Home em casos mais severos, onde múltiplas vulnerabilidades são detectadas e requerem mitigação rápida. Além disso, foi avaliado o tempo de adaptação (composicional) da UP-Home quando são detectados dispositivos com *firmwares* desatualizados. O tempo de detecção e mitigação de vulnerabilidades é um aspecto crítico para assegurar a proteção da *smart home*, sendo fundamental compreender como a

ferramenta responde em tempo hábil às ameaças.

O terceiro e último objetivo é avaliar a atuação da UP-Home em conjunto com o *Human-In-the-Loop*. Esse experimento inclui a interação com três diferentes perfis de usuários(as), destacando a importância do elemento humano em situações adversas que envolvem vulnerabilidades mais críticas. A abordagem visa ilustrar como o suporte humano contribui para a ferramenta, especialmente em cenários que requerem intervenção humana para decisões mais complexas.

Para alcançar esses objetivos, um cenário real de uma *smart home* foi utilizado em todos os experimentos, conforme mostrado na Figura 5.1.

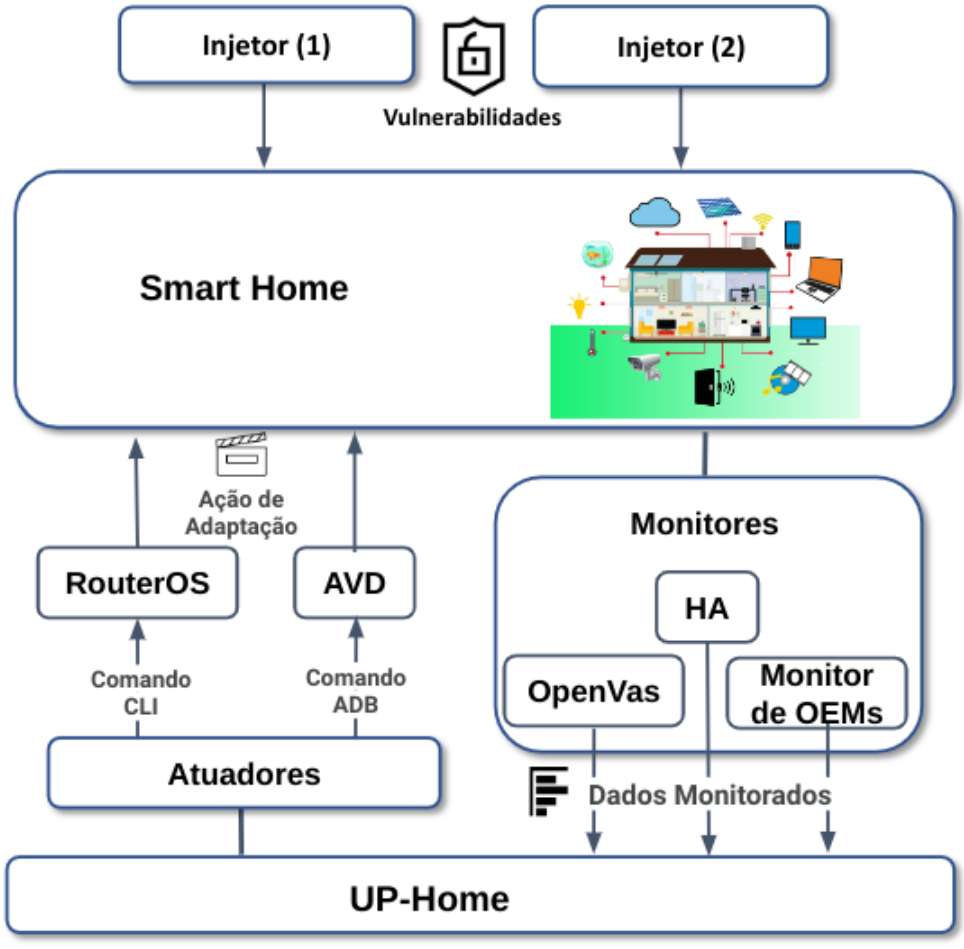

Figura 5.1 – Configuração usada na avaliação da UP-Home

Nesse cenário, a *smart home* inclui dispositivos conectados e sistemas integrados que podem estar sujeitos a diferentes tipos de vulnerabilidades de segurança. A *smart home* possui 33 dispositivos IoT, detalhados na Tabela 5.1.

Quantidade	**Dispositivo**	**Fabricante**
7	Câmeras de segurança	Intelbras (5), V380 (2)
6	Controladores de iluminação	Sonoff (4), Tuya (2)
4	Controladores de geração solar	Apsystems
4	*Smartphones*	Xiaomi (2), Samsung (1), Apple (1)
3	Notebooks	Samsung (2), Dell (1)
3	*Smart TVs*	Samsung (1), Sony (1), AOC (1)
1	Assistente de voz	Amazon
1	Alimentador do aquário	Ilonda
1	Controlador infravermelho	Tuya
1	Fechadura eletrônica	Tuya
1	Pi 3	Raspberry
1	*Routerboard*	Mikrotik

Tabela 5.1 – Dispositivos conectados da residência

No cenário da UP-Home está integrado o ambiente de *smart home*, destacando os diferentes componentes, seus papéis e as interações entre eles.

Na configuração, é importante destacar a presença de dois injetores de vulnerabilidade que desempenham um papel fundamental na avaliação da UP-Home. Esses injetores de vulnerabilidade atuam como dispositivos ou componentes específicos projetados para criar intencionalmente condições vulneráveis (vulnerabilidades) no ambiente da *smart home*. A introdução controlada de vulnerabilidades é uma prática importante em testes de segurança e visa avaliar a capacidade de resposta do sistema à exposição de ameaças. Os testes realizados por esses injetores de vulnerabilidade têm diversas finalidades, incluindo avaliar as medidas de segurança existentes e identificar vulnerabilidades potenciais que precisam ser corrigidas. É importante salientar que os testes de penetração, nos quais esses injetores de vulnerabilidade se enquadram, são práticas usadas na avaliação de segurança de sistemas e redes (SHEBLI; BEHESHTI, 2018) (VATS; MANDOT; GOSAIN, 2020). Com isso, os injetores simulam cenários de ataque de fontes maliciosas para avaliar a capacidade do sistema de defesa em identificar, responder e mitigar vulnerabilidades.

No monitoramento da *smart home*, são usados o *gateway* HA e o *scanner* OpenVAS. O HA é utilizado para a integração dos dispositivos da *smart home*, onde são coletadas informações sobre o *firmware* dos dispositivos da *smart home*. Já o *scanner* OpenVAS faz varreduras e fornece relatórios de vulnerabilidades detectadas na *smart home*. Para isto, OpenVAS realiza testes de penetração, como análise de portas abertas, verificação

de protocolos inseguros, avaliação de softwares desatualizados e exposição a vulnerabilidades conhecidas, como as listadas no CVE. Há ainda o Monitor de OEMs, que monitora atualizações e vulnerabilidades específicas de *firmwares* nos dispositivos da casa. Ambos (HA e OpenVAS) fornecem informações usadas para que adaptações possam ser iniciadas, caso haja violação de alguma meta de segurança.

Outro *gateway* utilizado no sistema é o RouterOS, responsável pela gestão das conexões dos dispositivos da *smart home*. Esse componente é configurado via comandos CLI para alterar parâmetros do *firewall*, permitindo a mitigação de vulnerabilidades identificadas. Já as atualizações de componentes são realizadas por meio de comandos ADB enviados a um AVD. Esses comandos acionam a aplicação correspondente ao dispositivo desatualizado, iniciando o processo de atualização automaticamente.

5.2 MÉTRICAS UTILIZADAS

Para fundamentar a avaliação da ferramenta UP-Home discutida nesta tese, foram estabelecidas métricas com o objetivo de quantificar o aumento da segurança em *smart homes*. Com base em práticas estabelecidas na área de sistemas autoadaptativos (SALEHIE; TAHVILDARI, 2009), (KEPHART; CHESS, 2003), este capítulo de avaliação focará nos parâmetros a seguir:

- O número de vulnerabilidades presentes na *smart home* serve como um indicador primário da capacidade de detectar e mitigar fraquezas de segurança ao longo do tempo. O monitoramento da redução no número de vulnerabilidades identificadas, detectadas por meio de ferramentas de teste de penetração, após a implantação e operação da solução, quantificará seu impacto na postura geral de segurança.

- O tempo de mitigação, medido em segundos, é uma métrica crítica para avaliar a capacidade de resposta da UP-Home no tratamento de diferentes números de vulnerabilidades. Isso será avaliado tanto para adaptação paramétrica, que envolve o ajuste de parâmetros do sistema (*e.g.,* bloqueio de acesso a IPs e portas específicas) quanto para adaptação composicional, que envolve a atualização de componentes de software como *firmware*. Um tempo de mitigação reduzido sugere que a ferramenta aprimorou a segurança.

- O tempo de mitigação assistida é considerado para entender a influência da intervenção humana (HIL). Essa métrica incorporará o tempo gasto para interação do usuário, como responder a notificações sobre vulnerabilidades críticas, no processo de adaptação de segurança, destacando o aspecto colaborativo da solução de segurança proposta.

Essas métricas visam avaliar a redução de vulnerabilidades, o tempo de adaptação e a eficácia da integração HIL. Ainda que outras métricas relevantes sejam levadas em conta nas avaliações de segurança da IoT, como a gravidade das vulnerabilidades (frequentemente avaliadas por meio de escore CVSS) e a eficácia das medidas de mitigação, a avaliação da UP-Home se concentra nesses três indicadores principais. A seleção dessas métricas é informada pelo cenário mais amplo de práticas de avaliação de segurança de IoT (ABDULLA et al., 2020), (HAMMI et al., 2022), e padrões de segurança (ETSI, 2019b).

Em todos os experimentos, analisou-se tanto o ambiente real da casa quanto a introdução intencional de vulnerabilidades por meio dos injetores, ambas detectadas em varreduras de vulnerabilidade com o OpenVAS. Durante os experimentos, UP-Home é observada em ação, permitindo avaliar como a ferramenta detecta essas vulnerabilidades (por meio do *Monitor*) e toma decisões com base nessa detecção (através do *Analisador*). Se a ferramenta UP-Home identifica a necessidade de uma adaptação de segurança, os outros componentes da ferramenta (*Planejador* e *Executor*) entram em ação.

Para alcançar o primeiro objetivo, utilizou-se a métrica do número de vulnerabilidades na *smart home*. Sempre que ocorriam mudanças significativas na rede, como a inserção de dispositivos, uma nova varredura era realizada pelo OpenVAS. Assim, a cada varredura, as vulnerabilidades detectadas eram catalogadas e mitigadas. Com isso, foram feitas dez varreduras durante os experimentos para verificar se o número de vulnerabilidades diminuía.

A avaliação dos mecanismos de adaptação foi dividida em duas partes principais: adaptação paramétrica e adaptação composicional. A adaptação paramétrica refere-se à capacidade da UP-Home de ajustar seus parâmetros operacionais em resposta ao aumento do número de vulnerabilidades detectadas. Esse mecanismo é importante em cenários em que novas vulnerabilidades são injetadas deliberadamente, para observar como a ferramenta reage em situações de carga elevada. Por outro lado, a adaptação composicional avalia a capacidade da UP-Home fazer a atualização de diferentes componentes, especial-

mente em casos em que dispositivos com *firmware* desatualizados estão presentes na *smart home*. Essa avaliação é relevante para determinar a flexibilidade da ferramenta em lidar com variações na infraestrutura de dispositivos da *smart home*, adaptando componentes para mitigar vulnerabilidades. Em ambos os casos, para atender ao segundo objetivo da avaliação, foi coletada a métrica *Tempo de mitigação*, que equivale ao tempo gasto (em segundos) pela UP-Home para detectar e mitigar a vulnerabilidade. Isto envolve a detecção e mitigação de vulnerabilidades da *smart home*, respondendo, assim, ao crescente número de vulnerabilidades injetadas. Por fim, para atender ao terceiro objetivo, foi avaliada a atuação da UP-Home em conjunto com o HIL. Neste último experimento, foi coletada a métrica *Tempo de mitigação assistida*, que equivale ao tempo de mitigação considerando os três perfis de usuários(as) estabelecidos.

5.3 AVALIAÇÃO DA REDUÇÃO DE VULNERABILIDADES

A natureza dinâmica de uma *smart home* pode levar à presença de diferentes dispositivos IoT em diferentes dias. Por essa razão, as varreduras de vulnerabilidades realizadas pelo OpenVAS foram conduzidas em 10 diferentes dias, visando obter uma visão abrangente da segurança da *smart home*. Nesses dez dias em que foram feitas as varreduras, havia diferentes números e tipos de dispositivos conectados à rede da *smart home*. Esta variação ocorre por conta da presença de dispositivos de visitantes (*e.g., smartphones, laptops*), e ainda, pela ativação de novos dispositivos (*e.g.,* alimentador de aquário, sensor de temperatura) ou desativação de dispositivos obsoletos ou defeituosos (*e.g.,* câmeras, fonte de alimentação inteligente). Como resultado dessas varreduras, relatórios detalhados foram gerados, compilando uma lista completa das vulnerabilidades encontradas no ambiente da *smart home*, naquele conjunto de dez varreduras.

A Tabela 5.2 apresenta a lista de vulnerabilidades encontradas.

Estas vulnerabilidades são provenientes de diferentes dispositivos, ampliando o potencial de ataques, uma vez que diversas partes da infraestrutura da *smart home* podem estar comprometidas, inclusive com mais de um tipo de vulnerabilidade no mesmo dispositivo. A primeira coluna CVSS define o escore da vulnerabilidade. Com isso, estão relacionadas aos valores do CVSS respectivo, categorizando escore da vulnerabilidade: 0.1 a 3.9 corresponde a *Baixo*, 4.0 a 6.9 a *Médio* e 7.0 a 10.0 a *Alto*. A coluna *Tipo de Vulnerabilidade* apresenta o nome das 21 diferentes vulnerabilidades encontradas. A coluna *Ação*

Tabela 5.2 – Vulnerabilidades encontradas

CVSS	Tipo de Vulnerabilidade	Escore	Ação	Detecções
10	libupnp Multiple Buffer Overflow Vulnerabilities	Alto	Bloquear	1
10	Operating System (OS) End of Life (EOL) Detection	Alto	Atualizar	2
8.1	MikroTik RouterOS RCE Vulnerability (CVE-2021-41987)	Alto	Atualizar	1
7.5	MikroTik RouterOS <6.46.7, <= 6.47.3, 7.x DoS Vulnerability	Alto	Atualizar	2
6.8	MikroTik RouterOS DoS Vulnerability (CVE-2022-36522)	Médio	Atualizar	2
6.5	MikroTik RouterOS < 6.48.2 Multiple DoS Vulnerabilities	Médio	Atualizar	1
6.5	MikroTik RouterOS <= 6.48.6 DoS Vulnerability	Médio	Atualizar	1
6.4	MQTT Broker Does Not Require Authentication	Médio	Bloquear	1
6.1	jQuery < 1.9.0 XSS Vulnerability	Médio	Bloquear	2
5.9	SSL/TLS: Report Weak Cipher Suites	Médio	Bloquear	1
5.4	SSL/TLS: Report 'Anonymous' Cipher Suites	Médio	Bloquear	1
5.3	Weak Host Key Algorithm(s) (SSH)	Médio	Bloquear	1
5.3	Weak Key Exchange (KEX) Algorithm(s) Supported (SSH)	Médio	Bloquear	1
4.8	Cleartext Transmission of Sensitive Information via HTTP	Médio	Bloquear	12
4.8	FTP Unencrypted Cleartext Login	Médio	Bloquear	2
4.8	Telnet Unencrypted Cleartext Login	Médio	Bloquear	2
4.3	SSL/TLS: Deprecated TLSv1.0 and TLSv1.1 Protocol Detection	Médio	Bloquear	2
4.3	Weak Encryption Algorithm(s) Supported (SSH)	Médio	Bloquear	2
2.6	TCP Timestamps Information Disclosure	Baixo	Bloquear	23
2.6	Weak MAC Algorithm(s) Supported (SSH)	Baixo	Bloquear	3
2.1	ICMP Timestamp Reply Information Disclosure	Baixo	Bloquear	12
Total				**75**

descreve as medidas tomadas pela UP-Home para mitigar cada vulnerabilidade. Existem duas ações possíveis: *Bloquear* e *Atualizar*. A ação *Bloquear* modifica um parâmetro no RouterOS (adaptação paramétrica), impedindo qualquer conexão a porta e IP do dispositivo afetado. Por outro lado, a ação *Atualizar* implica na substituição de um componente da pilha de software no dispositivo desatualizado (adaptação composicional), neste caso, o *firmware*. Por fim, na coluna *Detecções*, é apresentado o número de dispositivos afetados por cada vulnerabilidade.

Com a execução das varreduras, a análise final revelou a existência de 21 diferentes tipos de vulnerabilidades com três níveis de criticidade (escores). Desses 21 tipos de vulnerabilidades detectadas durante os testes de penetração, 4 têm escore *Alto* de criticidade, 14 são *Médio* e 3 *Baixo*. A coluna *Detecções* mostra quantos registros ocorreram de cada tipo de vulnerabilidade. Assim, das 75 detecções, 6 são de escore *Alto*, 31 são de escore *Médio* e 38 *Baixo*.

A partir desse ponto, UP-Home trata essas vulnerabilidades utilizando sua capacidade de detecção e aplicação de medidas corretivas, e por conseguinte cumprindo o primeiro

objetivo da avaliação. Assim, as vulnerabilidades identificadas foram mitigadas progressivamente após cada varredura na rede da *smart home*. A UP-Home analisou os resultados, criou um plano de ação específico e executou medidas para mitigar as vulnerabilidades encontradas. Os resultados dessas intervenções são apresentados na Figura 5.2, destacando as vulnerabilidades mitigadas ao longo de cada varredura.

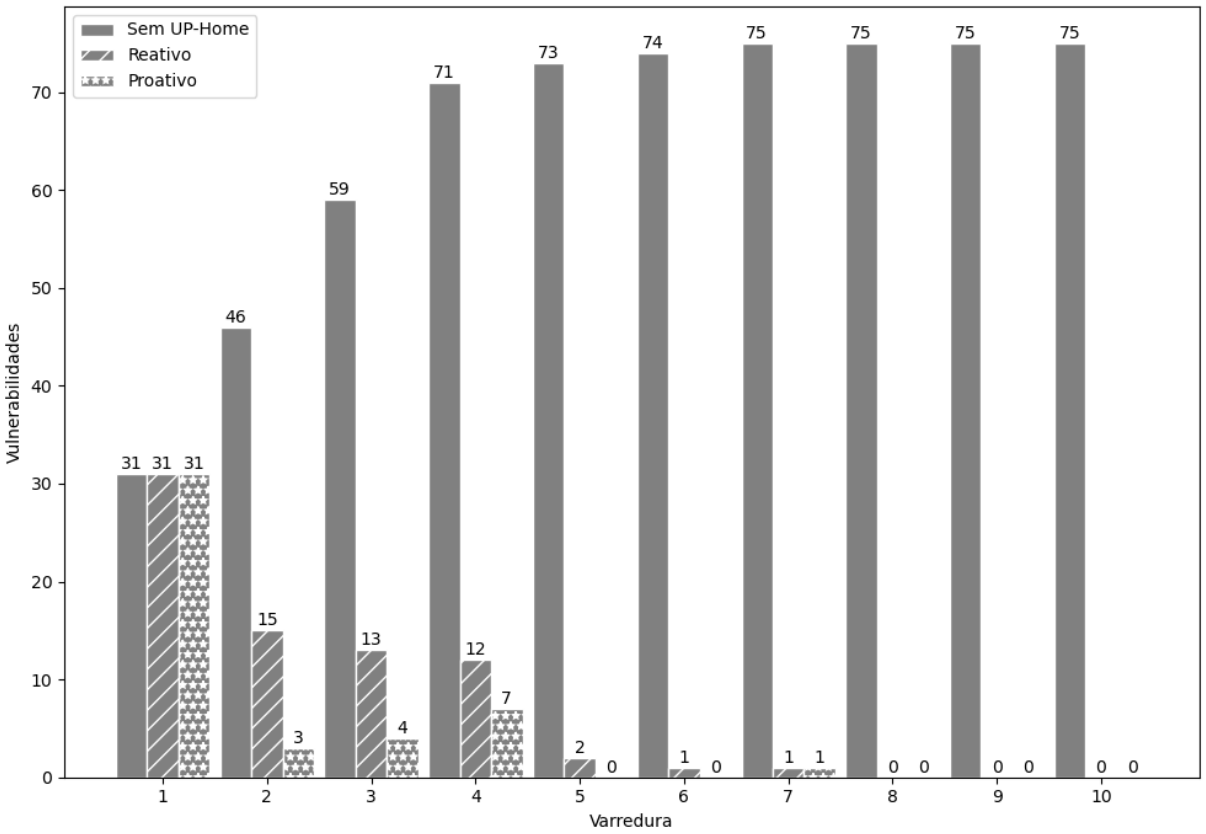

Figura 5.2 – Vulnerabilidades mitigadas

UP-Home atuou na mitigação das vulnerabilidades considerando três cenários: sem atuação da UP-Home (Sem UP-Home), mitigação reativa (Reativo) e mitigação proativa baseada no histórico (Proativo). Na primeira varredura, as três colunas apresentam valores iguais, pois todas as vulnerabilidades detectadas são novas e ainda não há histórico para medidas proativas. Um exemplo prático disto é na detecção de uma vulnerabilidade na porta 21 de um serviço FTP no IP *192.168.1.102*. Na abordagem reativa, essa vulnerabilidade necessita ser tratada sempre que for detectada em outro dispositivo, *e.g., IP 192.168.1.103*, em uma varredura subsequente. Já na abordagem proativa, a UP-Home cria, já na primeira detecção, uma regra preventiva que bloqueia qualquer serviço na porta 21, impedindo que a mesma vulnerabilidade afete outros dispositivos. Isso explica

por que a coluna *Proativo* apresenta reduções mais rápidas e efetivas ao longo das varreduras, enquanto a abordagem reativa permite que vulnerabilidades semelhantes reapareçam em outros dispositivos antes de serem mitigadas. O resultado mostra que a estratégia proativa não apenas elimina vulnerabilidades previamente detectadas, mas também diminui a probabilidade de novas ocorrências, garantindo maior segurança em ambientes de IoT. Através da análise, é possível destacar a superioridade do método proativo ao impedir a propagação de vulnerabilidades por meio de medidas preventivas, enquanto o método reativo, embora eficaz em resposta a detecções, não oferece a mesma capacidade de prevenção. Essas reduções evidenciam que a UP-Home pôde detectar e mitigar as vulnerabilidades de forma contínua. Na quinta varredura, o número de vulnerabilidades detectadas reduziu para apenas 2, e nas varreduras subsequentes (sexta e sétima) apenas 1 vulnerabilidade foi detectada, indicando que as ações de mitigação surtiram efeito. Um ponto importante é que, a partir da oitava varredura, não foram detectadas novas vulnerabilidades, o que sugere que as medidas implementadas pela UP-Home conseguiram estabilizar o ambiente da *smart home* e melhorar sua segurança.

Finalmente, a Figura 5.3 apresenta uma estratificação das ações de mitigação para as vulnerabilidades detectadas.

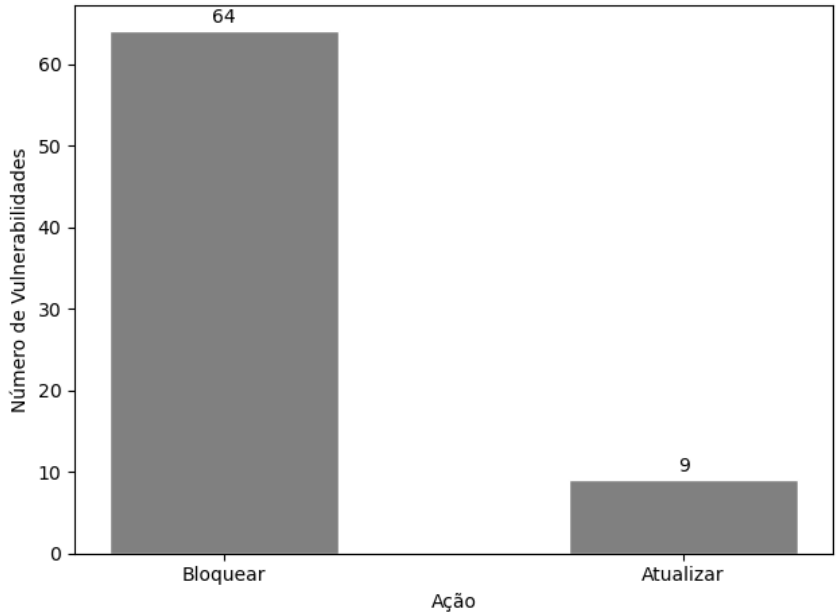

Figura 5.3 – Ações de mitigação das vulnerabilidades

Como mostrado no gráfico, a ação *Bloquear* foi responsável pela mitigação de 66 vulnerabilidades, representando aproximadamente 88% do total. Assim, a incidência da ne-

cessidade de atualização (adaptação composicional) foi menor, pois a relação dispositivo desatualizado em comparação às vulnerabilidades detectadas (adaptação paramétrica) por dispositivo foi consideravelmente menor.

Por outro lado, a ação de *Atualizar* foi aplicada em apenas 9 casos, correspondendo a cerca de 12% das situações de mitigação. Isso indica que, embora a atualização de *firmware* ou software seja uma medida importante para corrigir vulnerabilidades conhecidas (adaptação composicional), ela não foi tão necessária quanto o bloqueio de funcionalidades. A menor proporção de atualizações pode ser explicada pela relação observada entre dispositivos desatualizados e as vulnerabilidades detectadas, que foi relativamente baixa (adaptação paramétrica). Isso significa que a maioria das vulnerabilidades encontradas poderia ser mitigada sem necessidade de atualizar os dispositivos.

Portanto, a predominância do bloqueio como método de mitigação implementa uma estratégia focada em proteger a infraestrutura por meio de restrições de acesso aos IPs, portas e protocolos correspondentes às vulnerabilidades, enquanto a atualização foi utilizada complementarmente para melhorar a segurança dos dispositivos em casos específicos em que a atualização era essencial para a solução do problema.

5.4 AVALIAÇÃO DO TEMPO DE ADAPTAÇÃO

Para alcançar o segundo objetivo da avaliação da UP-Home, foram feitos experimentos em que foi medido o tempo de execução das adaptações. No ambiente mostrado na Figura 5.1, foram colocados dois injetores de vulnerabilidades. Inicialmente, foi usado o *Injetor 1* para criar a configuração mostrada na Figura 5.4.

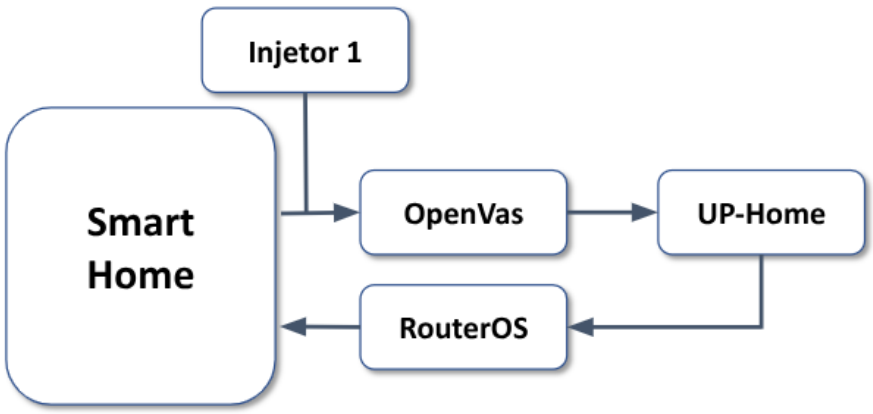

Figura 5.4 – Cenário de uso do Injetor 1

O (*Injetor 1*) possibilitou replicar e introduzir deliberadamente um tipo específico de vulnerabilidade na *smart home*. Com isto, foi avaliado o tempo gasto (em segundos) pela UP-Home, para detectar e mitigar vulnerabilidades da *smart home*. A mitigação (adaptação paramétrica) foi aplicada por meio do bloqueio do acesso ao IP e porta do dispositivo correspondente. Isto serviu para criar um ambiente controlado que permitiu avaliar a capacidade da UP-Home em detectar e mitigar vulnerabilidades em dispositivos IoT, quando confrontada com uma condição vulnerável definida.

A vulnerabilidade introduzida é conhecida como *jQuery < 1.9.0 XSS*, identificada pelo CVE-2012-6708[1]. Essa vulnerabilidade é classificada com um escore *médio* e possui escore CVSS de 6.1. O CVE-2012-6708 usa uma função no *jQuery* (*strInput*) que não faz uma diferenciação confiável entre seletores HTML. Em versões vulneráveis do *jQuery*, a função responsável por processar entradas (*strInput*) determina se a entrada contém código HTML verificando apenas a presença do caractere < em qualquer parte da *string*, dando mais flexibilidade aos invasores na construção de cargas maliciosas. Como resultado, essa vulnerabilidade pode ser explorada para inserir *scripts* de forma arbitrária na página, causando ataques de *Cross-Site Scripting* (XSS)[2]. Isso significa que um invasor pode manipular a entrada de dados, fazendo com que *scripts* não autorizados sejam executados no navegador da vítima, comprometendo a integridade e a segurança dos dados do usuário(a) e possibilitando o roubo de informações confidenciais ou a execução de ações indesejadas em nome do usuário(a).

A Figura 5.5 apresenta os resultados dos experimentos com as ações adaptativas considerando diferentes quantidades de vulnerabilidades injetadas. Os experimentos permitiram avaliar a capacidade de resposta da UP-Home frente a um aumento no número de vulnerabilidades.

Os resultados mostram que o tempo de mitigação tem correlação positiva de 0,996 em relação ao aumento progressivo no número de vulnerabilidades injetadas. Esse comportamento é esperado, pois mais vulnerabilidades, de forma geral, requerem mais tempo de processamento e adaptação por parte dos componentes da UP-Home. Embora o sistema consiga lidar com um aumento na carga de trabalho, é proporcionalmente afetado pelo volume de vulnerabilidades injetadas.

À medida que o número de vulnerabilidades se aproxima de valores mais altos (acima

[1]https://nvd.nist.gov/vuln/detail/cve-2012-6708
[2]https://owasp.org/www-community/attacks/xss/

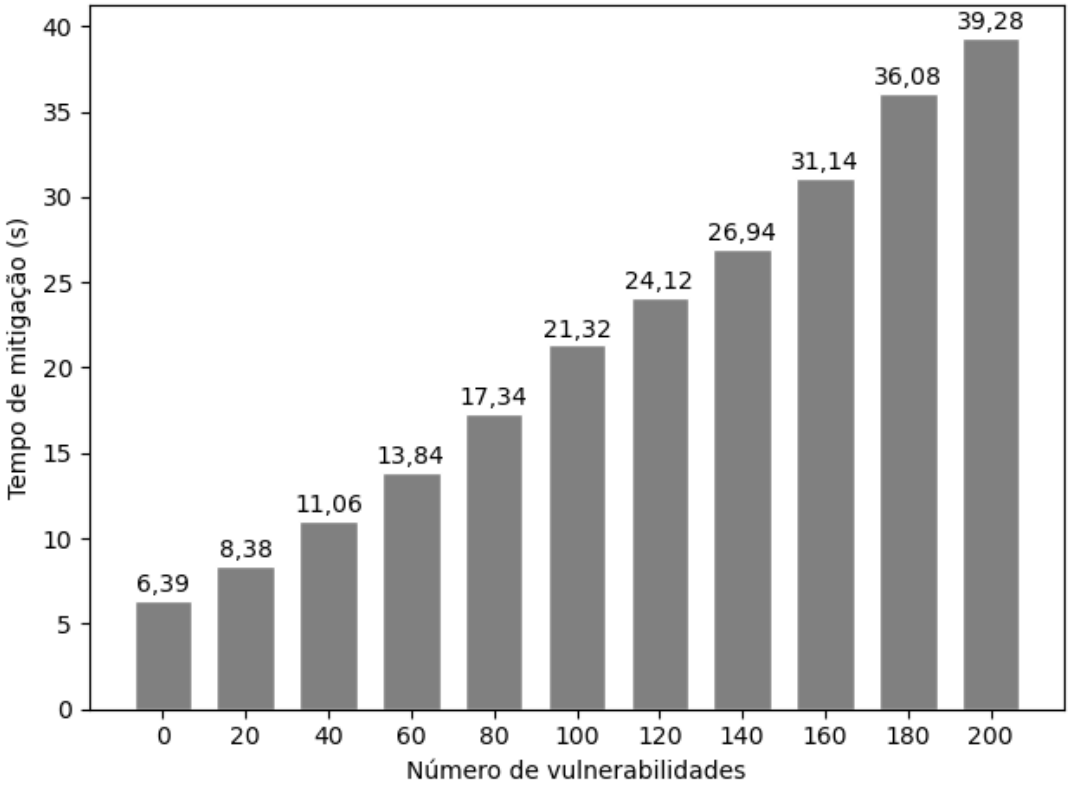

Figura 5.5 – Tempo de mitigação com diferentes cargas de trabalho (*Injetor 1*)

de 160), o tempo de mitigação atinge valores acima de 30 segundos, chegando a 39,28 segundos com 200 vulnerabilidades. Esse tempo elevado pode indicar um possível ponto de saturação do sistema, onde a eficiência operacional começa a diminuir e o tempo de resposta deixa de ser ideal para um ambiente crítico. Isso pode ser um sinal de que, em situações com mais de 200 vulnerabilidades, o sistema poderia precisar de otimizações adicionais ou recursos de processamento mais robustos. A análise sugere que existem oportunidades para melhorar a eficiência da UP-Home. Isso pode incluir a implementação de algoritmos de priorização de vulnerabilidades que tratem primeiro as ameaças mais críticas, reduzindo o impacto de múltiplas vulnerabilidades em um curto espaço de tempo. Além disso, a adoção de técnicas de processamento paralelo pode ajudar a mitigar múltiplas vulnerabilidades simultaneamente, diminuindo o tempo total de adaptação.

Para correlacionar os tipos de vulnerabilidades com o tempo de mitigação, foi considerada a possibilidade de analisar o impacto dessa variável na atuação da UP-Home. O experimento seguinte envolveu a introdução de vulnerabilidades com diferentes escore, visando avaliar se a complexidade dessas vulnerabilidades influencia o tempo necessário para sua mitigação.

Nestes experimentos, foi empregado um segundo injetor de vulnerabilidades, denominado *Injetor 2*, conforme ilustrado na Figura 5.6. Esse componente ampliou o escopo da avaliação da UP-Home, permitindo a introdução de diferentes tipos de vulnerabilidades nos experimentos. A Tabela 5.3 detalha as vulnerabilidades injetadas, juntamente com seus respectivos escores.

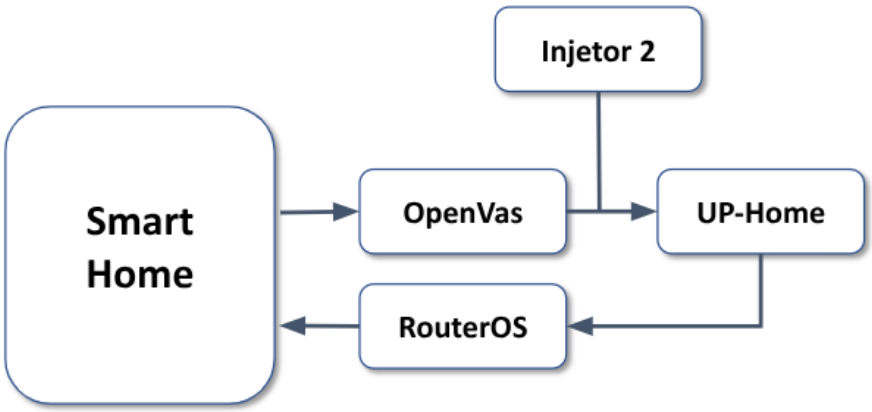

Figura 5.6 – Cenário de uso do Injetor 2

Nos experimentos conduzidos com o *Injetor 2*, foram injetadas vulnerabilidades para medir o tempo necessário para sua mitigação. Na prática, o *Injetor 2* complementou as varreduras realizadas pelo OpenVAS, introduzindo uma diversidade maior de vulnerabilidades a serem abordadas. Essa abordagem ampliou o escopo dos testes, fornecendo uma avaliação mais abrangente da eficácia da UP-Home em responder a diferentes tipos de ameaças. A Tabela 5.3 resume as vulnerabilidades utilizadas nesse experimento, destacando sua criticidade e características para uma análise mais detalhada.

É importante ressaltar que algumas vulnerabilidades podem aparecer em múltiplos dispositivos na *smart home*, indicando que o número total de vulnerabilidades detectadas pode ultrapassar o que está listado na Tabela 5.3. No experimento, foram consideradas 12 vulnerabilidades distintas, sendo uma classificada com criticidade *Alta* e 11 com escore *Médio*. Essas vulnerabilidades foram registradas em relatórios do OpenVAS, totalizando 36 detecções nos dispositivos da *smart home*, servindo como *baseline* para a análise.

A Figura 5.7 apresenta os resultados desses experimentos. Para simular cenários mais críticos, conjuntos adicionais de vulnerabilidades com 72, 108, 144 e 180 detecções foram criados aplicando um fator multiplicador. Isso permitiu a análise da UP-Home em situações com um aumento progressivo na quantidade de vulnerabilidades, desafiando

Tabela 5.3 – Vulnerabilidades detectadas

CVSS	**Vulnerabilidade**
10.0	libupnp Multiple Buffer Overflow Vulnerabilities
6.1	jQuery < 1.9.0 XSS Vulnerability
5.4	SSL/TLS: Report 'Anonymous' Cipher Suites
5.3	Weak Host Key Algorithm(s) (SSH)
5.3	Weak Key Exchange (KEX) Algorithm(s) Supported (SSH)
4.8	Cleartext Transmission of Sensitive Information via HTTP
4.8	Telnet Unencrypted Cleartext Login
4.8	FTP Unencrypted Cleartext Login
4.8	Telnet Unencrypted Cleartext Login
4.3	Weak Encryption Algorithm(s) Supported (SSH)
4.3	SSL/TLS: Deprecated TLSv1.0 and TLSv1.1 Protocol Detection
4.3	Weak Encryption Algorithm(s) Supported (SSH)

a ferramenta em termos de capacidade de resposta e mitigação em um ambiente mais exigente.

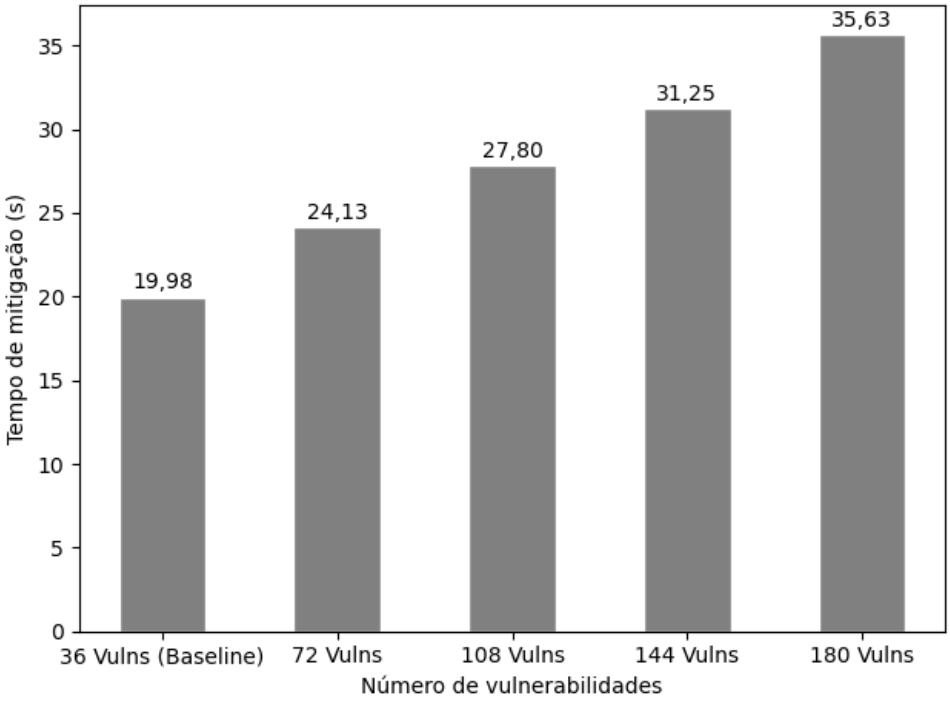

Figura 5.7 – Tempo de execução com diferentes cargas de trabalho (*Injetor 2*)

A análise dos experimentos com o *Injetor 2* revelou uma correlação linear entre o número de vulnerabilidades introduzidas e o tempo necessário para mitigá-las. Esse com-

portamento evidencia que a UP-Home pode escalar proporcionalmente ao aumento de ameaças, mantendo a consistência em sua resposta. Os resultados obtidos foram semelhantes aos observados com o *Injetor 1*, demonstrando que, independentemente do tipo de vulnerabilidade, o tempo de mitigação segue uma correlação positiva de 0,999.

A escalabilidade observada nos experimentos com os *Injetores 1 e 2* é um ponto positivo, e indica que a ferramenta UP-Home foi capaz de mitigar um número significativo de vulnerabilidades para uma *smart home*. No entanto, os resultados ressaltam a necessidade de otimizações futuras para garantir que o tempo de resposta não ultrapasse limites aceitáveis, especialmente em cenários com mais de 180 vulnerabilidades. Para manter o tempo de mitigação em uma janela segura, técnicas como paralelismo e balanceamento de carga devem ser consideradas, a fim de manter o tempo de resposta dentro de uma janela segura, de modo a limitar a exposição do sistema a possíveis ataques.

Esses resultados também ressaltam a importância de rápidas respostas em relação à presença de vulnerabilidades, uma vez que tempos de resposta prolongados podem comprometer a segurança da *smart home*. Embora a ferramenta tenha demonstrado uma atuação consistente, é fundamental investigar se vulnerabilidades de maior criticidade podem afetar a eficiência da mitigação diferenciadamente. Essa análise mais profunda pode levar a melhorias na arquitetura da UP-Home para garantir que o tempo de resposta seja aceitável, mesmo diante de ameaças mais críticas.

Ainda para atender o segundo objetivo da avaliação da UP-Home, foi feito um terceiro experimento, onde foi avaliado o tempo de adaptação da UP-Home utilizando a adaptação composicional. Para realizar essa avaliação, UP-Home monitorou a *smart home*, coletando informações sobre os parâmetros de segurança por meio do HA e dos sites dos fabricantes para identificar novas versões dos softwares instalados na *smart home*. Através dos dados coletados, UP-Home analisa a versão de *firmware* mais recente disponível por cinco fabricantes diferentes dos dispositivos que compunham a *smart home*: um smartphone com sistema operacional Android, duas câmeras de segurança de fabricantes distintos (Intelbras e V380), um *switch Sonoff*, e um *gateway* HA.

Para aplicar as adaptações, foram executados comandos ADB, que acessaram um AVD. O processo de atualização foi automatizado por um *script* Python que envia os comandos necessários para realizar a atualização do *firmware* por meio da aplicação do dispositivo. Vale ressaltar que os tempos de *download* e instalação das atualizações não foram considerados nesta avaliação, uma vez que, durante esse período, o dispositivo

geralmente reinicia e perde a capacidade de comunicação.

A Figura 5.8 apresenta os resultados destes experimentos. Nesta figura, quando não há atualização, o tempo para mitigação corresponde ao tempo de monitoramento e análise, uma vez que não existe a necessidade do planejamento e da execução da adaptação.

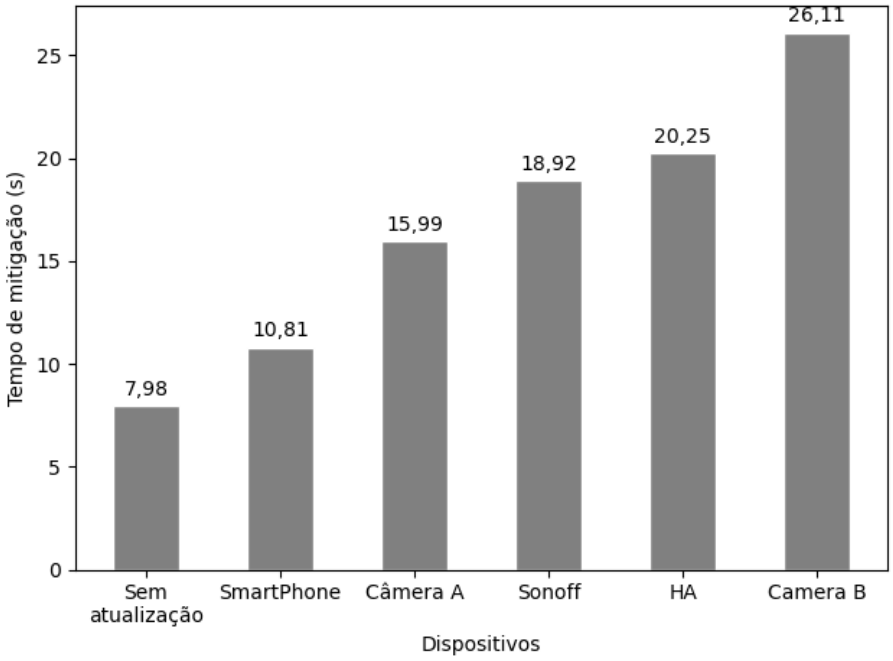

Figura 5.8 – Tempo de execução com cinco dispositivos

Os resultados obtidos mostram uma variação nos tempos de atualização, dependendo do tipo de dispositivo envolvido. Observou-se que o *smartphone* apresentou o menor tempo de atualização, enquanto a *Câmera B* teve o tempo mais longo para a conclusão do processo. Essa discrepância de tempos entre os dispositivos pode ser atribuída a características específicas de cada aplicação, como o tempo necessário para abrir o aplicativo, o acesso à interface do dispositivo e o número de interações exigidas pelo *script* de atualização. Esses fatores influenciam diretamente o tempo necessário para iniciar o processo de atualização.

5.5 AVALIAÇÃO COM O *HUMAN-IN-THE-LOOP*

A avaliação da UP-Home com suporte ao *Human-In-the-Loop* (HIL) considerou as seguintes condições: o usuário respondeu *sim* a *Q1* para aceitar notificações nas vulnerabilidades de escore *médio* (ver Tabela 4.2). O usuário também respondeu *sim* a *Q2* para

a aplicação de medidas preventivas, independentemente do escore das vulnerabilidades. O usuário foi consultado em *Q3* e decidiu adotar medidas severas para vulnerabilidades de escore *Alto*, bloqueando o dispositivo na rede da *smart home*.

Para a avaliação da terceira métrica (Tempo de mitigação assistida), foram definidos três perfis de usuários(as) com tempos de resposta variados: *Perfil A* (10 segundos), *Perfil B* (30 segundos) e *Perfil C* (50 segundos). Esses tempos representam diferentes interações dos usuários(as) com a UP-Home após receberem uma notificação de segurança. O tempo de resposta do usuário(a) foi então somado ao tempo necessário para mitigar a vulnerabilidade, notificar o usuário(a), aplicar medida preventiva e bloquear o dispositivo.

Além disso, foi estipulado que, em caso de inatividade do usuário(a) em resposta à *Q3*, seja por desconexão ou tempo de resposta excedido, a vulnerabilidade seria mitigada, mas o dispositivo não seria bloqueado, considerando o risco de interrupção de serviços essenciais. Na análise, foram selecionadas três vulnerabilidades com diferentes escores: Baixo, Médio e Alto. A Tabela 5.4 apresenta as vulnerabilidades usadas na atuação da UP-Home com suporte ao HIL.

Tabela 5.4 – Vulnerabilidades resolvidas com suporte ao *Human-In-the-Loop*

Vulnerabilidades	**CVSS**
libupnp Multiple Buffer Overflow Vulnerabilities	10 (Alta)
SSL/TLS: Report Weak Cipher Suites	5.9 (Média)
Weak MAC Algorithm(s) Supported (SSH)	2.6 (Baixa)

A escolha de vulnerabilidades com escores diferentes mostra como a interação humana pode influenciar na priorização e na tomada de decisão durante o processo de notificação, prevenção e bloqueio. Essa escolha destaca as especificidades da ferramenta e sua aplicação em cenários reais, nos quais a criticidade das vulnerabilidades varia significativamente. Com isso, o experimento avalia a ferramenta UP-Home em situações práticas e complexas, reforçando a relevância do HIL para a segurança de dispositivos IoT em *smart homes*. Assim, para mitigar qualquer uma das vulnerabilidades consideradas, realiza-se o bloqueio do IP, porta e protocolo associados à vulnerabilidade de qualquer escore. A medida preventiva impede acesso à porta associada a vulnerabilidade, independente do escore. Da mesma forma, a notificação é feita para vulnerabilidades de escores *Médio* ou *Alto*. Por fim, o bloqueio de dispositivo é voltado exclusivamente as vulnerabilidades de escore *Alto*.

No caso do bloqueio, após o usuário(a) ser notificado, e responder *Sim* à pergunta *Q3*, o dispositivo será completamente bloqueado. Com exceção da notificação, feita por meio da API do Telegram, as demais execuções impedem a exploração da vulnerabilidade, e são implementadas no *gateway* RouterOS.

Os resultados da interação entre a UP-Home e o HIL são apresentados na Figura 5.9.

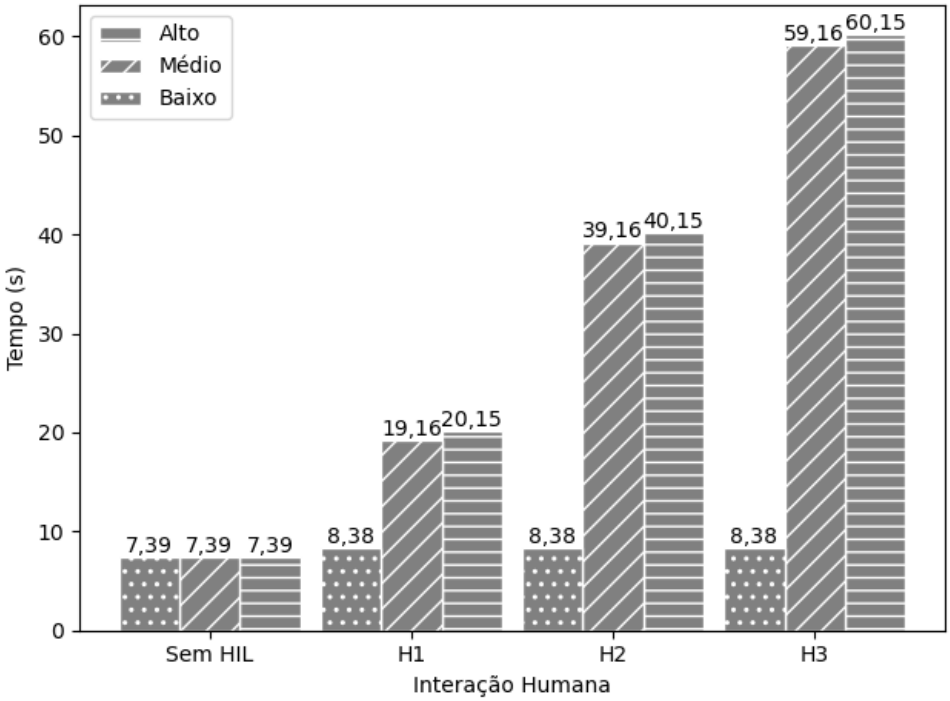

Figura 5.9 – *Human-In-the-Loop* com vulnerabilidades de diferentes escores CVSS

Os resultados mostram que UP-Home mitiga uma vulnerabilidade em 7,39 segundos, independente do seu escore (CVSS). Nesse tempo, a ferramenta UP-Home lê os relatórios de vulnerabilidade do OpenVAS e mitiga a vulnerabilidade detectada. Para notificar o(a) usuário(a) sobre a vulnerabilidade mitigada, UP-Home leva 0,78 segundo. Já a aplicação de uma medida preventiva ou de um bloqueio de dispositivo apresenta o mesmo tempo de execução, 0,99 segundo. Assim, para vulnerabilidades de escore *Baixo*, é feita a mitigação (obrigatória, 7,39 segundos) e medida preventiva (opcional, 0,99 segundos) apenas. Com vulnerabilidades de escore *Médio*, é feita a mitigação (obrigatória, 7,39 segundos), aplicada medida preventiva (opcional, 0,99 segundo) e notificação (opcional, 0,78 segundo). Por fim, as vulnerabilidades de escore *Alto*, são mitigadas (obrigatória, 7,39 segundos), notificadas (obrigatória, 0,78 segundo), aplicadas medidas preventivas (opcional, 0,99 segundo) e bloqueio (opcional, 0,99 segundo). Embora existam respostas opcionais, neste experimento foi considerado que o usuário(a) respondeu *sim* em todas as perguntas (*Q1, Q2 e Q3*).

Além disso, para todos os escores foi acrescido o tempo correspondente aos perfis de usuários (10, 30 e 50 segundos).

A interação da UP-Home com suporte humano viabilizou a gestão de vulnerabilidades da *smart home*, especialmente com vulnerabilidades mais críticas, reforçando que o uso do suporte humano aprimorou a tomada de decisão mais alinhada ao contexto do ambiente. No entanto, uma abordagem de mitigação totalmente automatizada pode, em situações específicas, afetar serviços essenciais da *smart home*. Por outro lado, a dependência do suporte humano (HIL) também apresenta desafios, como a possibilidade de decisões inadequadas ou atrasadas, que podem comprometer a segurança. Dessa forma, a integração equilibrada entre automação e intervenção humana é fundamental para otimizar a resposta a ameaças, minimizando riscos associados a ambas as abordagens.

Os resultados reforçam a importância de uma abordagem adaptativa que combina automação com a possibilidade de suporte humano, quando necessário, para oferecer uma ferramenta abrangente.

5.6 CONSIDERAÇÕES FINAIS

Este capítulo detalhou a avaliação da UP-Home. Primeiramente, os objetivos da avaliação foram apresentados na Seção 5.1. A Seção 5.2 abordou a avaliação dos mecanismos de adaptação, detalhando as métricas utilizadas em cenários específicos. Na Seção 5.3, a análise focou na quantidade de vulnerabilidades mitigadas pela UP-Home, verificando seu impacto na segurança geral da *smart home*. O segundo experimento, descrito na Seção 5.4, mediu o tempo de resposta da UP-Home em adaptações paramétricas e composicionais, avaliando tanto um tipo específico de vulnerabilidade quanto diferentes tipos de vulnerabilidades. Posteriormente, a Seção 5.5 examinou o suporte da UP-Home ao conceito de *Human-In-the-Loop*, explorando como a interação humana pode influenciar a eficácia da ferramenta em cenários mais complexos.

6 CONCLUSÕES E TRABALHOS FUTUROS

A IoT tem emergido como uma área de grande interesse em vários setores, tanto no âmbito acadêmico quanto na indústria. A capacidade de controle e interconexão oferecida pelos dispositivos IoT atrai usuários com diversos níveis de conhecimento em informática. Dentro desse contexto, as *smart homes* têm se destacado como um dos principais domínios de aplicação para a IoT. No entanto, a crescente adoção de dispositivos inteligentes na *smart home* torna esse ambiente suscetível a ameaças cibernéticas. Isto é motivado pela falta de segurança nos dispositivos da *smart home*, e atribuída a diversos fatores, como o gerenciamento inadequado da segurança, ausência de padronização nas atualizações e o abandono de produtos por parte de fabricantes. Além disso, as adaptações de segurança devem ser feitas de forma automática, mas permitindo suporte humano nas decisões mais críticas, para evitar que serviços sejam interrompidos sem uma análise prévia do usuário. Para enfrentar o avanço das ameaças à segurança, é necessário desenvolver soluções autônomas e dinâmicas que detectem e mitiguem ameaças em tempo de execução.

6.1 CONTRIBUIÇÕES

Esta tese apresenta o desenvolvimento da UP-Home, uma ferramenta autônoma voltada para aprimorar a segurança em *smart homes* no contexto de IoT. A UP-Home foca na identificação e no gerenciamento de parâmetros de segurança, implementando estratégias de monitoramento que detectam e mitigam ameaças em tempo de execução. A ferramenta combina atualizações automáticas com suporte ao HIL para decisões críticas, equilibrando automação e intervenção manual. Foi elaborado um modelo conceitual de segurança adaptativa e construída uma arquitetura funcional, culminando na implementação de um protótipo que demonstrou a atuação da abordagem proposta. A integração do modelo MAPE-K e atualizações OTA fortalece a capacidade da UP-Home em responder proativamente às vulnerabilidades em ambientes conectados.

As contribuições desta tese podem ser sumarizadas como segue:

- **Desenvolvimento de uma Ferramenta Autônoma.** Criação da UP-Home, uma arquitetura para aprimorar a segurança em *smart homes* focada na identificação e gerenciamento de parâmetros de segurança.

- **Estratégias de Monitoramento de Segurança.** Implementação de métodos para monitorar dispositivos da *smart home*, com foco na detecção de violações de segurança por meio de adaptações composicionais e paramétricas.
- **Modelo Conceitual de Segurança Adaptativa.** Elaboração de um modelo conceitual que estrutura a segurança adaptativa em *smart homes*, incluindo a lógica de adaptação e componentes que devem ser monitorados.
- **Notação de Plano de Adaptação.** Definição de uma notação para descrever planos de adaptação voltados à atualizações de segurança em *smart homes*.
- **Algoritmos para o MAPE-K** Definição de algoritmos para tomada de decisão e criação de planos de adaptação dos componentes Analisador e Planejador, respectivamente.
- **Projeto, Implementação da Arquitetura e Protótipo Funcional.** Criação e implementação da arquitetura UP-Home, que reúne elementos de adaptação e os componentes necessários para uma ferramenta de segurança em *smart homes*. Para avaliar a ferramenta, foi desenvolvido um protótipo utilizando *gateways* HA, RouterOS e o *scanner* OpenVAS. O protótipo permitiu o monitoramento das versões de *firmware* dos dispositivos, detectou e mitigou vulnerabilidades por meio do escaneamento de conexões.

Quando a UP-Home identifica uma nova versão (*release*), comandos ADB (*Android Debug Bridge*) são acionados para iniciar o processo de atualização OTA em dispositivos desatualizados. O uso do OpenVAS permitiu a identificação de vulnerabilidades na *smart home* por meio de varreduras detalhadas. O escaneamento das conexões estabelecidas pelos dispositivos possibilitou ações reativas e proativas para aprimorar a segurança. Os estudos de caso demonstraram a capacidade da ferramenta em evitar a persistência de vulnerabilidades.

A UP-Home aborda desafios de segurança em *smart homes* de forma reativa e proativa, mitigando vulnerabilidades e protegendo dispositivos contra ameaças. Diferente de sistemas que oferecem apenas soluções reativas, a UP-Home propõe um modelo de adaptação baseado em técnicas de computação adaptativa, ajustando componentes e parâmetros de segurança em tempo real. Essa capacidade adaptativa adiciona uma camada de defesa para ambientes IoT, onde vulnerabilidades podem surgir de forma imprevisível.

A abordagem da UP-Home é baseada no modelo MAPE-K (*Monitor, Analyzer, Plan e Executor - Knowledge*), permitindo que a ferramenta opere de forma dinâmica com suporte ao HIL para ajustes contínuos às ameaças e vulnerabilidades identificadas.

A integração de atualizações OTA é utilizada como método para aplicação das adaptações. Trabalhos como os de Lee e Lee (2017) e Zandberg et al. (2019) destacam a importância das atualizações OTA para manter a integridade de dispositivos IoT. A UP-Home combina OTA com sua arquitetura adaptativa, permitindo corrigir falhas de segurança e implementar melhorias sem interromper operações.

Outro aspecto da UP-Home é a integração do conceito de HIL com a autoproteção. Trabalhos como os de Gil et al. (2020), Taherisadr, Faruque e Elmalaki (2024) e Domaszewicz et al. (2016) ressaltam os benefícios da interação humana em sistemas ciberfísicos. A UP-Home combina essa interação com um modelo de autoadaptação centralizado, permitindo que decisões críticas sejam autorizadas com intervenção humana.

A análise comparativa apresentada na Seção 3.5 destacou as vantagens da UP-Home em relação a outras abordagens. A ferramenta aplica conceitos discutidos na literatura, como o uso de MAPE-K ((EL-MALIKI; SEIGNE, 2016), (RIBEIRO et al., 2016)) e OTA ((FRISCH; REISSMANN; PAPE, 2017), (ZANDBERG et al., 2019)), promovendo segurança adaptativa e colaborativa.

6.2 LIMITAÇÕES

As limitações desta tese são baseadas nos desafios enfrentados durante o desenvolvimento da UP-Home e nas restrições da ferramenta proposta. As principais limitações identificadas são:

- **Dependência da atualização de firmware pelos fabricantes.** A eficácia da UP-Home em mitigar vulnerabilidades por meio da atualização de *firmware* depende da disponibilidade de novas versões fornecidas pelos fabricantes dos dispositivos IoT. Muitos fabricantes abandonam o suporte a dispositivos antigos, deixando-os vulneráveis, mesmo quando a UP-Home identifica a necessidade de atualização.

- **Limitação da base de dados do OpenVAS.** A ferramenta utiliza o OpenVAS para detecção de vulnerabilidades, e a precisão dessa identificação depende da base

de dados do *scanner*. Se novas ameaças não estiverem catalogadas, a UP-Home pode não as detectar eficientemente.

- **Tempo de intervenção humana em decisões críticas (HIL).** A abordagem HIL permite que usuários tomem decisões sobre a mitigação de vulnerabilidades. No entanto, essa dependência pode causar atrasos em situações que exigem resposta rápida. Caso haja demora na resposta do usuário, a segurança da *smart home* pode ser temporariamente impactada.

- **Compatibilidade restrita a dispositivos com *Home-Assistant*.** A UP-Home foi projetada para operar com dispositivos compatíveis com *Home-Assistant*. A diversidade de dispositivos e protocolos no ecossistema IoT pode dificultar sua aplicação universal. Dispositivos que utilizam padrões proprietários podem não ser facilmente integrados, limitando o escopo da ferramenta.

6.3 TRABALHOS FUTUROS

Esta tese sugere oportunidades de aprimoramento para fortalecer a segurança adaptativa em *smart homes*.

- **Desenvolvimento de Modelos Cognitivos para Segurança Autoadaptativa.** A integração de modelos cognitivos baseados em aprendizado contínuo e raciocínio probabilístico pode permitir que a UP-Home identifique e antecipe padrões de ataque, ajustando suas políticas de segurança de forma proativa.

- **Autonomia Total em Segurança de *smart homes***. Atualmente, a UP-Home atua em um modelo híbrido, combinando automação com suporte ao HIL. Pesquisas futuras podem explorar o aprimoramento da participação humana nas tomadas de decisões de maior complexidade, sem comprometer a segurança.

- **Interoperabilidade e Padronização de Segurança para IoT.** Trabalhos futuros podem explorar estratégias de padronização e protocolos universais, permitindo que sistemas como a UP-Home se integrem a diferentes dispositivos e fabricantes.

- **Aplicação de Inteligência Coletiva na Defesa Cibernética.** A criação de redes colaborativas de dispositivos inteligentes, capazes de compartilhar informações

sobre ameaças e estratégias de mitigação, pode fortalecer a segurança de múltiplos ambientes da *smart home*.

Essas direções de pesquisa podem aprimorar a segurança adaptativa em *smart homes*, contribuindo para a evolução do campo de IoT e segurança cibernética.

6.4 PUBLICAÇÕES

Os resultados desta tese foram publicados em dois artigos científicos:

- Josival Silva, Nelson Rosa, and Fernando Lins. 2024. UP-Home: A Self-Adaptive Solution for Smart Home Security. Journal of Universal Computer Science (JUCS) 30, 4 (Abril 2024), 502-530. https://doi.org/10.3897/jucs.107050
- Josival Silva, Fernando Lins, and Nelson Rosa. 2025. A Human-In-the-Loop Approach for Self-Protecting Smart Homes. In Proceedings of the 39th IEEE International Conference on Advanced Information Networking and Applications (AINA 2025).

REFERÊNCIAS

ABDULLA, A. I.; ABDULRAHEEM, A. S.; SALIH, A. A.; SADEEQ, M.; AHMED, A. J.; FERZOR, B. M.; SARDAR, O. S.; MOHAMMED, S. I. Internet of things and smart home security. *Technol. Rep. Kansai Univ*, v. 62, n. 5, p. 2465–2476, 2020.

AKSU, M. U.; ALTUNCU, E.; BICAKCI, K. A first look at the usability of openvas vulnerability scanner. In: *Workshop on usable security (USEC)*. [S.l.: s.n.], 2019.

ALHAFIDH, B.; ALLEN, W. Design and simulation of a smart home managed by an intelligent self-adaptive system. *International Journal of Engineering Research and Applications*, v. 6, n. 8, p. 64–90, 2016.

ALMUSAED, A.; YITMEN, I.; ALMSSAD, A. Enhancing smart home design with ai models: A case study of living spaces implementation review. *Energies*, MDPI, v. 16, n. 6, p. 2636, 2023.

ARCAINI, P.; MIRANDOLA, R.; RICCOBENE, E. Smart home platform supporting decentralized adaptive automation control. In: *Proceedings of the 35th ACM Symposium on Applied Computing*. [S.l.: s.n.], 2020.

BADRAN, H. Iot security and consumer trust. In: *Proceedings of the 20th Annual International Conference on Digital Government Research*. [S.l.: s.n.], 2019. p. 133–140.

BAUWENS, J.; RUCKEBUSCH, P.; GIANNOULIS, S.; MOERMAN, I.; POORTER, E. D. Over-the-air software updates in the internet of things: An overview of key principles. *IEEE Communications Magazine*, IEEE, v. 58, n. 2, p. 35–41, 2020.

BHARDWAJ, A.; BHARANY, S.; ABULFARAJ, A. W.; IBRAHIM, A. O.; NAGMELDIN, W. Fortifying home iot security: A framework for comprehensive examination of vulnerabilities and intrusion detection strategies for smart cities. *Egyptian Informatics Journal*, Elsevier, v. 25, p. 100443, 2024.

BOOIJ, T. M.; CHISCOP, I.; MEEUWISSEN, E.; MOUSTAFA, N.; HARTOG, F. T. D. Ton_iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets. *IEEE Internet of Things Journal*, IEEE, v. 9, n. 1, p. 485–496, 2021.

CáMARA, J.; MORENO, G. A.; GARLAN, D. Reasoning about human participation in self-adaptive systems. In: *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. [S.l.]: IEEE Press, 2015. (SEAMS '15), p. 146–156.

CHANDRA, H.; ANGGADJAJA, E.; WIJAYA, P. S.; GUNAWAN, E. Internet of things: Over-the-air (ota) firmware update in lightweight mesh network protocol for smart urban development. In: IEEE. *2016 22nd Asia-Pacific Conference on Communications (APCC)*. [S.l.], 2016. p. 115–118.

DARBY, S. J. Smart technology in the home: time for more clarity. *Building Research & Information*, Routledge, v. 46, n. 1, p. 140–147, 2018. Disponível em: <https://doi.org/10.1080/09613218.2017.1301707>.

DEHRAJ, P.; SHARMA, A. A review on architecture and models for autonomic software systems. *The Journal of Supercomputing*, Springer, v. 77, p. 388–417, 2021.

DOMASZEWICZ, J.; LALIS, S.; PRUSZKOWSKI, A.; KOUTSOUBELIAS, M.; TAJMAJER, T.; GRIGOROPOULOS, N.; NATI, M.; GLUHAK, A. Soft actuation: Smart home and office with human-in-the-loop. *IEEE Pervasive computing*, IEEE, v. 15, n. 1, p. 48–56, 2016.

DORNERWORKS. *The Importance of Over-The-Air (OTA) Updates in IoT*. 2020. Disponível em: <https://www.iotforall.com/over-the-air-ota-updates-reduce-risk-win-customers>.

EL-MALIKI, T.; SEIGNE, J.-M. Efficient security adaptation framework for internet of things. In: IEEE. *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. [S.l.], 2016. p. 206–211.

ETSI. ETSI Technical Specification, *CYBER; Cyber Security for Consumer Internet of Things*. 2019. Disponível em: <https://www.etsi.org/deliver/etsi_ts/103600_103699/103645/01.01.01_60/ts_103645v010101p.pdf>.

ETSI. *ETSI TS 103 645 V2.1.2*. [s.n.], 2019. v. 1. 1–16 p. ISBN 9789292042363. Disponível em: <https://www.etsi.org/deliver/etsi_ts/103600_103699/103645/02.01.02_60/ts_103645v020102p.pdf>.

FERNANDES, J. M.; RIVADENEIRA, J. E.; RODRIGUES, A.; BOAVIDA, F.; SILVA, J. S. People 4.0—a model for human-in-the-loop cps-based systems. *Computer Standards & Interfaces*, Elsevier, v. 91, p. 103895, 2024.

FERREIRA, L.; OLIVEIRA, T.; NEVES, C. Consumer's intention to use and recommend smart home technologies: The role of environmental awareness. *Energy*, Elsevier, v. 263, p. 125814, 2023.

FRISCH, D.; REISSMANN, S.; PAPE, C. An over the air update mechanism for esp8266 microcontrollers. In: *Proceedings of the ICSNC, the Twelfth International Conference on Systems and Networks Communications, Athens, Greece*. [S.l.: s.n.], 2017. p. 8–12.

GIL, M.; ALBERT, M.; FONS, J.; PELECHANO, V. Engineering human-in-the-loop interactions in cyber-physical systems. *Information and Software Technology*, v. 126, p. 106349, 2020. ISSN 0950-5849.

GU, T.; FANG, Z.; ABHISHEK, A.; FU, H.; HU, P.; MOHAPATRA, P. Iotgaze: Iot security enforcement via wireless context analysis. In: IEEE. *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. [S.l.], 2020. p. 884–893.

GUISSOUMA, H.; KLARE, H.; SAX, E.; BURGER, E. An empirical study on the current and future challenges of automotive software release and configuration management. In: IEEE. *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. [S.l.], 2018. p. 298–305.

HAMMI, B.; ZEADALLY, S.; KHATOUN, R.; NEBHEN, J. Survey on smart homes: Vulnerabilities, risks, and countermeasures. *Computers & Security*, Elsevier, v. 117, p. 102677, 2022.

HARA, T.; OKADA, Y.; OTA, J. Reorganizing cyber-physical configurations using user activities for human-in-the-loop cyber-physical systems. *IFAC-PapersOnLine*, v. 56, n. 2, p. 9703–9708, 2023. ISSN 2405-8963. 22nd IFAC World Congress. Disponível em: <https://www.sciencedirect.com/science/article/pii/S240589632300633X>.

HASSIJA, V.; CHAMOLA, V.; SAXENA, V.; JAIN, D.; GOYAL, P.; SIKDAR, B. A survey on iot security: application areas, security threats, and solution architectures. *IEEE Access*, IEEE, v. 7, p. 82721–82743, 2019.

HELLAOUI, H.; BOUABDALLAH, A.; KOUDIL, M. Tas-iot: trust-based adaptive security in the iot. In: IEEE. *2016 IEEE 41st Conference on Local Computer Networks (LCN)*. [S.l.], 2016. p. 599–602.

HUSSIN, S. F.; ABDOLLAH, M. F.; AHMAD, I. B. Acceptance of iot technology for smart homes: A systematic literature review. In: SPRINGER. *International Conference on Information Systems and Intelligent Applications*. [S.l.], 2023. p. 187–202.

IANNUCCI, S.; ABDELWAHED, S. A probabilistic approach to autonomic security management. In: IEEE. *2016 IEEE International Conference on Autonomic Computing (ICAC)*. [S.l.], 2016. p. 157–166.

IBM. *An architectural blueprint for autonomic computing*. 2005. Disponível em: <https://www-03.ibm.com/autonomic/pdfs/AC\%20Blueprint\%20White\%20Paper\%20V7.pdf>.

Internet Society. *IoT Trust by Design Framework*. 2018. Acessado em: 22 de novembro de 2020. Disponível em: <https://www.internetsociety.org/resources/doc/2018/iot-trust-by-design/>.

IoTSF. IoT Security Architecture and Policy for the Enterprise- a Hub Based Approach Release 1. p. 37, 2018. Disponível em: <https://www.iotsecurityfoundation.org/wp-content/uploads/2018/11/IoT-Security-Architecture-and-Policy-for-the-Enterprise-a-Hub-Based-Approach.pdf>.

JACOBSSON, A.; BOLDT, M.; CARLSSON, B. A risk analysis of a smart home automation system. *Future Generation Computer Systems*, Elsevier, v. 56, p. 719–733, 2016.

KAFLE, V. P.; FUKUSHIMA, Y.; HARAI, H. Internet of things standardization in itu and prospective networking technologies. *IEEE Communications Magazine*, IEEE, v. 54, n. 9, p. 43–49, 2016.

KANG, W. M.; MOON, S. Y.; PARK, J. H. An enhanced security framework for home appliances in smart home. *Human-centric Computing and Information Sciences*, Springer, v. 7, n. 1, p. 1–12, 2017.

KAUSHIK, K.; BHARDWAJ, A.; DAHIYA, S. Framework to analyze and exploit the smart home iot firmware. *Measurement: Sensors*, Elsevier, v. 37, p. 101406, 2025.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. *Computer*, IEEE, v. 36, n. 1, p. 41–50, 2003.

KOLEHMAINEN, A. Secure firmware updates for iot: a survey. In: IEEE. *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. [S.l.], 2018. p. 112–117.

KOMNINOS, N.; PHILIPPOU, E.; PITSILLIDES, A. Survey in smart grid and smart home security: Issues, challenges and countermeasures. *IEEE Communications Surveys & Tutorials*, IEEE, v. 16, n. 4, p. 1933–1954, 2014.

KRUPITZER, C.; ROTH, F. M.; VANSYCKEL, S.; SCHIELE, G.; BECKER, C. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing*, Elsevier, v. 17, p. 184–206, 2015.

KUMER, S. A.; KANAKARAJA, P.; TEJA, A. P.; SREE, T. H.; TEJASWNI, T. Smart home automation using ifttt and google assistant. *Materials Today: Proceedings*, Elsevier, v. 46, p. 4070–4076, 2021.

LAPRIE, J.-C. Dependability: Basic concepts and terminology. In: *Dependability: Basic Concepts and Terminology: In English, French, German, Italian and Japanese*. [S.l.]: Springer, 1992. p. 3–245.

LEE, B.; LEE, J.-H. Blockchain-based secure firmware update for embedded devices in an internet of things environment. *The Journal of Supercomputing*, Springer, v. 73, n. 3, p. 1152–1167, 2017.

LEHMAN, M. M. Laws of software evolution revisited. In: SPRINGER. *European Workshop on Software Process Technology*. [S.l.], 1996. p. 108–124.

LIN, H.; BERGMANN, N. W. Iot privacy and security challenges for smart home environments. *Information*, Multidisciplinary Digital Publishing Institute, v. 7, n. 3, p. 44, 2016.

LIN, Y.-H. Novel smart home system architecture facilitated with distributed and embedded flexible edge analytics in demand-side management. *International Transactions on Electrical Energy Systems*, Wiley Online Library, v. 29, n. 6, p. e12014, 2019.

LIU, J.; ZHANG, C.; FANG, Y. Epic: A differential privacy framework to defend smart homes against internet traffic analysis. *IEEE Internet of Things Journal*, IEEE, v. 5, n. 2, p. 1206–1217, 2018.

MADAKAM, S.; LAKE, V.; LAKE, V.; LAKE, V. et al. Internet of things (iot): A literature review. *Journal of Computer and Communications*, Scientific Research Publishing, v. 3, n. 05, p. 164, 2015.

MARIKYAN, D.; PAPAGIANNIDIS, S.; ALAMANOS, E. A systematic review of the smart home literature: A user perspective. *Technological Forecasting and Social Change*, Elsevier, v. 138, p. 139–154, 2019.

MCKINLEY, P. K.; SADJADI, S. M.; KASTEN, E. P.; CHENG, B. H. Composing adaptive software. *Computer*, IEEE, v. 37, n. 7, p. 56–64, 2004.

MORAN, B.; MERIAC, M.; TSCHOFENIG, H. Firmware manifest format. *Internet Engineering Task Force, Internet-Draft draft-moran-suit-manifest-01*, 2018.

MORAN, B.; MERIAC, M.; TSCHOFENIG, H.; BROWN, D. A firmware update architecture for internet of things devices. *Internet Engineering Task Force, Internet-Draft*, 2019.

NAEEM, H.; ULLAH, F.; KREJCAR, O.; ALSIRHANI, A.; ZHAO, Y. Adversarial malware detection on consumer devices using optimized image-based ensembles. *IEEE Consumer Electronics Magazine*, IEEE, 2024.

NIELES, M.; DEMPSEY, K.; PILLITTERI, V. Y. An introduction to information security. *NIST special publication*, v. 800, p. 12, 2017.

NUNES, D. S.; ZHANG, P.; SILVA, J. S. A survey on human-in-the-loop applications towards an internet of all. *IEEE Communications Surveys & Tutorials*, IEEE, v. 17, n. 2, p. 944–965, 2015.

PASQUALE, L.; RAMKUMAR, K.; CAI, W.; MCCARTHY, J.; DOHERTY, G.; NUSEIBEH, B. Sustainable adaptive security. *arXiv preprint arXiv:2306.04481*, 2023.

PHILIP, S. J.; LUU, T. J.; CARTE, T. There's no place like home: Understanding users' intentions toward securing internet-of-things (iot) smart home networks. *Computers in Human Behavior*, Elsevier, v. 139, p. 107551, 2023.

POLK, T.; MCKAY, K.; CHOKHANI, S. Guidelines for the selection, configuration, and use of transport layer security (tls) implementations. *NIST special publication*, v. 800, n. 52, p. 32, 2014.

PPLWARE. *Mas afinal, qual é a diferença entre Firmware, Driver e Software?* 2020. Disponível em: <https://pplware.sapo.pt/tutoriais/diferenca-firmware-driver-software>.

PRIYA, S. B. D. et al. Building greater iot security and trust. *IJRAR-International Journal of Research and Analytical Reviews (IJRAR)*, IJRAR (www. ijrar. org), v. 5, n. 3, p. 62–67, 2018.

RIBEIRO, A. d. R. L.; ALMEIDA, F. M. de; MORENO, E. D.; MONTESCO, C. A. A management architectural pattern for adaptation system in internet of things. In: IEEE. *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*. [S.l.], 2016. p. 576–581.

SALEHIE, M.; TAHVILDARI, L. Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TAAS)*, ACM New York, NY, USA, v. 4, n. 2, p. 1–42, 2009.

SCHENKLUHN, M.; KNIERIM, M. T.; KISS, F.; WEINHARDT, C. Connecting home: Human-centric setup automation in the augmented smart home. In: *Proceedings of the CHI Conference on Human Factors in Computing Systems*. [S.l.: s.n.], 2024. p. 1–16.

SCHIEFER, M. Smart home definition and security threats. In: IEEE. *2015 ninth international conference on IT security incident management & IT forensics*. [S.l.], 2015. p. 114–118.

SEDGEWICK, P. E.; LEMOS, R. de. Self-adaptation made easy with blockchains. In: IEEE. *2018 IEEE/ACM 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. [S.l.], 2018. p. 192–193.

SHEBLI, H. M. Z. A.; BEHESHTI, B. D. A study on penetration testing process and tools. In: IEEE. *2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. [S.l.], 2018. p. 1–7.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Operating system concepts*. [S.l.]: John Wiley & Sons, 2006.

SILVA, G. R. da; SANTOS, A. L. dos. Adaptive Access Control for Smart Homes Supported by Zero Trust for User Actions. *IEEE Transactions on Network and Service Management*, p. 1–1, 2024.

SIVARAMAN, V.; GHARAKHEILI, H. H.; VISHWANATH, A.; BORELI, R.; MEHANI, O. Network-level security and privacy control for smart-home iot devices. In: IEEE. *2015 IEEE 11th International conference on wireless and mobile computing, networking and communications (WiMob)*. [S.l.], 2015. p. 163–167.

SLADE, P.; ATKESON, C.; DONELAN, J. M.; HOUDIJK, H.; INGRAHAM, K. A.; KIM, M.; KONG, K.; POGGENSEE, K. L.; RIENER, R.; STEINERT, M. et al. On human-in-the-loop optimization of human–robot interaction. *Nature*, Nature Publishing Group UK London, v. 633, n. 8031, p. 779–788, 2024.

STEGER, M.; DORRI, A.; KANHERE, S. S.; RÖMER, K.; JURDAK, R.; KARNER, M. Secure wireless automotive software updates using blockchains: A proof of concept. In: *Advanced Microsystems for Automotive Applications 2017*. [S.l.]: Springer, 2018. p. 137–149.

STOLOJESCU-CRISAN, C.; CRISAN, C.; BUTUNOI, B.-P. An iot-based smart home automation system. *Sensors*, MDPI, v. 21, n. 11, p. 3784, 2021.

TAHERISADR, M.; FARUQUE, M. A. A.; ELMALAKI, S. Erudite: Human-in-the-loop iot for an adaptive personalized learning system. *IEEE Internet of Things Journal*, IEEE, 2023.

TAHERISADR, M.; FARUQUE, M. A. A.; ELMALAKI, S. ERUDITE: Human-in-the-Loop IoT for an Adaptive Personalized Learning System. *IEEE Internet of Things Journal*, v. 11, n. 8, p. 14532–14550, 2024.

TAHERISADR, M.; STAVROULAKIS, S. A.; ELMALAKI, S. adaparl: Adaptive privacy-aware reinforcement learning for sequential decision making human-in-the-loop systems. In: *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation*. [S.l.: s.n.], 2023. p. 262–274.

TAN, C. J.; MOHAMAD-SALEH, J.; ZAIN, K. A. M.; AZIZ, Z. A. A. Review on firmware. In: *Proceedings of the International Conference on Imaging, Signal Processing and Communication*. [S.l.: s.n.], 2017. p. 186–190.

TAN, L.; WANG, N. Future internet: The internet of things. In: IEEE. *2010 3rd international conference on advanced computer theory and engineering (ICACTE)*. [S.l.], 2010. v. 5, p. V5–376.

TIINSIDE. *Segurança em IoT é um desafio permanente para o mercado*. 2021. Disponível em: <https://tiinside.com.br/24/02/2021/seguranca-em-iot-e-um-desafio-permanente-para-mercado/>.

VATS, P.; MANDOT, M.; GOSAIN, A. A comprehensive literature review of penetration testing & its applications. In: IEEE. *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*. [S.l.], 2020. p. 674–680.

WEYNS, D. Software engineering of self-adaptive systems: an organised tour and future challenges. *Chapter in Handbook of Software Engineering*, Springer, 2017.

YAR, H.; IMRAN, A. S.; KHAN, Z. A.; SAJJAD, M.; KASTRATI, Z. Towards smart home automation using iot-enabled edge-computing paradigm. *Sensors*, MDPI, v. 21, n. 14, p. 4932, 2021.

YUAN, E.; ESFAHANI, N.; MALEK, S. A systematic survey of self-protecting software systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, ACM New York, NY, USA, v. 8, n. 4, p. 1–41, 2014.

YUAN, E.; MALEK, S.; SCHMERL, B.; GARLAN, D.; GENNARI, J. Architecture-based self-protecting software systems. In: *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures*. [S.l.: s.n.], 2013. p. 33–42.

ZANDBERG, K.; SCHLEISER, K.; ACOSTA, F.; TSCHOFENIG, H.; BACCELLI, E. Secure firmware updates for constrained iot devices using open standards: A reality check. *IEEE Access*, IEEE, v. 7, p. 71907–71920, 2019.

ZHOU, W.; JIA, Y.; YAO, Y.; ZHU, L.; GUAN, L.; MAO, Y.; LIU, P.; ZHANG, Y. Discovering and understanding the security hazards in the interactions between iot devices, mobile apps, and clouds on smart home platforms. In: *28th {USENIX} Security Symposium ({USENIX} Security 19)*. [S.l.: s.n.], 2019. p. 1133–1150.

ZIKRIA, Y. B.; YU, H.; AFZAL, M. K.; REHMANI, M. H.; HAHM, O. *Internet of things (IoT): Operating system, applications and protocols design, and validation techniques*. [S.l.]: Elsevier, 2018.

APÊNDICE A – NOTAÇÃO BNF DO PLANO DE ADAPTAÇÃO

```
<start> ::= "Execute " (<adb_command> | <cli_command>) <end_planner>

<adb_command> ::= <launch_app> | <navigate_taps> | <force_stop>
<launch_app> ::= "adb shell monkey -p " <package> " -c android.intent.category.
    ↪ LAUNCHER 1"
<package> ::= "com.amazon.dee.app"
            | "io.homeassistant.companion.android"
            | "com.intelbras"
            | "com.android.settings"
            | "com.coolkit"
            | "com.macrovideo.v380"
<navigate_taps> ::= <input_tap> [<navigate_taps>]
<input_tap> ::= "adb shell input tap " <coordinate>
<coordinate> ::= <digit> <digit> <digit> " " <digit> <digit> <digit>
<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
<force_stop> ::= "adb shell am force-stop " <package>

<cli_command> ::= <login_api> <CLIexecute>
<login_api> ::= "apiros.login(" <user> "," <passw> ")"
<user> ::= <identifier>
<identifier> ::= <letter> | <digit>
<letter> ::= "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"
<passw> ::= <identifier>
<inputsentence> ::= "inputsentence = [" <action> "]"
<action> ::= <mitigate_one> | <prevent_all> | <disable_one>
<mitigate_one> ::= "/ip/firewall/filter/add action=drop chain=input dst-address="
    ↪  <ip> " dst-port=" <port> " protocol=" <protocol>
<ip> ::= <octet> "." <octet> "." <octet> "." <octet>
<octet> ::= <digit>
          | <digit> <digit>
          | "1" <digit> <digit>
          | "2" ("0" | "1" | "2" | "3" | "4") <digit>
          | "25" ("0" | "1" | "2" | "3" | "4" | "5")
<port> ::= <digit>
         | <digit> <digit>
         | <digit> <digit> <digit>
         | <digit> <digit> <digit> <digit>
         | <five_digits>
<five_digits> ::= <range_1_to_5> <digit> <digit> <digit> <digit>
                | "6" <range_0_to_4> <digit> <digit>
                | "6" "5" <range_0_to_5> <range_0_to_3> <range_0_to_5>
<range_0_to_5> ::= "0" | "1" | "2" | "3" | "4" | "5"
<range_1_to_5> ::= "1" | "2" | "3" | "4" | "5"
```

```
<range_0_to_4> ::= "0" | "1" | "2" | "3" | "4"
<range_0_to_3> ::= "0" | "1" | "2" | "3"
<protocol> ::= "HTTP" | "HTTPS" | "FTP" | "SSH" | "TCP" | "UDP" | "ICMP" | "
    ↪ TELNET"
<prevent_all> ::= "/ip/firewall/filter/add action=drop chain=input dst-address
    ↪ =0.0.0.0/0 dst-port=" <port>
<disable_one> ::= "/ip firewall filter add chain=forward action=drop src-address=
    ↪ " <ip>
<CLIexecute> ::= "apiros.writeSentence(" <action> ")"

<end_planner> ::= "Finish Planner"
<end> ::= "Adaptation process complete"
```