UNIVERSIDADE
BEIRA INTERIOR

Towards a Framework for System and Attack Modelling and Mapping of Requirements and Technology for the Internet of Things

João Bernardo Ferreira Sequeiros

Tese para obtenção do Grau de Doutor em
Engenharia Informática
(3º ciclo de estudos)

Orientador: Professor Doutor Pedro Ricardo Morais Inácio

setembro de 2024

(This page is intentionally left blank.)

Provas públicas realizadas em 4 de setembro de 2024 com início às 14:30 horas.

Composição do júri

Presidente

Doutor Mário Marques Freire

Professor Catedrático da Faculdade de Engenharia da Universidade da Beira Interior

Vogais

Doutor Miguel Nuno Dias Alves Pupo Correia

Professor Catedrático do Instituto Superior Técnico da Universidade de Lisboa

Doutor Pedro Ricardo Morais Inácio

Professor Associado da Faculdade de Engenharia da Universidade da Beira Interior

Doutor Tiago José dos Santos Martins da Cruz

Professor Associado da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Doutora Virgínia Franqueira

Professora Auxiliar da Universidade de Kent, Reino Unido

Doutor Bruno Miguel Correia da Silva

Professor Auxiliar da Faculdade de Engenharia da Universidade da Beira Interior

(This page is intentionally left blank.)

Declaração de Integridade

Eu, João Bernardo Ferreira Sequeiros, que abaixo assino, estudante com o número de inscrição 2055 do 3º ciclo de estudos em Engenharia Informática da Faculdade de Engenharia, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridade da Universidade da Beira Interior**.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã, 2024/09/10

Assinado por: **JOÃO BERNARDO FERREIRA SEQUEIROS**
Data: 2024.09.10 11:14:01+01'00'

(This page is intentionally left blank.)

Acknowledgements

It is said that wisdom begins with the statement *"I do not know"* when facing something unexpected or novel. This pursuit of knowledge and of learning something new, and the passion for sharing our knowledge with others have grown within me during this Ph.D., and this is, in no small measure, due to the dedication of my supervisor, Professor Pedro Inácio. The first *Thank You* is to him, for his patience, for what we have shared over these years, and for all his work, supervision, and motivation. It was him who encouraged me in entertaining the idea of pursuing a Ph.D., which, before knowing him, was not even a remote possibility in my head.

I would remiss, however, if I did not give the proper recognition to my parents, who have, through all these years, been the engine that kept me going. They have always been there for me, and were my rock during these years, even in the hardest moments. I cannot thank them enough, not only for the support during this period, but for everything, always.

I also leave a special *Thank You* to a few special friends, notably for all the times they put up with my long tirades, when things were going less smoothly. I cannot name all of you here, but rest assured you have my deep appreciation.

Last, but not least, I would like to acknowledge several of my lab colleagues and those who participated in the different unrelated projects I have been involved in, some of which comprising exotic adventures in regards to their topics and to what we needed to quickly learn to be able to handle them.

The work described in this thesis was carried out at the *Secure and Intelligent Networked Software Systems Laboratory* (**sins**-lab) and at the *Instituto de Telecomunicações* – Covilhã Delegation, while part of the *Network Applications and Services* research group (**nas**-cv), located at the Universidade da Beira Interior, Covilhã, Portugal. This research work was partially supported by the Ph.D. research grant from the *Fundação para Ciência e Tecnologia* (FCT) with reference SFRH/BD/133838/2017, by the SECUR IoT ESIGN Project, through FCT/COMPETE/FEDER funds (project with Reference Number POCI-01-0145-FEDER-030657), and also by operation Centro-01-0145-FEDER-000019 – C4 – *Centro de Competências em Cloud Computing*, co-financed by the European Regional Development Fund (FEDER) through the *Programa Operacional Regional do Centro* (Centro 2020), in the scope of the *Sistema de Apoio à Investigação Científica e Tecnológica - Programas Integrados de IC&DT*.

(This page is intentionally left blank.)

Abstract

The proliferation of Internet of Things (IoT) devices has been expanding several domains, offering unprecedented connectivity and convenience. However, this surge in interconnected devices has brought forth significant security challenges, as constrained budgets and development time leave security in a secondary role, or even non-existent. This is compounded upon by small design and development teams, where security expertise is reduced and lacking, creating a landscape of IoT systems that are unsecured and ripe for attack by malicious actors. The data gathered by these devices, their general lack of security, and the possibility of serving ans entry points to otherwise more secure systems, makes them increasingly tempting targets for exploration and exploitation.

This thesis attempts to bridge the gap of aiding in the secure IoT system development, by approaching the issue of security in IoT from a standpoint of low knowledge and/or low expertise in IoT security. The first step towards the main goal is the extensive survey of existing IoT architectures and modeling tools, to aid in identifying the main challenges in secure IoT development and what can be improved or built upon. The second phase advances upon what was surveyed, by proposing an IoT system model that encompasses a large set of IoT ecosystems, and that embeds security in its essence, by identifying, for each system component, what are its critical security requirements, and what are the most attractive targets for an attacker on the given component. This model is complemented by the creation of an attack taxonomy, that attempts to take the most common attacks on IoT, and identifying where in the system those attacks may occur.

To further aid the development process and provide a practical substrate to the Doctor of Philosophy (degree) (Ph.D.) work, an attack modeling tool named *Attack Trees for IoT (ATIoT)* is presented as a means to identify, starting from a system description given through a direct answer questionnaire, the attacks the system may be more susceptible to, providing the user with a set of attack trees, together with detailed node descriptions, of the identified attacks for the described system. Joining ATIoT, other existing tools are mapped to the proposed model, to further aid in identifying where security requirements, best practices, guidelines, security mechanisms and potential threats in an IoT system should be applied or can be found, further enhancing the usefulness of such tools.

Motivated by the profound transformation that Artificial Intelligence (AI) is causing in the technological world, and the always fast advancing security area, a series of experiments of applying different AI mechanisms to the developed tools are also detailed herein. They specifically concern the application of classification models to the elicitation of security requirements, and the use of Large Language Models (LLMs) for identifying potential attacks from a textual system description. The thesis presents the results of these experiments, which

show the promise of applying such methodologies to the process of security engineering.

Main conclusions include achieving the goal of creating a panoply of mechanisms and tools that aid the development of secure IoT systems, that were designed towards being used by developers with low or no security background and expertise. It was also concluded that AI methods can aid in the maintaining of such tools and mechanisms, ensuring their validity in a longer time period, a challenge always present in fast-paced, always evolving areas.

Keywords

Artificial-Intelligence, Attack-Modeling, Cybersecurity, IoT, Internet of Things, Security, Security-by-Construction, Security-by-Design, System-Modeling

Resumo

A proliferação de dispositivos para a Internet das Coisas (IoT) tem vindo a expandir vários domínios, oferecendo conectividade e conveniência sem precedentes. No entanto, este aumento de dispositivos interligados traz consigo significativos desafios de segurança, uma vez que orçamentos limitados e tempos de desenvolvimento insuficientes deixam a segurança num papel secundário, ou mesmo inexistente. Isto é agravado por equipas de desenho e desenvolvimento reduzidas, onde a experiência em segurança é limitada e escassa, criando um cenário de sistemas IoT não seguros e propícios a ataques por agentes maliciosos. Os dados recolhidos por estes dispositivos, a sua geral falta de segurança, e a possibilidade de servirem como pontos de entrada para sistemas geralmente mais seguros, torna-os alvos cada vez mais tentadores para ataques e exploração.

Esta tese procura colmatar a lacuna de auxílio no desenvolvimento seguro de sistemas IoT, abordando a questão da segurança em IoT a partir de um ponto de vista de baixo conhecimento e/ou baixa experiência em segurança. O primeiro passo rumo ao objetivo principal é o extenso levantamento das arquiteturas e ferramentas de modelação para IoT existentes, para ajudar a identificar os principais desafios no desenvolvimento seguro de sistemas IoT, e o que pode ser melhorado ou desenvolvido. A segunda fase avança sobre o que foi levantado, propondo umm modelo de sistema para a IoT que engloba um conjunto de ecossistemas da IoT, e que incorpora segurança na sua essência, identificando, para cada componente do sistema, quais são os seus requisitos de segurança, e quais são os alvos mais atrativos para um atacante num dado componente. Este modelo é complementado pela criação de uma taxonomia de ataques, que tenta identificar os ataques mais comuns na IoT e onde no sistema esses ataques têm maior probabilidade de ocorrerem.

Para auxiliar ainda mais o processo de desenvolvimento, e fornecer um substrato prático ao trabalho de doutoramento, é apresentada uma ferramenta de modelação de ataques, denominada *ATIoT*, como um meio de identificar, a partir de uma descrição do sistema, fornecida através de um questionário de resposta direta, os ataques aos quais o sistema pode ser mais suscetível, fornecendo assim ao utilizador um conjunto de árvores de ataque, juntamente com descrições detalhadas dos vários nós, dos ataques identificados para o sistema descrito. Juntamente com a *ATIoT*, outras ferramentas existentes são mapeadas para o modelo proposto, para auxiliar ainda mais na identificação de onde requisitos de segurança, melhores práticas, diretrizes, mecanismos de segurança e ameaças potenciais num sistema da IoT devem ser aplicados ou podem ser encontrados, aumentando ainda mais a utilidade dessas ferramentas.

Motivadas pela profunda transformação que a Inteligência Artificial (IA) está a causar no mundo tecnológico, e pela constante evolução das áreas de segurança e da IoT, uma série de

experiências de aplicação de diferentes mecanismos de IA às ferramentas desenvolvidas também são detalhadas neste documento. Especificamente, estas experiências incidem sobre a aplicação de modelos de classificação na elicitação de requisitos de segurança, e sobre uso de grandes modelos de linguagem para identificar potenciais ataques a partir de uma descrição textual do sistema. A tese apresenta os resultados destas experiências, que demonstram validade na aplicação destas metodologias no processo de engenharia de segurança.

As principais conclusões incluem o alcançar do objetivo de criar vários mecanismos e ferramentas que auxiliem no desenvolvimento de sistemas da IoT seguros, concebidos para serem utilizados por desenvolvedores com baixa ou nenhuma experiência em segurança. Também foi concluído que os métodos de IA podem auxiliar na manutenção dessas ferramentas e mecanismos, garantindo a sua validade num período de tempo mais longo, um desafio sempre presente em áreas em constante evolução e rápido desenvolvimento.

Palavras-Chave

Segurança; Internet-das-Coisas; Segurança-por-Construção; Modelação-de-Sistemas; Modelação-de-Ataques; Inteligência-Artificial

Resumo Alargado

Introdução

Motivação e Enquadramento

A Internet das Coisas (IoT) (expressão criada por Kevin Ashton em 1999) [1] pode ser descrita como o adicionar de conectividade a uma infinidade de dispositivos físicos (ou *coisas*), o que permite o controlo e a comunicação com esses dispositivos através de uma rede local e/ou da Internet. A maioria destes dispositivos possui uma coleção de sensores embutidos, sendo capazes de medir o ambiente em seu redor. Esta é uma das duas principais funções da IoT [2], denominada como *sentir*, sendo a segunda principal função denominada por *atuação*. A *atuação* diz respeito à capacidade de um dispositivo intervir e/ou interagir fisicamente no e com o espaço onde o dispositivo está localizado. Muitos dispositivos combinam estas duas facetas, sendo capazes tanto de sentir o mundo em seu redor, como interagir diretamente com ele, e podem ser aplicados a uma infinidade de ambientes diferentes, desde espaços domésticos e de escritório até indústria, saúde, rede elétrica, cidades, agricultura, e transporte, entre outros. É um dos principais pilares do que muitos consideram a quarta revolução industrial, devido à sua íntima conexão com os Sistemas Ciber-Físicos. Esta abrangência da IoT geralmente resulta num aumento de eficiência, confiabilidade e conforto, e as mudanças já são visíveis numa ampla gama de ambientes.

A proliferação de dispositivos IoT nos últimos anos, e sua potencial ubiquidade, tornam-os em alvos apetecíveis para atividades maliciosas. Isto, juntamente com a pressão de lançar novos produtos rapidamente no mercado, bem como as restrições a nível computacional de alguns destes dispositivos, fazem com que aspetos de segurança sejam muitas vezes deixados para segundo plano, enquanto a tecnologia amadurece. De acordo com a IoT *Analytics*, até o ano de 2027, mais de 29 mil milhões de dispositivos IoT estarão ligados à Internet [3]. Estes dispositivos terão os seus próprios problemas de segurança (por exemplo, *boot* seguro ou atualizações seguras) e herdarão muitos dos problemas conhecidos da Internet (por exemplo, problemas de controlo de acesso e autenticação). Como os dispositivos IoT são significativamente diferentes dos dispositivos tradicionais que utilizam protocolo IP, estão expostos a novas formas de ataque, incluindo ataques físicos (uma vez que muitos dispositivos IoT estão localizados em ambientes abertos) e ataque pela rede. Os desafios de segurança dos sistemas IoT vão desde autenticação até à confiança, integridade, privacidade de dados, comunicações em redes de sensores sem fio, tecnologia RFID, criptografia e segurança de ponta a ponta, ou controlo de acesso.

Não é trivial discernir quais as tecnologias, protocolos ou controlos a aplicar em termos de

segurança para cada caso de uso, ao desenvolver um dispositivo ou sistema para a IoT, e esta dificuldade, combinada com a experiência limitada em (ciber)segurança da maioria dos desenvolvedores, juntamente o ambiente em rápida evolução que envolve a IoT, pode ser vista como a principal motivação para este trabalho. Estas dificuldades decorrem do número de configurações, aplicações e tecnologias diferentes que compõem o ecossistema da IoT, o que dificulta o desenho e o desenvolvimento de sistemas IoT, embutindo segurança durante estes processos. A isto acresce o facto de, no processo de engenharia de software, adicionar ou efetuar mudanças profundas após o processo de desenvolvimento (ou seja, corrigir uma vulnerabilidade num sistema próximo da sua conclusão, ou já implementado) tem maiores implicações em termos de custo e tempo [4]. A criação de dispositivos seguros é um desafio difícil de enfrentar. Este é um dos fatores que dificultam a adoção de sistemas IoT [5], pois os problemas de segurança podem causar períodos sem serviço, perda de confiança, perda de reputação e até mesmo danos irreparáveis às operações.

Os seguintes desafios devem então ser resolvidos ou tidos em conta, para se atingir o objetivo de ter sistemas IoT *seguros por construção* e, como consequência, garantir maior adoção e melhorar a reputação destes sistemas:

- **Heterogeneidade** – a variedade de dispositivos, tecnologias de comunicação ou cenários de aplicação de sistemas IoT aumentam a dificuldade em criar solução genéricas úteis. Tanto o hardware como o software na IoT são heterogéneos, com sistemas diferentes a terem componentes, dispositivos e software diferentes, e diferentes capacidades computacionais e requisitos energéticos;
- **Falta de conhecimentos de segurança** – a vasta maioria dos desenvolvedores têm conhecimentos limitados em termos de (ciber)segurança e de engenharia de segurança. O tópico de segurança, em si mesmo, não é de abordagem simples, e os desenvolvedores, muitas vezes, não conseguem dedicar tempo a aprender cibersegurança e conceitos relacionados durante as diferentes fases do desenho e desenvolvimento de um projeto. É também uma área em constante desenvolvimento, aumentando a dificuldade se se manterem a par nas melhores práticas e tecnologias a utilizar;
- **Falta de uniformização** – estabelecer arquiteturas, mecanismos, boas práticas ou guias é de dificuldade acrescida sem uniformização. Muitas das ferramentas e arquiteturas para a IoT abordam casos específicos, e não são aplicáveis à maioria dos sistemas;
- **Falta de foco em segurança nas arquiteturas existentes** – as várias arquiteturas propostas para a IoT focam-se nas funcionalidades e no funcionamento dos sistemas como um todo, mas deixam a segurança para segundo plano, ou não a abordam de todo. Isto dificulta a tarefa de criar um sistema seguro, com base numa arquitetura, uma tarefa não-trivial;
- **Falta de ferramentas de apoio ao desenho e implementação** – olhando para

os desafios anteriores, é clara a falta de ferramentas e mecanismos que auxiliem o desenvolvimento de sistemas IoT seguros, através da disponibilização de, por exemplo, recomendações ou guias de boas práticas a seguir para este ecossistema.

Como subsequentemente discutido em maior detalhe, a investigação realizada e apresentada nesta tese foca-se na criação de uma estrutura de suporte ao desenho e criação de sistemas IoT seguros. Por abranger um conjunto variado de áreas de conhecimento, o enquadramento envolve diferentes áreas como criptografia, redes ou modelação de sistemas. As cinco categorias e descritores do Sistema de Classificação Informática (CCS) da *Association for Computing Machinery* (ACM) que melhor dão o enquadramento este trabalho são as seguintes:

- **Segurança e Privacidade ~Segurança de sistemas;**
- **Segurança e Privacidade ~Engenharia de segurança do software;**
- **Segurança e Privacidade ~Segurança do software e das aplicação;**
- *Segurança e Privacidade ~Segurança das redes;*
- *Segurança e Privacidade ~Segurança de sistemas embutidos.*

Definição do Problema e Formulação da Tese

As questões de segurança são comuns nos diferentes cenários da IoT e afetam as diferentes partes que compõem um sistema. Embora não haja uma arquitetura padronizada para a IoT, três camadas principais estão sempre presentes [6]: perceção, rede e aplicação, e a segurança é necessária em qualquer uma das três, já que ataques podem ser direcionados a todas elas. Embora uma grande parte dos ataques sejam herdados de outros paradigmas de computação, a IoT está particularmente exposta a uma quantidade variadade de ataques (alguns deles, novos e específicos) devido às suas características únicas. A camada de perceção, especificamente, é a mais vulnerável, pois consiste em nós computacionais com poucos recursos, com limitações em termos de capacidade de processamento e, muitas vezes, também de energia [7]. Devido a isto, os mecanismos de segurança comuns noutros paradigmas de computação não podem ser implementados da mesma forma nestes dispositivos. Os dispositivos IoT também são particularmente suscetíveis a ataques como Negação de Serviço (DoS), seja como alvo, ou quando infetados, utilizados como parte de *botnets* [8]. Como demonstrado por ataques e violações de dados recentes, os dispositivos IoT têm-se tornado num foco dos atacantes, devido à ubiquidade em termos de uso, e à falta de segurança e facilidade de exploração [9]. O desenvolvimento e a disseminação rápidos deste tipo de sistemas tornam este problema desafiante, já que novas ameaças surgem todos os dias.

O parágrafo anterior descreve o problema que, em última instância, necessita ser resolvido. No entanto, este pode ser ainda decomposto ou transformado noutros problemas. A falta de conhecimento em cibersegurança nas equipas de desenvolvimento é um exemplo, e é um problema que não pode ser facilmente resolvido num futuro próximo. Preencher essa lacuna no curto prazo pode ser alcançado por meio de ferramentas e estruturas que definem e, de alguma forma, forçam a incorporação da segurança em todo o processo de desenvolvimento. A principal problemática abordada nesta tese de Doutoramento é, portanto, a falta de ferramentas e mecanismos que possam auxiliar no desenvolvimento de sistemas IoT seguros.

A Proposta de Tese:

A proposta desta tese é que será possível projetar e desenvolver dispositivos IoT seguros por construção, desde que sejam fornecidas aos desenvolvedores ferramentas específicas para os diferentes aspetos da IoT, com foco na segurança, e concebidas para serem fáceis de utilizar. Estas ferramentas não devem exigir conhecimentos especializados em segurança, e devem apoiar nas diferentes etapas do processo de desenvolvimento e verificação.

Abordagem Adotada e Plano de Investigação

A abordagem adotada para resolver o problema foi segmentada em várias partes. Como o objetivo final é a criação de um conjunto de ferramentas, a ideia passou por abordar o problema identificando potenciais ataques, de acordo com as especificações do sistema, e técnicas de modelação de sistemas e ataques. Posteriormente, estas seriam integradas no processo de desenho e desenvolvimento.

A primeira fase consistiu num levantamento do panorama atual em termos de modelação de sistemas e ataques para a IoT. O objetivo era selecionar a melhor abordagem para definir, em primeiro lugar, os ataques mais comuns em sistemas IoT, como estes podem interagir com um sistema, tendo em conta as potenciais intrusões a que um dispositivo ou sistema está exposto. O foco incidiu sobre o que deveria ser uma ferramenta de modelação de ataques (e que técnica deveria utilizar, como o Modelo Diamante, as Árvores de Ataque ou a Cadeia de Eliminação), e na sua facilidade de utilização.

A segunda fase consistiu na integração da primeira parte do trabalho no processo de engenharia de dispositivos e sistemas IoT. Isto foi feito através da definição de um modelo de sistema pensado tendo considerações de segurança em mente para cada um dos seus componentes e conceitos, auxiliando na deteção e resolução de problemas de segurança, através de uma abordagem de segurança por desenho ou construção, onde todo o processo de desenvolvimento é feito tendo em conta a segurança. Através do modelo, o processo é simplificado,

permitindo que equipas com baixa experiência em segurança possam projetar e desenvolver um sistema que seja, por desenho, o mais seguro possível.

A terceira fase consistiu no desenvolvimento e adaptação de ferramentas, com base na investigação realizada durante as duas primeiras fases. O objetivo foi contribuir para os principais objetivos da segunda fase, e também para o objetivo principal desta tese. Isto foi feito através do mapeamento das características do sistema com diferentes definições relacionadas com a segurança, tais como ataques, requisitos, mecanismos e diretrizes, e o mapeamento destes com os conceitos definidos na segunda fase.

A quarta e última fase concentrou-se no estudo e experimentação com as possibilidades emergentes de adaptação da Inteligência Artificial (IA) às soluções desenvolvidas e adaptadas durante a terceira fase. A ideia foi aprofundar um dos objetivos secundários, relacionado com a longevidade das soluções desenvolvidas. Esta fase foi abordada através de trabalho experimental, com o foco na verificação, em primeiro lugar, da viabilidade real da aplicação de tais métodos (de IA) e, em segundo lugar, do desempenho efetivamente alcançado.

Principais Contribuições Científicas

Embora a motivação inicial e a hipótese para este projeto de Doutoramento tenham permanecido as mesmas ao longo de todo o trabalho, o ambiente dinâmico e em constante evolução (especialmente nas áreas de cibersegurança e IA) incentivou a exploração de diferentes abordagens e áreas ao longo do trabalho, o que também se reflete nas contribuições. As principais contribuições científicas resultantes deste trabalho podem ser descritas da seguinte forma:

- Uma **Revisão da Literatura sobre Modelação de Sistemas e Ataques para a IoT** – a primeira contribuição significativa da investigação é um estudo abrangente do panorama da modelação de ataques e sistemas para a IoT. A revisão produzida aborda e investiga diferentes técnicas e ferramentas existentes para modelação de sistemas e ataques, identificando lacunas nos tópicos específicos.

  Esta parte do trabalho foi o tema principal de um artigo submetido e publicado na revista científica internacional ACM *Computing Surveys* [10] e serviu adicionalmente como base para o Capítulo 2 desta tese.

  Publicação:

  – João B. F. Sequeiros, Francisco Chimuco, Musa G. Samaila, Mário M. Freire, and Pedro R. M. Inácio, Attack and System Modeling Applied to IoT, Cloud and Mobile Ecosystems: Embedding Security by Design, ACM Computing Surveys (CSUR), 53(2):25, March 2020.

- **Uma Proposta de um Modelo de Sistema para a IoT** – a segunda contribuição consiste num novo modelo de sistema global para a IoT, desenvolvido com cibersegurança em mente, identificando ainda os requisitos de segurança e as interconexões entre os diferentes componentes, e definindo uma série de ecossistemas de IoT aos quais o modelo pode ser aplicado. Este trabalho foi um dos temas de um artigo submetido e aceite na Conferência AINA 2024 [11], e também foi objeto de um artigo submetido à revista internacional *Springer* International Journal of Information Security (IJIS) [12]. Esta parte do trabalho serviu como base para a primeira parte do Capítulo 3 desta tese.

  Publicações:

  - João B. F. Sequeiros, Francisco T. Chimuco, Tiago M. C. Simões, Mário M. Freire, and Pedro R. M. Inácio, An Approach to Attack Modeling for the IoT: Creating Attack Trees from System Descriptions, in Proceedings of The 38th International Conference on Advanced Information Networking and Applications (AINA-2024), Kitakyushu, Japan, April 17-19, 2024;
  - João B. F. Sequeiros, Francisco T. Chimuco, Tiago M. C. Simões, Mário M. Freire, and Pedro R. M. Inácio, Secure System Model for IoT, and Application Cases of AI in Security Engineering, submitted to the *Springer* IJIS journal.

- **Proposta de uma Taxonomia de Ataque Para a IoT** – a terceira contribuição é composta por uma taxonomia de ataques para a IoT, que partiu da proposta de modelo acima mencionado, onde os ataques são categorizados, fornecendo uma abordagem estruturada para compreender as diversas ameaças colocadas aos sistemas IoT. Esta contribuição também estabelece as bases para a criação de ferramentas que abordem os ataques definidos na taxonomia. Partir de uma forte ortogonalização do trabalho é de particular importância na área, uma vez que parte da investigação em cibersegurança carece deste aspeto específico. Esta contribuição foi objeto de parte do artigo submetido e aceite na Conferência AINA 2024 [11], e do artigo submetido à revista internacional *Springer* IJIS [12] (mencionados anteriormente). Esta contribuição serviu como base para a segunda parte do Capítulo 3 desta tese. Uma contribuição secundária relativa à adaptação da taxonomia proposta para aplicações móveis baseados na nuvem também foi objeto de parte de um artigo publicado na *Springer* IJIS [13], embora não seja mais elaborada aqui devido ao seu papel menos proeminente no referido artigo e por estar fora do âmbito desta tese.

  Publicações:

  - Francisco Chimuco, João B. F. Sequeiros, Carolina Lopes, Tiago M. C. Simões, Mário M. Freire and Pedro R. M. Inácio, Secure Cloud-based Mobile Apps: Attack Taxonomy, Requirements, Mechanisms, Tests and Automation, International Journal of Information Security (IJIS), 22(4): 833–867, February 2023.

- **Desenho e Implementação da Ferramenta ATIoT** – a quarta contribuição é a criação e teste da ferramenta ATIoT, que gera conjuntos de árvores de ataque e re-

spetivas descrições a partir de uma descrição de sistema fornecida. Esta contribuição inclui o desenho e criação de um conjunto de árvores de ataque, bem como a representação do conhecimento na forma de lógica de tomada de decisão. Esta ferramenta foi apresentada e descrita no artigo aceite na Conferência AINA 2024 [11] (mencionada anteriormente). Esta é a fundação sobre a qual elabora a primeira parte do Capítulo 4.

- **Adaptação das Ferramentas Security Requirements Engineering (SRE), Security Best Practices and Guidelines (SBPG), Lightweight Cryptographic Algorithm Recommendation (LWCAR) and Threat Modeling Solution (TMS) para o Modelo de Sistema Proposto** – a quinta contribuição diz respeito ao mapeamento das recomendações, guias e sugestões de várias ferramentas diferentes do projeto SECURIoTESIGN, do qual o autor desta tese foi coordenador da Atividade 2 – *Engenharia de Segurança para a IoT*. Esta adaptação é o tema da segunda parte do Capítulo 4 desta tese. Da coordenação da Atividade resultaram contribuições secundárias para dois artigos submetidos e publicados nas revistas internacionais IEEE *Access* [14] e *Elsevier* COMNET [15], na forma de auxílio na criação da lógica e do mapeamento das ferramentas durante o seu desenvolvimento. As principais contribuições do autor desta tese para estas duas publicações focaram-se especificamente na lógica, tomada de decisão e desenvolvimento das ferramentas SRE e SBPG. Embora não elaboradas com detalhe na tese, estas contribuições precederam a referida adaptação das ferramentas.

  Publicações:

  - Musa G. Samaila, João B. F. Sequeiros, Tiago Simões, Mário M. Freire, and Pedro R. M. Inácio, IoT-HarPSecA: A Framework and Roadmap for Secure Design and Development of Devices and Applications in the IoT Space, IEEE Access, 8:16462-16494, January 2020;
  - Musa G. Samaila, João B. F. Sequeiros, Carolina Lopes, Edi Aires, Tiago Simões, Mário M. Freire, and Pedro R. M. Inácio, Performance Evaluation of the SRE and SBPG Components of the IoT Hardware Platform Security Advisor Framework, Elsevier Computer Networks (COMNET), 199, November 2021.

- **Utilização de Inteligência Artificial na Automação do Mapeamento de Requisitos** – a sexta contribuição gira em torno do estudo de possíveis abordagens para automatizar o mapeamento de requisitos de segurança através do uso de técnicas de IA. O objetivo é garantir que, dado um pequeno conjunto de exemplos de treino fornecidos por um especialista, o sistema, dotado de inteligência artificial, seja capaz de extrapolar os requisitos com alta precisão, para que desenvolvimentos futuros e adições à base de conhecimento possam ser diretamente incorporados em ferramentas com mínima interação humana no processo de atualização. Esta abordagem foi tema de um artigo submetido e aceite na conferência INForum 2021 [16], e é detalhada na primeira parte do Capítulo 5 desta tese.

Publicação:

- Carolina Lopes, Joana C. Costa, João B. F. Sequeiros, Tiago M. C. Simões, Mário M. Freire, and Pedro R. M. Inácio, Machine Learning Applied to Security Requirements Elicitation: Learning From Experience, in Atas do 12º Simpósio de Informática (INForum 2021), Lisboa, Portugal, September 9-10, 2021, pp. 0-12.

- **Utilização de Grandes Modelos de Linguagem (LLMs) para Facilitar a Descrição de Entradas e Criação de Recomendações e o seu Mapeamento** – a sétima contribuição deste projeto de Doutoramento centra-se no estudo do uso de LLMs. Através de processos de ajuste fino, pretende-se capacitá-los para realizar processos de modelação, mapeamento de requisitos e recomendações de segurança, expandindo-os através da sua vasta base de conhecimento. Este trabalho foi incluído no artigo (anteriormente mencionado) submetido à revista *Springer* IJIS [12], e serve de base para a segunda parte do Capítulo 5 desta tese. Uma contribuição secundária inicial, em termos de suporte experimental, também foi realizada como precursora do trabalho que culminou na contribuição principal, publicada na revista internacional IEEE *Access* [17]. Este último trabalho, especificamente, é apenas mencionado brevemente no capítulo 5 desta tese.

  Publicação:

  - Joana Cabral Costa, Tiago Roxo, João B. F. Sequeiros, Hugo Proença, and Pedro R. M. Inácio, Predicting CVSS Metric Via Description Interpretation, IEEE Access, 10: 59125-59134, June 2022.

Estado da Arte

O capítulo 2 apresenta alguns aspetos importantes a considerar e trabalhos relacionados sobre modelação de sistemas e de ataques na IoT. Foca-se e descreve resumidamente várias das estratégias comuns mencionadas na literatura e aborda diversas soluções criadas para estes tópicos específicos de engenharia de software no contexto da IoT. Apresenta uma visão geral da modelação de sistemas de IoT, incluindo a descrição de arquiteturas e de várias ferramentas para esse efeito e nesse ecossistema, e aborda a modelação de ataques e ameaças, em termos de metodologias, classificações e ferramentas específicas para IoT. Este capítulo apresenta uma comparação dos vários trabalhos relacionados, e conclui com algumas lições aprendidas. Este capítulo baseia-se fortemente em [10].

Modelo de Sistema e Taxonomia de Ataque na Internet das Coisas

O capítulo 3 aprofunda o tema específico da modelação de sistemas e de ataques na IoT, de forma a contribuir para uma estrutura capaz de abordar um cenário de ameaças em constante evolução. O objetivo deste capítulo é duplo: primeiro, introduzir uma taxonomia abrangente de ataques na IoT e, segundo, propor um modelo de sistema sofisticado que auxilie investigadores, engenheiros e desenvolvedores na modelação de sistemas IoT e na identificação das suas vulnerabilidades. Este capítulo é baseado nas publicações [11], [12].

A primeira parte deste capítulo apresenta e elabora sobre um novo modelo de sistema projetado para apoiar a criação de sistemas IoT robustos, seguros e eficientes. Este artefacto de modelação desempenha um papel crítico na identificação de requisitos de segurança, dado que a segurança normalmente fica fora do âmbito das propostas de modelos de sistema para IoT. Nesta definição, a segurança não é considerada como um conceito de sistema, mas sim parte integrante de todos os conceitos do sistema. Isto é aprofundado na definição de cada um dos componentes, onde diferentes requisitos de segurança são mapeados para os diferentes componentes do sistema.

A natureza dinâmica dos cenários de aplicação da IoT, aliada à diversidade de dispositivos e protocolos de comunicação, exige uma compreensão precisa dos potenciais vetores de ataque. A segunda parte deste capítulo apresenta uma taxonomia de ataques especificamente adaptada à IoT. Vários vetores de ataque, os seus impactos e o *modus operandi* de atores maliciosos são explorados e descritos aqui, fornecendo uma visão abrangente do panorama de ameaças na IoT.

Considerando que a IoT promete avanços transformadores em quase todas as dimensões da sociedade, a segurança permanece uma preocupação constante. Este capítulo procura agregar o conhecimento necessário para enfrentar estas preocupações. A abordagem escolhida é abrangente, reconhecendo a natureza em constante evolução dos sistemas IoT e o panorama de ameaças dinâmico que os acompanha. Através de uma exploração profunda da taxonomia de ataques e do modelo do sistema, o conteúdo deste capítulo visa servir como uma base robusta para a segurança dos ecossistemas IoT. O trabalho apresentado neste capítulo foi fulcral para a compreensão e conceptualizaão da IoT, permitindo assim o desenvolvimento de ferramentas e a progressão geral do trabalho.

Modelação de Ataques e de Sistema na Internet das Coisas

O capítulo 4 baseia-se no modelo de sistema IoT e na taxonomia de ataques apresentados no capítulo 3 para propor soluções de modelação de ataques e sistemas. O modelo fornece uma estrutura para compreender os diferentes componentes e interações dentro de um sistema

IoT, enquanto que a taxonomia de ataques classifica e categoriza várias técnicas de ataque na IoT. A partir destas contribuições, é possível aplicá-las para desenvolver e adaptar soluções de modelação de sistemas e ataques.

Ferramentas e técnicas existentes para modelação de ataque e sistema podem ter limitações para capturar a complexidade dos sistemas IoT [10]. Ferramentas tradicionais, que podem ser genéricas ou personalizadas para outras áreas, muitas vezes têm dificuldades em lidar com a heterogeneidade dos sistemas IoT, enquanto que as técnicas de modelação podem não representar adequadamente a dinâmica e as interações entre os componentes do sistema. Este capítulo apresenta trabalhos anteriores e ferramentas desenvolvidas para o objetivo principal da engenharia de segurança da IoT, bem como novos trabalhos que visam especificamente a modelação de ataques e sistemas IoT, precisamente para abordar estas limitações.

Este capítulo detalha os processos de modelação e os mapeamentos desenvolvidos entre as características do sistema e as definições e propostas de modelação de ataque e sistema, apresentando a estrutura desenvolvida no âmbito do projeto SECURIoTESIGN [18], onde este projeto de Doutoramento foi parcialmente desenvolvido. Aborda ainda o tópico de modelação de ataques, e como pode ser utilizada para auxiliar ainda mais no desenvolvimento de sistemas IoT seguros. A ferramenta ATIoT é apresentada neste capítulo, e foi desenhada a partir da taxonomia de ataques definida anteriormente. A ferramenta é capaz de gerar conjuntos de árvores de ataque dadas as características do sistema. Esta parte do capítulo descreve o trabalho mencionado no artigo [11]. Por fim, é contextualizada a modelação de sistemas e como as suas diferentes técnicas auxiliam na estruturação do desenvolvimento. Várias ferramentas da estrutura apresentada são adaptadas ao modelo apresentado no capítulo 3, para melhorar a usabilidade dos seus resultados no desenvolvimento seguro de sistemas IoT.

Utilização de Técnicas de Inteligência Artificial para o Aprimoramento de Ferramentas

O capítulo 5 apresenta um trabalho exploratório sobre o potencial da utilização de técnicas de Aprendizagem Automática (ML), Processamento de Linguagem Natural (NLP) e o ajuste fino de um LLM (Grande Modelo de Linguagem) para automatizar, atualizar e transformar ferramentas descritas no capítulo anterior. A utilização destas técnicas é promissora para resolver um dos principais problemas que todas as ferramentas baseadas no conhecimento atual sofrem: o avanço (rápido) da tecnologia. Ao inferir conhecimento e lógica, e com acesso a grandes quantidades de dados, estas técnicas podem fornecer meios para garantir que estas ferramentas sejam automaticamente mantidas, atualizadas e relevantes. Por outro lado, o trabalho realizado no âmbito deste doutoramento poderá ser utilizado num futuro próximo

para construir os conjuntos de dados necessários para treinar *chatbots* orientados para a segurança.

O capítulo detalha várias experiências realizadas para explorar estas técnicas, e os resultados nelas obtidos. É descrita a aplicação de ML à ferramenta SRE e as experiências realizadas para avaliar o desempenho de diferentes modelos e classificadores de ML. Esta parte do capítulo é fortemente sustentada por [16]. A segunda parte do capítulo apresenta o uso de técnicas de NLP para aprimorar as ferramentas, principalmente a ferramenta ATIoT. O trabalho apresentado nesta parte do capítulo encontra-se em [12].

Conclusões e Perspetivas Futuras

O capítulo 6 sintetiza os resultados da investigação desenvolvida durante o projeto de Doutoramento, e apresentados nesta tese, identificando as principais contribuições que a mesma produziu e, através delas, reavalia os objetivos inicialmente propostos e verifica se foram atingidos. A avaliação crítica do trabalho realizado permite determinar se a investigação se encontra em concordância com o esperado. Uma das principais conclusões é que o conjunto de ferramentas e artefactos produzidos no contexto do trabalho apresentado são um contributo efetivo na direção do desenvolvimento de sistemas IoT seguros por construção. Estas conclusões confirmam a premissa de que é possível criar sistemas IoT seguros, desde que os desenvolvedores disponham de ferramentas adequadas e fáceis de usar, que não exijam um vasto conhecimento e experiência em segurança. Contribuições posteriores expandiram este conceito ao focarem-se na validação da possibilidade de utilizar IA para manter e potencialmente aprimorar estas ferramentas. Este é um desafio constante em cenários de rápido avanço, como é o caso da IoT e da segurança. A tese termina com a identificação de algumas limitações da investigação e sugestões para trabalhos futuros que possam aprofundar o conhecimento nesta área.

(This page is intentionally left blank.)

Contents

(This page is intentionally left blank.)

List of Figures

(This page is intentionally left blank.)

List of Tables

(This page is intentionally left blank.)

Acronyms and Abbreviations

6LowPAN IPv6 over Low power Wireless Personal Area Networks

ABC AdaBoost Classifier

ACM Association for Computing Machinery

ACOSO Agent-based Cooperative Smart Object

AI Artificial Intelligence

AINA Advanced Information Networking and Applications

API Application Programming Interface

AR Augmented Reality

ASF Application Security Frame

ATIoT Attack Trees for IoT

BLE Bluetooth Low Energy

BR Binary Relevance

BRkNN Binary Relevance k-Nearest Neighbors

CC Classifier Chains

CCS Computing Classification System

CoAP Constrained Application Protocol

COMNET Computer Networks

CPS Cyber Physical System

CSS3 Cascading Style Sheets 3

CT Category Theory

CVSS Common Vulnerability Scoring System

CWE Common Weakness Enumeration

DCS Distributed Control System

DFD Dataflow Diagram

DNS Domain Name System

DDoS Distributed Denial of Service

DoS Denial of Service

DREAD Damage Reproducibility Exploitability Affected users Discoverability

DSL Domain Specific Language

DTC Decision Tree Classifier

EMI Electromagnetic Interference

ETC Extra Trees Classifier

ETC2 Extra Tree Classifier

ETM Enterprise Threat Modeling

ETSI European Telecommunications Standards Institute

FPGA Field Programmable Gate Array

GDPR General Data Protection Regulation

GPT Generative Pre-Trained Transformer

GPU Graphical Processing Unit

GSM Global System for Mobile Communications

HTML5 HyperText Markup Language 5

HTTP HyperText Transfer Protocol

IBM International Business Machines

ID IDentification

IEEE Institute of Electrical and Electronics Engineers

IIoT Industrial Internet of Things

IJIS International Journal of Information Security

I/O Input/Output

IoT Internet of Things

IP Internet Protocol

kNC K-Neighbors Classifier

kNN k-Nearest Neighbors

Llama Large Language Model Meta AI

LLM Large Language Model

LoRA Low Rank Adaptation

LoRa Long Range

LoRaWAN Long Range Wide Area Network

LP Label Powersets

LRC Logistic Regression Classifier

LSP Label Space Partitioning

LTE Long Term Evolution

LWCAR Lightweight Cryptographic Algorithm Recommendation

M2M Machine-to-Machine

MAC Media Access Control

MaP Maximum *a Posteriori*

MDA Model Driven Architecture

MitM Man-in-the-Middle

ML Machine Learning

MLkNN Multi-Label k-Nearest Neighbors

MQTT	Message Queuing Telemetry Transport
MV	Majority Voting
NFC	Near Field Communications
NLP	Natural Language Processing
NS	Needham-Schroeder
OS	Operating System
OTA	Over-the-air
OWASP	Open Web Application Security Project
PLC	Programmable Logic Controller
Ph.D.	Doctor of Philosophy (degree)
QLoRA	Quantized Low Rank Adaptation
RFC	Random Forest Classifier
RFI	Radio Frequency Interference
RFID	Radio Frequency Identification
RNC	Radius Neighor Classifier
SAM	Security Advising Modules
SBC	Single Board Computer
SBPG	Security Best Practices and Guidelines
SCADA	Supervisory Control And Data Acquisition
SDL-TM	Software Developed Life-Cycle Threat Modeling
SDN	Software Defined Network
SGDC	Stochastic Gradient Descent Classifier
SMT	Satisfiability Modulo Theories
SRE	Security Requirements Engineering
SQL	Structured Query Language
SSE	Systems Security Engineering
SSN	Smart Sensor Networks
STRIDE	Spoofing Tampering Repudiation Information disclosure Denial of service Elevation of privilege
SUS	System Usability Scale
TAM	Threat Analysis and Modeling
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TMS	Threat Modeling Solution
UDP	User Datagram Protocol
UML	Unified Modeling Language
VR	Virtual Reality
WSN	Wireless Sensor Network

(This page is intentionally left blank.)

Chapter 1

Introduction

1.1 Thesis Scope and Motivation

The Internet of Things (IoT) (a term coined by Kevin Ashton in 1999) [1] can be described as the addition of connectivity to a multitude of physical devices (or *things*), which allows control and communication with these devices through a local network, and/or the Internet. The majority of these devices have a collection of sensors embedded into them, being capable of measuring the environment about them. This is one of the two major functions of IoT [2], denominated as *sensing*, with the second major function being *actuation*. *Actuation* pertains to the capability of a device physically intervene in the space the device is located in. Many devices combine these two facets, being both able to sense the world around them, and directly interacting with it, and can be applied to a multitude of different environments, from home and office spaces, to industry, healthcare, power grid, cities, agriculture, transportation, among others. It is one of the main pillars of what many consider the fourth industrial revolution, due to its intimate connection with Cyber-Physical Systems. This overarching reach of IoT typically results in an increase of efficiency, reliability and comfort, and changes are already visible across a wide range of these environments. IoT systems, most often, and as their name implies, are connected to the Internet, using a variety of communication protocols, most commonly including those within the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol stack.

The proliferation of IoT devices over the last few years and their potential ubiquity is turning them into prime targets for malicious activities. Moreover, the pressure to quickly bring them to the market and the computational constraints of some of those devices are leaving security aspects aside, while the technology matures. According to IoT Analytics, by the year 2027 more than 29 thousand million IoT devices will be connected to the Internet [3]. These devices will have their own security issues (e.g., safe boot or secure updates) and inherit many of the well known Internet related problems (e.g., access control and authentication issues). Since IoT devices are significantly different from the traditional Internet Protocol (IP)-connected devices, they are exposed to new forms of attack, including physical attacks (since many IoT devices are located in open environments) and network based attacks. The security challenges on IoT systems range from authentication to trust, integrity, data privacy, in communications in wireless sensor networks, Radio Frequency Identifica-

tion (RFID) technology, end-to-end encryption and security, or access control.

It is non-trivial to know what technologies, protocols or controls to apply in terms of security for each use case when developing a device or system for the IoT, and this difficulty, combined with the limited expertise on security of most developers, and the fast evolving environment surrounding IoT, can be seen as the main motivation for this work. These difficulties stem from the number of different architectures, applications and technologies that encompass the IoT ecosystem, and also difficult the design and development of secure IoT systems from the ground up. However, it is also a given that in software engineering, adding or making changes deep in the development process (i.e., patching a vulnerability in a system nearing completion or already deployed) has larger implications in terms of cost and time [4]. The creation of secure devices is then a difficult challenge to tackle. This is one of the factors that hampers the adoption of IoT systems [5], as security issues can bring downtime, loss of trust, loss of reputation, and even irreparable damage to the operations.

The following challenges, therefore, need to be resolved or taken into consideration if IoT development is to become more secure by construction, and as a consequence, ensure greater adoption and reputability:

- **Heterogeneity** – the variety of devices, communication technologies and scenarios in which IoT can be deployed increases the difficulty in creating generic meaningful solutions. Both hardware and software are heterogeneous in the IoT, with different systems requiring different components or sub-devices and software stacks, with different computational capabilities or energy requirements;

- **Lack of knowledge in security** – most developers have reduced knowledge in terms of security and security engineering. The subject of security itself is not easily approachable, and developers often cannot afford to dedicate time to learn and master it during the design and development phases of a project. It is also an always-evolving field, increasing the difficulty of staying up-to-date on the appropriate practices and technologies to use;

- **Lack of standardization** – non-standardization makes it more difficult to properly establish architectures, mechanisms and best practices and guidelines. The majority of IoT frameworks and architectures for development are disjoint, and only approach a narrow space of systems and use cases;

- **Missing security focus in current architectures** – the various proposed IoT architectures focus on the functionalities and workings of the system as a whole, but either have security non-present or as an afterthought, not being embedded in each layer or component of the architecture. This makes understanding how to create a secure system, based on an architecture, inherently non-trivial;

- **Lack of tools to aid in design and implementation** – taking the previous challenges into consideration, it is clear that there is a lack of tools that can aid developers to develop secure IoT systems, by giving them recommendations, guidelines and best practices to follow during the design and development stages of an IoT system.

As further discussed below, the research presented in this thesis is focused on creating a framework to support the design and creation of secure IoT systems. Because it touches several different areas of expertise, its scope encompasses different areas such as cryptography, networking or system modelling. Five of the categories and descriptors of the Association for Computing Machinery (ACM) Computing Classification System (CCS) that provide the scope of this work are as follows (structured and using the notation for priorities defined by the classification system):

- **Security and privacy ~Systems security;**
- **Security and privacy ~Software security engineering;**
- **Security and privacy ~Software and application security;**
- *Security and privacy ~Network security;*
- *Security and privacy ~Embedded systems security*.

1.2 Problem Definition and Thesis Formulation

Security issues are commonplace in the IoT landscape and touch the different parts of the systems. While there is no standardized IoT architecture, three main layers are always present [6]: *perception*, *network* and *application*, and security is required in either of the three, as attacks can be directed towards all of them. While a large section of attacks are carried over from other computing paradigms, IoT is particularly exposed to a varied amount of (sometimes, new and specific) attacks due to its unique characteristics [27]. The *perception layer*, specifically, is the most vulnerable, as it consists of smaller nodes, with constraints in terms of processing capabilities and, many times, energy as well [7]. Due to this, the usual security mechanisms implemented in other computing paradigms are not possible to be implemented in the same manner in these devices. IoT devices are also particularly susceptible to Denial of Service-type of attacks, either being the target, or when infected, used as part of *botnets* [8]. As shown by recent attacks and data breaches, IoT devices are becoming a strong focus of attackers due to ubiquity in terms of use, and lack of security and easy exploitation [9]. The rapid development and spread of this type of systems makes this problem a challenging one, as new threats appear every day.

The previous paragraph elaborates on the problem that ultimately needs to be solved. Nonetheless, it can further be decomposed or transformed into other problems. The issue of lack of cybersecurity expertise in development teams is an example, one that cannot be easily solved in the near future. Address that gap in the meantime might be achieved via tools and frameworks that define and somehow force the embedding security in the entire development process. The main problem approached by this doctoral thesis is thus the lack of tools and mechanisms that can aid in the development of secure IoT systems.

The Thesis Proposal:
The thesis proposal is that it will be possible to design and develop IoT devices which are secure-by-design, if developers are provided with tools that are tailored towards IoT-specific aspects, with focus on security and designed to be easy to use, do not require expertise in security, and support the development and verification steps thoroughly.

1.2.1 Research Objectives

The main purpose of this Doctor of Philosophy (degree) (Ph.D.) is the advancement of scientific knowledge in the areas of IoT security and system and software engineering, namely through the contributions towards the proposal, development and evaluation of a framework for guiding a user or manufacturer through all phases of the security engineering of an IoT device or system, including identification of security requirements, assisted generation of attack models, mapping of requirements, constraints and technologies, producing high quality documentation, and designing of assurance tests.

Secondary objectives include creating a comprehensive system model for IoT that takes security considerations in every of its definitions; creating a comprehensive panoply of attack models for the main attacks applicable to diverse IoT application domains; prototyping tools for proof-of-concept of the framework; and investigating the potential of Artificial Intelligence (AI) techniques learning the logic behind the mapping done by a tool, as a way to further improve them down the line, or (partially) automate their updating process. Research on the creation of attack models was based on an in-depth study of the state-of-the-art on attacks, attack modelling techniques and IoT system architectures. Subsequently, the objective was the assessment of IoT system properties, and determining how a model could be proposed, that interconnects with the different security requirements, mechanisms, and the generated attack models, while being compatible with the majority of IoT systems in different domains.

1.3 Adopted Approach and Research Plan

The approach for solving the problem that was adopted during the course of this programme was segmented in several parts. As the final objective is towards the creation of a framework, the idea was to approach the problem by identifying potential attacks, according to system specifications, and system and attack modelling techniques, and then integrate these in the design and development process.

The first phase consisted in surveying the current landscape in terms of system and attack modeling for the IoT, to then define what the best approach would be on first defining the most common attacks on IoT systems, how they can interact with a system, and taking into account the possible intrusions a device or system is exposed to. The focus was on what a tool for attack modelling should be (and what technique it should use, such as the Diamond Model [23], Attack Trees [28], or Kill Chain [23]), and on its ease of use.

The second phase consisted on the integration of the first part of the work in the engineering process of IoT devices and systems, through the definition of a model that took security considerations into each of its integral components and their definitions, to aid in detecting and solving security issues, through a security-by-design approach, where the entire process of development is done taking security into consideration. The process is, through the model, streamlined, so that it is possible for teams with low security expertise to be able to design and project a system that is, by design, as secure as possible.

The third phase consisted in the development and adaptation of tools, based on the research performed during the first two phases, for the purpose of contributing towards the main objectives of the second phase, and also towards the main objective of this thesis. This was done through the mapping of system characteristics with different security-related definitions, such as attacks, requirements, mechanisms and guidelines, and the mapping of these with the concepts defined in the second phase.

The fourth and final phase focused on studying and experimenting with the emerging possibilities of adapting AI to the solutions developed and adapted during the third phase, with the purpose of elaborating upon one of the secondary objectives, and also pertaining to the longevity of the developed solutions. This was approached through experimental work, with the focus of verifying, firstly, the actual feasibility of applying such methods, and second, the actual achieved performances.

1.4 Main Scientific Contributions

While the initial motivation and hypothesis for this Ph.D. programme remained the same along the entire work, the very dynamic and evolving environment (specially in the areas of cybersecurity and AI) has encouraged the exploration of different approaches and areas along the work, which is also reflected in the contributions. The main scientific contributions resulting of this work can thus be described as follows:

- A **Survey on System and Attack Modeling for the IoT** – the first significant contribution of this research is a comprehensive examination of the landscape of attack and system modeling for the IoT. The produced survey approaches and investigates on different techniques and existing tools for both system and attack modeling, identifying existing gaps on the specific topics.
  This part of the work was the main subject of a paper submitted and published in the *ACM Computing Surveys* international scientific journal [10] and serves additionally as the basis of Chapter 2 of this thesis.

  Publication:

  – João B. F. Sequeiros, Francisco Chimuco, Musa G. Samaila, Mário M. Freire, and Pedro R. M. Inácio, Attack and System Modeling Applied to IoT, Cloud and Mobile Ecosystems: Embedding Security by Design, ACM Computing Surveys (CSUR), 53(2):25, March 2020.

- **A Proposed System Model for the IoT** – the second contribution consists of a novel global architectural model for the IoT, which embeds security in its core, with a further identification of security requirements and interconnections between the different components, and a definition of a series of IoT ecosystems that the model may be applied to. This work was one of the subjects of a paper submitted and accepted at the *Advanced Information Networking and Applications (AINA)* 2024 Conference [11], and was also subject of a paper submitted to the international journal *Springer* International Journal of Information Security (IJIS) [12]. This part of the work serves as the basis of the first part of Chapter 3 of this thesis.

  Publications:

  – João B. F. Sequeiros, Francisco T. Chimuco, Tiago M. C. Simões, Mário M. Freire, and Pedro R. M. Inácio, An Approach to Attack Modeling for the IoT: Creating Attack Trees from System Descriptions, in Proceedings of The 38th International Conference on Advanced Information Networking and Applications (AINA-2024), Kitakyushu, Japan, April 17-19, 2024;

  – João B. F. Sequeiros, Francisco T. Chimuco, Tiago M. C. Simões, Mário M. Freire, and Pedro R. M. Inácio, Secure System Model for IoT, and Application Cases of

AI in Security Engineering, submitted to the *Springer* IJIS journal.

- **A Proposed Attack Taxonomy for the IoT** – the third contribution takes the previously mentioned model into consideration and presents an attack taxonomy for the IoT, where attacks are categorized, providing a structured approach for understanding the diverse threats posed to IoT systems. It also lays the foundations for the creation of tools approaching the attacks defined in the taxonomy. Starting from a strong orthogonalization of the work is of particular importance in the field, since part of the research on cybersecurity lacks on this particular aspect. This contribution was the subject of part of the paper submitted and accepted at the *AINA* 2024 Conference [11] and of the paper submitted to the international journal *Springer* IJIS [12] (previously mentioned). This contribution serves as the basis of the second part of Chapter 3 of this thesis. A secondary contribution concerning the adaptation of the proposed taxonomy to cloud-based mobile apps was also subject of part of a paper published on the *Springer* IJIS [13], though it is not further elaborated herein because of its less prominent role in the said paper and falling out of scope of this thesis.

  Publication:

  – Francisco Chimuco, João B. F. Sequeiros, Carolina Lopes, Tiago M. C. Simões, Mário M. Freire and Pedro R. M. Inácio, Secure Cloud-based Mobile Apps: Attack Taxonomy, Requirements, Mechanisms, Tests and Automation, International Journal of Information Security (IJIS), 22(4): 833–867, February 2023.

- **Design and Implementation of the Attack Trees for IoT (ATIoT) Tool** – the fourth contribution is the creation and testing of the ATIoT tool, that generates sets of attack trees and their descriptions from a given system description. This contribution includes both the design and creation of a set of attack trees, and the knowledge representation in the form of decision-making logic. This tool was presented and described in the paper accepted at the AINA 2024 Conference [11] (previously mentioned), and it was released as an open-source project, with an Apache license, in [29]. This is the foundation upon which the first part of Chapter 4 elaborates on.

- **Adaptation of the Security Requirements Engineering (SRE), Security Best Practices and Guidelines (SBPG), Lightweight Cryptographic Algorithm Recommendation (LWCAR) and Threat Modeling Solution (TMS) Tools to the Proposed System Model** – the fifth contribution concerns the mapping of the outputs of several of the different tools of the SECURIoTESIGN project, of which the author of this thesis was coordinator for Activity 2 – *Security Engineering for the IoT*. This adaptation is the subject of the second part of Chapter 4 of this thesis. From the Activity coordination resulted secondary contributions to two papers submitted and published in the international journals Institute of Electrical and Electronics Engineers (IEEE) Access Journal [14] and *Elsevier Computer Networks (COMNET)* [15], in the form of aiding in the logic and mapping of the tools during their development. The main con-

tributions of the author of this thesis to these two publications were focused specifically on the logic, decision making and development of the SRE and SBPG tools. While not elaborated in detail in this thesis, these contributions preceded the previously mentioned adaptation of the tools.

Publications:

  - Musa G. Samaila, João B. F. Sequeiros, Tiago Simões, Mário M. Freire, and Pedro R. M. Inácio, IoT-HarPSecA: A Framework and Roadmap for Secure Design and Development of Devices and Applications in the IoT Space, IEEE Access, 8:16462-16494, January 2020;
  - Musa G. Samaila, João B. F. Sequeiros, Carolina Lopes, Edi Aires, Tiago Simões, Mário M. Freire, and Pedro R. M. Inácio, Performance Evaluation of the SRE and SBPG Components of the IoT Hardware Platform Security Advisor Framework, Elsevier Computer Networks (COMNET), 199, November 2021.

- **Use of Artificial Intelligence in the Automation of Requirements Mapping** – the sixth contribution revolves on the study on possible approaches to automate the mapping of security requirements through the use of AI techniques, attempting to ensure that, given a small training set of examples given by an expert, the AI system is capable of extrapolating with strong precision and accuracy, so that future developments and additions to the knowledge base can be directly incorporated into tools with minimal human interaction in the update process. This approach was the subject of a paper submitted and accepted at the INForum 2021 conference [16], and is detailed in the first part of Chapter 5 of this thesis.

  Publication:

  - Carolina Lopes, Joana C. Costa, João B. F. Sequeiros, Tiago M. C. Simões, Mário M. Freire, and Pedro R. M. Inácio, Machine Learning Applied to Security Requirements Elicitation: Learning From Experience, in Atas do 12º Simpósio de Informática (INForum 2021), Lisboa, Portugal, September 9-10, 2021, pp. 0-12.

- **Use of Large Language Models (LLMs) to Facilitate Description Input and Creation of Recommendations and Their Mapping** – the seventh contribution of this Ph.D. programme is the study on the use of LLMs to imbue them, through fine-tuning processes, of the capability of performing the modeling processes, mapping of requirements, and security recommendations, expanding upon them through their vast knowledge base. This work was included in the (previously mentioned) paper submitted to the *Springer* IJIS journal [12], and is the basis for the second part of Chapter 5 of this thesis. A first, secondary study was also contributed to, as a secondary contribution, in terms of experimental support, as a precursor to the work that culminated into the main contribution, which is published in the international journal IEEE Access [17]. This later work, specifically, is only briefly mentioned in chapter 5 of this thesis.

Publication:

- Joana Cabral Costa, Tiago Roxo, João B. F. Sequeiros, Hugo Proença, and Pedro R. M. Inácio, Predicting CVSS Metric Via Description Interpretation, IEEE Access, 10: 59125-59134, June 2022.

1.5 Thesis Organization

This document is divided in six main chapters. The contents of these chapters can be briefly described as follows:

- Chapter 1 – **Introduction** – presents the motivation and scope of the work proposed in this thesis project, the problem that it attempts to solve, the main objectives and the approach to be adopted to solve the presented problem. The organization of the document is also summarily described here;
- Chapter 2 – **State-of-the-Art** – discusses different IoT security issues and different works on both surveying vulnerabilities of this type of devices and modelling approaches for attacks and system models. A brief revision of literature on this area and related works are also discussed in this chapter;
- Chapter 3 – **System Model and Attack Taxonomy for the IoT** – presents the proposal of a global model for IoT systems, applicable to different application fields and specifically considering security in the building of an IoT system, and a comprehensive attack taxonomy for the IoT;
- Chapter 4 – **System and Attack Modeling in the IoT** – elaborates on the topics of system and attack modeling in the IoT, focusing on the developed attack tree generation tool named ATIoT, and detailing the application of the model to different tools;
- Chapter 5 – **Leveraging Artificial Intelligence Techniques for Tool Enhancement** – describes studies on the application of AI techniques to different security related tools, and how these can aid in improving and maintain them;
- Chapter 6 – **Conclusions and Topics for Future Research** – examines some of the implications of the work developed throughout the Ph.D. programme and discusses its main conclusions. Furthermore, it unveils and briefly elaborates on potential avenues for future continuation and exploration.

(This page is intentionally left blank.)

Chapter 2

State of the Art

2.1 Introduction

This chapter presents some important aspects to consider and related works on IoT System and Attack Modeling, briefly focusing and describing several of the common strategies mentioned in the literature, and approaching several solutions that have been created for these specific topics of software engineering in the IoT. Section 2.2 presents an overview of system modeling in the IoT, including the description of IoT architectures and several system modeling tools for IoT. Section 2.3 approaches attack and threat modeling, in terms of methodologies, classifications and tools specific for the IoT ecosystem. Section 2.4 concludes the chapter. This chapter is strongly based on [10], as presented in section 1.4.

2.2 System Modeling on IoT

System modeling plays a crucial role in software development, particularly in complex environments like IoT. It helps organize and structure the design process, reducing complexity and facilitating understanding of system modules, interactions, behavior, and requirements. This approach enhances user satisfaction and increases the likelihood of detecting errors early in the development cycle.

Given the heterogeneity of IoT systems, with their diverse characteristics and functionality, various IoT architectures have emerged to define the main components or layers of these systems. The following is an overview of three prominent IoT architectures commonly found on the specialized literature:

- **The 3-Layer Model** – The most widely adopted IoT architecture is the three-layer model [19], comprising three distinct layers: (i) the *Perception Layer*, which deals with data sensing and acquisition, encompassing both data collectors and actuators (components). This layer is responsible for capturing real-time data from sensors and devices, enabling the system to perceive its surroundings; (ii) the *Network Layer*, responsible for facilitating communication and data transfer within the IoT system and with exter-

nal networks. It manages data routing, transportation, and security, ensuring seamless data exchange; and (iii), the *Application Layer*, which processes and analyzes the data gathered from the perception layer, providing insights and actionable information to users. It also supports applications and services that leverage the collected data, driving decision-making and automation. Figure 2.1 shows a representation of the model.

- **The IoT Reference Model** – The IoT Reference Model [20], proposed by Cisco, International Business Machines (IBM) and Intel, encompasses a comprehensive seven-level architecture: (i) the *Physical Devices and Controllers Layer*, which comprises the physical IoT devices, sensors, actuators, and other "things" that generate and interact with data; (ii) the *Connectivity Layer*, responsible for providing network connectivity for devices to communicate with each other and with processing units, encompassing various network technologies, such as Wi-Fi, Bluetooth, and cellular networks; (iii) the *Edge Computing Layer*, responsible for handling initial data processing and transmission, reducing network congestion and improving data responsiveness, enabling localized data processing and analysis near the source; (iv) the *Data Accumulation Layer*, which stores and manages the massive data generated by the IoT devices, employing various data storage technologies, such as cloud storage and databases, ensuring data integrity and accessibility; (v) the *Data Abstraction Layer*, which transforms raw data into meaningful insights and actionable information, through data aggregation, structuring, and analysis techniques to extract patterns and trends; (vi) the *Applications Layer*, through which user-facing applications and services that utilize the analyzed data are provided, supporting decision-making, automation, and real-time monitoring; and (vii) the *Collaboration and Process Layer*, which integrates business and operational processes across the IoT ecosystem. It ensures seamless data exchange and collaboration among different stakeholders. Figure 2.2 presents a scheme of the reference model.

- **ETSI M2M Architecture** – The European Telecommunications Standards Institute (ETSI) has proposed the Machine-to-Machine (M2M) architecture [21], which categorizes IoT devices into two main domains: (i) the *Device and Gateway Domain*, which integrates devices, their local area networks, and gateways that facilitate communication between devices and other parts of the system (it focuses on device management, data collection, and local processing), and (ii), the *Network Domain*, which includes applications, the core network, and management and access functions, handling data transmission, routing, security and inter-domain communication. Figure 2.3 schematizes the proposed architecture.

Figure 2.4 establishes a comparison between the three different models. There is clear common ground between them, especially in the division of the three base layers of the three layer model, where the different parts of the other models can be directly integrated into. This shows that, in spite of the IoT heterogeneity, it is possible to identify enough common

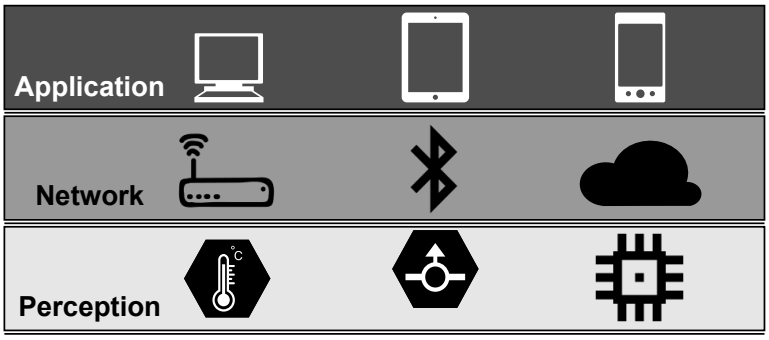


Figure 2.1: Graphical representation of the three layer model (partially adapted from [19]).

elements to define an architecture that can be applied to the majority of IoT systems.

Integrating security engineering into the development process is paramount for modern computing environments. Systems Security Engineering (SSE) systematically addresses security issues as an integral part of software development, aiming to create secure software systems. As Nunes et al. [30] highlights, the majority of attacks targeted at modern systems focus on the application layer, highlighting the importance of SSE for security professionals. This has driven the need for improvement in software development processes. Devanbu [31] emphasizes that SSE practices can bring about changes in both architectures and development practices, making it a viable option for reducing development time and costs.

SRE plays a critical role in SSE by enriching and cost-effectively streamlining the initial development phase. This is achieved through various techniques, rules, and methods employed in SRE. Mellado [32] underscores the importance of security by design, emphasizing the need for systematic and repeatable strategies throughout the development process. This approach ensures that requirements are comprehensive, consistent, and readily understandable. Both functional (directly related to services) and non-functional (encompassing characteristics like performance or security) requirements should be included in the specification document. Given the pervasiveness of software systems in daily life, Devanbu advocates for incorporating security from the earliest stages of system conception.

The advent of the Unified Modeling Language (UML) revolutionized system modeling. UML provides a standardized language for specifying the artifacts that constitute a software system. It allows for modeling the objects of the system and their interactions, enabling the division of complex systems into smaller, more manageable modules. Although it is not a programming language, UML is designed to facilitate code generation from the defined models. Over time, various UML variants and offshoots with enhanced capabilities have emerged, catering to specific application domains. One such example is SecureUML, specifically tailored for security engineering. It integrates UML with the Role-Based Access Control model, enabling the inclusion of access control concepts like roles, role permissions, and user-role assignments directly into the system model. This, in turn, facilitates the creation of access

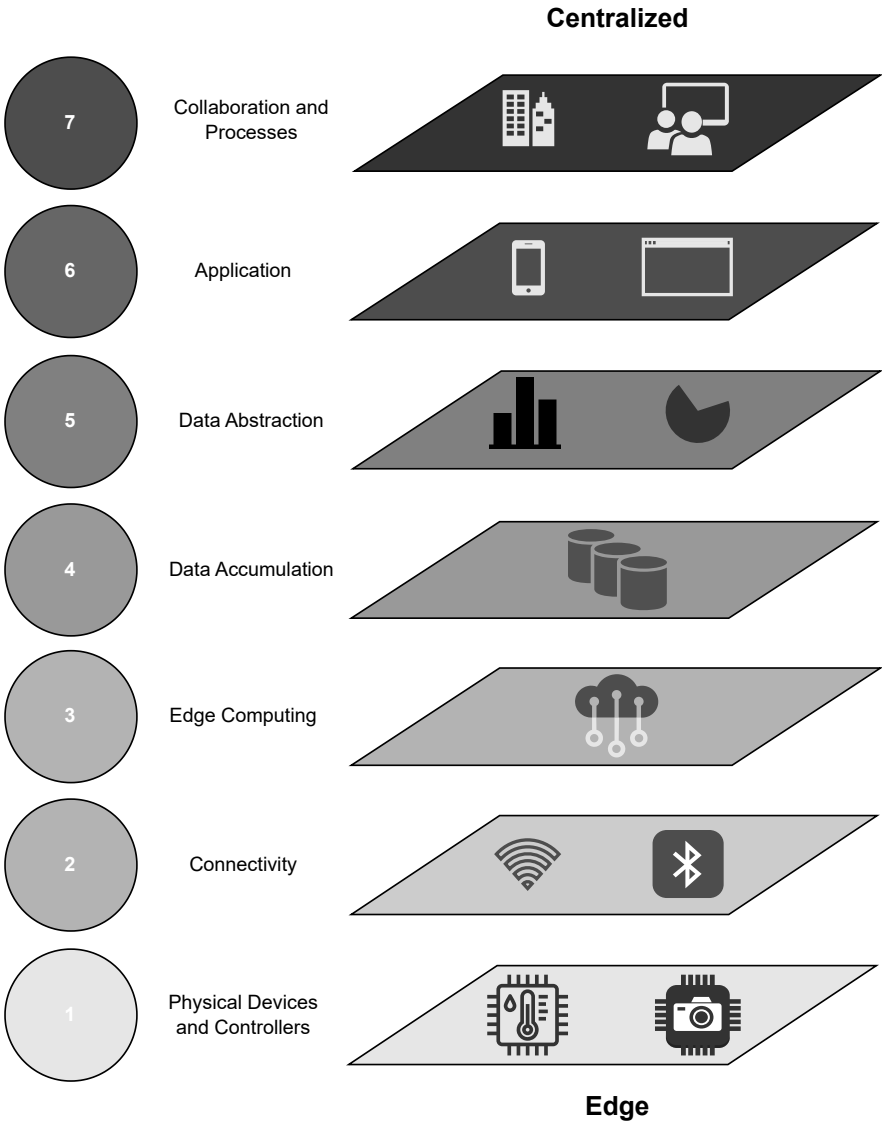


Figure 2.2: Graphical representation of the Cisco IoT Reference Model (partially adapted from [20]). *Edge* refers to to the parts of the system that are dispersed across the network (even geographically), while *Centralized* relates to the concentrated parts of infrastructure that support the system, such as cloud services or processing servers.

control infrastructures.

Additionally, the Model Driven Architecture (MDA) has emerged as a standardized approach to improve and streamline system modeling. MDA, proposed by the *Object Management Group*, integrates various standards, including UML, to aid in system design. This approach aims to separate design and architecture, allowing for independent selection of the most suitable modeling techniques for functional requirements related to system design and non-functional requirements, which are handled by the architecture.

However, most commonly used system modeling tools are primarily geared towards traditional computing systems. This section explores tools and approaches specifically tailored for IoT, systems. These systems present unique challenges due to their distinct architectures

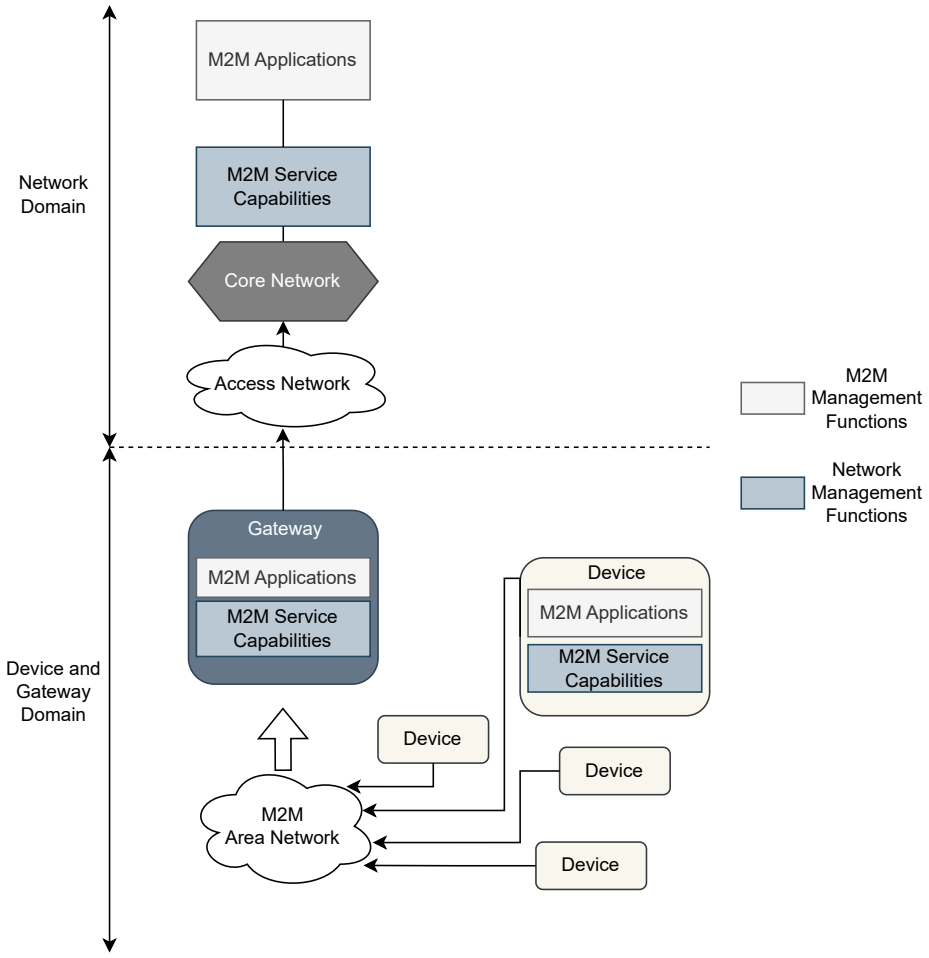


Figure 2.3: Graphical representation of the ETSI M2M Architecture for IoT (partially adapted from [21]).

and limitations, requiring specialized modeling tools and techniques.

2.2.1 System Modeling Tools for IoT

Morin et al. [33] present a model-based software engineering for the IoT based on a ThingML approach, a modeling language, based on UML, which comprises a modeling language, tools, and methodology. This allows the modeling of IoT systems and architectures without the need for specifying the used technologies or components, which usually are not defined in the early stages of a design process. The integrated tools also enable code generation through a framework that can be tinkered with to be adaptable to the needs and specifics of the system. While the integration of language and tools, including code generation, is one of the main advantages of ThingML, it still requires further work to be applicable to smaller IoT nodes, and applications that require resource sharing or software/firmware updating.

Fortino *et al.* [34] introduce a modeling and simulation approach for the IoT based on a framework developed by the same authors, Agent-based Cooperative Smart Object (ACOSO),

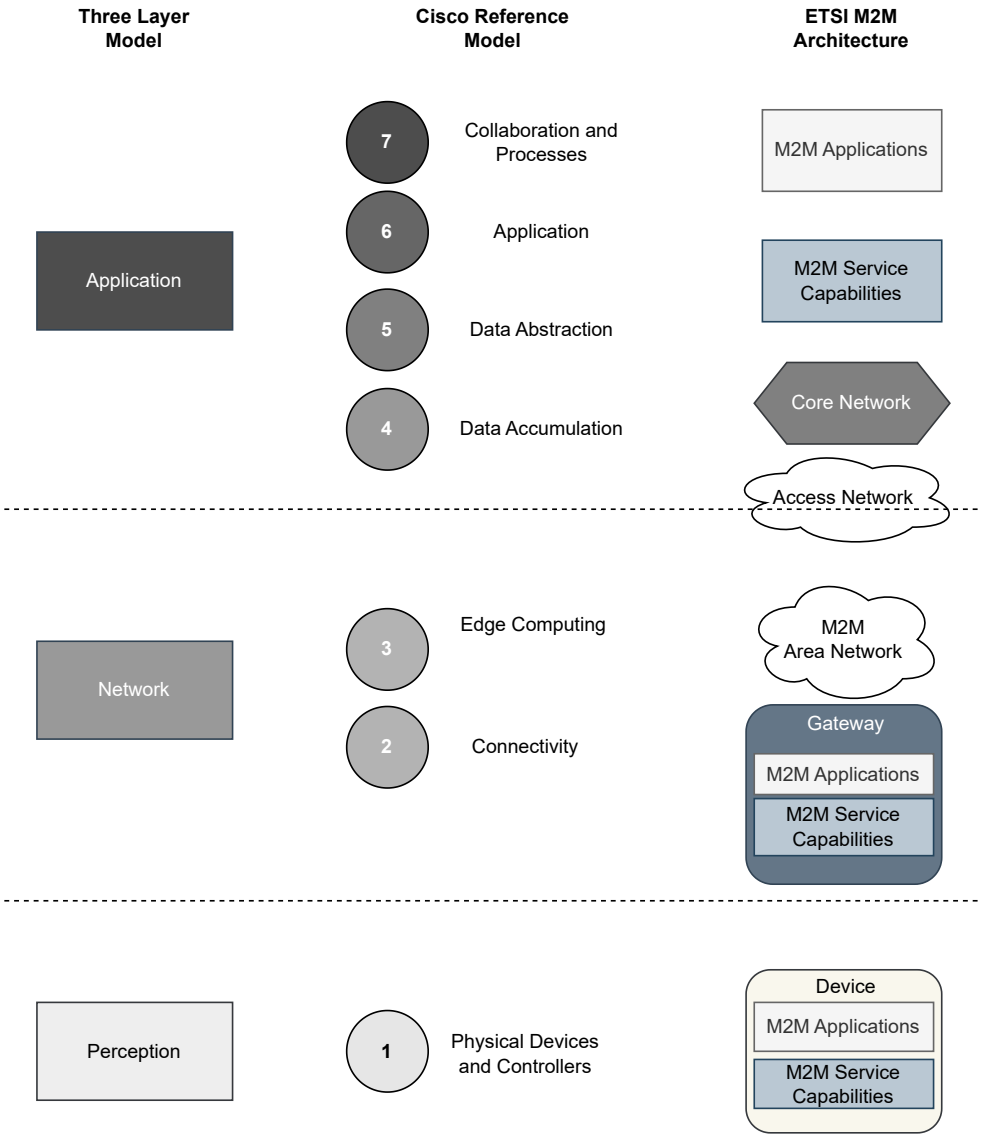


Figure 2.4: Graphical comparison between the ETSI M2M Architecture, Cisco Reference Model, and the three layer model for IoT.

which can model smart *things* functionalities and requirements, and on OMNeT++, an event simulator capable of simulating interactions and communications between entities in a distributed system. The framework can model the behavior and goals of a system agent based on event occurrences and system states. The event simulator is supported by an extension, INET, capable of simulating wireless nodes in Wireless Sensor Networks (WSNs). This approach can facilitate the design of both smart objects, or *things*, and the network to which they will be connected, with the simulator facilitating the design process by allowing the testing of different design decisions and their impact on the system. The main limitation of the system, however, is that it is designed for WSNs and wireless nodes, and therefore, its simulations are not easily adaptable for other types of IoT systems and networks.

In [35], Chen et al. propose a hardware-based model for Smart Sensor Networks (SSN) in Industrial Internet of Things (IIoT) applications, aiming to design efficient hardware con-

trollers for the SSN. The proposed model excels in achieving efficiency in both its design and final implementation. While not specifically focused on secure system design, it incorporates fault detection and error reporting mechanisms. However, the Field Programmable Gate Array (FPGA) implementation limits the reconfiguration and adaptability of the model for reuse in diverse scenarios.

Breiner et al. [36] propose a modeling approach for the IoT based on Category Theory (CT), a mathematical theory of abstract processes. This approach enables the creation of formal diagrams, aligning with class diagrams commonly employed in system design. It allows for the development of logical diagrams that can formally represent and define the structure of a system. Additionally, it facilitates the analysis of interactions between distinct components or information models. CT can also generate axioms in dynamical systems. Another advantage is that CT directly connects to functional programming, a prevalent software development paradigm. However, the authors highlight the steep learning curve associated with CT. Even simple examples necessitate advanced mathematical knowledge, primarily due to the sophistication of the employed tools, which are independent of system complexity. Additionally, software support for CT remains limited. Nevertheless, the uniformity of CT methods ensures that the same tools can be applied to a broad range of problems and systems.

Nguyen et al. [37] introduced a framework to assist in the development of IoT applications, primarily targeting sensor-based networks. They propose an architecture for sensor nodes employing a rule-based programming model and a Domain Specific Language (DSL) to facilitate system architecture definition, enabling abstraction and separation from the chosen Operating System (OS). This flexibility allows for the application of the proposed architecture to various applications and scenarios, though primarily confined to sensor node networks. The decoupling of the programming language from the model and the automated code generation are the two primary strengths of the framework.

Yan et al. [38] introduce a system model that aids in defining trust management in the IoT, comprising three layers: physical perception, network, and application. These layers collectively define the interactions between their respective constituents and associated technologies. Based on this model, the authors outline objectives and characteristics to achieve trust in an IoT system. Trust is established for data across various scenarios and properties, including communication, processing, or storage. However, a significant shortcoming of this framework lies in the apparent absence of integration between security and trust.

Babar et al. [39] introduce an IoT security model constructed upon security requirements derived from IoT system properties. They identify the core characteristics of an IoT system as mobility, wireless connectivity, embedded deployment, heterogeneity, and scalability. Based on these properties, they define three primary challenges: device management and scalability, networked knowledge, context and privacy, and security and trust. The latter challenge,

encompassing security, privacy and trust, converges in a secure IoT system. Regarding requirements, the authors define several, including resilience to attacks, data authentication, access control, client privacy, identity management, and tamper resistance. As previously mentioned, the application of these challenges in privacy, security, and trust concepts enables the delineation of the secure model of an IoT system. The presented security model is conceptualized around the notion of a cube, owing to the intersection of these three dimensions. The model, therefore, remains a concept under development, requiring further refinement.

Bau and Mitchell [40] proposed a framework for security modeling. As illustrative examples, the authors modelled the Needham-Schroeder (NS) public-key protocol using a finite-state modeling approach, along with various system components, encompassing employed cryptography protocols, security properties (logical expressions representing the state that must have been maintained), hardware security (the Execute-only-memory architecture in this case), and web security. For the latter, the authors utilize *Alloy* [41], a logical language that facilitates the creation of models and subsequent verification of their properties for correctness, representing a more high-level approach than that employed in the NS protocol example. Threat behavior is also modeled for distinct components, where attackers are characterized based on their capabilities and knowledge. By analyzing the models of each component, the existence of vulnerabilities can be identified and rectified during the design process. The framework is primarily tailored for traditional computing environments and not specifically designed for domains like IoT, although the authors leave open the possibility of further development in this direction.

Lazarescu [42] introduces the design of a WSN platform tailored for environmental monitoring. The author proposes a guideline for outlining and enhancing WSN platform development within the IoT realm. Based on the purpose of the network and deployment conditions, several requirements are extracted to construct the structure of the platform, consisting of four categories: sensor nodes, gateways and repeaters, application servers, and backend and alerts. Drawing upon these requirements, the author suggests designs for each network component, including sensor nodes, gateways, deployment devices, and application servers. Priority is given to availability, reliability, and the ability to recover from errors, along with fast deployment, low cost, and minimal maintenance requirements, characteristics that hold paramount significance in IoT setups. However, security remains an oversight in the design process.

Vučinić et al. [43] introduced an architecture to achieve security in the IoT domain, specifically addressing networking aspects. The authors propose a security architecture centered on data producers and consumers, built upon the Constrained Application Protocol (CoAP). Authorization servers, designated trusted entities, manage access to data producers from data consumers, guaranteeing the trustworthiness of producer entities. Proxy servers, re-

sponsible for ensuring data producer availability to consumers under constraints, further enhance the architecture functionality. By employing this system architecture, the overhead imposed by security protocols is shifted from IoT devices to external servers, optimizing performance while maintaining robust security measures. Data streaming remains a primary use case where the proposed architecture falls short of its intended effectiveness. This limitation stems from the reliance on CoAP, a protocol not readily adaptable to other communication protocols.

Ye and Qian [44] devised a secure architecture for IoT, centered on a security controller equipped to perform security auditing, secure network resource management, device authentication, traffic encryption, and other security-related tasks. Additionally, the architecture incorporates a monitoring system to oversee communications between the IoT network of devices and the security controller. The design of the architecture is flexible and can be adapted to various deployment scenarios, such as smart homes or field-based WSNs, while maintaining the security controller as a crucial component. The security controller assumes responsibility for managing network resources, disconnecting compromised devices, securing communications between devices and the controller, authenticating users, and establishing a secure link between the network and the end-user. This architecture enables the implementation of security measures on a node cluster without the need for each node to possess cryptographic capabilities. However, the requirement for separate monitoring and security controller systems could pose challenges in certain scenarios, particularly in terms of cost and configuration complexity.

Sosa-Reyna et al. [45] presented a four-phase methodology for IoT application development, comprising system requirements analysis, business logic definition, integrated services solution design, and technological solution generation. The methodology operates by translating initial requirements into business requirement models, which guide the construction of business solution models. These models, in turn, feed into the system architecture model, leading to the generation of technological solution models. This approach also enables the derivation of meta-models tailored to specific system types, serving as a foundation or base architecture adaptable to diverse implementation platforms. The models are sufficiently abstract to represent various IoT domains and facilitate code generation in multiple programming languages, a benefit arising from the heterogeneous nature of IoT. The main shortcoming lies in the absence of security integration throughout the modeling process.

In [46], an IoT system architecture suggests three layers: *perception*, *network*, and *application*, where *things* exist as physical devices and cyberentities. These cyberentities receive emphasis in the study due to their direct impact on system security. Managing the domains in which these identities operate and their interactions is crucial for ensuring security. Given that cyberentity interactions occur in three phases (preactive, active, and postactive), distinct security solutions are employed for each phase to achieve an all-encompassing security ap-

proach. The architecture outlines countermeasures for attack categories. However, these, alongside the mentioned cyberentities, are conceptual and do not specify particular implementation methods or define the technologies used in a defined system.

Robles-Ramirez et al. [47] introduce an UML extension tailored for IoT to enhance security modeling, facilitating the inclusion of security modeling during the system modeling phase in IoT system design development. In comparison to other security-focused UML extensions (e.g., UMLSec), IoTSec is specifically crafted for IoT, capable of representing components, deployment scenarios, communications, activities, or system layers. This enables the incorporation of security within the development processes of IoT system design and system modeling, offering the advantage, as highlighted by the authors, of not demanding extensive expertise in cybersecurity in general or IoT security in particular. Nevertheless, the work still necessitates the extension of additional UML tools, such as code generation from models, to integrate it into a comprehensive development cycle process.

Ge and Kim [48] introduced an IoT security modeling and assessment framework capable of assessing the security level of a given network when encountering an attack. The framework generates a security model derived from the user-input IoT network architecture, which must consider network topology, its nodes, and vulnerability details. Attack paths are then derived from this security model, and the framework conducts security analysis, yielding a network security metric. The final phase involves updating the network model based on the previous security analysis, allowing for the application of diverse strategies to reassess their effectiveness within the model. The primary constraint of the framework lies in its limited coverage of attack targets, defense strategies, IoT network varieties, and network mobility aspects (e.g., mobile nodes in smart cities).

Kirichek et al. [49] introduce a modeled network, a prototype representing an envisioned network for IoT systems. This network comprises five segments: a WSN, a ubiquitous sensor network (comprised of mobile sensor nodes gathering data), an indoor location system, a Software Defined Network (SDN) for network resource management, and a Cloud-IoT platform facilitating user interaction with the IoT network, covering configuration, data access, and processing. The platform allows for the testing of diverse scenarios, configurations, and communication technologies within IoT networks. However, the description does not encompass methods for testing attack scenarios or exploiting network vulnerabilities to observe potential variations in behaviors and defense mechanisms.

Table 2.2, presented in Section 2.4, contains a resume and comparison of the surveyed works of this section, along with their advantages and disadvantages.

2.3 Attack and Threat Modeling on IoT

While system modeling is an essential component in the design and development of complex systems, the structured approach of attack modeling offers a powerful methodology for identifying and quantifying security risks within a given system. The incorporation of attack modeling into the development process represents a proactive and comprehensive strategy, ensuring that security is not an afterthought but a core consideration right from the inception. The documentation produced through the attack modeling process serves as a valuable asset, providing a detailed and holistic overview of the security landscape of the system.

Attack modeling takes a unique perspective by examining the system from the viewpoint of a potential attacker. This vantage point allows for a deeper understanding of potential vulnerabilities and exploits that could compromise the integrity (in the sense of partial to full compromise) of the system. In contrast, threat modeling, while sharing similar objectives, focuses on fortifying the defenses of the system against potential attacks.

Generic risk models, which are adaptable to both security and software-centric approaches, play a pivotal role in the attack modeling process. These models facilitate the creation of a threat priority list, enabling project teams to make informed decisions about which threats to address first. Threats are typically categorized as *High*, *Medium*, or *Low risk*, based on the evaluation of generic risk factors.

A comprehensive approach to threat risk modeling involves the consideration of multiple factors, which collectively contribute to a holistic assessment of risk. This multifaceted perspective allows project teams to gain a more nuanced understanding of the security landscape and prioritize their efforts effectively [26].

According to [26], the attack modeling process can be divided into three high-level stages:

1. *Application Decomposition* – in the first phase of the threat modeling process, the primary objective is to gain a deep understanding of the behavior of the system and its interactions with external entities. This is accomplished by leveraging use cases, which help identify areas or components that might attract the interest of potential attackers. Moreover, during this phase, it is crucial to determine the privileges granted by the system to external entities based on established trust levels. This valuable information is typically documented within the attack model and subsequently used to create Dataflow Diagrams (DFDs). These DFDs serve as powerful visual representations illustrating how data flows within the system, with a specific focus on delineating access boundaries [22], [26]. For a more concrete understanding, consider Figure 2.5, which provides an illustrative example of a level 0 DFD that depicts a webmail application and its interactions with two distinct types of external entities [22];

2. *Determining and Classifying Attacks* – various categorization techniques are available for identifying potential threats. One such technique is STRIDE, an acronym introduced by Microsoft, encompassing six fundamental vulnerability classes: *Spoofing identity*, *Tampering with data*, *Repudiation*, *Information disclosure*, *Denial of service*, and *Elevation of privilege*. This classification system greatly simplifies the process of identifying potential attack vectors. For a more organized approach, Table 2.1 offers a comprehensive list of generic threats, along with examples and the associated security controls, as defined by STRIDE. Another valuable classification technique is the Application Security Frame (ASF), which covers multiple threat classes, including data validation, authorization, auditing and registry, configuration management, authentication, and data protection in both transit and storage. The primary objective of categorizing threats is to identify potential attacks (following the STRIDE model) and threats (in line with the ASF model). Use and abuse cases play a vital role in revealing how existing protective measures might be bypassed or areas where protection measures may be lacking. To determine the severity of each threat, risk models like DREAD are employed. DREAD takes into account factors such as probability and impact, or a qualitative risk assessment that considers various risk factors. The DREAD model, for instance, factors in elements like damage and affected users as technical impact risk factors, while reproducibility and discovery are categorized as ease of exploitation factors. These risk factors can be weighted and combined to assign values to different aspects. In contrast to manual attack modeling processes, tools have been developed to automate these procedures. An example is the tool created by Frank Swiderski [50], which supports the attack modeling process, making it easier to create multiple attack model documents;

3. *Countermeasure Determining and Mitigation* – the absence of protective measures against potential attacks can increase the likelihood of vulnerability exploitation. Introducing countermeasures is an effective approach to reducing the exposure to risks. In this regard, attack and countermeasure mapping lists prove invaluable for identifying these vulnerabilities. As attacks are categorized based on their risk level, it becomes possible to prioritize efforts to mitigate these threats. When it comes to selecting an appropriate risk mitigation strategy, factors such as the potential commercial impact of a threat come into play. Options for risk mitigation may include risk assumption, where an acceptable level of commercial impact is considered; informing users about identified threats; or, in certain cases, deciding to take no action at all [22], [26]. It is worth noting that the verification list presented in [26] is a concise starting point for identifying countermeasures for specific web application threats.

In addition to the methodologies provided, various other threat and attack modeling approaches are available. Two methodologies from *Microsoft*, namely, Threat Analysis and Modeling (TAM) and Software Developed Life-Cycle Threat Modeling (SDL-TM), offer ef-

fective ways to tackle threats within software applications. However, these methodologies may have limitations when dealing with interconnected systems [51]. To address the unique challenges of threats and attacks in the cloud and mobile ecosystems, a specialized methodology known as Enterprise Threat Modeling (ETM) is introduced in [52]. This approach creates a customized deployment diagram, representing, e.g., servers, endpoints or databases as nodes, and showing their interconnections, to later better apply the STRIDE and DREAD methodologies.Al-Mohannadi et al. [23] shed light on various attack modeling techniques that are technology-agnostic and can be applied to diverse network topologies and architectures. These techniques focus on three primary modeling methods: the *Diamond Model*, the *Kill Chain*, and *Attack Graphs*. The *Diamond Model* revolves around four key elements: *adversary*, *victim*, *capability*, and *infrastructure*. By assessing these elements, it is possible to discern the relationships between adversaries, victims, infrastructure, and capabilities, the Diamond Model facilitates the identification of how the different elements of an attack interact with and influence each other, condensing large amounts of data into a simple diagram, simplifying the identification and exploration of different links and patterns. Figure 2.6 exemplifies the Diamong Model. The *Kill Chain* model breaks down attacks into a series of actions taken by an attacker. This chain comprises seven stages: *reconnaissance*, *weaponization*, *delivery*, *exploitation*, *installation*, *command and control* and *action on objectives*. The first three stages are part of the initial phase of an attack, while the latter three stages belong to the second phase. The exploitation is typically depicted at the center of the flow chart for this model on purpose (Figure 2.7), representing precisely the point that separates the two phases of of the model. *Attack Graphs* are instrumental for analyzing how a target can be compromised. They employ tree diagrams to identify vulnerabilities, attack paths, and actions that can thwart attackers from achieving their objectives. This approach is invaluable in improving system defenses and preventing potential attacks, as depicted in Figure 2.8. In the realm of automated tools for attack modeling, Frank Swiderski proposed a tool [50] facilitates the creation of multiple attack model documents. This tool is freely available at [53]. An enhanced version of the tool offers additional resources and is specifically tailored for use by software developers. Finally, IriusRisk [54] represents an integrated solution that primarily focuses on threat modeling and specifying security requirements. It provides insights into all stages of development and even offers a test suite. This approach is ideal for providing developers with the tools to incorporate security by design, even when they lack an in-depth understanding of security.

To determine the classification of a threat/attack in Microsoft DREAD, the threat/attack specialist should answer several questions for each risk factor, e.g.:

- Damage – what would a successful attack cause in terms of the extent of the damage?
- Reproducibility – if an attack works, how simple is it to reproduce it?
- Exploitability – how much needs to be spent, in terms of time and effort, and what

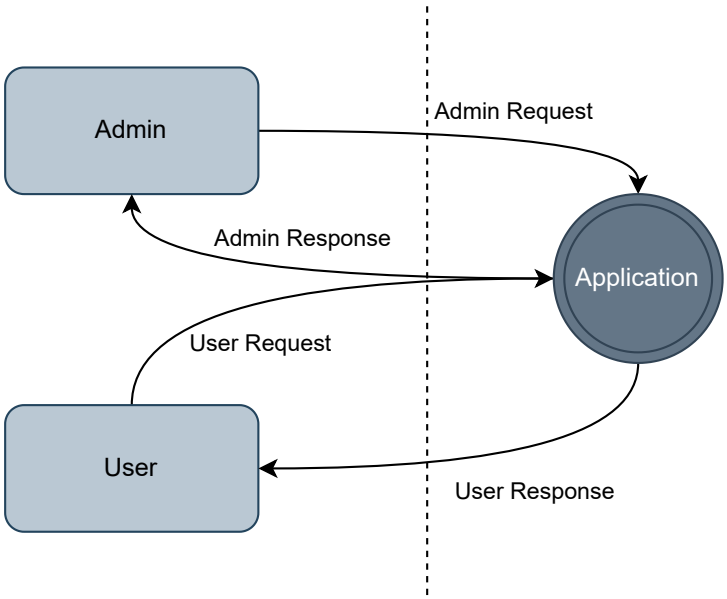


Figure 2.5: Representation of the first level of Webmail application decomposition using DFD (based on [22]).

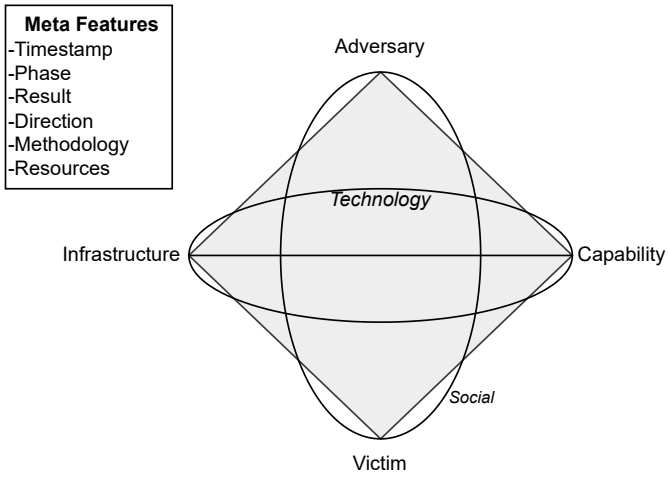


Figure 2.6: Graphical high level representation of the *Diamond Model* (based on [23]).

technical experience is needed for the threat to be successfully explored?

- Affected Users – how many users would be affected by the exploitation of a threat?
- Discoverability – how simple is it for an attacker to uncover the existence of the threat?

In DREAD, the risk value is the average of the values attributed to each of the aforementioned parameters. The values are approximate, having to be coherent between the different vulnerabilities. Values are attributed taking into consideration the detailed description of the potential attacks against vulnerabilities given by the *attack trees*, introduced by Guy Helmer in [55] (the expression *Attack Tree* was first coined by Bruce Scheiner, making this approach the most known [28] nowadays). Other attack modeling mechanisms include *Vulnerability Cause Graphs*, *Misuse Cases* [56], *Petri Nets* [57], [58] (the *Attack Net* model was presented

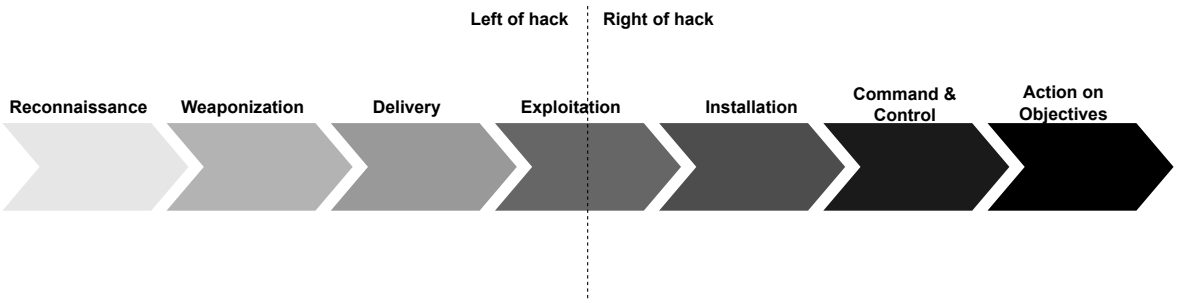


Figure 2.7: Flowchart that typically represents the *Kill Chain Model* (based on [23]).

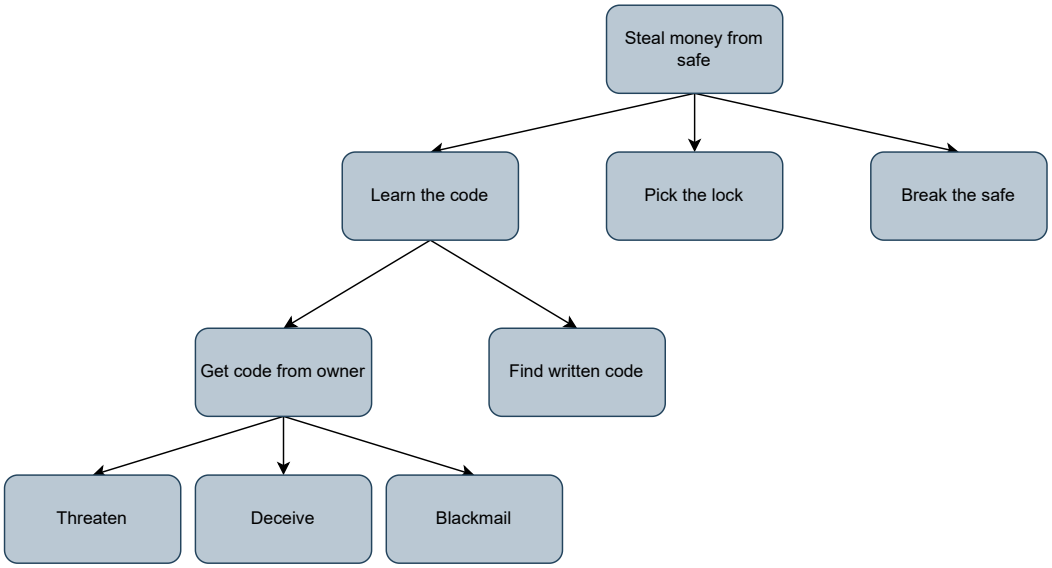


Figure 2.8: A toy example for an *Attack Tree*, used herein to introduce the Model (based on [24]).

by McDermott in [59]), *Attack Patterns* [60] and, more recently, *Boolean Logic Based on Markov Processes* [61].

Since the aforementioned methodologies are not specific to any type of computing paradigm, the next section will now focus and discuss threat and attack modeling approaches for the IoT ecosystem, which take into account the different attack surfaces and their specific limitations and characteristics, to attempt to model the potential attacks and threats that a given system may be subject to, in an effort to supply the design phase in its security modeling processes.

2.3.1 Attack and Threat Modeling Tools for IoT

Li and Xin [25] introduce a threat modeling approach for IoT systems, focusing on the perception layer (also known as the sensing layer) where sensing nodes operate. The approach utilizes the threat tree model to identify and analyze threats to the perception layer, particularly in WSNs, RFID devices, and mobile terminals. Relationships are established between threats and security goals, and threat trees are constructed to represent these relationships.

Threat	Examples	Security Control
Spoofing	Illegal access and usage of credentials of other user.	Authentication
Tampering	Modification of persistent data or data in transit in an open network.	Integrity
Repudiation	Performing illegal operations that cannot be tracked.	Non repudiation
Information disclosure	Accessing information for which access was not allowed.	Confidentiality
Denial of Service	Denying a valid user of access to a service.	Availability
Elevation of privilege	Obtaining unauthorized access by privilege escalation in an illegal manner.	Authorization

Table 2.1: STRIDE threat list (based on [26]).

An excerpt of the threat tree for WSNs is provided in Figure 2.9, and was included to better explain and differentiate it from attack trees. This approach provides in-depth analysis for RFID and WSNs, recommending preventive or nullifying techniques to mitigate threats. However, further exploration is needed in the transport and application layers, along with comprehensive threat modeling for other IoT device types.

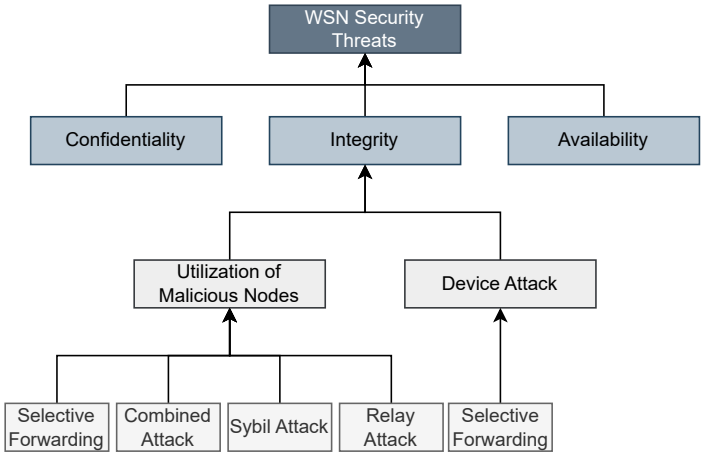


Figure 2.9: Excerpt of the Threat Tree scheme for WSNs, as presented in [25]).

Andrea et al. [62] propose an IoT security attack classification system that categorizes attacks based on their origin: *physical*, *network*, *software* or *encryption*-based. The authors begin by describing the enabling technologies of IoT and then classify IoT devices into three layers: *Application*, *Network* and *Physical*. They also define the main security goals for IoT: *data security and privacy*, which encompass the protection of data collected and stored by the system, and *trust*, which encompasses trust between device layers, trust in data at each layer, and trust in the system from the perspective of the user. The classification system divides attacks into physical, software, network, or encryption-based categories. Each category contains descriptions of different types of attacks, such as node tampering for physical attacks and man-in-the-middle attacks for network attacks. Recommendations are provided for trust and protection across all IoT layers, including risk assessment, physical device security, trust management, and intrusion detection. The classification system focuses on single attacks rather than considering interactions between different devices (e.g., between a node

and a gateway).

Agadakos et al. [63] introduce an approach to modeling *attack paths* in IoT systems by modeling interactions between IoT devices on a network and identifying potential unexpected and harmful events. The system employs sniffers to detect devices and verify interactions in real-time. It compares the network interactions to defined user policies and updates the model in real-time to detect the impact of device additions or removals and potential attack paths that may arise. The model is verified using the Alloy model-checking tool. This approach can identify potential attack paths from intercepted communication channels but has limitations in device detection and fingerprinting, as well as in the generalization of the threat model, which assumes attackers with unlimited resources, leading to false-positive alerts.

Ashraf and Habaebi [64] define autonomous schemes to mitigate threats in the IoT [64]. They utilize an autonomic taxonomy to examine threat mitigation approaches in IoT. The authors identify the five main information security goals of a system: *confidentiality*, *integrity*, *availability*, *privacy* and *authenticity*. Based on these goals and the definition of three layers: *M2M*, *network*, and *cloud*, their approach suggests the use of autonomic computing and self-security to enable systems to autonomously identify, mitigate, and recover from attacks or faults (through self-protection and self-healing). This approach ensures security even for unattended systems. However, it requires sufficient processing power in end devices to perform autonomous defense tasks and complex software design.

Stepanova and Zegzhda [65] propose adaptive graphs for IoT security modeling [65]. They define network properties as control, resilience, tenacity, scalability, and operational constancy. A formal description of the network is provided using adaptive graphs, aiming to achieve balance between different devices under hostile conditions. The system adapts to maintain stability, sustain itself, manage risk, and reduce threat exposure. Efficiency is crucial to ensure connectivity and data integrity in challenging environments.

Sarigiannidis et al. [66] introduce a framework for modeling IoT attacks [66]. Their adaptable model utilizes a G-network to model two generic attack types: *light* and *heavy* attacks. Negative arrivals, representing data losses due to attacks (e.g., jamming attacks), are considered. A threat impact metric assesses the impact of attacks on the network. The framework focuses on the devices/sensing layer and the communication layer, leaving the application layer for future development. Extensive testing and analysis demonstrate the robustness, correctness and performance of the framework.

Kotenko and Chechulin [67] present a framework for modeling and assessing cyberattacks [67]. The framework generates attack graphs, performs real-time event analysis, predicts future attacker steps, and assesses the impact of attacks. Attacks are added to the attack graph when three conditions are verified: system vulnerabilities exist, the attacker has the knowledge and

resource to perform the attack, and the attacker is likely to achieve their goal. Attack graphs are generated through an anytime approach, through different timelines and precision in the applied set of algorithms.Real-time event analysis identifies events that fit the path of the attack graph so that they remain relevant during system runtime. With a known attacker goal, the framework predicts the next steps of the attacker based on successful attacks and the final goal.

Mohsin et al. [68] introduce a framework for performing a formal security analysis of IoT systems, capable of modeling their behavior based on device configurations, network topologies, user policies, and attack surfaces. This framework aims at identifying threat vectors, attack techniques, and the resilience of a system against attacks, all while considering the goals of the attacker. The framework utilizes Satisfiability Modulo Theories (SMT), a decision problem for logical formulas in first-order logic, to formalize interactions between system entities and classify potential threats. The system dynamically adjusts the capabilities of the attacker until a successful attack is simulated, leading to the identification of threat vectors, attack techniques, and system resilience. The framework models system behavior by considering the environment, sensors, controllers, actuators, and services within the system. A threat propagation system is defined, classifying and actuating threats based on entry points and propagation paths within the system. The security analysis performed by the framework involves first defining an attack model and then verifying whether an attacker can achieve their goal within the defined constraints. The framework assesses the capabilities of the attacker and identifies the steps required to achieve their objective. This framework specifically addresses active threats, evaluates system resilience, and identifies complex attack vectors that a system may be susceptible to.

In Section 2.4, table 2.2 resumes and compares the different works surveyed in this section, their advantages and disadvantages. Additionally, to keep the size of the chapter manageable, the previous discussion did not include figures for most of the presented models or approaches, but more details and representations can be found in the paper [10].

2.4 Conclusions

This chapter presented an overview of the IoT attack and system modeling landscape, which aided in providing a brief perspective on the main issues present in this area: the expert-directed approaches that the area mainly focuses on, leaving developers without a security background or with low security expertise outside of their scope, which makes most IoT development projects leaving security as an afterthought, and the overall heterogeneity of IoT, which increases the difficulty of describing an overarching architecture for IoT systems. Table 2.2 presents a resumed comparison of the different main works surveyed in this chapter,

through which it is possible to achieve the conclusions mentioned *infra* and *supra*: most approaches are focused towards highly knowledgeable security experts and developers, and many approaches, especially in terms of system modeling, focusing on specific types of IoT systems, not taking into consideration the heterogeneity of this area, and therefore limiting the scope of their applicability. Several symbols are used in the table: ✓is used to denote that the subject is addressed on the work, – is used to represent a topic that is briefly or only lightly mentioned or applicable, and × is used to denote the absence of the subject. The review of the literature also emphasizes that the system abstract model is many times at the core of attack modeling, with many authors using different system models or proposing their own when addressing attack modeling in IoT. For this and other reasons, and while this work adopted one of the approaches for attack modelling, it was decided to propose a system model for IoT within the scope of this Ph.D. project.

The following chapter (chapter 3) attempts to define an overarching security-aware architecture that encompasses the majority of IoT systems, as well as an attack taxonomy, that will attempt to define a broad set of attacks for IoT.

Work	System	Attack	Type	Security	Req. Expertise	Adaptable
Morin et al. [33]	✓	×	Model	×	Medium	✓
Fortino *et al.* [34]	✓	×	Model and Simulation	×	High	×
Chen et al. [35]	✓	×	Hardware-based Model	–	High	×
Breiner et al. [36]	✓	×	Model, Functional Programming	×	High	✓
Nguyen et al. [37]	✓	×	WSN architecture and DSL	×	High	–
Yan et al. [38]	✓	×	Trust Management	–	Medium	✓
Babar et al. [39]	✓	×	Security Model	✓	High	✓
Bau and Mitchell [40]	✓	–	Security Modeling through Modeling Language	✓	High	✓
Lazarescu [42]	✓	×	WSN Design	×	Medium	×
Vučinić et al. [43]	✓	×	Security Architecture using CoAP	✓	High	×
Ye and Qian [44]	✓	×	Security Architecture	✓	High	✓
Sosa-Reyna et al. [45]	✓	×	Development Methodology, Generation of Models	×	High	✓
Ning et al. [46]	✓	×	Conceptual Security Architecture	✓	High	✓
Robles-Ramirez et al. [47]	✓	×	UML IoT Security Extension	✓	High	✓
Ge and Kim [48]	✓	–	Security Modeling Framework	✓	High	×
Kirichek et al. [49]	✓	×	Prototype Modeled IoT Network	×	Medium	✓
Li and Xin [25]	×	✓	Threat Modeling through Threat Trees for WSNs	✓	Medium	×
Andrea et al. [62]	×	✓	Attack Classification System	✓	Medium	✓
Agadakos et al. [63]	×	✓	Attack Path Modeling	✓	High	–
Ashraf and Habaebi [64]	×	✓	Autonomic Taxonomy for Threat Identification	✓	High	–
Stepanova and Zegzhda [65]	×	✓	Security Modeling with Attack Graphs	✓	High	✓
Sarigiannidis et al. [66]	×	✓	Attack Modeling Framework	✓	High	✓
Kotenko and Chechulin [67]	×	✓	Attack Graph Modeling	✓	High	✓
Mohsin et al. [68]	×	✓	Formal Security Analysis Framework	✓	High	✓
This Ph.D.	✓	✓	System Modeling, Taxonomy, Attack Modeling	✓	Low	✓

Table 2.2: Comparison table for the works surveyed in the chapter. This comparison takes into consideration the area the works focus on, and the expertise level they are designed towards.

Chapter 3

System Model and Attack Taxonomy for the IoT

3.1 Introduction

This chapter dives into the specific subject of IoT system and attack modeling, so as to provide a framework to address the evolving threat landscape. The purpose of this chapter is twofold: first, to introduce an expansive taxonomy of IoT attacks, and second, to propose a sophisticated system model that aims at aiding researchers, engineers, and cybersecurity experts in the modeling of IoT systems and their vulnerabilities.

The dynamic nature of IoT ecosystems, coupled with the diversity of devices and communication protocols, requires a precise understanding of potential attack vectors. This chapter begins by presenting an attack taxonomy specifically tailored to the IoT. Various threat vectors, their impacts, and the *modus operandi* of malicious actors are explored and described herein, providing a comprehensive view of the IoT threat landscape.

While a detailed understanding of IoT attacks is essential, it is only one piece of the puzzle. The second part of this chapter contains and elaborates on a novel system model designed to support the creation of robust, secure, and efficient IoT environments. This system modeling artefact plays a critical role in identifying security requirements, and how security is normally left outside the scope of architectural model proposals for IoT. In this definition, security is not considered a system concept, but an integral part of all system concepts. This will be further developed in the definition of each of the components, where different security requirements are mapped to the different system components.

Considering that the IoT promises transformative advancements in nearly every aspect of daily life, security remains an enduring concern. This chapter seeks to aggregate the necessary knowledge to address these concerns heads-on. The chosen approach is not only comprehensive but forward-looking, acknowledging the ever-evolving nature of IoT systems and the dynamic threat landscape that accompanies them. Through a deep exploration of the attack taxonomy and system model, the contents of this chapter aim at serving as a robust foundation for securing IoT ecosystems. The following section, Section 3.2, focuses on the main ecosystems that are considered for the latter definition of the attack taxonomy. In section 3.3, a system model developed with security concerns in mind for each of its components, is presented. Section 3.6 presents a proposed taxonomy for IoT attacks, immediately

and dully mapped to the previously defined model. Finally, Section 3.7 finalizes the chapter with its conclusions.

3.2 Sub-Ecosystems

IoT is inherently heterogeneous, encompassing a vast and diverse array of devices, sensors, communication protocols, and applications. This inherent diversity is what makes IoT so powerful and adaptable to various use cases, yet it also poses one of the most significant challenges in the realm of security. This heterogeneity is not just a matter of technological variance; it extends to the applications themselves, spanning domains as diverse as, e.g., smart homes, healthcare, industrial automation, or smart cities. To navigate this multifaceted IoT universe effectively, it must be recognized that IoT is not a singular entity but a composition of numerous sub-ecosystems, each with its unique characteristics and security requirements.

To develop robust taxonomies and models for IoT security, it is important to dissect this intricate IoT ecosystem into more manageable, well-defined sub-ecosystems because they represent clusters of devices, technologies, and applications that share commonalities in terms of functionality, use cases, or operational environments. By categorizing the IoT landscape into these sub-ecosystems, deeper insight into the security challenges unique to each category can be gained. This approach allows the development of an attack taxonomy, system model, and security measures that are specifically designed to address the nuances and vulnerabilities of each sub-ecosystem. Table 3.1 gives an overview of the main communications technologies and security challenges of each sub-ecosystem, while figure 3.1 presents a graphical representation of the seven sub-ecosystems. The remaining part of this section is devoted to the description of the main characteristics and identification of types of devices of IoT sub-ecosystems.

3.2.1 Healthcare (e-Health)

Healthcare is an area where pervasive integration of IoT systems can be of great use [69]–[71], e.g., to gather real-time data to aid in the decision process, patient monitoring and event detection, or simply have medical aiding devices that can be remotely adjusted. Sensors play a major role here, enabling remote monitoring of the status of a given patient, correct prescription compliance, detection of events, enabling a faster response, and issue prevention. Not only directly related to patient health, IoT can also be used in healthcare to increase hospital/clinic workflow and organization, reduce incidents such as wrong dosage or wrong patient identification, and automate several care tasks. In this interconnected landscape, a multitude of smart devices, sensors, and data analytics tools collaborate to enhance the effi-

ciency, accessibility, and quality of healthcare services. IoT in healthcare comprises a diverse array of components that collectively transform how medical data is collected, transmitted, and utilized.

At the core of the IoT healthcare ecosystem are wearable devices, such as fitness trackers and smartwatches, that continuously monitor vital signs, physical activity, and sleep patterns. These devices transmit real-time data to healthcare providers and patients, enabling the remote monitoring of chronic conditions and the early detection of health issues. Alongside these wearables, medical devices like smart inhalers, glucose monitors, and remote patient monitoring systems extend the capabilities of traditional healthcare by offering personalized and data-driven care. Moreover, IoT-enabled infrastructure within healthcare facilities ensures seamless asset management, inventory control, and optimized energy usage. This includes smart hospital beds, connected medical equipment, and ambient sensors that streamline hospital operations and enhance patient experiences.

The IoT healthcare ecosystem is not confined to hardware alone; it also helps enabling telemedicine, ambient assisted living, or smart implants. However, the widespread integration of IoT devices in healthcare also raises significant security challenges. Protecting the vast amount of sensitive patient data and ensuring the integrity of connected medical devices is paramount. Healthcare institutions must implement robust cybersecurity measures to safeguard against data breaches, unauthorized access, and potential threats to patient safety. Additionally, as IoT devices often have longer lifespans, ensuring that they receive timely security updates and patches is crucial to mitigate evolving security risks. Balancing the benefits of IoT in healthcare with these security concerns remains a key challenge for the industry, necessitating ongoing efforts to develop and enforce robust security protocols and standards.

3.2.2 Smart Wearables

Wearables are some of the most common IoT devices that users have had access to over the last years [72]. From fitness trackers to smart watches and augmented reality glasses, wearables are devices that can incorporate a number of sensors, that allow for measuring exercising, health data, help in staying connected, and enjoying a more immersive digital experience. Wearables most commonly utilize the Bluetooth protocol for communication, but most are also capable of operating independently, having some processing capabilities. Wearables can sometimes be included in the healthcare category, as some of the gathered data can come from body sensors (most commonly, heart rate), though they have enough differences to warrant the different classification.

One of the standout features of smart wearables is their ability to collect and transmit a wealth of personal data, ranging from heart rate and sleep patterns to location and activity levels.

However, the collection and storage of such sensitive data raise concerns about privacy and security. As these devices become an integral part of our daily lives, they can become attractive targets for attacks. Ensuring the privacy and security of personal data is an ongoing challenge that the smart wearables ecosystem must address. This ecosystem also continues to evolve with advancements in augmented reality (AR) and virtual reality (VR) technology. AR glasses and VR headsets are enabling immersive experiences for entertainment, training, and remote collaboration. These innovations are poised to reshape industries beyond consumer markets, such as healthcare, where surgeons can access real-time information during procedures, or industrial sectors, where workers receive hands-free instructions and access critical data. While these advancements offer substantial benefits, they also pose concerns about data protection, user safety, and potential vulnerabilities that must be diligently managed to ensure a secure and reliable ecosystem.

3.2.3 Agriculture and Environmental Monitoring

The application of the IoT in agriculture and environmental monitoring brings several advantages to these sectors [73], improving efficiency, sustainability, and data-driven decision-making. In the context of agriculture, IoT technologies are used to monitor and manage crops, livestock, and agricultural resources, while in environmental monitoring, they help track and address various ecological concerns.

In agriculture, smart sensors, drones, and autonomous machinery are deployed to gather real-time data on soil conditions, weather patterns, and crop health. This information enables farmers to make more informed decisions about irrigation, fertilization, and pest control, thereby optimizing resource usage and increasing crop yields. However, security challenges arise in protecting this data, which is often valuable proprietary information. Unauthorized access, data breaches, and other attacks can have financial and operational consequences for farmers and businesses.

Environmental monitoring also benefits from the IoT. Smart sensors and remote monitoring devices are deployed in ecosystems, helping scientists and environmental agencies track, e.g., air and water quality, climate change, wildfires, or wildlife behavior. The collection of such sensitive environmental data poses unique security challenges. Ensuring the integrity of data is essential for sound policymaking and conservation efforts. In some cases, the security of monitoring equipment itself is crucial to protect against vandalism or tampering in remote or ecologically sensitive locations.

Addressing these security challenges will be essential to ensure the potential of these technologies. Collaboration between technology providers, regulatory bodies, and stakeholders in these sectors is key to developing and implementing robust security measures that safe-

guard data, protect critical infrastructure, and ensure the sustainable and responsible use of these technologies.

3.2.4 Smart Grids

Smart Grids concern the adaptation an integration of IoT technologies in the electrical power grid management [74], [75]. IoT represents a major transformation on the energy sector, in terms of how electricity is generated, distributed, and consumed. Smart grids integrate digital technologies into the traditional energy infrastructure to enhance reliability, efficiency, and sustainability.

In smart grids, sensors, meters, and communication networks allow for real-time monitoring of energy consumption and grid conditions. This capability enables providers to optimize energy distribution, reduce outages, and integrate renewable energy sources more effectively. Moreover, consumers can access detailed data about their energy consumption, enabling them to make more informed choices about energy use. However, the amounts of data generated and transmitted through smart grids create significant security challenges. Protecting this data from threats and ensuring the integrity of the grid is crucial. A breach in the security of smart grids could disrupt power supply, potentially causing widespread outages and other serious consequences, highlighting the need for robust measures and contingency plans. Furthermore, the interconnection of smart grids with various energy sources, including renewables and energy storage, presents opportunities for more sustainable and resilient energy systems. Nevertheless, it also introduces security concerns related to the control systems governing these assets, which are part of a critical infrastructure. Protecting against potential attacks that could compromise energy generation and storage facilities is essential.

IoT systems can be applied in several points of the electricity supply chain, from production to transmission, distribution and consumption. The greater challenge on enabling IoT in this scenario is that, depending on the point it is applied, the aggregate technologies differ, due to specific characteristics of each case. Also, most of the devices will be exposed to major electromagnetic interference, especially those used on power lines, substations and generators, due to the high currents and voltages present in these areas. Monitoring power transmission lines, for example, is one of the most important and challenging aspects on a smart grid, as there are several areas to monitor: the conductors on the tower, the tower status (e.g., if it is leaning or oscillating), or the general environment the tower is located in.

3.2.5 Transportation and Logistics

Both the handling of logistics in moving cargo and monitoring of the transportation means can have their efficiency improved with IoT [76], [77]. Either in terms of locating either cargo or transport in real time, monitoring through IoT devices can help in reducing costs, increase time efficiency, prevent mechanical issues and contribute to predictive maintenance. The cargo itself, especially if sensitive, can also be similarly monitored (e.g., temperature-sensible cargo, such as perishable goods).

Both the handling of logistics in moving cargo and monitoring of the transportation means can have their efficiency improved with IoT, improving connectivity, efficiency, and safety in these sectors. IoT technologies have significantly improved the management of transportation networks, vehicles, and the supply chain, offering a range of benefits.

In transportation, IoT-enabled systems are used to monitor and optimize the movement of vehicles. Telematics devices installed in vehicles collect data on location, speed, fuel consumption, and maintenance needs, providing real-time insights that can be used to enhance safety, reduce fuel costs, and improve route planning. However, the vast amounts of data generated and transmitted within this ecosystem create security challenges. Protecting sensitive data from threats and ensuring the integrity of transportation networks are crucial concerns, particularly given the potential risks associated with the manipulation of traffic systems or vehicle control.

In the logistics sector, IoT technologies are used to monitor the supply chain from end to end. This includes tracking the location and condition of goods, managing inventory levels, and optimizing warehouse operations. IoT devices like RFID tags, sensors, and cameras help improve the visibility of goods in transit and enable efficient inventory management. Nonetheless, the security of data and control systems within logistics and supply chain operations is a critical issue. Unauthorized access or tampering with data could lead to costly disruptions in the movement of goods, presenting significant challenges that need to be addressed to ensure the smooth operation of these systems.

3.2.6 Smart Home/Office/Building

Home automation, or domotics, are terms frequently used to describe the automation and integration of IoT into common homes or offices [78], [79]. These interconnected spaces leverage IoT technologies to enhance comfort, energy efficiency, and security while providing greater convenience for occupants. The goal is to ease the usage (e.g., enabling remote control) or automate most common attributes, such as lighting, energy spending control, air quality, smoke, CO_2 and other types of detectors and sensors, automated cleaning systems

(e.g., vacuum robots), security and intrusion detection, automated blinders, or occupancy detection (e.g., detecting if a division has occupants and automatically turning the lights on or off). This results in energy savings, reduced operational costs and increased comfort. In terms of security (physical security of the building, not the systems themselves), it also allows users to monitor when away from the spaces, and even trigger automated routines that can prevent outsiders from noticing a vacancy (e.g., turning lights on or off, closing shutters during the evening and opening them in the morning, or playing music).

However, the extensive network of devices and data streams in these settings raises security challenges. Protecting sensitive data about building operations and their occupants, and ensuring the integrity of control systems is crucial to prevent unauthorized access, data breaches, potential disruptions to building functions, or physical damage to systems.

3.2.7 Smart Manufacturing

The integration of IIoT in manufacturing can be characterized by improved efficiency, reduced downtime, and enhanced product quality [80]. IIoT technologies enable manufacturers to connect machines, sensors, and systems across the production process, transforming traditional factories into automated and data-driven environments. In these environments, sensors and devices are employed to monitor and optimize various aspects of the production line, including equipment performance, quality control, and predictive maintenance, and has a strong emphasis in M2M communication and big data. It also integrates well with industrial control systems, such as Supervisory Control And Data Acquisition (SCADA), Cyber Physical Systems (CPSs), Programmable Logic Controllers (PLCs) and Distributed Control Systems (DCSs). The real-time insights provided by IIoT devices empower manufacturers to make data-driven decisions that result in cost reductions and improved productivity. However, the extensive interconnectivity and data exchange within these smart factories create significant security challenges. Protecting critical manufacturing data and ensuring the operational integrity of machinery is a paramount concern. Attacks or breaches in smart manufacturing environments could lead to production disruptions, data theft, and even equipment damage. Furthermore, the remote control of industrial equipment presents an added challenge. Ensuring that unauthorized parties cannot gain control over automated machinery is of critical importance to prevent accidents or malicious interference.

3.2.8 Smart Cities

The concept of smart cities (powered by IoT) presents a vision of urban development that harnesses data and technology to improve the quality of life for citizens, enhance sustainability, and optimize city operations [81], [82]. Smart cities use IoT technologies to create

connected, efficient, and data-driven urban environments that address a wide range of challenges.

In smart cities, sensors and networks collect vast amounts of data from various sources, including traffic management systems, public transportation, waste management, and environmental monitoring. This data is utilized to optimize city services, reduce traffic congestion, improve energy efficiency, and enhance public safety. Additionally, smart cities often involve the deployment of devices like surveillance cameras and sensors for public safety and monitoring. While these technologies provide valuable insights and enhance security, they raise concerns about individual privacy. The urban scenario is also complex one, with a multitude of devices intercommunication, through M2M protocols, WSNs, and other technologies that enable them to sense and actuate on a large scale. Unifying the urban environment through IoT into a single, large scale platform is one of the most ambitious IoT applications, and one of the more complex ones, as it can fuse several of the other sub-ecosystems herein described, and a multitude of different technologies, communication protocols, and system components.

Striking the right balance between public safety and personal privacy is an ongoing challenge that requires the development of responsible data collection and usage policies. This, combined with the interconnected nature of these systems, poses significant security challenges. Protecting sensitive data and the integrity of city infrastructure is paramount to prevent disruptions, data breaches, and potential attacks that could impact the day-to-day lives of the residents. To fully realize the potential of smart cities, addressing these security and privacy challenges is essential. Implementing robust security measures, encryption protocols, and data access controls are critical to protect sensitive urban data. Furthermore, collaborative efforts involving city planners, technology providers, and regulatory authorities are required to establish and enforce security and privacy standards that can safeguard the digital infrastructure of smart cities while providing residents with the benefits of enhanced urban living.

3.3 Proposed IoT System Model

The IoT system model proposed herein is a versatile model designed with modularity in mind, with the objective of accommodating a broad range of IoT system designs. It recognizes that IoT systems can vary widely, encompassing physical and logical components, and that not all architectural elements are mandatory for each implementation. This adaptability is a key feature, ensuring applicability to diverse IoT development projects. The primary objective of this model is to provide a holistic view of the system, encompassing data, communications, and security considerations, thus facilitating the design and construction process. An intellectual exercise is undertaken to map different security requirements to the rele-

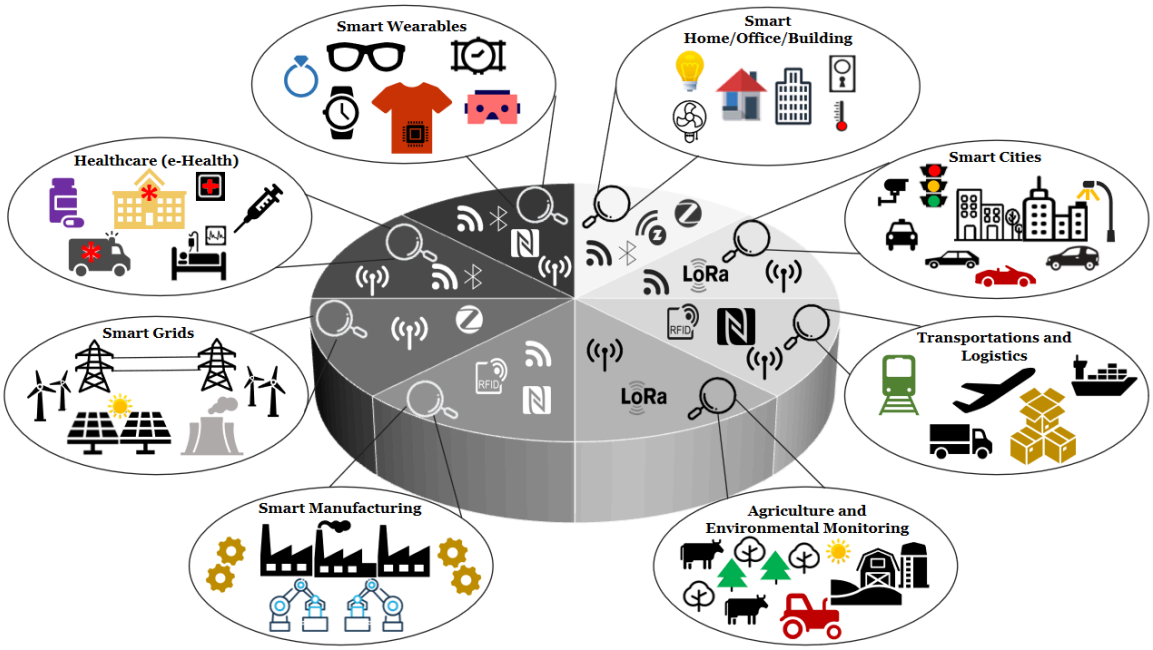


Figure 3.1: High-level graphical representation of the different IoT application scenarios.

vant system components, enhancing the understanding of where and how security measures should be integrated. The idea is that this model serves as a valuable tool for system designers and developers, guiding them in creating robust and secure IoT systems. Figure 3.2 presents a graphical representation of the proposed model. The following subsections describe the several components in the system model with more detail. Also, it is important to mention that a high-level model, such as the one proposed herein, defines concepts and components of different abstraction levels. This mixing of different levels in concepts or layers is done accordingly to the objective the model is designed towards achieving.

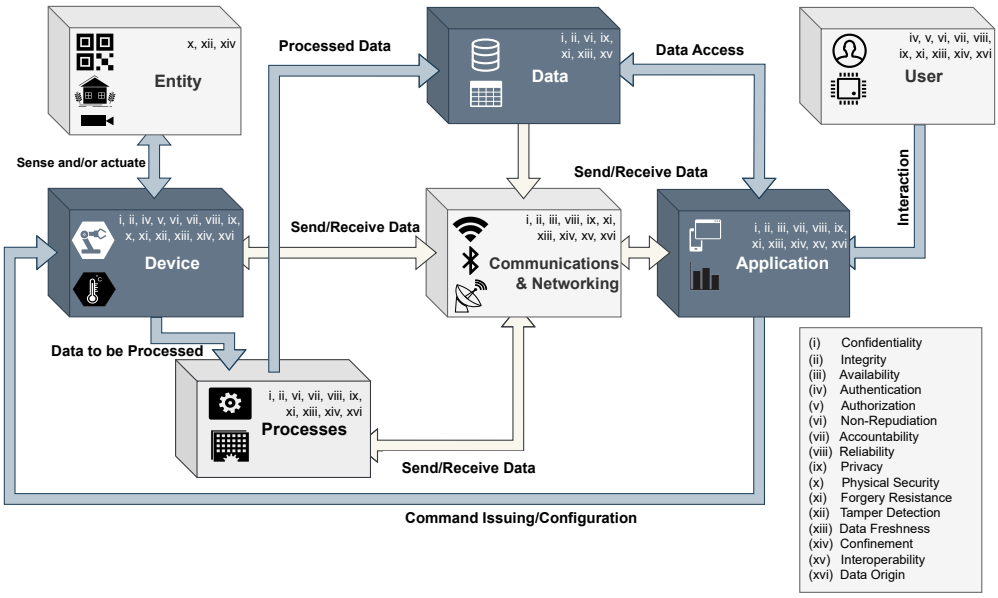


Figure 3.2: High-level graphical representation of the proposed global IoT model, representing its components, their interactions, and identifying which security requirements may be applied to each component.

Sub-Ecosystem	Main Characteristics	Device Types	Communications Technologies	Security Challenges
Healthcare	Embedded sensors, control of medical devices, sensitive data transmission	Heartrate monitors, Glucose and insulin monitors and pumps, Hearing aids, Pacemakers	Bluetooth, Wi-Fi, Cellular	Sensitive Data, Patient safety, Device integrity
Smart Wearables	Day-to-day devices, clothing, accessories, embedded sensors	Fitness trackers, Smart glasses, Smart rings, Smart watches	Bluetooth, Wi-Fi, NFC, Cellular	Data protection, user safety, vulnerabilities
Agriculture and Environ. Monitoring	Large sensor networks, spread across vast areas, difficult connection coverage	Temperature and humidity sensors, Irrigation sprinklers, Cattle trackers	Cellular, LoRa	Data integrity, Vandalism, Tampering, Theft
Smart Grids	Critical infrastructure monitoring and control, hostile conditions for electronics	Power meters, Battery monitors, Equipment status sensors	Cellular, ZigBee, Power Line Communication	Critical infrastructure, Electromagnetic interference, tampering
Transportation and Logistics	Cellular-connected sensors and trackers, cargo monitoring sensors, fleet tracking and data collection	RFID tags, GPS trackers, Vehicle status sensors	Cellular, RFID, NFC	Sensitive data, Vehicle control, Traffic control
Smart Home/Office/Buildings	Sensors and automated devices for building functions control, HVAC, lighting, access control, indoors limitations for wireless technologies	Temperature sensors, Occupancy sensors, Lighting, Blinds, Locks, HVAC	Wi-Fi, Bluetooth, ZigBee, Z-Wave	Control systems integrity, Data breaches, Service disruption
Smart Manufacturing	Machinery sensors, logic control for automated machinery and SCADA systems, quality and safety control	Sensors, Air quality monitors, PLCs, CPSs	Wi-Fi, NFC, RFID	Operational integrity, Disruption, Data theft, Accidents
Smart Cities	Traffic control, parking control, large sensor networks spread across large areas, heterogeneous environment	Air quality monitors, Surveillance systems, Traffic control systems, Parking sensors, Public lighting, Gateways	Cellular, Wi-Fi, LoRa	Service disruption, Data breaches

Table 3.1: Comparison of the different identified sub-ecosystems, their main characteristics, example devices, communications technologies, and main challenges faced in terms of security.

3.3.1 Device

In the proposed model, the *device* is an end-point of an IoT system, meaning that this component can either refer to sensors or actuators and it is crystallized by actual hardware that is

deployed. As such, it often constitutes a limiting factor in terms of what can be implemented in terms of security, as these devices are the ones suffering from the most constraints, be it processing power, power usage, footprint, or location, which make them one of the points that should receive greater focus. A particularly unique issue here is physical security: as these devices can be located in an exposed environment, they should be able to resist being tampered with (or at least, if accessible, to be protected so that the network, data, and remaining system remain secure, even if this one device is physically compromised). On the standard IoT model layers, they are part of the *Perception layer*. Many models also specifically consider the device layer. This *Perception*, as implied by its name, is the part of the device that senses, or collects data. It is the actual sensor part of an IoT device, e.g., a thermostat or a humidity sensor. In here, the main dataflow to take into account is internal to the device, it is the captured data that is then processed to be sent by the communication part of the device. *Actuation* is the other potential part of the *Device*, which can directly interact with the physical world. Any component that is capable of altering the physical world is part of the *Actuation* capabilities of the device. In *Perception* and *Actuation*, specifically, the only specific type of security that can be feasibly implemented is physical security, as the only actual location where data is flowing are the physical traces on the device.

3.3.2 Data

Data flows across an IoT system and is required at every component of the model, be its creation, processing, storage, use or transfer. Data can have different definitions according to the information it stores, such as sensor data, which could be temperature, pressure, motion, device data, which includes, e.g., device status, location, and firmware versions, or user data, i.e., access logs or personal information. This data can be subject to breaches, which can, among others, expose vulnerabilities, enable remote attacks, be used to gain unauthorized access, disrupt operations, and compromise privacy. To safeguard data throughout its lifecycle, security measures need to be implemented at every stage, in the devices, during transmission, and when stored. Every part of the system where data flows through is then required to have security measures implemented. Most commonly, encryption is the main requirement, but the type of encryption possible to be used will be defined and limited by the previous and next sections of system components.

3.3.3 Communications & Networking

The connection to the Internet and the ability to send and receive data is part of the definition of IoT. Devices need to communicate with each other, a centralized infrastructure, or a specific transmitter node. The *Communications & Networking* component of the model represents this connections in the model. This connection is usually done via wireless technolo-

gies. There are many different protocols that can be used for this, which range from the more traditional Wi-Fi, Bluetooth, Global System for Mobile Communications (GSM) and Cellular (3G, 4G or 5G), to protocols created specifically for IoT, e.g., Zigbee, Long Range Wide Area Network (LoRaWAN) or IPv6 over Low power Wireless Personal Area Networks (6LowPAN). However, the chosen communication protocol is not the only concern here. The way data are transmitted has a major importance in terms of security; if this transmission is done without any security measures, external entities can either read, modify, or block data transmission. It is then of critical importance to define security protocols that are, at the same time, compatible with the transmission means and supported by the devices. Most security measures present a trade-off in terms of overhead, be it time, power consumption or the amount of transmitted data. What must be ensured is that the most appropriate protocols are chosen, as there must be a choice in terms of which trade-offs should or should not be made.

3.3.4 Entity

In the model, an *Entity* represents a real-world *something* that plays a crucial role in the operation of the system. The existence of this entity makes it either an objective of the system, or an integral part of it. To define the *Entity* in this model, both its physical and virtual existences in the system need to be taken into account. The physical entity is the actual *something in the real world* (e.g., an object, a device or a person), while the virtual entity is its digital twin within the system. This digital twin contains data and state information about the physical entity, allowing for monitoring and control. The digital part of the entity is represented by information stored in data on the system, but in terms of defining what security measures an *Entity* may require, the physical side should be the one considered in the model, with the protection of the *something* either from the system itself, or through the system detecting outside tampering with the *Entity*.

3.3.5 User

A *User*, by its definition in the model, is someone or something who interacts with a system. This does not define the user, however, as an actual human user. Human users are usually external and interact with the system directly through its main means (e.g., an interface), i.e., through a high level process of the application layer. In M2M communication, another system can be a user as well, and interact with the system through lower level processes, by directly communicating with the devices, through network protocols or Application Programming Interfaces (APIs), and without the interference of a human user. Security is, nonetheless, required for both types of *Users*. For *Human Users*, strong authentication protocols, user access controls, and security awareness training is required. For *Machine Users*, enforcing encryption on communications, employing least privilege access controls

and monitoring for suspicious activity to prevent unauthorized access and potential exploitation are the norm to be followed.

3.3.6 Application

The *Application* is the component of the model through which *Users* are able to interact with the system, i.e., the bridge between *Human Users* and the system. Usually, the application layer, in most models, relates to traditional and mobile computing, as the end-devices that users use for these interactions do not have the same amount of constraints as those in the other previously mentioned components. However, the application layer does need to ensure defensive mechanisms against the most common flaw in any computer system: the human component. As such, and specifically depending on the type of ecosystem, several of the common mechanisms need to be ensured, such as as authentication and authorization, and other good practices such as input validation, regular security testing, least privilege access control, network segmentation and use of secure communication protocols. Another important aspect here is that the application component is, many times, the actual tangible perspective of the system and its data, even the part that is remotely processed and stored in the cloud, or other remote solutions, which makes it a tempting target for attacks, especially in situations where the system is remotely accessible through this application layer.

3.3.7 Processes

Processes is another component of the proposed model. They are related to the need to treat data and reduce data volume. As data is gathered in bulk by the system and due to the aforementioned constraints in most of the system components, there needs to be a selection and processing (or, in some cases, pre-processing, especially when done at the edge) of the data, to ensure that system resources and bandwidth are not fully hogged with irrelevant data. Usually, this processing is done not directly in the devices, but in the main connection nodes (e.g., gateways and routers) located at the edge. This ensures the main connection with the application layer does not get fully overloaded by the amount of data produced by sensing nodes (when applicable). Similarly, in the case of actuator nodes, the main nodes at the edge need to ensure that instructions are kept at a minimum number, to avoid overloading the end devices. This part of the model also enables, besides the logical security component, potential energy and communications efficiency increases.

3.4 Identification and Mapping

This section elaborates on the mapping of concepts and security requirements, each dealt with in a dedicated subsection. This exercise complements the definition of the system model.

3.4.1 Identification and Mapping for Concepts

Mapping refers to the association of the different identified parts of the system to the previously defined concepts. As the name implies, it is when association and identification are done, so that a further mapping of requirements and mechanisms to the different concepts and parts can be done, and the global system model can be further defined. Taking the defined concepts into consideration, the mapping can thus be defined for each component as follows:

- **Identifying Users** – identifying the *Users* in a system is a straightforward task, but goes beyond simply recognizing those who interacts with it. Any external item or individual that is capable of interacting or communicating with the system, without expressly being a part of it, should be considered as a user. As previously mentioned, this can either be a human or another system/device that is not part of the system itself. To manage this diverse user base, a system needs to differentiate between them. Assigning unique identifiers, like serial numbers, Media Access Control (MAC) addresses, or usernames, is the first step. For human users, secure identification comes through credentials like, e.g., passwords, tokens, or biometrics like fingerprints or facial recognition. Non-human Users, on the other hand, may rely on certificates for secure communication, preventing unauthorized devices from infiltrating the system;
- **Identifying Entities** – *Entities* are a more complex concept in this model; an *Entity* is any objective of the system, be it what it measures or interacts with. The easiest way to identify the system *Entities* is to define all which the system either senses or actuates upon. This can be a physical object, person, or the environment itself that the system is integrated into. The virtual representation of the entity must also be taken into account, by defining how that physical entity is represented virtually inside the system, and which components it is then related to, from the *Device* to the virtual entity storage location. As each physical entity has this virtual twin within the system, its digital representation acts as a mirror, reflecting the real-world counterpart. Defining this virtual entity involves understanding how the physical details are translated into the system language through data. Connections need to be established in this context defining which components interact with the physical entity, where this virtual representation is stored within the system, and how the system logic performs to achieve its objective

relative to the entity. By creating this intricate web of connections, the physical world and the digital world become seamlessly intertwined;

- **Identifying Devices** – after identifying the *Entities*, *Devices* can be identified by observing what are the parts of the system that interact directly with these entities. Identifying devices requires following the threads that connect to the entities. For instance, if the temperature of a room is the entity, the device could be composed of the physical sensor itself, the circuit board, chips and antenna that transmits the temperature data. This is where classic IoT devices may be identified – smart thermostats, wearables, environmental monitors – anything that directly gathers information or influences the previously identified entities. By pinpointing these interaction points, the entire device network that will bring the IoT system to life can be mapped and defined.
  Interestingly and for example, while smartphones are not typically a *Device* in the proposed system model, they may pose as one in scenarios where they are monitoring the environment or the user. As such, the context and purpose of the IoT system is of utmost importance to identify *Devices*;

- **Identifying Processes** – *Processes* represent the functional operations performed within the IoT system. They include data manipulation, analysis, and decision-making activities that are distributed across various system components. Unlike entities and devices, processes are inherently distributed across the system, making their identification more intricate. The key to mapping processes lies in data flow analysis. It is essential to pinpoint locations where data is processed, transformed, and stored, both temporarily and permanently. Data storage locations often represent potential process execution points. Security considerations become paramount when data is processed or temporarily stored at any point within the system, including on the devices themselves. By tracing the data flow and identifying these processing points, a comprehensive process map for the IoT system can be established;

- **Identifying Applications** – identifying the application part of the system is one of the easier tasks. Any software developed to interact indirectly with the system will be part of the *Application* component. Applications can be defined as the user-facing software of the system. Unlike entities, devices or processes, that work behind the scenes, applications take center stage. Identifying them is a easier when compared to untangling the complexities of the inner workings of the system. Applications are any software programs designed for human interaction with the IoT system, but not directly. They rely on well-defined pathways, like APIs, to communicate with the system, using system capabilities to deliver value to the user. Applications can take different forms, such as mobile applications, web dashboards, or even machine learning applications that analyze data to give insights. By establishing a clear application component, the system model creates a clear division of labor. The core functionalities of the IoT system focus on interacting with entities, managing devices, and processing data internally.

Applications, on the other hand, leverage these core functionalities through standardized interfaces to deliver user-centric experiences;

- **Identifying Communications** – to allow developers to define certain security mechanisms, there needs to be a definition of which communications technologies the system implements, and how networking is done overall. It is the communications that bind all the components together, allowing them to exchange data and orchestrate actions. To ensure system security, developers need a clear map of this communication landscape. This map will define how communication occurs within the system. It will detail the specific communication technologies employed, and outline the overall network architecture, pinpointing how different parts of the system connect and interact. The communication map reveals which components are connected by which technologies, and what the data throughput is like on each route. It also identifies the communication protocols used. By specifying all these elements – technologies, network architecture, protocols, and throughput – developers can then select the most appropriate security mechanisms to safeguard the system. A well-defined communication map paves the way for robust security, ensuring the smooth flow of data within the system.

3.4.2 Mapping of Security Requirements

To ensure that a set of established security requirements are interpreted within the correct components of the model, so that mechanisms can later be chosen and utilized for each of the system components, a mapping of those two parts is presented below. The set of security requirements considered herein was based on the work presented in [14], and is composed of a total of 16 requirements: *confidentiality, integrity, availability, authentication, authorization, non-repudiation, accountability, reliability, privacy, physical security, forgery resistance, tamper detection, data freshness, confinement, interoperability* and *data origin*. What follows is a description of each one of those requirements, and in which conceptual system component they are applicable:

- **Confidentiality** – *confidentiality* is a property that, when applied, ensures that information is not disclosed or made available to unauthorized parties. It ensures that data, in whichever part of the system, is only accessible to those with the proper privileges (e.g., a registered user, or the data owner). Taking this into consideration, *confidentiality* is needed in the *Device, Processes, Communications & Networking*, *Data* and *Application* concepts, as these are the main components where information is either stored on moved through, and where any unauthorized party should not be able to access or disclose it;
- **Integrity** – *integrity* is the property of safeguarding the correctness and completeness

of assets in a system. It ensures that, e.g., data stays consistent and trustworthy during its lifecycle, i.e., that it is not altered by unauthorized parties or even due to errors. Similarly to *confidentiality*, this property is applicable to the *Device, Processes, Communications & Networking, Data*, and *Application* concepts, since it needs to apply to all components where data is either stored or passing through;

- **Availability** – *availability* ensures that the system is accessible and can be used whenever an authorized user demands it. It ensures that the system is able to respond to requests and accesses. *Availability* is a global property (as it refers to the system as a whole), but can be considered to be directly applicable to the *Application* and *Communications & Networking* components;

- **Authentication** – the *authentication* property ensures that the *User* or *Device* processing or transacting information is the entity they claim to be. A *Device* or *User* will need to authenticate themselves before receiving or transmitting any information in the system. Through this property, it is assured that information generated or read in the system is done so by actual authenticated parties. This property is applicable to the *Device* and *User* components of the system;

- **Authorization** – the *authorization* property determines the privileges of accessing resources or issuing commands. With Authorization, access to a given resource (e.g., information), or command issuing can only be done by an authorized party. *Authorization* is often related with *authentication* (e.g., authorizations are applied to authenticated entities) and is applicable to the *Device* and *User* system concepts;

- **Non-Repudiation** – the *non-repudiation* property guarantees proof of communication between two parties. Neither party can later claim they did not send or receive the message. This allows for holding both parties accountable for their role in the communication (sending or receiving). This requirement is applicable to the *Device*, *Data*, *Processes*, and *User* concepts;

- **Accountability** – *accountability* ensures that actions can be traced back to a specific source within the system. It is particularly important for transactions (e.g., sending data). This allows the identification of the responsible party, be it a device or user, in case of, for example, a malicious message. This is applicable to the *Device*, *Processes*, *Application*, and *User* system concepts;

- **Reliability** – the *reliability* property ensures that behaviors in the system are consistent and intended. A reliable component performs as expected, and does not stray from its defined operations. It is a system-wide property, and as such, it is applicable to all of the active concepts, i.e., *Device, Processes, Application, Communications & Networking*, and *User*;

- **Privacy** – *privacy* can be seen as the control of data disclosure related to humans. Users should always have control over their sharing of data, and if it should be kept

public, private, or accessible by third parties (under data owner authorization). Privacy is required in all parts of a system where data is either stored or sent through, and as such, it is applicable in the *Device, Processes, Data, Application, Communications & Networking*, and *User* concepts;

- **Physical Security** – in regards to IoT devices, *physical security* is a property that ensures that measures are implemented in order deny unauthorized access to physical devices and equipment, to avoid damage and/or access to devices and system information. It also avoids the possibility of physically compromising a device, and therefore, compromise the system (tampering). This property is applicable to the *Entity* and *Device* concepts;

- **Forgery Resistance** – this property ensures that a third party cannot forge information typically shared between two parts to try to damage or harm the system, or make it behave erroneously or return false values. Messages cannot be sent by unauthorized third parties impersonating internal system parties, or trusted third parties. This property applies to the *Device, Processes, Data, Communications & Networking, Application*, and *User* concepts;

- **Tamper Detection** – this property is related to the physical security property, as it ensures that devices are physically secured, so that any tampering attempt on a physical part of the system is detected. This property is applicable to the *Entity* and *Device* system concepts;

- **Data Freshness** – though it shares some similarities with *forgery resistance*, *data freshness* is another security property that ensures that the data is the most recent, and that old messages are not replayed to be mistakenly used by the system as a potential attack. *Freshness* is especially important in the context of IoT, since may systems are used to sense the, or act on the environment. This property is applicable to the *Entity, Device, Data, Processes, Communications & Network, Application*, and *User* concepts, as it can be a system-wide issue;

- **Confinement** – the *confinement* property, as its name implies, ensures that if a part of the system is compromised or corrupted, its effects are confined as much as possible, not compromising or corrupting other parts of the system. This is another system-wide property, applicable to the *Entity, Device, Processes, Communications & Network, Application*, and *User* concepts.

- **Interoperability** – *interoperability* ensures that different software can communicate and work with each other. This concerns, in particular, systems that are able to interact directly with third party systems (e.g., through a Low Level Access User), and this property ensures that the communications protocols, data formats, and interactions and permissions are well established and defined. This property is applicable to the *Data, Communications & Networking* and *Application* concepts;

- **Data Origin** – *data origin* ensures that the source of any received data is what it claims to be, i.e., that the sender is legitimate and that the date arrives as it was sent (compromising of a message is otherwise detected). This can apply to an internal system sender (e.g., a *Device)* or data coming from a third party. This property is applicable to the *Device, Processes, Communications & Network, Application*, and *User* concepts.

3.5 Specifying Concepts and Eliciting Technologies from the Global Model

Designing and characterizing the model, by defining which parts of the system apply to each concept, its main specificities, and how they interconnect, is of paramount importance to ensure a correct definition of each concept. A general specification is made herein for the different concepts.

3.5.1 Devices

Devices comprise the majority of the pieces of an IoT system, with many being populated by a multitude of devices, as is the case of, e.g., WSNs. IoT devices are characterized by having low power, low processing capacity, and are directly related to entities (which are subject to sensing and actuation). While there are many different types of devices, their two main functions are the same: sensing and/or actuating. Moreover, an important aspect for each of the ecosystems is defining the environment the device is deployed at, its capabilities, communications suite, other system components with which they communicate, types of data being collected and number of total devices deployed. Some examples of items that a device is composed of are presented below:

- Microcontrollers – low-power, low-cost processors that handle basic tasks and interact with sensors and actuators in IoT devices;
- FPGAs – programmable chips used in IoT devices for custom hardware acceleration and low-power processing tasks;
- Single Board Computers (SBCs) – a compact electronic device containing all the fundamental components of a computer on a single printed circuit board. These components typically include a microprocessor, memory, and Input/Output (I/O) interfaces;
- Sensors – convert physical or environmental conditions into electrical signals. Examples include temperature sensors, pressure sensors, motion sensors, and image sensors;

- Actuators – responsible for converting electrical signals into physical actions. Examples include motors, solenoids, and relays used to control lights, valves, or other physical systems;

- Connectivity modules – enable communication between devices and the rest of the network.

3.5.2 Data Flows

Data flows, as their name implies, are the record of where data is generated, transmitted, and stored in the system. As data is the ultimate goal in terms of most attacks, and is where most defensive efforts are focused, it is important to know exactly how data behaves and moves between different system components. This helps in identifying where the different security requirements are relevant, and where security mechanisms should be implemented. The work in identifying start with perceiving where in the system data enters or is generated; then, where it is sent to from these points, and in each component it passes through, what happens to it (if it is stored, processed, or duplicated and sent to more than one system component). Finally, the endpoints for the data should also be identified, as these are where it will be able to ”leave” the system, and should always be considered as higher potential targets for attacks.

3.5.3 Communications and Connectivity

Connectivity and communication technologies are one of the main enablers for IoT systems. All the different system components need to integrate some form of connectivity, so for correctly defining communications and connectivity, first the main types of connectivity used in IoT need to be defined, together with the corresponding advantages and usage scenarios. As different data throughput, energy consumption, and range are needed, depending on each component, their circumstances and deployment environment, the definition of the appropriate communications technology then allows the developers to further define the network protocols, other hardware and software characteristics, and further on deciding on which security and privacy measures should (and can) be implemented.

For IoT, most connectivity is done through wireless means. To stratify the most common communication technologies, they are herein present by range and throughput:

- Long Term Evolution (LTE) and 5G – Cellular based networks [83] have the advantage of having nearly unlimited range: as long as the location has a signal, the device can

communicate. Throughput is also high, especially for 5G applications. There is a constraint, however, in terms of power, and having to rely on a third party for connection, as the infrastructure needs to already be in place, and cannot be easily deployed. It is also affected by network congestion when a large amount of devices are trying to access the network (as is the case in many WSNs).

- Wi-Fi – Wi-fi [84] is one of the more ubiquitous wireless communication technologies that exist today. Commonplace in most devices and homes, it can also have a very high throughput, and has a good range in the vicinity of up to 100 meters.
- Bluetooth (and Bluetooth Low Energy (BLE)) – Bluetooth [85] is another wireless communication protocol, with a range of around 10m, that works on the 2.4GHz frequency band.
- Zigbee – Zigbee [86] is a low power communications technology stack prepared to operate in noisy RF environments. It is low-power and low-latency, and supports up to 65,000 individual nodes in a network. While its range is limited to around 75m, and a low bandwidth between 20 and 250Kbps, depending on the used frequency. Zigbee works in the 2.4GHz or in the 868 and 915 MHz (depending on region) bands.
- Z-Wave: Z-Wave [87] is a low power communication protocol that covers up to about 30 meters of range for communication, and is specific for applications that need very small data transmissions. It uses either the 908.42MHz band in North America, or the 868.42MHz band in Europe.
- Near Field Communications (NFC) – NFC [88] is a low power, low rate communication, that works in very near proximity (approximately 4cm or less). It can be used to bootstrap other communications, or simply read/write small amounts of information.
- Long Range (LoRa) – LoRa [89] is a low power communication technology that enables long range communications with low power usage, and used in the, e.g., LoRaWAN protocol. It works in the sub-Ghz spectrum, and can achieve ranges exceeding 10Km, but has low data transmission rates, below 30Kbps.
- RFID – RFID [90] is one of the most simple communications technologies that aid in enabling IoT. Using radio frequency to identify objects, read information, transmit simple messages, among others, RFID is one of the most cost-effective solutions for communications, with the other main advantage being that passive tags do not require any type of power source.

In terms of communications protocols, the following are some of the most common in IoT:

- Transmission Control Protocol (TCP) [91] – a connection-oriented protocol, it establishes a virtual circuit between sender and receiver, which ensures ordered delivery

of data packets, error checking, and retransmission requests for missing information. These characteristics are critical for scenarios demanding reliability;

- User Datagram Protocol (UDP) [92] – a connectionless protocol, it transmits data packets without establishing a dedicated connection or guaranteeing delivery or order, it excels in scenarios where low latency and minimal resource consumption are needed. It is simpler also, when compared to TCP, but has reliability as a trade-off, as data integrity and transmission success are not guaranteed;
- CoAP [93] – this protocol, designed specifically for the IoT, uses a connectionless model, similar to UDP, prioritizing efficiency and minimal resource consumption, and a simplified design philosophy employing methods like `GET`, `POST`, `PUT`, and `DELETE`, like, e.g., HyperText Transfer Protocol (HTTP). Its unique feature is called *observability*, which allows a device to proactively push updates, without the need to constantly receive requests, enabling a more efficient and responsive data flow;
- Message Queuing Telemetry Transport (MQTT) [94] – a lightweight protocol that works, unlike traditional client-server models, on a publish-subscribe architecture. Devices can publish information to a broker, while others can subscribe to that information from the broker to receive relevant updates. It is flexible, in terms that it can optionally support message delivery guarantees;
- LoRaWAN [95] – a protocol designed for connecting battery-powered IoT devices over long distances, using LoRa as its communications technology. It uses a star topology, where devices communicate with gateways, which then relay the information to a central server connected to the Internet.

3.5.4 Entities

With an *Entity* being a subject of interest of the system, entities will always be defined as a target of measurement or actuation. In any IoT system, an *Entity* will have its physical and its virtual representation, with the virtual representation being the amalgamation of data that is stored pertaining to that entity, taken from measurements in the physical world. Similarly, if the entity is not measured, but actuated upon, it will still have a virtual representation stored, along with a log of actuations.

3.5.5 Users

In any IoT system, a *User* is any entity who/that interacts with the system, be it human or a third party system engaging in M2M communications. For most systems there will be at least one type of human user that has the higher privileges; in some, a second type of user

may exist, with limited privileges. In systems relying on data to complete tasks, or systems where processing is offloaded to a cloud or fog infrastructure, there will be ”machine” users in addition, which will also communicate with the system overall.

3.5.6 Applications

In terms of *Applications*, these are the means through which the previously identified *Users* are able to interact with the system. For *human Users*, the system will need to define which level of access each type of *User* will have, what information (s)he will be able to access, localization in terms of allowed access (access control), among others. For *non-human Users*, the system will need to provide the correct API for interaction, together with documentation and access control policies. The data disclosure done for each type of users needs to be well defined, to ensure data privacy for sensible or critical information.

The following are examples of types of applications and associated technologies:

- Mobile Applications – mobile apps comprise a common place for interacting with IoT systems. These apps leverage APIs to communicate with the IoT backend (very frequently on the cloud), allowing users to monitor and/or control systems, and visualize data on the go. Popular frameworks like React Native and Flutter facilitate the development of cross-platform mobile apps, making them accessible on various operating systems;
- Web Applications – for a more comprehensive user experience, web applications provide a familiar browser-based (and often easily accessible in different user devices) interface for interacting with IoT systems. Web apps can offer functionalities like data visualization dashboards, remote device management tools, or real-time data analytics. Technologies like HyperText Markup Language 5 (HTML5), Cascading Style Sheets 3 (CSS3), and JavaScript, along with frameworks such as Angular or Vue.js, form the building blocks for creating web applications seamlessly integrated with IoT systems;
- API Management Platforms – as the number of connected devices and the complexity of interactions grow, managing APIs is essential for ensuring smooth communication between applications and an IoT system. API management platforms provide a centralized hub for defining, securing, publishing, and monitoring APIs, streamlining the development and deployment of IoT applications. Popular choices include, e.g., Apigee, Mulesoft Anypoint Platform, and Amazon API Gateway;
- Cloud Platforms – the cloud acts as a powerful enabler for developing and deploying IoT applications. Cloud platforms offer a comprehensive suite of services including storage, compute, databases, analytics tools, and machine learning capabilities.

3.5.7 Processes

In any IoT system, there needs to be a definition of where processing occurs, i.e., which components will be responsible for it, and if the system relies on, e.g., the device itself, a local gateway, edge, fog or the cloud to perform that task [96]. Some systems may use a combination of those paradigms for its *Processes*. Data in IoT also presents several unique challenges in terms of processing, many times having several of the following characteristics: heterogeneous, multidimensional, inconsistent, dynamic, continuous, has varying quality over time, different types of data structure or type, and has strong dependency both in terms of time and space. Depending on data characteristics and system architecture, several challenges may arise, such as the limited processing power, the restraints in terms of communications, the need for real-time processing, the high throughput of data, the need to preserve context awareness, or the need to ensure security and privacy. Then, identification of where data is located is needed, as well as temporary or more permanent storage. Starting from this point, the previously identified concepts for performing processing can be observed and chosen for the specific system case. Edge computing can be defined by bringing processing power closer to the source of the data (the devices themselves or local gateways as these sources), which reduces latency by minimizing data transmission to the cloud and enabling real-time decision-making for devices with limited Internet connectivity. Fog computing sits between edge devices and the cloud, acting as a distributed processing layer, aggregating and pre-processing data from multiple edge devices, filtering out irrelevant information and optimizing data for cloud processing, reducing the overall data volume sent to the cloud and improving efficiency. Cloud computing provides a scalable, cost-effective environment for handling massive datasets and complex analytics tasks that often overwhelm traditional on-premise solutions, but, as identified for edge and fog, it is remote both in terms of networking and physical distance, having added latency, and depending of the application scenario, it may raise privacy issues.

3.6 Attack Taxonomy on the IoT

The IoT is subject to a wide panoply of different attacks, due to its complexity and heterogeneity. Adversaries may target different parts of a system, and resort to a wide range of attacks to achieve their objective. An attack taxonomy attempts to categorize different attacks scenarios, depending on what part of the system they primarily target. Taking into consideration the model proposed in the previous section, a corresponding attack taxonomy for the IoT is proposed in this section. This taxonomy attempts to correlate attacks with the more relevant parts of the model, to provide insight into these attacks and what parts of the system they may affect. The proposal of the system model was key to the development of this taxonomy, with five of the seven main components being chosen to organize the taxonomy. The cho-

sen components were *Device, Data, Communications & Networking, User* and *Application*. This decision takes as a basis that these are the components that are a more objective target for attackers, as they are either the more vulnerable, or more related to hardware and software that the system may be composed of. This taxonomy then allows to classify and detect the most prevalent and plausible threats to a given system, and through it, it will be possible to provide attack modelling approaches that developers can then follow to aid in the decision process on what security requirements, and corresponding security mechanisms, should be chosen. A total of 28 attack scenarios were taken into consideration. Figure 3.3 presents the taxonomy of attacks for the IoT proposed herein.

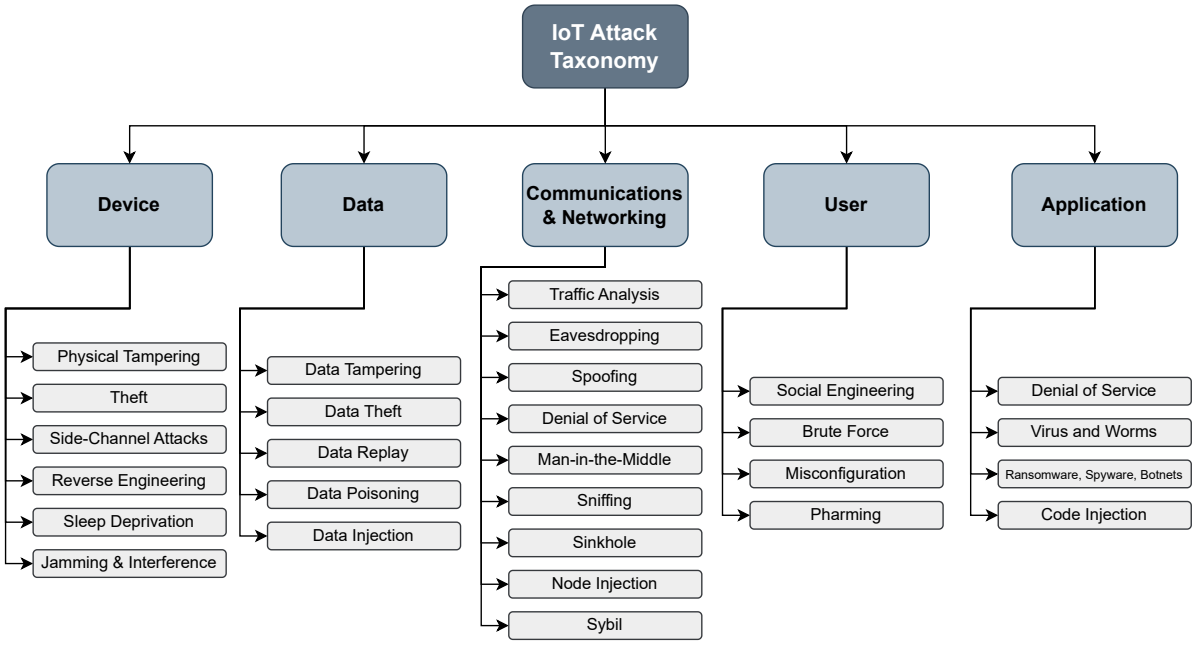


Figure 3.3: Representation of the proposed attack taxonomy for the IoT, divided across the five selected system model components.

The following subsections present the set of chosen attacks for the taxonomy for each one of the categories (sometimes also referred to as *domains*). Each attack is also briefly described.

3.6.1 Device

The device-centric part of the attack taxonomy encompasses a spectrum of threats targeting the physical and operational aspects of IoT systems. From tampering and reverse engineering to the subtler perils of sleep deprivation and interference, these attacks exploit vulnerabilities within hardware, software, and operational characteristics. Understanding and mitigating device threats is imperative for bolstering the security and resilience of IoT systems, as each unique attack vector poses distinct challenges and demands tailored protective measures. The following list presents the attacks falling into the *Device* category:

- **Physical Tampering** – *physical tampering* attacks [62] on IoT devices constitute a significant threat to the integrity and security of interconnected systems. Such attacks involve the deliberate manipulation or interference with the physical components of IoT devices, transcending the boundaries of traditional threats. One manifestation of *physical tampering* entails unauthorized access to device hardware, wherein adversaries exploit vulnerabilities in the physical enclosure of the device to gain entry. This may involve bypassing or circumventing security mechanisms, such as breaking seals, opening casings, or manipulating connectors. The breach of physical barriers grants attackers the opportunity to directly manipulate internal components, potentially compromising the confidentiality, integrity, and availability of sensitive data processed by the IoT device (e.g., directly reading encryption keys from memory units).

  A more sophisticated form of *physical tampering* involves the modification of existing components within the device. Attackers may introduce firmware alterations or malicious payloads onto the device. This can cause data gathered and sent by the device to be modified or destroyed. Furthermore, *physical tampering* attacks can extend beyond manipulation, encompassing destructive measures that render devices inoperable. Sabotage may involve the deliberate destruction of critical components, such as microprocessors or communication modules. This form of attack not only compromises the targeted device but can also have cascading effects on interconnected systems, leading to potential service outages, operational disruptions, or systemic failures;

- **Theft** – *theft* [97] of IoT devices refers to the unauthorized acquisition or physical stealing of these interconnected devices. IoT devices, due to their connectivity and often portable nature, are susceptible to theft, leading to potential data breaches, misuse of devices, or compromising the integrity of the IoT system.

  The primary objective of theft targeting IoT devices is to obtain the hardware itself, which might contain sensitive data, access credentials, or proprietary information. Once stolen, these devices can be reverse-engineered, reprogrammed, or manipulated by unauthorized individuals or groups. This could lead to unauthorized access to data, breach of privacy, or even exploitation of the device to gain entry into the broader network;

- **Side-Channel Attacks** – *side-channel* attacks [9] aim at unintended leaking of information through various channels like power consumption, electromagnetic emanations, or timing variations. These attacks rely on analyzing the observable behavior of a device during its operation to glean sensitive information, such as cryptographic keys. Power analysis attacks, for instance, scrutinize power consumption patterns during cryptographic operations to deduce key information. Similarly, electromagnetic emanations from the device can be intercepted to discern internal processing activities. Timing attacks exploit variations in the execution time of operations to deduce critical information.

The nature of *side-channel* attacks lies in their capacity to circumvent traditional cryptographic defenses, as they focus on the physical implementation rather than cryptographic algorithms. Countermeasures against *side-channel* attacks include the implementation of secure coding practices, cryptographic algorithm diversification, and the incorporation of noise or randomization techniques to obfuscate observable patterns. Additionally, the adoption of hardware-based defenses, such as constant-time algorithms and the utilization of shielding techniques to mitigate electromagnetic emanations, plays a pivotal role in enhancing the resistance of IoT devices to *side-channel* attacks. As the IoT landscape evolves, continual research and innovation are imperative to stay ahead of the ever-adapting techniques employed by malicious actors in exploiting side channels for unauthorized access and data compromise;

- **Reverse Engineering** – *reverse engineering* [98] involves the deconstruction and analysis of device firmware, software, or hardware to understand its inner workings. Malicious actors may uncover proprietary algorithms, protocols, or sensitive data, leading to unauthorized access and potential compromise of device functionality. This process allows adversaries to exploit vulnerabilities, discover security flaws, or clone devices for nefarious purposes. Counteracting reverse engineering necessitates the implementation of obfuscation techniques, code encryption, and the use of secure boot processes to hinder the extraction of critical information. Notice that *reverse engineering* may be a second step to device *theft*, but it may also be achieved via inappropriate access to code or schematics, thus herein presented as a separated type;

- **Sleep Deprivation** – *sleep deprivation* [99] refers to the intentional or unintentional prolongation of active operational states without adequate rest periods or contrarily energy conservation measures. This persistent activity can lead to adverse consequences, affecting device performance, reliability, battery life and security. Extended operation without sufficient downtime increases the risk of system fatigue, potentially causing malfunctions, resource depletion and early downtime, and increased susceptibility to attacks.

  Prolonged wakefulness can also increase hardware degradation, accelerating wear and tear, and contribute towards increased power consumption. These factors may compromise the long-term functionality of IoT devices, leading to decreased operational lifespan and heightened vulnerability to physical and software-based attacks. Moreover, *sleep deprivation* in IoT devices may impact security protocols, as continuous operation can create predictable patterns and opportunities for exploitation by malicious actors;

- **Jamming & Interference** – *jamming and interference* [100] represent pervasive threats to the reliability and security of devices, with the potential to disrupt communication channels and compromise data integrity. Jamming involves the intentional transmission of interference signals on the same frequency bands used by a device, causing communication collisions and rendering the device unable to send or receive

data. This form of attack can lead to service disruptions, data loss, and operational inefficiencies.

Interference, a broader category encompassing unintentional disruptions, can be caused by various sources such as Radio Frequency Interference (RFI) or Electromagnetic Interference (EMI). RFI and EMI may emanate from nearby electronic devices, power lines, or other wireless communication systems, introducing noise that degrades signal quality and compromises data transmission. Deliberate jamming attacks, on the other hand, involve the injection of malicious signals to overwhelm or drown out legitimate communication.

3.6.2 Data

The *Data* category focuses on threats and vulnerabilities associated with the handling, transmission, and storage of sensitive information within a system. This encompasses a wide array of risks, including data breaches, interception, and manipulation, all of which pose substantial challenges to the confidentiality, integrity, and availability of data, as well as the overall objectives of a given system. Securing this critical aspect is fundamental to preserving privacy, trust, and the overall reliability of data-driven services and applications. Understanding the nuances of data related threats is essential for devising comprehensive safeguards to protect the integrity and confidentiality of information flowing through interconnected IoT systems. The following list presents the attacks falling into the *Data* category:

- **Data Tampering** – *data tampering* attacks [62] threaten the integrity and reliability of information exchanged between the different system components. These attacks include unauthorized alterations or manipulation of data traversing the system, aiming to compromise the accuracy, reliability, and trustworthiness of the information.

  Te main form of *data tampering* involves the interception and modification of data packets exchanged between components. Attackers exploit vulnerabilities within communication protocols or insecure transmission channels to intercept, alter, or inject spurious data packets. By manipulating the content of these packets, adversaries can deceive receiving devices or systems, leading to erroneous interpretations of sensor readings, misleading control commands, or unauthorized access to sensitive information.

  Moreover, *data tampering* attacks can occur at the storage level, targeting databases or repositories where data is stored. Attackers may gain unauthorized access to these repositories and modify stored data, introducing inaccuracies or malicious content. Such manipulations can corrupt historical records, compromise analytics, or trigger incorrect decision-making processes based on flawed or falsified data. The ramifications of *data tampering* attacks extend beyond data integrity, potentially impacting the

reliability of automated systems, safety-critical operations, and the trustworthiness of services;

- **Data Theft** – *data theft* [101] represents a threat wherein sensitive information transmitted or stored in or between interconnected components is illicitly accessed and appropriated by unauthorized entities. This insidious attack vector involves the exfiltration of valuable data, including personally identifiable information, proprietary algorithms, intellectual property, or confidential operational details, compromising the confidentiality and privacy of individuals or organizations.

  Data theft exploits vulnerabilities in devices, networks and services to gain unauthorized access to sensitive data, namely weak authentication mechanisms, unencrypted communication channels, or inadequate access controls. The stolen data can subsequently be misused for various malicious purposes, including identity theft, financial fraud, corporate espionage, or compromising the competitive advantage of organizations through the unauthorized acquisition of proprietary data, those being the main motivations for these type of attacks;

- **Data Replay** – *data replay* attacks [102] concern a form of attack where previously intercepted, legitimate data packets are maliciously re-transmitted or replayed within the network. This form of attack capitalizes on vulnerabilities in communication protocols or insecure transmission channels to re-inject previously captured data packets back into the network. By re-transmitting these packets, adversaries aim to deceive devices or systems, causing erroneous responses, unauthorized access, or manipulation of device functionalities.

  The objective of *data replay* attacks is to exploit the trust or behavior established between devices or between the different components, by replaying legitimate commands or sensor readings at a later time. Adversaries may employ this tactic to mimic authentic device behavior, manipulate automated systems, or gain unauthorized access to sensitive areas within the infrastructure. Such attacks can lead to erroneous decision-making processes, compromised data integrity, and disruptions in the functionality of interconnected devices, undermining the reliability and trustworthiness of the entire system;

- **Data Poisoning** – *data poisoning* [103] is a type of attack that aims to manipulate the data collected by IoT devices. This can be done by introducing malicious or inaccurate data into the system, which can then be used to train machine learning models decision-making processes about how devices operate or providing analytics over the monitored *Entities*. The data in IoT devices is often an attractive target for attackers, due to the lack on security controls, data validation, and data protection during transit and at rest. *Data poisoning* attacks can have a variety of consequences, including misclassification of data and incorrect decisions and actions being made by devices, or denial of service, where large amounts of malicious data can overload the system;

- **Data Injection** – *data injection* [104] refers to a malicious attack where unauthorized data is introduced into the IoT system with the intent to manipulate device behavior or compromise system integrity. Adversaries exploit vulnerabilities in devices, communication protocols, or network interfaces to insert falsified or unauthorized data packets into the data flow. This injected data may include fabricated sensor readings, spurious control commands, or misleading information, aiming to deceive devices, systems, or end-users.

  Data injection attacks disrupt normal operations or manipulate device functionalities. By injecting misleading data, attackers can cause devices to execute unauthorized actions, convey inaccurate information to central servers, or trigger incorrect responses in interconnected systems. Such attacks can lead to compromised data integrity, erroneous automation, and potential safety hazards, undermining the reliability and trustworthiness of the system.

3.6.3 Communications & Networking

The *Communications & Networking* domain focuses on threats and weaknesses in the transmission, routing, and exchange of data among interconnected devices within IoT systems. This encompasses various risks such as eavesdropping or traffic analysis, all of which challenge the confidentiality, integrity, and availability of data in transit. Safeguarding this aspect is pivotal in ensuring secure and seamless communication among IoT devices and it clearly frames a series of attacks that are distinct from others in previous categories. The following list presents the attacks falling into the *communications & networking* domain:

- **Traffic Analysis** – *traffic analysis* [62] concerns methods wherein adversaries analyze and interpret patterns, volume, or characteristics of data transmissions within a network. This attack does not involve accessing the actual content of the transmitted data but focuses on scrutinizing the metadata and traffic patterns to infer sensitive information. By observing the timing, frequency, or size of data packets exchanged between devices or with components and/or servers, attackers aim to discern behavioral patterns, user activities, or system operations, thus impacting more user privacy than confidentiality, although affecting the overall system security. Adversaries exploit vulnerabilities, undocumented behaviour or abnormal scenarios in communication protocols (e.g., *idle scans)*, network architectures, or transmission channels to intercept and analyze metadata associated with data transmissions. Through the aggregation and analysis of this metadata, attackers can deduce patterns of behavior, identify endpoints, discern operational schedules, or extract contextually sensitive information without accessing the actual contents of the transmitted data;
- **Sniffing** – *sniffing* [105] represents a form of passive eavesdropping, where an at-

tacker intercepts and monitors data transmitted over the network. This attack does not involve altering, interrupting, or injecting on the communication flow, but focuses on surreptitiously capturing unencrypted or improperly secured data packets, while being outside of the system and network. By employing specialized hardware or software tools, adversaries can intercept and analyze these transmitted packets, potentially extracting sensitive information (such as authentication credentials, personally identifiable information) or command sequences. The main difference between *sniffing* and *eavesdropping* is that the latter requires the attacker to be a system insider, while the former is performed from the perspective of an outsider, without ever needing to intrude the system;

- **Eavesdropping** – *eavesdropping* [106] refers to the act of intercepting and monitoring communications between interconnected devices, specifically from within the network (either being an unauthorized party in the network, or an unprivileged one), with the intent to obtain sensitive information. This form of attack involves infiltrating and listening in on data transmissions or conversations without the knowledge or consent of the communicating parties. Adversaries exploit vulnerabilities in wireless communication protocols, unencrypted channels, or inadequate security measures to eavesdrop on data exchanges, potentially extracting confidential data, commands, or personally identifiable information, and through analyzing them, compromise the confidentiality, integrity, or privacy of data exchanged within IoT systems;

- **Spoofing** – *spoofing* [107] refers to the deceptive practice of falsifying data, identities, or network addresses to impersonate legitimate entities within a system. Attackers perpetrate spoofing attacks by masquerading as trusted devices, users, or systems, typically making use of the fact that the origin of addresses or IDentifications (IDs) is not *data authenticated*, or by exploiting vulnerabilities in authentication protocols or communication channels. This malicious tactic can take various forms such as ID forgery, IP or MAC address spoofing, or Domain Name System (DNS) spoofing. By falsifying identities or network addresses, attackers attempt to bypass authentication measures, infiltrate secure networks, escalate privileges and thus conduct unauthorized actions within IoT systems;

- **Sinkhole** – a *sinkhole* [108] attack revolves around a strategy wherein malicious actors manipulate network traffic by redirecting it to a centralized point controlled by the attackers, known as the sinkhole. This method involves rerouting legitimate traffic from its intended destination to a controlled server or device under the command of the attacker. By exploiting vulnerabilities in network routing protocols, in the assigning or resolving of addresses, adversaries redirect traffic towards the sinkhole, allowing them to intercept, analyze, or manipulate the data passing through. The primary objective of a sinkhole attack in IoT is then to gain unauthorized access to sensitive information, disrupt communication flows, or compromise the integrity of data exchanged between devices or systems. This attack might be one of the predecessors of other attacks men-

tioned herein, even to *sleep deprivation*, as the inclusion of a sinkhole may result in added communication and energy costs;

- **Denial of Service** – *Denial of Service (DoS)* attacks [109] refer to malicious activities aimed at disrupting or degrading the availability and functionality of IoT devices, networks or services, causing the systems to become unresponsive, slow down, or crash. DoSs can either be achieved through *flooding* or through the exploitation of weaknesses, e.g., in the implementation of the network stacks. *Flooding* attacks overwhelm network resources by inundating the network or target devices with an excessive volume of data packets or requests, aiming to disrupt normal operations by consuming available bandwidth, server resources, or device capacities.
  Given that many IoT devices of a system are resource constrained, they are particularly vulnerable to exhaustion. Additionally, due to their pervasive nature, IoT is nowadays seen by malicious actors as an attractive means to build massive botnets and conduct Distributed Denial of Services (DDoSs);

- **Node Injection** – *node injection* [62] concerns a malicious actor inserting unauthorized or rogue nodes into an established IoT system aiming to compromise the integrity and functionality of the network. These injected nodes can masquerade as legitimate devices, allowing adversaries to gain unauthorized access, manipulate data, or disrupt communication between genuine IoT devices, leading to potential data breaches, unauthorized control over devices, or disruption of critical services. *Node injection* is particularly relevant in the IoT ecosystem, especially in the WSNs domain;

- **Man-in-the-Middle** – a *Man-in-the-Middle (MitM)* attack [110] is a form of attack in which an unauthorized third party intercepts and alters communication between two parties, often without their knowledge. Such an attack occurs when an attacker inserts themselves between the communication path of devices, including in one of the endpoints (e.g., man-in-the-browser). By positioning themselves as an intermediary, the attacker can block, monitor, manipulate and inject data exchanged between the communicating parties, or impersonating legitimate entities to gain unauthorized access or control. Attackers exploit vulnerabilities in communication channels, insecure protocols, or weak authentication mechanisms to intercept and modify data packets exchanged between IoT devices or systems;

- **Sybil** – *sybil* attacks [111] involve the potential injection or compromise of a malicious node and the creation of multiple fabricated identities or (virtual) nodes, puppeteered by that single malicious entity. The attacker generates numerous fake identities or nodes, known as *Sybil* nodes, to deceive the authentication and trust mechanisms. These fabricated identities may appear as legitimate devices, allowing the attacker to gain disproportionate influence (e.g., in consensus mechanisms), control, or privileges within the IoT system, which undermines the trust and integrity of the network, and may lead to potential data manipulation, unauthorized control, or misinterpretation

of network behavior. Attackers exploit vulnerabilities in the authentication protocols or trust establishment mechanisms to introduce the *Sybil* nodes. *Sybil* attacks apply especially to WSNs.

3.6.4 User

The *User* domain focuses on the human element within an IoT system, encompassing threats and vulnerabilities arising from user behavior, actions, or interactions with devices or applications. This includes social engineering attacks, weak authentication practices, or misconfiguration of the system. Securing this domain involves educating users, enforcing strong authentication measures, and promoting security awareness to mitigate risks associated with human vulnerabilities. Understanding the human factor in IoT security is crucial for fostering a culture of security consciousness and ensuring responsible user behavior to fortify the overall resilience of IoT systems. The following list describes the attacks included in the *User* domain of the taxonomy:

- **Social Engineering** – *social engineering* [112] is an expression used to refer the manipulation of users or stakeholders to divulge sensitive information or perform actions that compromise the security of the IoT infrastructure. This form of attack exploits human psychology, often through deceptive techniques, including phishing (and its many variants, such as smishing), pretexting, or impersonation, to persuade individuals into revealing credentials, providing access to devices, or unwittingly aiding malicious actors in compromising the security of IoT networks (e.g., installing malware), through human vulnerabilities such as trust, curiosity or fear;
- **Misconfiguration** – *misconfiguration* [113] concerns the incorrect setup or flawed configuration of devices, networks, or security settings within the system. It involves human errors or oversights in configuring devices, cloud services, network components, or access controls, leaving weaknesses that can be exploited by attackers. Misconfigurations can lead to unintended exposure of sensitive data, insecure access points, or flawed authentication mechanisms, compromising the confidentiality, integrity, or availability of the system as a whole. Vulnerabilities arising from misconfigurations might include default or weak passwords, improperly configured access controls, unpatched firmware, or exposed network ports, all of which can then be leveraged by attackers to conduct many other actions;
- **Brute Force** – *brute force* attacks [114] involve automated attempts by adversaries to gain unauthorized access to devices, systems, or networks by systematically trying numerous combinations of usernames, passwords, or encryption keys. This attack method relies on the sheer volume of attempts to guess or crack authentication credentials, exploiting weak or default settings within systems, and easily guessable

passwords. Attackers use specialized software or scripts to repeatedly try different combinations until they successfully break through security barriers. It was decided to include *brute force* in this major category because the success of this type of attacks is directly related with the User and the way (s)he configures the system;

- **Pharming** – *pharming* [115] is an attack that manipulates the DNS to redirect users from legitimate websites or services to fraudulent ones without their knowledge. This attack involves compromising DNS servers or altering DNS records to redirect web traffic to malicious websites that mimic legitimate ones. Attackers exploit vulnerabilities in name or address resolution services or devices to redirect users to counterfeit websites, exploiting user trust in legitimate websites, and stealing sensitive information, such as login credentials or financial data.

3.6.5 Application

The *Application* category tries to capture threats and vulnerabilities stemming from the development, deployment, and operation of software applications within IoT systems. It concerns software vulnerabilities, infections with malicious software, and insecure coding practices. Securing this domain entails adopting secure coding methodologies, implementing robust software development practices, and regularly updating applications to address known vulnerabilities. A solid idea of application-related threats is essential for building resilient and secure IoT applications, safeguarding data integrity, and ensuring the reliable functionality of IoT systems. The attacks included in the part of the taxonomy for the *Application* category are presented *infra*:

- **Denial of Service** – *DoS* [109] attacks target the software and applications running on IoT devices, cloud services, or servers supporting the IoT infrastructure. These attacks aim to disrupt the availability and functionality of applications, rendering them inaccessible or non-operational. Attackers exploit vulnerabilities in applications, flood them with excessive requests, or manipulate their functionalities to exhaust resources, causing service degradation or complete unavailability. Attackers may target specific application endpoints or APIs, initiate resource-intensive tasks, or manipulate application functionalities to consume excessive computing resources, causing the application to become unresponsive or crash. DoSs were also included for *communications & networking*, thought the focus there was on the network, allowing to differentiate attacks in different domains;

- **Virus and Worms** – *viruses and worms* [108] are malicious software programs that can affect (operating) systems, propagating and causing various forms of damage or disruption. The following is a description of each of these types of programs:

  - **Viruses** – self-replicating malicious programs that attach themselves to legitimate files or programs within an operating system. These viruses spread when infected files or software are shared or transferred between devices. Once activated, they can execute unwanted actions, compromise device functionalities, or steal sensitive information. For instance, a virus might corrupt firmware, disrupt operations, or even render an IoT device inoperable. *Viruses* require human intervention in given parts of their spreading process;
  - **Worms** – standalone malicious programs that can replicate themselves and spread across networks independently, unlike viruses that need a host file and human interaction to spread. Worms can exploit vulnerabilities in devices or network protocols, rapidly spreading through interconnected devices. They cause disruptions, compromise data integrity, or execute destructive actions on vulnerable devices.

- **Ransomware, Spyware and Botnets** – these specific types of *malware* [116] are herein proposed to be in a different class to *viruses and worms*, to enable practitioners to better segregate these types of threats, as there is also some implicit historical pertinence, with the significant increase in ransomware more recently. They encompass a broad category of software programs specifically designed to damage, or gain unauthorized access to a system or ask for ransoms, causing various forms of harm, such as data theft, device manipulation, or network disruption. Following is a description of these types of malware:
  - **Ransomware** – this type of malware encrypts or steals important data from a system, rendering it inaccessible until a ransom is paid. It can disrupt operations, compromise device functionalities, and lead to data loss;
  - **Spyware** – *spyware* infiltrates the system to covertly gather sensitive information, such as passwords, system or personal data, compromising privacy;
  - **Botnets** – *botnets* are networks of compromised devices infected with malware (e.g., trojans) and controlled by attackers. They can be used to carry out DDoS attacks or execute other malicious activities.

- **Code Injection** – *Code injections* [117] are a type of attack that exploits vulnerabilities in applications associated with a given system. They involve inserting malicious code into input fields or parameters of an application, exploiting vulnerabilities in poorly coded or unprotected applications within the IoT system, tricking the application into executing unintended commands, bypassing authentication mechanisms, and allowing attackers to gain unauthorized access to data and/or perform unauthorized operations. As a a particular type of code injection, *Structured Query Language (SQL) injections* are highlighted as a frequent attack vector where poorly coded database operational commands can be exploited to trick the application into execute unintended SQL commands, manipulating database queries, and allowing an attacker to gain access and/or perform modifications to databases.

3.7 Conclusions

This chapter presented the main foundations of the proposed model and attack taxonomy. With these definitions, the ground is set for the work that will be presented in chapter 4, focusing on system and attack modeling for the IoT. Both tasks of devising a system model and a taxonomy are hard and the resulting artefacts presented herein are perhaps better defined as a work-in-progress so as to conveniently capture the evolution of IoT and receive improvements that benefit their orthogonality.

One of the main motivations for proposing a new system model for IoT was related with the conclusion that many authors were also using *add-hoc* models in some of the surveyed works. Though clearly inspired in security requirement and properties, the proposed model was designed to be simultaneously generic, so that the model is both applicable to the majority of IoT application scenarios (which were also defined, in subsection 3.2). From the definition of each component, an identification of where a given set of identified security requirements needs to be met was provided, with the objective of constituting a valuable resource in the development process, by giving better suggestions on where security mechanisms will need to be implemented to fulfill the security requirements.

The designing of an attack taxonomy for the IoT was a challenging process for many reasons: (i) the attack surface of IoT overall; (ii) the number of different concepts and similarity of several attacks; (iii) the very prolific and evolving nature of many technologies surrounding IoT; and (iv), the difficulty associated with defining completely orthogonal and meaningful classes that developers may relate with. The work presented in this chapter was key to understand and conceptualize IoT, enabling the development of tools and the overall progression of the work.

Chapter 4

Attack and System Modeling on the IoT

4.1 Introduction

This chapter builds upon the IoT system model and attack taxonomy presented in chapter 3, to propose attack and system modeling solutions. The IoT system model provides a structured framework for understanding the different components and interactions within an IoT ecosystem, while the attack taxonomy classifies and categorizes various IoT attack techniques. From these contributions, it is possible to apply them to develop and adapt system and attack modeling solutions.

Existing tools and techniques for attack and system modeling can have limitations in capturing the complexity of IoT systems [10]. Traditional tools, which may be generic or custom-purpose for other areas, often struggle to handle the heterogeneity of IoT systems, while modeling techniques may not adequately represent the dynamics and interactions between system components. To address these limitations, this chapter introduces both previous works and tools developed towards the main goal of security engineering of IoT, as well as novel work that specifically targets IoT attack and system modeling.

This chapter then serves to detail the modeling processes and developed mappings between system characteristics and attack and system modeling definitions and proposals. The following section, Section 4.1, presents the framework developed within the scope of the SECURIoTESIGN project [18], where this Ph.D. programme was partially developed. Section 4.3 delves into the attack modeling topic and how it can be leveraged to further aid in the development of secure IoT systems. Section 4.4 presents the ATIoT tool starting from the defined attack taxonomy. The tool is capable of generating sets of attack trees given system characteristics. This section describes the work mentioned in the paper [11]. Section 4.5 contextualizes system modeling and how its different techniques aid in the structuring of the development. Section 4.6 approaches several tools from the framework presented in Section 4.1, through their adaptation to the model presented in chapter 3, to better improve the usability of their outputs in the secure development of IoT systems. Finally, Section 4.7 finalizes the chapter with its conclusions.

4.2 SAM (Security Advisory Modules)

The work developed during this programme was part of the SECURIoTESIGN [18], [29] project, whose main practical output was Security Advising Modules (SAM). SAM is a (software) framework that aggregates and connects different modules, and works as a personal, relatable security assistant, capable of communicating with the end-user through a series of easily understandable questions, which can be answered directly by the user. Interestingly, the part of SAM that interacts with the humans is a chatbot with a limited set of interactions (questions) that was designed and developed prior to the dawn of generative chatbots, since the aforementioned project ended in January 2022. Some of the modules included in the framework were developed under the guidance of the author of the thesis (e.g., as part of master programmes). Section 4.4 presents one of the tools of the framework, the one that the author researched and developed individually.

SAM [29], whose conceptual architecture can be seen in figure 4.1, was designed with full modularity in mind, so that new modules can be added, and existing ones can be updated with ease. The framework has a web-based front-end, which then communicates with the back-end through a well documented API [118]. The back-end integrates all modules, functionality, and interconnections. The framework is designed so that modules work through the use of questionnaires, from whose answers a set of recommendations are outputted. These questions are built through a question/answer tree, being possible to create a tree of questions with dependencies, where the appearance of several questions depends on previous given answers. SAM is also prepared to accept modules that generate custom responses, such as diagrams or personalized recommendations, beyond those stored in its database. It also stores user answers and sessions, so that modules can be fed with answers previously given by the user, or with outputs of other modules (i.e., some modules outputs can be used as inputs in other modules).

Through SAM, a user without security expertise has a support to rely on to ensure that security is embedded by design since the beginning of the development process, minimizing potential vulnerabilities and risks. For example, for the SRE module, the user is asked, among other questions, if the system being developed will store some type of information; if a positive answer is given, the user will be provided with the recommendation of the need to ensure *Privacy* and *Confidentiality*, along with a brief description of what each one of these requirements entails.

SAM integrates a set of different modules whose purpose is to aid in the different tasks of designing a secure IoT system. The modules encompass a large set of the entire development process, including: security requirements elicitation, which elicits the main security requirements and properties (e.g., confidentiality, integrity, authentication, access control, availability or accountability), that the system should implement from the basic information

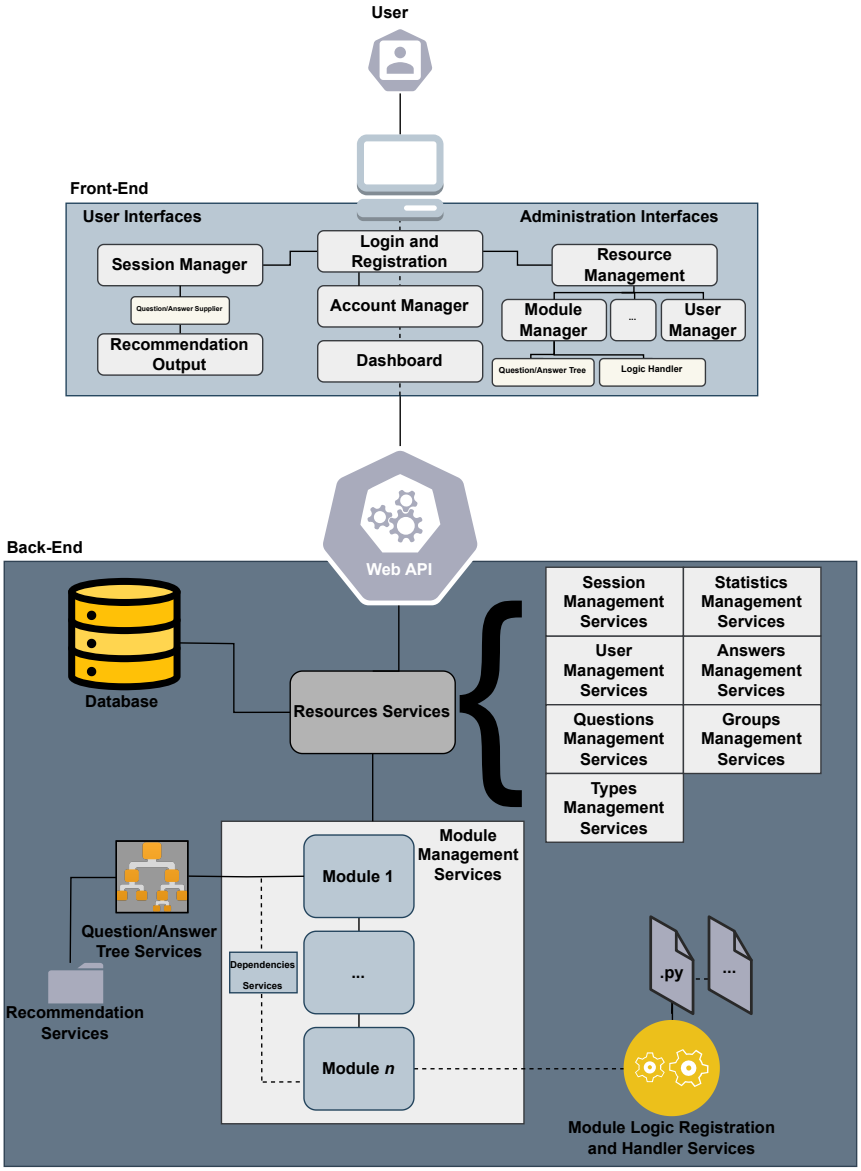


Figure 4.1: High-level view of SAM framework, including its API, database, modules, services, front-end and back-end.

of what composes it and defines it; definition of best practices and guidelines, that highlight common potential vulnerabilities that need to be taken into account during development, and provide information on secure practices that should be implemented; hardware platform advising, which takes the information on the system hardware requirements and defined security requirements, and proposes algorithms that can be implemented to ensure the security requirements are fulfilled, specifically lightweight cryptographic algorithms, for both software and hardware implementations; threat modeling, which presents common weaknesses found in software and hardware components, during design and development phases, and provide the respective Common Weakness Enumeration (CWE) identifier and its description; attack modeling, which presents a set of attack trees of the attacks the system is most likely to be exposed to; and security tests specification for verification of implementation, which provides guidelines on how to design tests to ensure that previous recommendations

and guidelines were followed and correctly implemented.

4.3 Attack Modeling

Attack modeling involves the systematic and structured analysis of potential threats, vulnerabilities, and attack scenarios that a system might face. It provides a detailed and organized overview of the potential attack surfaces within a system, aiding in the prioritization of security efforts. This structured approach allows development teams to focus resources on addressing high-risk areas, thereby maximizing the effectiveness of security measures and minimizing potential damage from attacks. Moreover, attack modeling fosters a proactive security stance throughout the development process. By anticipating potential threats and attack scenarios early in the design and development phases, developers can integrate security measures into the core architecture of their systems. This proactive approach significantly reduces the likelihood of successful attacks, as security considerations are woven into the foundation of the system, rather than added as an afterthought.

Attack modeling incorporates various methodologies and tools, among which *attack trees* [28] stand out due to their effective visual representation. These *attack trees* present a hierarchical model that illustrates the possible paths an attacker might take to compromise a system. By breaking down the system into components and mapping out potential attack vectors, developers gain a comprehensive understanding of its security landscape. This visual representation aids in identifying weak points and enables proactive measures to mitigate these risks before they are exploited. They also facilitate risk assessment and prioritization. Through the hierarchical structure, developers can evaluate the likelihood and impact of various attack scenarios. This assessment helps in allocating resources more effectively by focusing on the most critical vulnerabilities, thereby enhancing the overall security of the system. Moreover, *attack trees* foster collaboration among development teams and security professionals. They provide a common language and framework for discussing potential threats, allowing different stakeholders to contribute their expertise in identifying vulnerabilities and devising countermeasures. Through them, there is a direct aid in the design of effective security controls and defenses. By understanding potential attack paths (inherently embedded in the branching structure), developers can proactively implement security measures at different levels of the architecture, such as encryption protocols, authentication mechanisms, and access controls. Another advantage of *attack trees* is that they are easily (graphically) represented using text with a pinch of ASCII art, which results in the particular collateral of enabling the expansion of the *tree* as needed, since adding new branches is the same of adding newlines in the textual representation. Furthermore, *attack trees* support ongoing security assessments and improvements post-deployment. They serve as a reference point for evaluating the effectiveness of implemented security measures and assist in identifying

new vulnerabilities that may arise due to system changes or evolving threats. This iterative process of assessment and refinement helps in maintaining the resilience of the system over time. Lastly, *attack trees* contribute to raising awareness about security risks among developers and stakeholders involved in the development. By visualizing potential threats and attack paths, these models highlight the importance of incorporating security measures early in the development lifecycle. This heightened awareness encourages a security-centric mindset, leading to better decision-making and more secure systems.

Taking into consideration the attack taxonomy proposed in the previous chapter, it is possible to describe a set of *attack trees* from the attacks raised in the taxonomy, by eliciting where in the system each of these attacks can be applied to. The total of 28 attacks identified in the taxonomy give way to a set of 28 different *attack trees*. These *trees* were designed to be as system-agnostic as possible, so that they may be applicable to any IoT system, regardless of which sub-ecosystem or specific characteristics they may have. This was done to be able to present pre-created *attack trees* to developers, by choosing those which may be relevant for the given system that is being developed. The *trees* are adapted for five of the seven components present in the model defined in the previous chapter, as also explained in section 3.6. Part of the research work thus included the intellectual exercise of generalizing the developed *attack trees*. Developers may subsequently and easily add details (branches) to the trees.

4.4 ATIoT

The ATIoT system generates a series of *attack trees* based on the description of an IoT system, including its ecosystem and various characteristics. These *trees* outline potential attacks relevant to the system, detailing the diverse avenues and targets for each specific attack. In order to be informative and useful, each *tree* includes node descriptions illustrating possible attacker actions and potential access points. Designed for users lacking substantial security expertise, ATIoT automates tree generation from user-provided descriptive system information. The tool comprises three primary components: *Input Gatherer*, *System Analyzer*, and *Tree Generator*, as depicted in Figure 4.2.

4.4.1 Input Gatherer

The process of data collection employs a structured questionnaire, incorporating a total of 24 questions, which serves as a comprehensive guide for users to methodically describe their system. These queries, delineated in table 4.1, have been formulated to aid in the description of the core characteristics of the system, and through them, map and identify requirements,

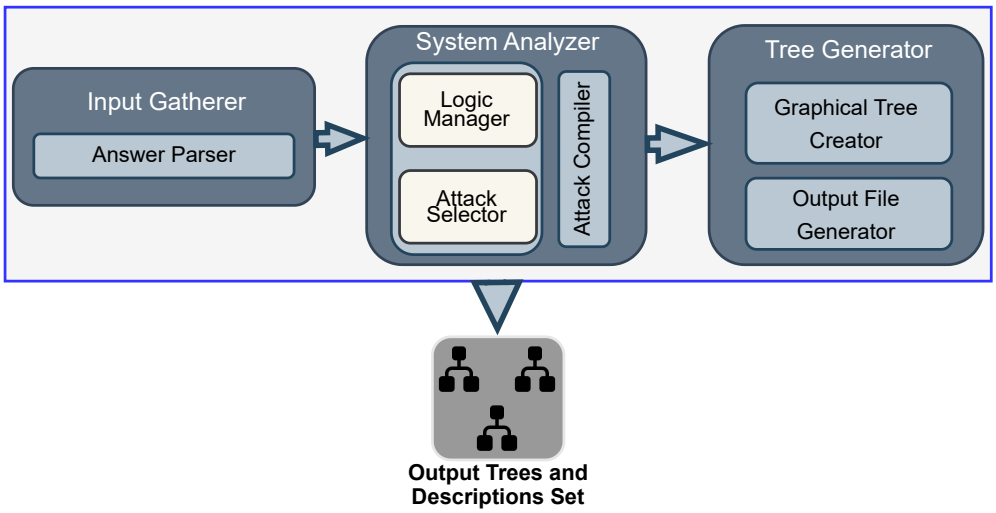


Figure 4.2: High-level architecture of the ATIoT tool.

guidelines or attacks. The questionnaire structure facilitates an efficient exploration of the system properties, steering users towards a correct description of the system, which can then be used to map requirements, establish guidelines, identify threats (all these in other tools of the SAM framework), and select potential attacks (for ATIoT). Each question within the questionnaire was created to maintain simplicity and ensure clarity. Emphasizing a predominantly binary `yes/no` response format, the questions are crafted to be straightforward, aiding in streamlining the data collection process. For questions necessitating more elaborate answers, an array of clear options has been created. The total selection of options within table 4.2 is intended to provide users with a concise set to articulate and contextualize the specificities of the system under analysis. This multifaceted approach in query design not only fosters coherence in information provision but also amplifies the efficacy of subsequent analysis, ultimately contributing to a more comprehensive understanding of the overarching system.

4.4.2 System Analyzer

The central module within the system framework, denoted as the *system analyzer*, and positioned as the core element in Figure 4.2, embodies the main functionality required for discerning and categorizing relevant attacks. This module codes all the required logic designed to identify and delineate the most pertinent attacks potentially impacting the described system. Drawing upon the established taxonomy outlined earlier, this component maps out the prospective attacks most likely to befall the given system.

By assimilating and processing the data gathered from user-provided responses to the questionnaire, the system analyzer identifies the array of potential attacks into a subset specifi-

Q. N.	Question
1	State the domain type for your IoT system.
2	Will the system have a user?
2.1	Will the system have user login?
3	Will the system hold any user information?
4	Will the system store any kind of information?
4.1	What will the level of information stored be?
4.2	Will this information be sent to an external entity?
5	How will the system transmit data?
6	Will the system be connected to the internet?
7	Will it send its data to a cloud?
8	Will it store data in a database?
9	How will the system store data?
10	Will the system receive regular updates?
10.1	How will the system receive software updates?
11	Will the system work with third-party software?
12	Could the messages sent between the system components be captured and resent?
13	Can someone try to impersonate a user to gain access to private information?
14	Can someone gain physical access to the system/components and/or perform any modifications?
15	What type of processing hardware will the devices use?
16	What type of operating system will the devices run?
17	What type of network connectivity will the devices use?
18	What is the expected lifespan of the system?
19	What is the expected usage pattern of the system?
20	What are the expected communication protocols and/or data formats used by the system?

Table 4.1: Set of questions that comprises the questionnaire used by ATIoT.

cally chosen from the unique attributes of the system in focus. This subset of attacks, coupled with comprehensive descriptive details, serves as the foundational input for subsequent phases of analysis.

The process of attack selection operates on a rule-based system. Each rule correlates with a specific answer and identifies potential attacks to which the system might be vulnerable. The rules take into consideration the results of the *Input Gatherer* and determine which *attack trees* should be presented to the user. For instance, if the user confirms the possibility of physical access to the system hardware or components, particularly in an ecosystem like environmental monitoring with accessible physical sensors, the *System Analyzer* will generate attack trees such as *Physical Tampering*, *Theft*, and *Reverse Engineering*.

These rules function akin to conditional statements, shaping the selection process for potential attacks based on user-provided information. They guide the system analyzer in determining the appropriate attacks to present, aligning them with the provided system characteristics and potential vulnerabilities. The following are some examples of rules the system analyzer utilizes:

- `IF` system belongs to the *Environmental Monitoring* ecosystem, `THEN` select *Physical Tampering, Theft, Reverse Engineering, Sleep Deprivation;*
- `IF` system is connected to the Internet, `THEN` select *Traffic Analysis, Denial-of-Service,*

Q. N.	Possible Answers
1	Healthcare/Wearable/Agriculture or Environmental Monitoring/Smart Grid/Transportation and Logistics/Smart Home, Office or Building/ Manufacturing/Smart City
2	Yes/No
2.1	Yes/No
3	Yes/No
4	Yes/No
4.1	Normal/Sensitive/Critical
4.2	Yes/No
5	Device-gateway-Cloud/Device-Application/Device-cloud
6	Yes/No
7	Yes/No
8	Yes/No
9	Local/Cloud/Hybrid
10	Yes/No
10.1	Manually/Over-the-air
11	Yes/No
12	Yes/No
13	Yes/No
14	Yes/No
15	Microcontroller/Single-Board Computer/Custom
16	Real-Time OS/Embedded Linux/Linux/Custom
17	Wi-fi/Cellular/Bluetooth/Zigbee/Z-wave/NFC/RFID
18	Short(1-2 years)/Medium (3-4 years)/Long (5+ years)
19	Continuous/Intermittent/Occasional
20	CoAP/MQTT/HTTPS/Proprietary

Table 4.2: Set of possible answers to the questionnaire used by ATIoT.

Malware;

- `IF` system does not have a user, `THEN` do NOT select *Brute Force* and *Social Engineering;*
- `IF` someone can attempt to impersonate a user to gain access to private information, `THEN` select *Data Tampering, Data Theft, Data Poisoning*, and *Data Injection;*
- `IF` system stores information, `AND` level of information is *Sensitive* `OR` *Critical*, `THEN` select *Data Tampering* and *Data Theft;*
- `IF` someone is able to gain physical access to the physical device, `THEN` select *Physical Tampering, Side-Channel Attacks, Reverse Engineering, Jamming* and *Interference*.

These attacks are then fed to the tree generator component — the last main constituent of ATIoT. This module will then be tasked with converting the subset of attacks into visual attack trees. These graphical representations encapsulate not only the identified attacks but also elucidate their interrelationships, hierarchies, and potential fully describe each tree node for better comprehension.

Table 4.3: Ruleset for ATIoT.

Q. N.	Answer	Potential Attacks
1	Healthcare	Sleep Deprivation, Jamming & Interference, Data Tampering, Data Theft, Data Replay, Data Poisoning, Data Injection
1	Wearables	Sleep Deprivation, Jamming & Interference, Social Engineering
1	Agriculture/Environmental M.	Physical Tampering, Theft, Reverse Engineering, Sleep Deprivation
1	Smart Grids	Physical Tampering, Reverse Engineering, Jamming & Interference, Data Tampering, Data Theft, Data Replay, Data Poisoning, Data Injection, Traffic Analysis, Denial of Service, Sinkhole, Node, Sybil
1	Transportation and Logistics	Physical Tampering, Theft, Data Theft, Jamming & Interference
1	Smart Home/Office/Building	Physical Tampering, Theft, Data Poisoning, Jamming & Interference, Node Injection
1	Smart Manufacturing	Phsyical Tampering, Theft, Data Injection, Data Poisoning, Jamming & Interference
1	Smart Cities	Physical Tampering, Theft, Sleep Deprivation, Jamming & Interference, Denial of Service, Sinkhole, Sybil, Pharming
2	Yes	-
2	No	No Brute Force, No Social Engineering
2.1	Yes	Brute Force, Social Engineering, Code Injection
2.1	No	No Brute Force, No Social Engineering
3	Yes	Data Theft
3	No	-
4	Yes	Data Theft
4	No	No Data Theft
4.1	Normal	Data Theft
4.1	Sensitive	Data Tampering, Data Theft, Social Engineering, Data Poisoning, Data Injection
4.1	Critical	Data Tampering Data Theft, Social Engineering, Data Poisoning, Data Injection
4.2	Yes	Data Theft, Data Poisoning, Data Injection
4.2	No	-
5	Device-Application	Eavesdropping, Jamming & Interference
5	Device-Cloud	Traffic Analysis, Spoofing, Denial of Service, Man-in-the-Middle
5	Device-Gateway-Cloud	Denial of Service, Man-in-the-Middle, Sniffing, Sinkhole, Node Injection
6	Yes	Traffic Analysis, Denial of Service, Malware
6	No	-
7	Yes	Data Theft
7	No	-
8	Yes	Data Theft
8	No	-
9	Local	Physical Tampering, Theft, Data Theft
9	Hybrid	Physical Tampering, Theft, Data Theft
9	Cloud storage	Data Theft, Man-in-the-Middle
10	Yes	-
10	No	-
10.1	OTA updates	Misconfiguration, Man-in-the-Middle, Pharming, Spoofing

Continued on next page

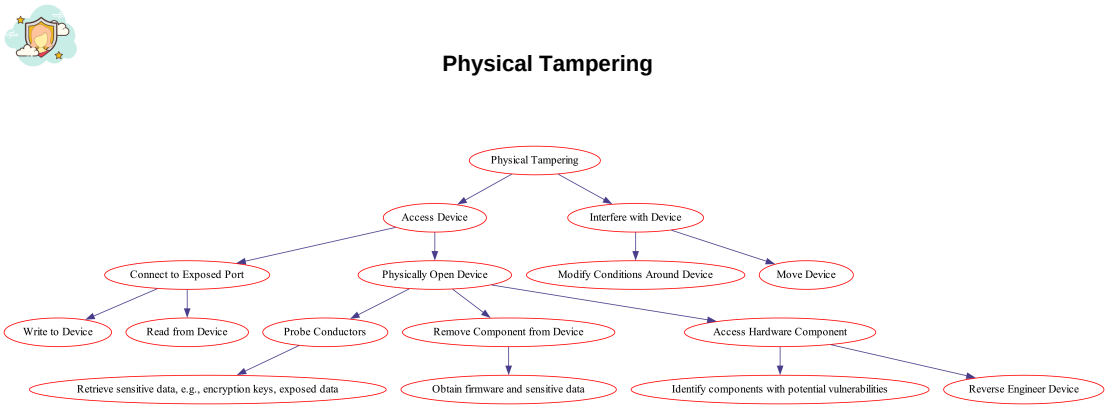
Table 4.3: Ruleset for ATIoT. (Continued)

Q. N.	Answer	Potential Attacks
10.1	Manual updates	Social Engineering, Pharming, Virus and Worms
11	Yes	Misconfiguration, Social Engineering
11	No	-
12	Yes	Data Replay
12	No	-
13	Yes	Data Tampering, Data Theft, Data Poisoning, Data Injection
13	No	-
14	Yes	Physical Tampering, Side-Channel Attacks, Reverse Engineering, Jamming & Interference
14	No	No Physical Tampering, No Theft, No Side-Channel Attacks, No Reverse Engineering
15	Microcontroller	Physical Tampering, Side-Channel Attacks
15	Single-board computer	Physical Tampering, Side-Channel Attacks
15	Custom hardware	Reverse Engineering, Side-Channel Attacks
16	Real-time operating system	Sinkhole, Denial of Service
16	Linux	Sleep Deprivation, Denial of Service, Malware
16	Embedded Linux	Sleep Deprivation, Denial of Service, Malware
16	Custom	Reverse Engineering
17	Wi-Fi	Eavesdropping, Sniffing, Denial of Service
17	Bluetooth	Eavesdropping, Sniffing, Denial of Service
17	Cellular	Eavesdropping, Sniffing, Denial of Service
17	Zigbee	Jamming & Interference, Denial of Service, Node Injection, Sinkhole, Sybil
17	Z-Wave	Jamming & Interference, Denial of Service, Node Injection, Sinkhole, Sybil
17	NFC	Jamming & Interference, Side-Channel Attacks, Physical tampering
17	RFID	Physical tampering, Theft, Social Engineering
18	Short (1-2 years)	-
18	Medium (3-5 years)	Reverse Engineering, Misconfiguration
18	Long (5+ years)	Reverse Engineering, Misconfiguration, Brute Force
19	Continuous operation	Denial of Service, Eavesdropping, Traffic Analysis
19	Intermittent use	Sleep Deprivation, Denial of Service
19	Occasional use	Sleep Deprivation, Spoofing, Node Injection
20	HTTP/HTTPS	Eavesdropping, Traffic Analysis, Man-in-the-Middle
20	CoAP	Sniffing, Denial of Service
20	MQTT	Eavesdropping, Traffic Analysis, Man-in-the-Middle
20	Proprietary protocol	Side-Channel Attacks, Reverse Engineering, Traffic Analysis, Misconfiguration

4.4.3 Tree Generator

The *tree generator* component is responsible for translating the information provided by the *system analyzer* into a comprehensive attack tree visualization. It receives the selected at-

tacks and their associated node structures and descriptions as a key-value store. This store effectively encodes the hierarchical relationships between the attack nodes. The component parses the input key-value store to extract the attack nodes and their children. It then constructs the *attack tree* as an internal representation, maintaining the hierarchical relationships between the nodes. Simultaneously, it retrieves the node descriptions from the provided key-value store and stores them alongside the corresponding nodes. It constructs the graphical elements of the tree, such as nodes, edges, and labels, according to the internal representation of the *attack tree*. Additionally, it inserts the extracted node descriptions into the output document, providing detailed explanations for each attack step. This output, which encapsulates the visual and textual descriptions of the *attack tree*, is then presented to the user. The output format is two-fold: (i) as a PDF document, and (ii) text-based. The trees are generated dynamically during execution time (using the `graphviz` Python package) to benefit the longevity of the tool, allowing for the introduction of new *attack trees* in the future in the form of the appropriate Python data structure, waiving the contributor from the need to worry about the graphical representation. These output documents are readily understandable visualizations and text descriptions, facilitating effective communication of security risks and enables developers to grasp the attack scenarios thoroughly. Figures 4.3 and 4.4 exemplify PDF outputs generated by ATIoT for the *physical tampering* and the *data tampering* attacks. Figure 4.5 represents the same tree for the *physical tampering* attack, but as a text-based output.



Access Device: Physically access the device. This is possible in environments where the device is not fully monitored, or left in the environment unattended.

Interfere with Device: Situations where the attacker does not physically interact with the device (i.e., opening it or connect to it), but can physically move it or alter the environment around it.

Physically Open Device: The attacker opens the shell of the device, and directly interacts with its internal components.

Connect to Exposed Port: The attacker utilizes an active management or access port, e.g., USB or Ethernet. Through it, the attacker may be able to gain access to e.g., configurations, sensitive data, or inject malicious code.

Remove Component from Device: The attacker physically removes parts of the device, such as chips, memory or sensors. From these, the attacker may extract firmware, software, and sensitive data.

1

Figure 4.3: Example of the PDF output of the ATIoT tool for the *physical tampering* attack.

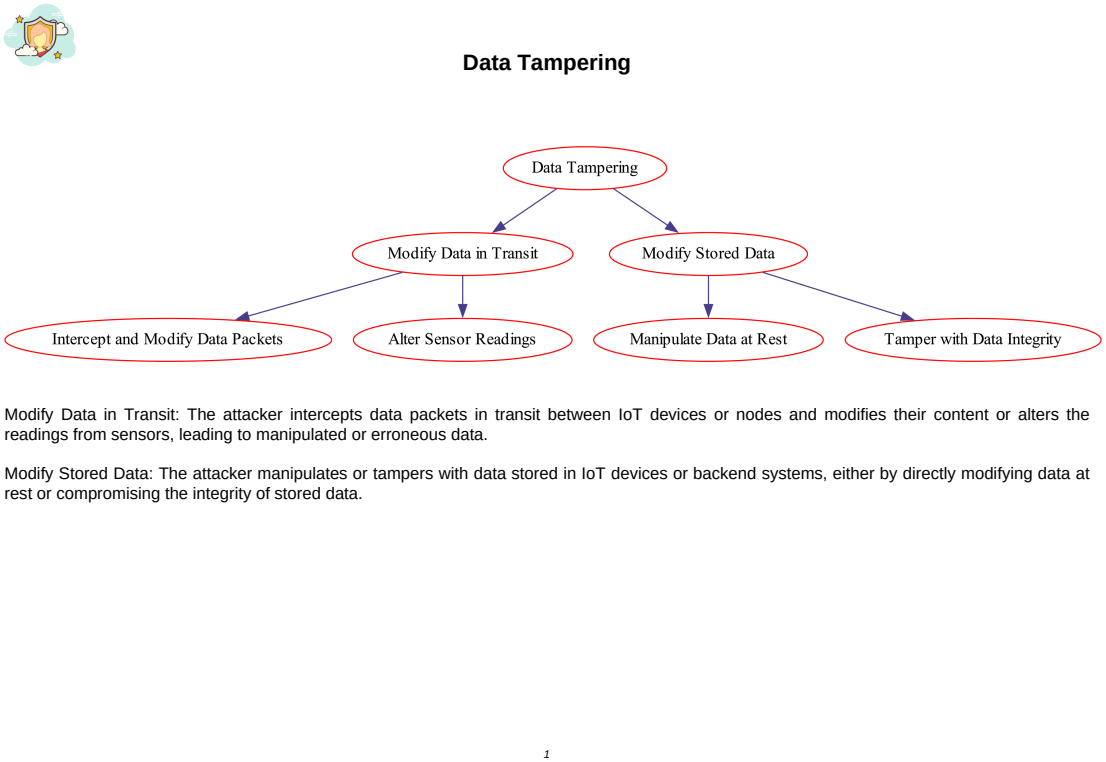


Figure 4.4: Example of the PDF output of the ATIoT tool for the *data tampering* attack.

4.4.4 Usability Evaluation of ATIoT

Testing, with performance and usability in mind, ensures that a given system or tool is working according to the proposed requirements and objectives, as well as allowing to evaluate and assess the overall user experience when using said system or tool. Usability can be described as a general quality of the appropriateness to the task it was designed to perform [119].

ATIoT, and other tools of this kind, are inherently difficult to validate, as the attack modeling process is, in some of its aspects, partially subjective, and is usually specific to the intrinsic details of the system being developed. As such, the decision fell on evaluating the usability, which can be seen as either ease of use, or the ability to use a tool for its intended purpose [120] and usefulness, which can be defined as how applicable the tool is to achieving the objective it was designed and developed for [121], aspects of ATIoT, under the guise of its initial premise: that such a tool should be simple to use, and should benefit the work of developers with low to no security expertise. This usability testing and evaluation was performed on a focus group, comprised of software developers and engineers, with several of them working specifically in the design and development of IoT systems. The core knowledge of ATIoT was presented in the previous parts of this section, and as such, it is considered to be based on the gathered knowledge from exhaustively surveying the existing literature and was additionally consolidated through extensive modeling. That part of the tool was not under

```
| Physical Tampering
|   OR|
|      | Interfere with Device
|      |  AND|
|      |      | Modify Conditions Around Device
|      |      | Move Device
|      | Access Device
|      |   OR|
|      |      | Connect to Exposed Port
|      |      |  AND|
|      |      |      | Read from Device
|      |      |      | Write to Device
|      |      | Physically Open Device
|      |      |   OR|
|      |      |      | Access Hardware Component
|      |      |      |  AND|
|      |      |      |      | Reverse Engineer Device
|      |      |      |      | Identify components with potential vulnerabilities
|      |      |      | Remove Component from Device
|      |      |      |      | Obtain firmware and sensitive data
|      |      |      | Probe Conductors
|      |      |      |      | Retrieve sensitive data, e.g., encryption keys, exposed data
```

Figure 4.5: Example of the text output of the ATIoT tool for the *physical tampering* attack.

analysis in this part of the work, since it would require having access to a pool of cybersecurity experts using the tool, whose scarcity is partially the motivation behind this work. The System Usability Scale (SUS) methodology [119] was chosen and adapted to obtain a representative usability and usefulness rating of the tool. This methodology presents users with a questionnaire, comprised of a total of ten questions, each scoring from 1 to 5, as defined by the Likert scale [122], which SUS applies. The questionnaire developed for this evaluation is based on the SUS questionnaire, but is adapted from it, and is presented in appendix 6.2. It is comprised of a total of 12 questions, which are presented as a series of statements that users then rank according to their level of agreement or disagreement. The evaluation was performed following the steps described below:

- Potential users were approached, to gauge interest in experiencing the tool, and participating in its usability evaluation;
- A small explanation was given to each user, after accepting, explaining the steps that should be taken, and what was required of them;
- The users installed the tool, following the given instructions;
- Users used the tool, being given full freedom on how they used it, with no instructions given. A set of scenarios for IoT systems was made available as inspiration, but users were given freedom to create their own scenarios and imagine system characteristics;
- In the moments following the use of the tool, and before any discussion or debriefing, users were given access to the questionnaire, and the answers were collected, with full anonymity from the users.

The questionnaire was applied to a stratified random sample, which was composed of a total of 17 people, all with software development and computer science and engineering back-

Question	Average Score	Standard Deviation
Q1	4.11	0.86
Q2	2.09	1.14
Q3	4.35	0.93
Q4	1.94	1.25
Q5	4.47	0.80
Q6	1.24	0.44
Q7	3.82	0.88
Q8	1.70	0.85
Q9	4.18	0.95
Q10	1.94	1.25
Q11	4.82	0.39
Q12	1.76	0.83
SUS Score	81.36	13.5

Table 4.4: Questions average scores, and standard deviation for the SUS questionnaire, and total SUS scores.

ground, and some working in the industry developing IoT solutions but with no elements in the sample having a security background. The main results of the SUS questionnaire are presented in table 4.4. The average score of the questionnaire was calculated by adapting from the formula defined for this methodology:

- for each odd numbered question, subtract 1 from the answer value;
- for each even numbered question, subtract the answer value from 5;
- add all values for each user, and divide by 2.08 (to convert to a scale 0-100);
- a score above 68 is considered a positive result.

The achieved SUS score was of 81.36, an overall positive usability score for ATIoT, which shows users found the usability of the tool to be, overall, very positive. Interestingly, the answer to question 11 ("I found the tool to be useful in development work") achieved the highest average value, and the lowest standard deviation, pointing towards not only the good usability of the tool, but its usefulness, in the perspective of the focus group.

Figure 4.6 presents the spread of the given answers for each of the questions.

4.5 System Modeling

System modeling involves a structured and comprehensive approach to defining, analyzing, and illustrating the architecture, components, interactions, and behaviors within a system. It serves as a blueprint that aids in understanding, designing, and optimizing the functioning of complex systems. System modeling encompasses various methodologies and tools,

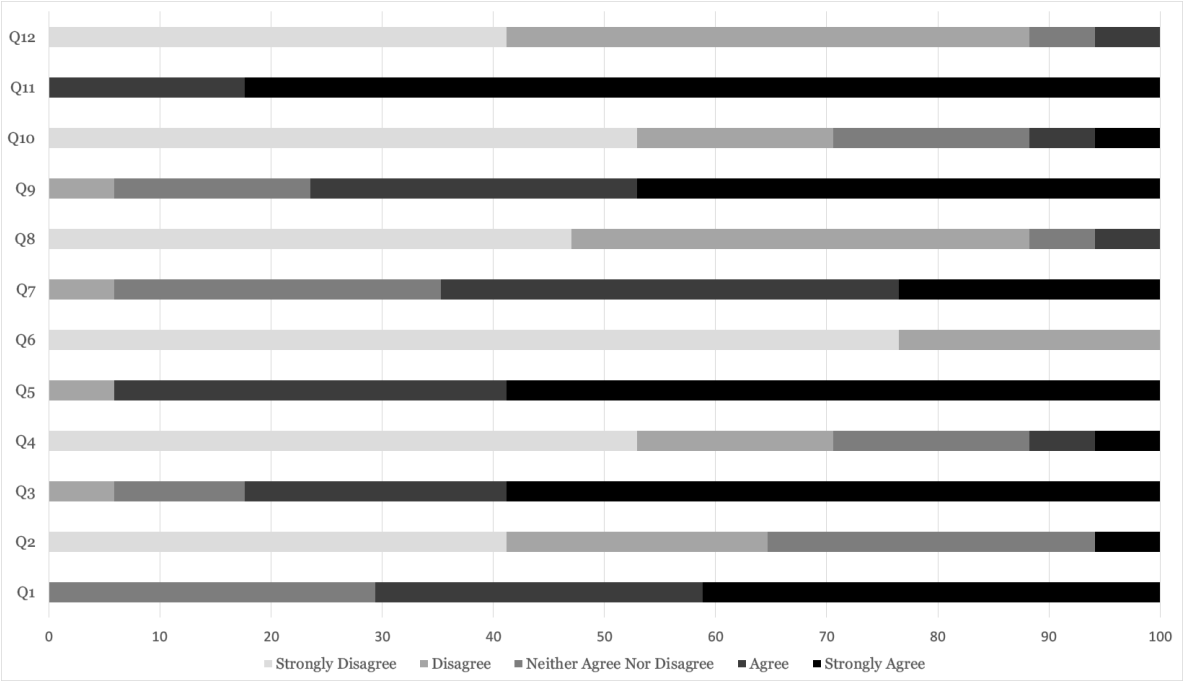


Figure 4.6: Answer spread for each question during the SUS questionnaire.

each contributing to a holistic view of the structure, functionalities, and interdependencies of a given system. One prominent aspect of system modeling is its capacity to offer a clear representation of system components and their relationships. This includes architectural schemes and diagrams, e.g., those drawn using UML, that can depict structure, modules, interfaces, components, and data flows, among others. Through these visualizations, development teams gain insights into their system, facilitating better decision-making in design, development, and maintenance phases. Moreover, system modeling supports the identification and mitigation of potential issues or inefficiencies within the system. By simulating system behaviors, performance, and potential edge cases, developers can anticipate challenges and fine-tune the architecture and functionalities to enhance overall performance and reliability. This proactive approach helps in addressing bottlenecks and potential failure points before they impact functionality or security. System models aid in the allocation of resources and prioritization of efforts during development. They enable teams to pinpoint areas that require more attention or refinement, guiding the allocation of time and resources effectively. This structured approach ensures that development efforts are focused on enhancing core functionalities. System modeling also supports iterative development and continuous improvement. As the system evolves, models can be updated to reflect changes, enabling developers to assess the impact of modifications and enhancements. This iterative process promotes agility and adaptability, ensuring that the system remains aligned with evolving requirements and technological advancements.

4.6 Combining System Model and SAM Tools for System Modeling

The different SAM tools, as exemplified in section 4.1, output their recommendations and suggestions to be applied on the overall system, never specifying which system components should be prioritized, or where mechanisms and/or suggestions should be implemented on. Through the system model presented in chapter 3, it is possible to further identify, for several of these tools, where each of their outputs focuses on, and where they are properly applicable to. This will, as a consequence, improve the quality of the outputs of the tools, improving the overall system modeling that they provide and achieve on the given described system. To better emphasize this, the different tools are herein described in greater detail.

The first tool developed for SAM was the SRE, which is focused on eliciting security requirements given a system description (this is done through a questionnaire, such as with ATIoT, as presented in section 4.4). The identification of security requirements is paramount in providing the developers with a standardized set of requirements that aids them in defining security controls, best practices and mechanisms to implement, and it concerns the inception point in security engineering. The tool, as such, identifies the overall security requirements that a system should fulfill given its characteristics. As one of the main direct outputs of the model, the relation between components and appropriate requirements can be properly applied to the SRE tool, through which it is possible to directly assess in which system component a given security requirement needs to be met, so that developers can then define and decide on which proper guideline or security mechanism they should implement to assure the fulfillment of that security requirement. This correspondence can be seen in table 4.5.

Component	Requirements
Entity	Physical Security, Tamper Detection, Confinement
Device	Confidentiality, Integrity, Authentication, Authorization, Non-Repudiation, Accountability, Reliability, Privacy, Physical Security, Forgery Resistance, Tamper Detection, Data Freshness, Confinement, Data Origin
Processes	Confidentiality, Integrity, Non-Repudiation, Accountability, Reliability, Privacy, Forgery Resistance, Data Freshness, Confinement, Data Origin
Data	Confidentiality, Integrity, Non-Repudiation, Privacy, Forgery Resistance, Data Freshness, Interoperability
Communications & Networking	Confidentiality, Integrity, Availability, Reliability, Privacy, Forgery Resistance, Confinement, Interoperability, Data Origin
Application	Confidentiality, Integrity, Authentication, Accountability, Reliability, Privacy, Forgery Resistance, Data Freshness, Confinement, Interoperability, Data Origin
User	Authentication, Authorization, Non-Repudiation, Accountability, Reliability, Privacy, Data Freshness

Table 4.5: Correspondence between the security requirements outputted by SRE and model components.

The next tool in the framework is the SBPG, which outputs a series of recommendations, guidelines and guides on best practices, by taking as input several system characteristics.

This tool focuses more on practical and technical development decisions and system characteristics, such as, e.g., used programming languages, type of data and data storage, logging or file upload functionality. Applying the model defined in chapter 3, it is possible to identify where more appropriately the guidelines and best practices can be applied to in the context of an IoT system, herein shown in table 4.6.

Component	Recommendation Guides
Device	Access Control, API, Cryptography, File Upload, Logging, Session Management
Processes	Cryptography, Logging
Data	Access Control, Cryptography, Logging
Communications & Networking	Cryptography
Application	Access Control, API, Authentication, Cross-Site Scripting, SQL Injection, Cryptography, File Upload, Input Validation, Logging, Session Management
User	Authentication, Input Validation

Table 4.6: Correspondence between the recommendations outputted by SBPG and model components.

The LWCAR tool of the framework is responsible for, given the set of requirements identified using the SRE tool, identify the different security mechanisms that should be implemented to fulfill those requirements, and given a set of technical system characteristics (mainly device characteristics), such as, e.g., processing power, available memory, throughput or type of data transmission (provided by the user), output the corresponding lightweight algorithms that can be implemented as those security mechanisms. Table 4.7 shows the mapping between the recommendations and the system components.

Requirement	Components	Mechanism	Algorithms
Confidentiality	Device, Processes, Data, Application, Communications & Networking,	Encryption	Grain_v1-80, Grain_v1-128, Enocoro_-v2-128, Trivium-80, MICKEY_2.0-80, TWINE64/80, TWINE64/128, Midori64/128, Midori128/128, SIMON64/96, SIMON64/128, Piccolo64/80, Piccolo64/128, Clefia128/128
Integrity	Device, Processes, Data, Communications & Networking, Application	Hashing	PHOTON-80/20/16, PHOTON-256/32/32, Keccak-f[100], SPONGENT-88/80/8, SPONGENT-128/128/8, SPONGENT-160/160/16, U-QUARK, S-QUARK
Confidentiality & Authentication	Device, Application	Authenticated Encryption	ACORN, AES-GCM, AES-OTR, SILC-AES, SILC-PRESENT, Deoxys, Ascon, CLOC-AES, JAMBU-AES
Privacy	Device, Processes, Data, Communications & Networking, Application, User	Encryption	Grain_v1-80, Grain_v1-128, Enocoro_-v2-128, Trivium-80, MICKEY_2.0-80, TWINE64/80, TWINE64/128, Midori64/128, Midori128/128, SIMON64/96, SIMON64/128, Piccolo64/80, Piccolo64/128, Clefia128/128

Table 4.7: Correspondence between the model components, SRE requirements and LWCAR recommendations.

The TMS tool intends to identify possible weaknesses potentially found in the software and hardware of an IoT system. To define the weaknesses, the vulnerabilities and weaknesses enumerated by CWE [123] were used, and from it, taking also into consideration the Open Web Application Security Project (OWASP) top 25 vulnerabilities to aid in the selection in

terms of relevance, a set of 90 weaknesses to be presented was chosen, and the selection logic was created, using a questionnaire to aid in the system definition, in a similar fashion to the other previously described tools. Through the model, it is possible to further map in which components each vulnerability may apply to, as seen in table 4.8.

Component	**Weakness**
Entity	CWE-200, CWE-1221
Device	CWE-22, CWE-78, CWE-119, CWE-125, CWE-170, CWE-190, CWE-200, CWE-248, CWE-252, CWE-269, CWE-287, CWE-306, CWE-327, CWE-330, CWE-400, CWE-404, CWE-416, CWE-434, CWE-469, CWE-476, CWE-477, CWE-479, CWE-515, CWE-522, CWE-544, CWE-573, CWE-668, CWE-705, CWE-732, CWE-758, CWE-778, CWE-798, CWE-840, CWE-862, CWE-913, CWE-1069, CWE-1109, CWE-1209, CWE-1220, CWE-1221, CWE-1224, CWE-1242, CWE-1244, CWE-1245, CWE-1247, CWE-1251, CWE-1253, CWE-1253, CWE-1261, CWE-1262, CWE-1263, CWE-1267, CWE-1268, CWE-1271, CWE-1273, CWE-1274, CWE-1276, CWE-1277, CWE-1278, CWE-1279, CWE-1280, CWE-1281, CWE-1282, CWE-1291, CWE-1294, CWE-1295, CWE-1296, CWE-1297, CWE-1298, CWE-1299, CWE-1304, CWE-1310, CWE-1311, CWE-1313, CWE-1319, CWE-1326, CWE-1331, CWE-1334, CWE-1338
Processes	CWE-170, CWE-190, CWE-200, CWE-248, CWE-252, CWE-327, CWE-469, CWE-476, CWE-479, CWE-502, CWE-515, CWE-544, CWE-573, CWE-705, CWE-758, CWE-778, CWE-787, CWE-787, CWE-1069, CWE-1109, CWE-1209, CWE-1242, CWE-1244, CWE-1291
Data	CWE-190, CWE-200, CWE-269, CWE-327„ CWE-477, CWE-522, CWE-778, CWE-1280
Communications & Networking	CWE-327, CWE-330, CWE-441, CWE-477, CWE-522, CWE-544, CWE-573, CWE-778, CWE-1069, CWE-1242, CWE-1253, CWE-1331
Application	CWE-20, CWE-22, CWE-74, CWE-78, CWE-79, CWE-89, CWE-94, CWE-98, CWE-119, CWE-125, CWE-170, CWE-190, CWE-200, CWE-248, CWE-252, CWE-269, CWE-287, CWE-306, CWE-327, CWE-330, CWE-352, CWE-416, CWE-434, CWE-441, CWE-469, CWE-476, CWE-477, CWE-479, CWE-515, CWE-522, CWE-544, CWE-573, CWE-611, CWE-625, CWE-668, CWE-705, CWE-732, CWE-758, CWE-778, CWE-787, CWE-798, CWE-840, CWE-862, CWE-913, CWE-1069, CWE-1109, CWE-1209, CWE-1220, CWE-1221, CWE-1242, CWE-1280, CWE-1291, CWE-1295
User	CWE-200, CWE-269, CWE-287

Table 4.8: Correspondence between TMS weaknesses and model components.

4.7 Conclusions

This chapter defined strategies for attack and system modeling for IoT, taking into consideration the main premises of this work: that this modeling should be security-driven and that it should target developers with low security expertise.

The work of the previous chapter, chapter 3, served as the basis for the work presented in this chapter, by utilizing both the defined model and attack taxonomy as a basis for the attack modeling, including ATIoT, as presented in sections 4.3 and 4.4, and the system modeling,

presented in sections 4.5 and 4.6. With these defined strategies and tools, it is possible to plan secure designs and development processes for IoT systems, ensuring that security is embedded from the start.

The work described herein started (and mostly ended) before the dawn of LLMs, generative AI and the new chatbots. Nonetheless, it is interesting to notice that a simple chatbot was the method that was selected as the interface of the tools of the SECURIoTESIGN framework, precisely to make them easier to use. The lengthy work to define the rules behind the tools might actually provide now the basis to fine-tune language models to teach security to these AI bots and create adaptable and more powerful tools. Chapter 5 elaborates on how the logics and mappings presented in this chapter were used to prepare and train AI models, to investigate their potential in discerning the decision-making logic of the tools and, for LLMs, their reliability in terms of answers, and using their knowledge bases for expanding the scope of the potential given answers.

(This page is intentionally left blank.)

Chapter 5

Leveraging Artificial Intelligence Techniques for Tool Enhancement

5.1 Introduction

This chapter presents exploratory work on the potential usage of Machine Learning (ML) techniques and fine-tuning a LLM to automate, update and transform tools described in the previous chapter. The use of these techniques shows promise in solving one of the main issue that tools based on current day knowledge all suffer: the (rapid) advancement of technology. By inferring knowledge and logic, and tapping into large amounts of data, these techniques may provide a means to ensure such tools are automatically kept up-to-date and relevant. On the other hand, the work performed in the scope of this Ph.D. work may be used in the near future to build the datasets needed to train security oriented chatbots.

The chapter, therefore, details some experiments performed to explore such techniques and the results achieved with those experiments. Section 5.2 describes the application of ML to the SRE tool, and the experiments performed to assess the performance of different ML models and classifiers. This section is strongly sustained by [16]. Section 5.3 presents the use of Natural Language Processing (NLP) techniques for enhancing the tools, mainly the ATIoT tool. The work presented in this section is presented in [12]. Finally, Section 5.4 finalizes the chapter with its conclusions.

5.2 Applying Machine Learning to Security Requirements Elicitation

In the dynamic realm of IoT, changes and new approaches appear at a fast pace, potentially making much of the work presented in the previous chapters limited terms of usefulness and applicability. While the basis can be considered sound, and therefore iterated upon when, in the future, it is possible to look back and see how technological advancements and new approaches can be integrated into it, without such effort, any developed tool, and any proposed methodologies, architectures, or taxonomies, are susceptible to be no longer considered valid and useful due to the aforementioned advancements.

With AI becoming more and more a hot topic in terms of its application to different areas, training a model to be capable of inferring the underlying logic of the tools conceptualized along this project can be a potential solution for aiding the issue of updating a tool. A model of this kind, having undergone training on an extensive dataset comprising a large group of training examples, can capture the decision-making process and discern relationships between input data (the questionnaires) and output data (requirements, recommendations, models. etc.). As novel advancements are detected, the integration of new training data into the model becomes feasible, which will then be reflected in updated outputs associated with these technological advancements. Subsequently, upon assimilating the new data into its dataset, the model autonomously adjusts its internal representations, incorporating the latest knowledge sans manual intervention. This adaptive mechanism could then ensure the pertinence and contemporaneity of the tool, even amidst technological evolution.

The advantages of integrating an AI-driven model into the update protocol of a tool are several. First, it reduces or even removes the necessity for manual updates, thereby streamlining the process and mitigating the potential for human error. Second, it facilitates a continuous enhancement paradigm by seamlessly assimilating novel knowledge as it emerges. Third, it ensures a seamless adaptation to dynamic technological landscapes, thereby sustaining its efficacy.

5.2.1 Analysis and Identification of the Approach

To verify the feasibility of using an approach such as the one herein described, a methodology was developed and applied to one of the first tools developed for SAM, the SRE tool. The main underlying idea was to replace the (hardcoded) engine of SRE with a box that learned its functioning with ML.

The inference machine is designed to rely exclusively on a singular type of input, which are the responses given by the user. In the SRE tool, this input is utilized to derive a set of requirements, each associated with a binary indicator denoting if the requirement should be present or not. Considering that this machine will output a series of security requirements, in terms of machine learning the problem can be characterized as multi-label, given that a single input prompts the generation of multiple requirements. Additionally, the output is configured as a multi-output schema, wherein each label is associated with two potential integer values: `0` signifies the lack of necessity for a specific security requirement, whereas `1` designates the need for recommending the identified security requirement in the target system.

The fundamental concept underlying this methodology includes the construction of the machine that receives user responses, formulates a set of requirements deemed suitable for the specified scenario, and subsequently presents the set of security requirements. Concurrently,

the machine initiates an inquiry with the user regarding the accuracy of the generated output. At this point, should the user opt to do so, they possess the capability to give feedback to the machine, thereby contributing to the refinement of the requirements elicitation process. This feedback mechanism serves as a means for the user to signal the appropriateness or lack thereof for each of the outputted requirements.

With the problem being multi-label and multi-output, it was possible to narrow down the number of possible models needed to test and apply to achieve the desired outcome. A total of 7 different models were chosen for testing, to then assess their applicability and performance in the given task. The chosen models were Classifier Chains (CC), Binary Relevance (BR), Label Powersets (LP), Multi-Label k-Nearest Neighbors (MLkNN), Binary Relevance k-Nearest Neighbors (BRkNN), Label Space Partitioning (LSP) and Majority Voting (MV).

The BR model [124] is a classification approach commonly employed in the context of multi-label classification, where each instance can be associated with multiple labels simultaneously, as opposed to traditional binary or multi-class classification where each instance is assigned to a single label. BR addresses this by transforming the problem into multiple independent binary classification tasks. Essentially, it treats each label as a separate and independent binary classification problem. For each label, a binary classifier is trained to predict whether that specific label is relevant or not for a given instance. For each transformed binary label, a separate binary classifier (e.g., logistic regression, support vector machine, or any other binary classification algorithm) is trained. During the prediction phase, each binary classifier independently predicts whether its corresponding label is relevant for a given instance. The final multi-label prediction is obtained by combining the individual binary predictions for all labels.

The CC model [125] is another approach used in the context of multi-label classification, addressing situations where labels may exhibit dependencies. It is a BR approach in this context, but it differs from BR in the sense that the attribute space for each binary model is expanded with the 0/1 label relevance of all previous classifiers, resulting in a classifier chain. It aims to capture label dependencies by considering the order in which the labels are processed. This can be beneficial when labels are not mutually independent. This order is determined prior to training the model. One common strategy is to use a predefined order or to employ a data-driven approach to identify label dependencies. The order in which the classifiers are trained corresponds to the predetermined label sequence. The first classifier is trained to predict the presence or absence of the first label in the sequence. The second classifier takes into account the features of the input data and the prediction of the first classifier, and it predicts the presence or absence of the second label. This process continues until all labels are considered.

The LP method [126] is a straightforward and efficient problem-solving strategy, where each

collection of labels in a training set is treated as one of the classes in a new single-label classification task. For example, if there are three labels (A, B, C), the LP model considers all possible combinations (e.g., A, B, C, AB, AC, BC, ABC) as individual classes. The original multi-label dataset is transformed into a new dataset, where each instance is associated with a unique combination of labels. If an instance originally had labels A and B, it is now associated with the class "AB" in the transformed dataset. A multi-class classification algorithm (e.g., decision trees, support vector machines, or neural networks) is then trained on the transformed dataset. The objective is to learn a model that can classify instances into the various label combinations.

MLkNN [127] extends the traditional k-Nearest Neighbors (kNN) algorithm to handle instances associated with multiple labels simultaneously. MLkNN is particularly useful in scenarios where each data point may belong to multiple classes, and there may be dependencies between these classes. For each instance in the transformed dataset, MLkNN identifies its k-nearest neighbors based on the feature space. The neighbors are determined using a distance metric, such as Euclidean distance, and the k nearest neighbors are selected. For each instance, it then assigns labels based on the majority labels of its k-nearest neighbors. The model takes into account the frequency of each label in the neighbors and assigns the labels with the highest frequencies to the instance. The parameter k determines the number of neighbors considered. The Maximum *a Posteriori* (MaP) principle is then used to evaluate the label set based on the probabilities to convert them into binary predictions. For example, if the estimated probability for a label is above a certain threshold, that label is considered present; otherwise, it is considered absent.

The BRkNN [128] model is designed to handle instances associated with multiple labels, taking into account the relationships between labels and utilizing the kNN algorithm for classification. Similarly to other multi-label models, BRkNN starts by transforming the multi-label problem into a multi-class problem using the LP transformation. Each unique combination of labels in the training dataset is treated as a separate class. For each instance in the transformed dataset, just like MLkNN, BRkNN identifies its k nearest neighbors based on the feature space. For each label in the transformed dataset, a binary classifier is trained independently. For a given instance, labels are assigned based on the majority labels of its k nearest neighbors. The binary classifiers associated with each label contribute to the label assignments. The labels with the highest frequency among the neighbors are assigned to the instance.

LSP [129] can be defined as a set of simple, sub-optimal classifiers that can achieve similar efficiency to a strong complex classifier. The label space is partitioned into smaller multi-label problems. Instead of the split being done randomly, or for a given grid of values, a label space clusterer is used, that separates the labels into subsets, given their characteristics. The classifiers are then applied to the subsets. When decoding, an appropriate weighting

technique is used to focus on the most significant partitions, resulting in a list of scores for each label.

The MV model [130] combines the predictions of multiple individual models and makes a final prediction based on the majority decision of the ensemble. It begins with a set of individual models, each trained on the same dataset, but using different algorithms, hyperparameters, or subsets of the data. When a new instance needs to be classified, each individual model in the ensemble independently makes its prediction based on the input features. The predictions of all individual models are then combined through a voting mechanism. The final prediction for the ensemble is the class that receives the majority of votes from the individual models.

To start the training and testing phase of the chosen models, a dataset was generated through all possible input combinations and outcomes of the SRE tool. The dataset comprises 17458 test and 70094 train instances, each one containing 17 answers (inputs) and 16 requirements (outputs).

To select the optimal model and classifier combination, a comprehensive set of tests was conducted, evaluating each potential pairing based on various metrics, including accuracy, mean accuracy, processing time, and the minimum dataset size required for acceptable performance.

For evaluating the best combination of model and classifier, the list of employed classifiers [131], [132] was as follows:

- AdaBoost Classifier (ABC) [133] – an ensemble learning method that combines multiple weak classifiers to create a strong classifier, with higher accuracy than any of the individual classifiers alone. It is an iterative method that involves training a series of weak classifiers, each designed to focus on classifying the misclassified instances from the previous iteration. This process of weighting and re-training the classifiers continues until a satisfactory level of accuracy is achieved;
- Extra Trees Classifier (ETC) [134] – this classifier falls under the category of decision trees. It is characterized by its extreme randomization in the construction of individual trees, leading to reduced variance and improved predictive performance. The combination of randomization techniques makes it less prone to overfitting and exhibiting improved predictive accuracy compared to traditional decision trees or random forests;
- Random Forest Classifier (RFC) [135] – the RFC achieves its effectiveness by combining multiple decision trees, each trained on a different subset of the training data and using a random subset of features at each split. The randomness introduced in the tree-building process prevents overfitting and enhances its generalization ability;

- Decision Tree Classifier (DTC) [136] – this classifier splits data into smaller subsets based on a set of rules until each subset is homogeneous. Its name is due to its structure of decision nodes and branches, which resembles a tree. It starts by selecting the most informative feature to split the data on. Once a feature is selected, the data is split into two or more subsets based on the value of the feature. This process is repeated recursively for each subset until the desired level of granularity is achieved;

- Extra Tree Classifier (ETC2) [134] – a classifier that combines decision trees and gradient boosting to achieve improved classification performance. It iteratively creates and trains decision trees, each tailored to focus on the misclassified instances from the previous iteration. This process of sequential learning, which utilizes gradient boosting to adjust the weights of training instances based on their misclassifications, allows it to gradually refine its understanding of the data and improve its ability to distinguish between different classes;

- K-Neighbors Classifier (kNC) [137] – kNC classifies a data point based on the majority class of its k nearest neighbors in the training data. It operates by first calculating the distance between the new data point and each of the data points in the training set. Then, it identifies the k data points that are closest to the new data point, based on the chosen distance metric. Finally, it assigns the new data point to the majority class of the k nearest neighbors;

- Radius Neighor Classifier (RNC) [138] – an extension of the kNC that considers all points within a certain radius of a new data point instead of just the k closest points. This makes the radius neighbor classifier more robust to outliers and noise in the data, as it takes into account a wider range of points when making a prediction;

- Stochastic Gradient Descent Classifier (SGDC) [139] – the SGDC iteratively adjusts the weights of the linear model based on the predicted class and the actual class of the data point. This process of adjusting weights is guided by the gradient of the loss function, which indicates the direction of steepest descent towards the minimum;

- Logistic Regression Classifier (LRC) [140] – this classifier uses a linear model that uses a logistic function to convert the linear combination of input features into a probability between 0 and 1. These probability coefficients are estimated using a training dataset of labeled examples. The goal of the training process is to find a set of coefficients that minimize the error between the predicted probabilities and the actual labels.

5.2.2 Achieved Results

Measuring the accuracy of each combination is required to identify the best models/classifiers pairs. To perform this assessment, each model-classifier combo was trained with 3000

instances from the dataset and then tested with another 600 instances. This evaluation is presented in table 5.1. *Accuracy* was calculated utilizing the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad (5.1)$$

where TP are *True Positives*, TN *True Negatives*, FP *False Positives*, and FN *False Negatives*.

Since the MLkNN and the BRkNN models do not require a classifier, they were only trained and tested once. This test produced an accuracy of 90.4% for the MLkNN and 93.1% for the BRkNN.

Models	Classifiers (%)								
	AB	ETC	RFC	DTC	ETC2	kNC	RNC	SGDC	LRC
Classifier Chains	96.8	95.8	95.7	95.0	89.3	94.3	80.7	N/A	N/A
Binary Relevance	96.8	95.7	96.5	95.2	91.7	94.2	80.7	N/A	N/A
Label Powerset	96.8	96.8	96.8	95.0	91.0	94.0	80.7	N/A	N/A
Label Space Partitioning	96.8	95.7	96.7	95.3	90.8	94.2	80.7	N/A	N/A
Majority Voting	96.8	96.2	96.5	95.5	92.2	94.2	80.7	N/A	N/A

Table 5.1: *Accuracy* of each model-classifier combo for the experiments concerning the SRE tool automation.

From the results presented in table 5.1, it can be seen that the ETC2, kNC, and RNC classifiers showed an overall lower accuracy when compared to the remainders.

Another conclusion that arose from testing was the imbalance of the datasets, given that the proportion of 1s was higher than the proportion of 0s, as it is more common for a security requirement to be outputted than to not be outputted (security requirements are very typically necessary in most applications nowadays). This imbalance was such that it impossibilitated the running of the SGDC and LRC algorithms. This imbalance made it necessary to reevaluate the classifiers, as the high number of false positives eschewed the results. To correctly assess the classifier performances, mean accuracy was used.

To calculate the *Mean Accuracy* (mA), or balanced accuracy, the following formula was used [141]:

$$mA = \frac{1}{2N} \sum_{i=1}^{L} (TP_i/P_i + TN_i/N_i), \quad (5.2)$$

where L is the number of attributes, N is the number of examples, P_i and TP_i are the amount of positive examples and correctly predicted positive examples, respectively, and N_i and TN_i the amount of negative examples and correctly predicted negative examples, respectively.

The next step was thus the measurement of the *Mean Accuracy* of each combination for each label. Considering that the dataset is imbalanced, only suitable classifiers were taken into consideration (in this case, DTC, ETC2, SGDC, and LRC). An average of the results of

the *Mean Accuracy* for each label was calculated. The summarized results can be observed in table 5.2 (a).

As previously mentioned, classifiers are not needed for models such as MLkNN or BRkNN; their results were of 91.3% and 97.1%, respectively. Considering the results detailed in table 5.2 (a), it is possible to conclude that DTC achieved the best results for all the models, being the one chosen to be used in the remaining set of tests.

Models	Classifiers (%)			
	DTC	ETC2	SGD	LR
Classifier Chains	100	94.3	98.0	97.8
Binary Relevance	100	97.9	97.6	97.9
Label Powerset	100	92.0	57.0	90.5
Label Space Partitioning	100	97.0	97.6	97.9
Majority Voting	100	96.1	97.2	97.9

(a)

Models	Time(s)
Classifier Chains	16.445
Binary Relevance	14.863
Label Powerset	21.614
Label Space Partitioning	21.242
Majority Voting	29.295
Multi-label k-Nearest Neighbor	847.902
Binary Relevance k-Nearest Neighbor	227.357

(b)

Table 5.2: Average of the *Mean Accuracy* for each label (a) and temporal analysis of each model for the time consumed to train and test each model, when using the DTC (b).

With DTC chosen as the highest performing classifier, a new set of experiments was performed. The first test intended to evaluate time consumption of each model for each classifier. Not losing the scope and main purpose of the experiment, which is determining if such a model has the capacity of learning the decision-making logic through a small subset of examples, additional testing was performed to verify mean accuracy of each model given a smaller training set. This was done with samples between 250 and 1000 instances. Each test was performed a total of five times for each of the models. All tests were performed on a machine running Windows 11, using an AMD Ryzen 2700x CPU, and 32GB of DDR4 3200MHz memory (even though all the tests described in this section were performed using the same computer, the description of the specifications was deliberately left to this part, as this would only impact and be of interest in terms of time efficiency).

The main objective of the first test is to infer the most efficient model in terms of consumed time, as the objective is for the retraining to be expediently done after the addition of user inputs, without requiring an initial, large size dataset. This retraining will need to happen after each new user input, requiring the training to not be excessively tasking in terms of time spent. Table 5.2 (b) presents the consumed time results for each model, with the BR model the fastest to train and test, consuming only a total of 14.863 seconds, with the remainder models, with the exception of the kNN-based models, performing well in this respect.

The main purpose of the second test was evaluating the capacity of the model to infer the decision-making structure of the SRE tools, i.e., to be capable, from a small subset of outputs, to understand their logic and replicate it, so that, if a future change is required due to an advancement in the area, a small subset created by knowledgeable experts can be used to automatically update the tool. Considering the results of the first test, it was decided that

MLkNN and BRkNN would be excluded from the second test, due to the time consumed by these models and their lower mean accuracy results.

Table 5.3 presents the results achieved for this test. Training was done with a series of different training set sizes, from 250 instances up to 1000 instances, with an increase of 50 instances between each test. For all runs, for the different models, the same training set was used in each size, to ensure that the achieved results were directly comparable between the different models. Results above 800 instances are omitted from the table, as they all achieved the same 100% mean accuracy for 850, 900, 950 and 1000 instances.

Results show that with 400 instances the performance of all models, with the exception of the LP, which showed a significantly lower performance, is such that they show being capable of outputting recommendations for the larger test set with a very strong mean accuracy (of 99.7%). Overall, from the different classifiers, CC, BR, LSP and MV achieve good performance when utilizing smaller amounts of training instances, with CC and MV slightly ahead in terms of mean accuracy, and, also according to the results in table 5.2, with CC being the second fastest model overall, with an average of 98.94% mean accuracy for a training size of 250 instances. This strongly favors the hypothesis of the labels in the dataset (the security requirements) being correlated, as this is one of the key features of this type of model.

Overall, the performed testing showed that the ML models are capable of correctly learn the decision-making logic behind the original SRE tool, fulfilling that objective with a (relatively) small subset of instances, showing that the relearning process through user-provided feedback of the set of outputted requirements is possible, meaning that it can learn from its interactions with the users.

5.3 Natural Language Processing Applied to Security Modeling

NLP [142] is an area of AI that deals with human language. The focus of this area is twofold: to interpret human language; and be able to answer in kind. Due to the way human language works, its subtleties, imprecision, and underlying subjectiveness, it is a non-trivial task to understand and write or produce human language. NLP has many different applications nowadays, such as, e.g., spellchecking, information retrieval, aggregation, question answering, translation, summarization or speech recognition. For a NLP model to be able to fulfill these types of tasks, it typically needs to be able to lemmatize and stem words (i.e., be capable of reducing words to their base form, e.g., *"working"* to *"work"*, or *"done"* to *"do"),* filter words with no contribution to the real meaning of the text (e.g., connection particles such as *"a", "an"* or *"the"),* and tokenize words and word fragments, so the text can then be trans-

Models	Instance Size or Training Size												
	250	300	350	400	450	500	550	600	650	700	750	800	
Classifier Chains	98.4	99.3	99.1	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	Mean Accuracy (%)
Binary Relevance	98.4	99.0	99.1	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Label Space Partitioning	98.4	99.1	98.6	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Label Powerset	77.2	78.3	78.5	78.8	79.7	79.7	80.0	80.3	80.9	80.9	81.8	82	
Majority Voting	98.4	99.0	99.2	99.7	99.8	99.7	99.8	99.8	99.9	99.9	99.9	100	
Classifier Chains	98.9	99.3	99.1	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Binary Relevance	99.6	99.0	98.6	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Label Space Partitioning	98.4	99.6	98.4	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Label Powerset	78.0	78.1	78.8	78.1	79.2	80.3	79.8	80.6	80.8	81.3	81.9	82.1	
Majority Voting	99.0	99.1	98.4	99.7	99.8	99.7	99.8	99.8	99.8	99.9	99.9	100	
Classifier Chains	99.5	99.1	98.4	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Binary Relevance	99.6	99.1	98.6	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Label Space Partitioning	99.6	99.3	98.5	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Label Powerset	78.6	78.5	78.5	78.8	79.7	79.7	80.0	80.3	80.9	80.9	81.8	82	
Majority Voting	99.5	99.2	99.2	99.7	99.8	99.7	99.8	99.8	99.9	99.9	99.9	100	
Classifier Chains	98.3	99.1	99.1	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Binary Relevance	98.4	99.3	98.4	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Label Space Partitioning	98.4	99.1	99.1	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Label Powerset	77.9	78.3	78.0	78.1	80.0	79.7	79.9	80.3	80.7	80.9	81.9	81.6	
Majority Voting	98.4	99.0	99.1	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Classifier Chains	99.6	99.7	99.1	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Binary Relevance	98.4	98.4	98.5	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Label Space Partitioning	99.6	99.7	98.5	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	
Label Powerset	78.0	78.5	78.9	78.9	79.7	79.7	80.1	80.1	80.8	81	81.9	82.3	
Majority Voting	99.0	99.0	98.4	99.7	99.8	99.8	99.8	99.8	99.9	99.9	99.9	100	

Table 5.3: *Mean Accuracy* of the models when trained with datasets with different sizes. Each grouped row of models corresponds to one series of tests (total of 5 series performed).

formed in a way different techniques and models can use them as information for training. These models then have the challenge of understanding lexical and semantic characteristics, identifying language constructs, and be capable of representing abstract concepts, for example.

Considering the work presented in section 5.2, it is possible to assume the possibility of applying other strategies with the same concept: if it is possible to train a model to understand the logic and knowledge base of a given security tool, then the model can be used to evolve the tool, and make it self-updateable. The following subsection (subsection 5.3.1) will explore the use of LLMs to take human inputs, such as system descriptions, and generate and output correct and useful security-related contents.

The remaining part of this section will be briefly describing a related work performed in the scope of this Ph.D. project using NLP in collaboration. It is interesting to mention this work, as it can be seen as a precursor to the use of LLMs in this context. This work analyzed the use of NLP to predict Common Vulnerability Scoring System (CVSS) scores from the textual description of the vulnerabilities. These CVSS prediction tests utilized a combination of text pre-processing and vocabulary addition as an attempt to improve the model accuracy and its prediction reasoning, using Shapley values to gauge the importance of the different words. The tested models, transformers, encode the inputted text sequences into three vectors, query, key, and value, and through them, the transformer models utilize an attention

mechanism that computes a weighted sum based on the resemblance between the query and key vectors, which is then, together with the initial input, processed through a feed-forward neural network, to assess dependencies and capture the relevant information from the input. A total of 8 different scores were considered for calculation, with five different models being tested, and the best results were attained with the *DistilBERT* model, whose achieved accuracy is presented in table 5.4, for each of the scores. With a lowest achieved accuracy of 86.42%, and an average of 90.74%. This type of model shows a strong capacity for these tasks. These results show that the use of transformer models for the purpose of correctly identifying security properties is viable, which in turn leads to the next section, where the use of a more advanced type of NLP techniques, LLMs, are tested upon.

Category	Accuracy(%)
Attack Vector	91.41
Attack Complexity	95.20
Privileges Required	86.42
User Interaction	93.33
Scope	96.40
Confidentiality	86.71
Integrity	87.61
Availability	88.81

Table 5.4: Category accuracy for the *DistilBERT*-Enhanced model.

5.3.1 Using LLMs and Fine-Tuning With ATIoT Logic

LLMs [143], [144] are a specific subset of the NLP field of AI and concern the generation of general-purpose text. These models generate text by evaluating the statistical distribution of *tokens (tokens* can be,e .g., words, parts of words, or individual characters), and are designed to comprehend and generate human-like text, constructed through the utilization of deep neural networks with millions or even billions of parameters. LLMs are pre-trained on vast corpora of diverse textual data, enabling them to capture intricate patterns, contextual nuances, and syntactic structures inherent in natural language. The pre-training process equips these models with the capability of predicting, given the statistics of the analyzed human language, what are the words that are most likely to continue that specific inputted text. This gives them the ability to perform a wide array of natural language processing tasks, including text generation, translation, summarizing, and question answering. The architecture of LLMs, particularly the transformer architecture, is central to their capabilities. The attention mechanism allows LLMs to focus on relevant parts of input sequences, fostering long-range dependencies, making the model context-aware and capable of retracing previously given inputs and generated outputs, and capturing linguistic relationships. This architecture also addresses challenges associated with contextual understanding and semantic coherence.

LLMs are trained, as previously mentioned, with a corpora of textual data, which is generic in terms of its actual context, to ensure the best possible performance of the models, which

makes them capable of generating text in multiple contexts or knowledge areas. However, these models are, as such, of general purpose and use. To improve the capabilities of a model for a given domain, or improve their capabilities for a specific task, it is possible to take the previously trained model, and fine-tune its outputs, by training it further on a smaller, specific dataset catered towards the intended task or knowledge area. While the initial training is mostly unsupervised, fine-tuning is, in most situations, a case of supervised learning, i.e., each input data is given to the model together with an example of the expected output.

5.3.1.1 Analysis of the Approach

The main purpose of the experiment presented herein is to assess the ability of LLMs of recognizing the logic that entails the ATIoT tool, and verify if they are capable of outputting information that is equivalent in value to the direct outputs of the ATIoT tool starting from natural language inputs. To that end, and considering that training a LLM from its inception requires massive computing capabilities, and vast datasets (e.g., ChatGPT Generative Pre-Trained Transformer (GPT)-4, according to leaked information, used over 570GB of data, and took over 90 days to train on over 100 thousand Graphical Processing Units (GPUs) [145]), fine-tuning, i.e., the process of specializing a LLM through supervised learning, was the utilized method. This process adapts an already trained model to a specific domain by training it further on task-specific data. This allows the model to learn specific patterns and improve its performance on the target task, without requiring extensive computational resources or large amounts of annotated data.

While being more time and resource-efficient than fully retraining a model, the process is still very heavy in terms of memory requirements: fine-tuning a model such as Large Language Model Meta AI (Llama) 65B can require up to 780GB of GPU memory, which is out of reach for most systems, which do not have hundreds of GPUs to pool their memory to achieve such amounts. To reduce this required amount, a process called quantization is applied, which can be described as a discretization method that changes the representation of a given input to one that has less information. Computationally, this equates on converting data types to ones using less bits. To exemplify, consider a 32 bit float, $x \in [a, b]$, which will be quantized to 8 bit integer. The process is represented in the following quantization scheme [146]:

$$x = S \times (x_q - Z), \quad (5.3)$$

where x_q is the quantized integer value of x, S is the scale, represented as a 32 bit float, and Z is the zero-point, an 8 bit integer that represents 0 in a 32 bit float.

The quantized value x_q of x can then be computed as follows:

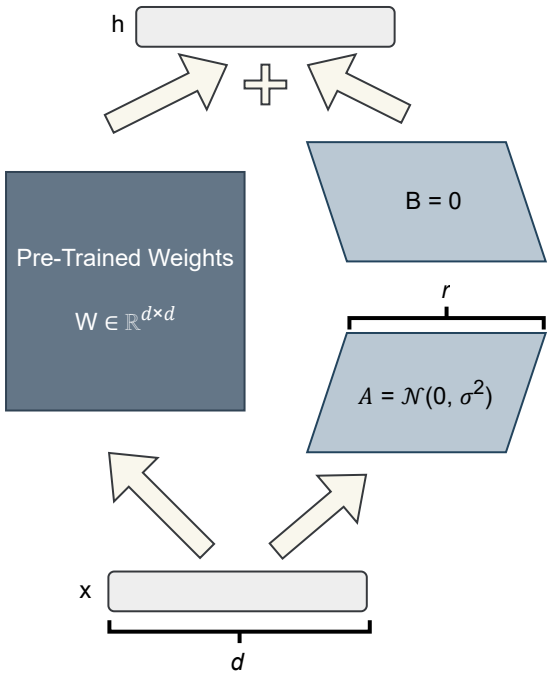
$$x_q = [x/S + Z]. \quad (5.4)$$

32 bit float values that fall outside the interval of x, e.g., $y \notin [a, b]$, are clipped to the closest value that can be represented:

$$y_q = \begin{cases} [a/S + Z], & \text{if } y < a \\ [y/S + Z], & \text{if } a \leq y \leq b \\ [b/S + Z], & \text{if } y > b. \end{cases} \quad (5.5)$$

For the specific case of this experiment, Quantized Low Rank Adaptation (QLoRA) [147] was used as the fine-tuning approach, to further reduce the necessary amount of memory during the training process. QLoRA quantizes a model to 4 bits precision, freezing its parameters, which reduces the memory footprint of the model. QLoRA then adds a set of learnable Low Rank Adaptation (LoRA) [148] weights to the quantized model. These weights are what is then updated during the fine-tuning. Figure 5.2 exemplifies the QLoRA scheme. LoRA works by freezing the trained model weights and injects trainable rank decomposition matrices into each layer of the transformer architecture, reducing the amount of trainable parameters. Figure 5.1 exemplifies LoRA, with W representing the weight matrix of $d \times d$ dimension, decomposed into two low-rank matrices, A and B. The dimension of A is $r \times d$, and that of B is $d \times r$. When x is inputted into the LoRA fine-tuned model, it is multiplied with W, and with the resulting matrix of $A \times B$. The two outputs are then summed, generating h.

To be able to fine-tune models, the first requirement is the creation of a dataset for that purpose. In the case of LLMs, datasets are constituted by human-inputted text and by the corresponding expected outputted text samples. For the ATIoT tool, to generate the human text, a set of statements that mimic answering to the questions in table 4.1, with a set of variations for each of the sentences, was generated. To define the set of texts to be generated, for each human-like input, the sentence variation was chosen, randomly, utilizing a uniform distribution. As the questionnaire has a total of 24 questions, and a total of 69 possible answers between them, the full combination of possible answers numbers at 10 thousand million. The created subset was of around 0.00001% of the total, which, while appearing to be a very reduced subset, is still a subset of over 1200 different combinations. The selection of samples for the subset was done using the Monte Carlo method, based on a uniform distribution, to randomly sample the full set. The set of attacks was also added to each of the generated human sentences, as an answer example of what the model should output. The following is an excerpt of an entry of the dataset used in the fine-tuning (notice that the texts have a certain colloquial tone, so as to potentially better mimic natural language interactions):



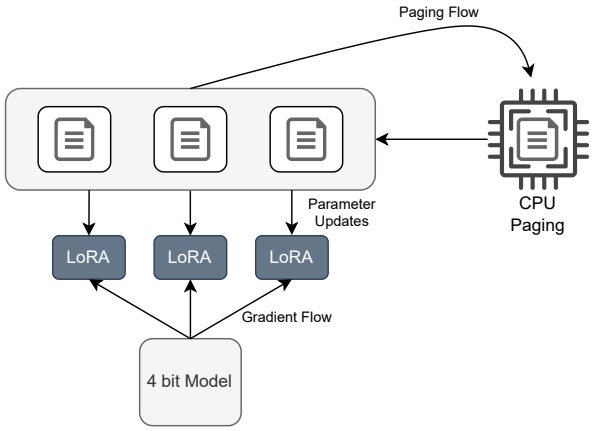


Figure 5.2: QLoRA scheme representation.

Figure 5.1: LoRA scheme representation.

- "Human: `I'm designing an IoT system, and I need to know what attacks it can suffer and also obtain the attack trees for each of those attacks. Here is a description of the system: This system will be used in the context of healthcare. It will support users. Login functionality will be implemented. It will...";`
- "Example Output: `The following attacks and respective attack trees should be considered: Physical Tampering, Side-Channel Attacks, Reverse Engineering, Sleep Deprivation, Jamming & Interference, Data Tampering, Data Theft...";`

The LLM model chosen for fine-tuning was Llama 2-chat, part of the Llama 2 [149] collection, released by Meta in July 2023. Llama 2-chat is a model already fine-tuned for human interaction, further fine-tuned for the specific purpose of embedding it with ATIoT logic in this work. This model comes in four sizes, differing in the number of parameters used (7, 13, 33 and 70 thousand million) . This experiment made use of the Llama 2-Chat-7B model, with 7 thousand million parameters, due to constraints in required GPU memory capacity.

Fine-tuning was then performed for the model. For this purpose, the previously generated dataset was split, once again by randomly (with an uniform distribution) choosing samples using the Monte Carlo method, with an 80/20 split. The 80% part was used for the fine-tuning of the models, while the remainder 20% were used to assess the correctness of the achieved results. Table 5.7 shows the used parameters for the fine-tuning process. Given the memory related requirements, the experiments were performed using Google Colaboratory,

on an A100 instance, equipped with 83.5GB of RAM, and a Nvidia A100 GPU, with 40GB of video memory. Table 5.5 and figure 5.6 present the loss function [150] results during the fine-tuning steps, used to infer the difference between predicted and actual values, showing the desired regular reduction in cross entropy during the advancement of the fine-tuning process. The used loss function was the cross entropy function, which is calculated as follows:

$$H(p,q) = -\sum_{x \in classes} p(x) log(q(x)), \quad (5.6)$$

where p represents the target distribution and q represents the approximation to the target distribution.

Total fine-tuning time was of 2616 seconds, i.e., approximately 44 minutes, with a maximum GPU memory usage of 34.6 GB.

Step	Training Loss
250	0.2335
500	0.2101
750	0.1977
1000	0.1900
1250	0.1877
1500	0.1849
1750	0.1804
2000	0.1789
2250	0.1759
2425	0.1784

Table 5.5: Train-Loss values for Llama 2 fine-tuning process, using the ATIoT related dataset.

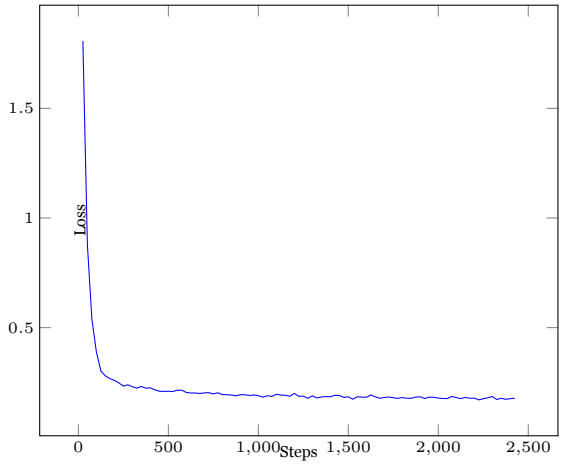


Table 5.6: Train-Loss graph for Llama 2 fine-tuning process, using the ATIoT related dataset.

Parameter Name	Parameter
Training Epochs	2
Gradient Accumulation Steps	1
Learning Rate	$2e^{-4}$
Saving Steps	25
Threshold Value for Gradient Norm	0.3
Weight Decay	0.001
Warm-up Ratio	0.03
Optimizer Algorithm	Adam 32bit
Learning Rate Scheduler Type	Constant

Table 5.7: Selected final fine-tuning parameters for training on Llama 2.

To assess the results of fine-tuning, a set of human texts, structured similarly to those used to fine-tune the model, were prompted to the chatbot executing that model after the training. For each response, the outputted attacks were compared to the expected output. This was performed automatically using a Python script. Similarly to the experiment presented in section 5.2, a comparison was performed between the expected suggested attacks and the ones predicted by the model. Table 5.8 presents the achieved results for the experiment,

showing an overall balanced accuracy of 84%, and a F_1 score of 80%. F_1 score is the harmonic mean of precision and recall, and is calculated as follows:

$$F_1 = \frac{2TP}{2TP + FP + FN}, \quad (5.7)$$

where TP is the number of true positives, FP the number of false positives, and TN the number of false negatives.

These results are in line with the final loss value during training (17.84%), but with a particular noteworthy observation: the model, while strong in terms of identifying attacks, is including many more attacks than what it should, which is emphasized by the 10 true negatives. This means the model is very rarely removing attacks when they should not be present. This issue has some similarities with what was found during the application of classifiers to the SRE tool, which root cause was related with the imbalance in terms of positive and negative cases in the dataset, although the imbalance is not as pronounced in the ATIoT dataset. This may point towards LLMs preference to (abundantly) generating text, at the cost of not omitting information when they should. While this is an issue on the overall accuracy of the generated outputs, from a security perspective, it is preferrable that the model selects attacks that are irrelevant, than omitting relevant ones. The higher number of false positives, when compared to the total of false negatives, emphasizes this characteristic of the trained model. The overall results show that the LLM is successful in apprehending the ATIoT logic to a certain extent, but future training and different experiments with modified datasets will be required to attempt to achieve higher balanced accuracy results.

Metric	Achieved Value
True Positives	1449
True Negatives	10
False Positives	461
False Negatives	272
F1 Score	0.80
Balanced Accuracy	0.84

Table 5.8: Fine-tuning results for ATIoT logic with Llama 2. The comparison was done on what attacks were identified by the fine-tuned model, in comparison to what was expected from the logic of ATIoT.

5.4 Conclusions

This chapter presented the work performed to leverage the use of AI to improve security tools, with the purpose of ensuring their maintenance, longevity and their overall improvement. Taking on the work presented in chapter 4, this chapter presented the work towards the automation and update capabilities of security-related tools, as shown by the experiences and results attained and presented in sections 5.2 and 5.3. The achieved results from the exper-

iments show promise in the use of applying different AI techniques to security engineering. With these finalized results, and taking all which was presented previously, the following chapter, chapter 6, presents the final conclusions of this thesis and explores future avenues that can be explored on this subject.

(This page is intentionally left blank.)

Chapter 6

Main Conclusions and Future Work

This chapter presents the summarized main conclusions of the research work presented in this thesis and developed during this Ph.D. project, and discusses potential research directions identified or directly deriving from the work presented herein, that could and should be pursued on further advancements elaborated upon the presented, developed work.

6.1 Main Conclusions

The first main conclusion is that there is still much work to be done in security engineering. With the development of software and systems being naturally guided by functional requirements, cybersecurity and software development evolved at very different paces and, in spite of an increased awareness, they are still unbalanced by the time this project was ending. This new era of automated programming and system design may actually stretch that gap once again.

During the course of the development of this project, the digital realm witnessed a profound transformation, particularly in the domains of cybersecurity and AI. The introduction of significant regulatory measures such as the General Data Protection Regulation (GDPR), for example, instigated a paradigm shift in data privacy and security practices, leading first to a new era of tentatives to adherence to compliance and accountability standards. The COVID-19 pandemics then (forcibly) motivated the very fast adoption of remote working, mostly supported by computer systems, applications and networks. More recently, we observed the emergence of what can be characterized as the *new hot summer of AI*, mostly fostered by the emergence of advanced generative models in 2021. These AI breakthroughs, which include deep learning architectures such as GPT-3 and GPT-4, have revolutionized diverse domains, including natural language processing, computer vision, and various others, illustrating an unparalleled convergence of theoretical foundations with pragmatic implementation, which underpins the contextual evolution of this project.

This thesis approached the issue of aiding in the secure development of IoT systems, and presented a concise, complete, yet approachable set of contributions towards the main proposed objective at the start of the program: that these contributions would aid in the design of IoT systems, and that they would be approachable by designers and developers with low to no se-

curity expertise. These contributions included an architectural model for IoT, that identifies, in detail, for each common component of an IoT system, its main security concerns, requirements that may be applied, and steps that should be followed in the design and development process to attempt to embed security from inception. The proposed attack taxonomy identified a series of attacks that IoT systems may be vulnerable to, mapped to the components of the proposed model, describing each attack in detail and what they affect, and their common means of achieving success. From the attacks, the ATIoT tool was developed and herein described as being able of mapping a series of questions to the most probable attacks a system may be susceptible to. The tool, beyond identifying the attacks, outputs the set of attack trees that can then be used by developers to identify the potential vulnerable entry points of their system. The mapping of the outputs of the SRE, SBPG, LWCAR and TMS tools is another contribution of this work, aiding in the process of identifying where each recommendation applies to. Finally, and in a sense coming back to the beginning of these conclusions, the last contributions focused on the application of different AI and machine learning techniques to the previously developed work. The presented results show promise in leveraging these technologies to bring two main benefits to the original proposed objective: (i) *the ease of interaction and approachability* through the use of language models, which may allow users to simply input an overall description of their system in a more natural manner and then obtain a trustworthy set the results; and (ii) *the capability of learning and updating*, which is one of the main limitations of any work that proposes to map security suggestions, recommendations or mechanisms, since they are necessarily tied to a given moment in time, thus in risk of losing some practical usefulness if left unattended.

To finalize, this thesis presents a series of contributions towards the original premise of aiding the development of secure IoT systems, corroborating the initial statement that it is possible to develop secure IoT systems, if developers are given the appropriate tools and mechanisms, designed to be simple to use and that do not require a strong security background and expertise. This was further expanded upon through the latter contributions, that focused on validating the possibility of utilizing AI to maintain and, potentially, enhance the tools, a challenge always present in fast-paced, and quickly advancing scenarios, as IoT and security are.

6.2 Future Work

Computer science evolves at a significant fast pace and many potential research lines unfolded along this project. This section discusses some of the main potential research directions that could closely follow the work described herein.

Eventually, the most general (higher level) future research direction concerns the training of

AI systems with focus on security engineering, since it has become evident that AI has the potential to address numerous security gaps across various phases of the engineering cycle. While previous efforts were primarily concentrated on training and integrating algorithms to execute specific tasks, such as intrusion detection, the current focus should expand to higher-level objectives involving system design, requirements elicitation, security integration and testing. While the training of large language models for many specific areas (such as written human language or computer programming) has benefited from very large corpora readily available in the Internet, the same does not apply to security engineering, mostly because this area was not mature or consolidated enough, thus hindering the availability of datasets. It is important to conduct research aimed at addressing this gap in the short term. This specifically concerns finding out how human differently express security requirements or cybersecurity problems and how do these translate into technical specifications, mechanisms and protocols.

Even though many advances were made in recent years towards a better definition of many concepts and mechanisms involving cybersecurity, there is still much work to be done before security is an intrinsic part of software and (computer) system engineering. A future research direction is to continue contributing towards that objective, and perhaps even challenge some of the current models or approaches to cybersecurity, as it might be beneficial to rethink some of them in the scope of a so much larger and rich digital ecosystem.

Still within the scope of using AI to assist (or perhaps fully perform) the security engineering process, it will be interesting to study the potential impact of the existence of hallucinations not only on the cybersecurity of a software or system, but also in their set of functionalities. Given the general state of cybersecurity at the time of writing this thesis, having a system or software implementing some security mechanisms, even the ones resulting from hallucinations, might be better than not having them at all. As such, it will be interesting to specifically understand if the security of a software or system will get better or worst in such cases.

When assuming that the tools are being used solely by people or systems performing security engineering, the topic of *AI hallucinations* warrants exploration. The possibility of a model outputting recommendations or guidelines that are nonsensical or inaccurate is an issue that needs to be tackled, lest the reliability of the models be compromised. Hallucinations reduce the trust users may have on such tools, and may have direct influence on decisions made during the security engineering process, introducing potential flaws that may not be detected during the full development process.

In the medium to long term, being able to ascertain how a given AI model reaches its outputs, through *AI explainability*, attaining understanding of how it performed its decision process, may be an invaluable advance, as it may aid in finding hidden correlations and patterns that experts may not have yet been able to reach. Simultaneously, this may aid in the

previous topic of identifying cases of AI hallucination. The topic of *adversarial attacks* is another avenue open to exploration in the future, as it is a threat to differing AI models in different areas, including, e .g., their previously mentioned use in security engineering and self-updating mechanisms for security-based tools.

Bibliography

[1] K. Ashton, "That 'internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009. xiii, 1

[2] S. Misra, A. Mukherjee, and A. Roy, *IoT Sensing and Actuation*. Cambridge University Press, 2021, p. 97–114. xiii, 1

[3] S. Sinha, "State of IoT 2023: Number of connected IoT devices growing 1616.7 billion globally," https://iot-analytics.com/number-connected-iot-devices/, accessed: 2024-01-18. xiii, 1

[4] dig8ital, "Poor Application Security Can Increase Costs by 3,000%," https://dig8ital.com/post/poor-application-security-can-increase-costs-by-3000/, accessed: 2024-03-02. xiv, 2

[5] A. Padyab, A. Habibipour, A. Rizk, and A. Ståhlbröst, "Adoption barriers of iot in large scale pilots," *Information*, vol. 11, no. 1, 2020. xiv, 2

[6] P. Sethi and S. R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," *Journal of Electrical and Computer Engineering*, vol. 2017, 2017. xv, 3

[7] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the Internet of Things: perspectives and challenges," *Wireless Networks*, vol. 20, pp. 2481–2501, 2014. xv, 3

[8] K. Angrishi, "Turning Internet of Things (IoT) into Internet of Vulnerabilities (IoV): IoT Botnets," *arXiv preprint arXiv:1702.03681*, 2017. xv, 3

[9] J. Deogirikar and A. Vidhate, "Security Attacks in IoT: a Survey," in *Proceedings of the 2017 International Conference on IoT in Social, Mobile, Analytics and Cloud(I-SMAC)*. IEEE, 2017, pp. 32–37. xv, 3, 56

[10] J. B. Sequeiros, F. T. Chimuco, M. G. Samaila, M. M. Freire, and P. R. M. Inácio, "Attack and System Modeling Applied to IoT, Cloud, and Mobile Ecosystems: Embedding Security by Design," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–32, 2020. xvii, xx, xxii, 6, 11, 28, 67

[11] J. B. Sequeiros, F. T. Chimuco, T. M. C. Simões, M. M. Freire, and P. R. M. Inácio, "An Approach to Attack Modeling for the IoT: Creating Attack Trees from System Descriptions," in *Proceedings of the 38th International Conference on Advanced Information Networking and Applications (AINA)*, Kitakyushu, Japan, April 2024. xviii, xix, xxi, xxii, 6, 7, 67

[12] J. B. F. Sequeiros, F. T. Chimuco, T. M. C. Simões, M. M. Freire, and P. R. M. Inácio, "Secure System Model for IoT, and Application Cases of AI in Security Engineering," *submitted for publication in Springer IJIS*, 2024. xviii, xx, xxi, xxiii, 6, 7, 8, 87

[13] F. T. Chimuco, J. B. F. Sequeiros, C. G. Lopes, T. M. C. Simões, M. M. Freire, and P. R. M. Inácio, "Secure cloud-based mobile apps: attack taxonomy, requirements, mechanisms, tests and automation," *International Journal of Information Security*, pp. 1–35, 2023. xviii, 7

[14] M. G. Samaila, J. B. F. Sequeiros, T. Simões, M. M. Freire, and P. R. M. Inácio, "IoT-HarPSecA: A Framework and Roadmap for Secure Design and Development of Devices and Applications in the IoT Space," *IEEE Access*, vol. 8, pp. 16 462–16 494, 2020. xix, 7, 46

[15] M. G. Samaila, C. Lopes, É. Aires, J. B. F. Sequeiros, T. Simoes, M. M. Freire, and P. R. M. Inácio, "Performance evaluation of the sre and sbpg components of the iot hardware platform security advisor framework," *Computer Networks*, vol. 199, p. 108496, 2021. xix, 7

[16] C. Lopes, J. C. Costa, J. B. F. Sequeiros, T. M. C. Simões, M. M. Freire, and P. R. M. Inácio, "Machine Learning Applied to Security Requirements Elicitation: Learning From Experience," in *Atas do 12º Simpósio de Informática (INForum 2021)*, Lisboa, Portugal, September 2021. xix, xxiii, 8, 87

[17] J. C. Costa, T. Roxo, J. B. F. Sequeiros, H. Proenca, and P. R. M. Inácio, "Predicting CVSS Metric via Description Interpretation," *IEEE Access*, vol. 10, pp. 59 125–59 134, 2022. xx, 8

[18] SECURIoTESIGN Team, "SECURIoTESIGN Project Page," https://lx.it.pt/SECURIoTESIGN, accessed: 2023-12-05. xxii, 67, 68

[19] S. A. Al-Qaseemi, H. A. Almulhim, M. F. Almulhim, and S. R. Chaudhry, "IoT architecture challenges and issues: Lack of standardization," in *Proceedings of the 2016 Future Technologies Conference (FTC)*. IEEE, 2016, pp. 731–738. xxix, 11, 13

[20] Cisco, "Cisco IoT Reference Model," http://cdn.iotwf.com/resources/72/IoT_Reference_Model_04_June_2014.pdf, Cisco, 2014, accessed: 2018-06-24. xxix, 12, 14

[21] H. Olivier, B. David, and O. Elloumi, *The ETSI M2M Architecture*. Wiley-Blackwell, 2011, ch. 14, pp. 237–267. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119958352.ch14 xxix, 12, 15

[22] M. P. Correia and P. J. Sousa, *Segurança no Software*, 2nd ed. FCA - Editora da Informática, Lda., 2017. xxix, 21, 22, 24

[23] H. Al-Mohannadi, Q. Mirza, A. Namanya, I. Awan, A. Cullen, and J. Disso, "Cyber-Attack Modeling Analysis Techniques: An Overview," in *Proceedings of the 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. IEEE, 2016, pp. 69–76. xxix, 5, 23, 24, 25

[24] A. L. Opdahl and G. Sindre, "Experimental comparison of attack trees and misuse cases for security threat identification," *Information and Software Technology*, vol. 51, no. 5, pp. 916–932, 2009. xxix, 25

[25] Z. Li and T. Xin, "Threat modeling and countermeasures study for the Internet of Things," *Journal of Convergence Information Technology*, vol. 8, no. 5, 2013. xxix, 25, 26, 30

[26] V. Drake, "Application Threat Modeling," February 2017, accessed: 2018-05-17. [Online]. Available: https://www.owasp.org/index.php/Application_Threat_Modeling xxxi, 21, 22, 26

[27] M. J. Covington and R. Carskadden, "Threat Implications of the Internet of Things," in *Proceedings of the 5th International Conference on Cyber Conflict (CYCON 2013)*. IEEE, 2013, pp. 1–12. 3

[28] B. Schneier, "Attack trees," *Dr. Dobb's journal*, vol. 24, no. 12, pp. 21–29, 1999. 5, 24, 70

[29] SECURIoTESIGN Team, "SECURIoTESIGN Project Github page," https://github.com/SECURIoTESIGN, accessed: 2024-01-08. 7, 68

[30] F. J. B. Nunes, A. D. Belchior, and A. B. Albuquerque, "Security Engineering Approach to Support Software Security," in *Proceedings of the 2010 6th World Congress on Services*. IEEE, July 2010, pp. 48–55. 13

[31] P. T. Devanbu and S. Stubblebine, "Software Engineering for Security: A Roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, ser. ICSE '00. New York, NY, USA: ACM, May 2000, pp. 227–239. 13

[32] D. Mellado, C. Blanco, L. E. Sánchez, and E. Fernández-Medina, "A systematic review of security requirements engineering," *Computer Standards & Interfaces*, vol. 32, no. 4, pp. 153 – 165, 2010. 13

[33] B. Morin, N. Harrand, and F. Fleurey, "Model-Based Software Engineering to Tame the IoT Jungle," *IEEE Software*, vol. 34, no. 1, pp. 30–36, 2017. 15, 30

[34] G. Fortino, R. Gravina, W. Russo, and C. Savaglio, "Modeling and Simulating Internet-of-Things Systems: a Hybrid Agent-Oriented Approach," *Computing in Science & Engineering*, vol. 19, no. 5, pp. 68–76, 2017. 15, 30

[35] C.-H. Chen, M.-Y. Lin, and X.-C. Guo, "High-level modeling and synthesis of smart sensor networks for Industrial Internet of Things," *Computers & Electrical Engineering*, vol. 61, pp. 48–66, 2017. 16, 30

[36] S. Breiner, E. Subrahmanian, and R. D. Sriram, "Modeling the Internet of Things: A Foundational Approach," in *Proceedings of the Seventh International Workshop on the Web of Things*. ACM, 2016, pp. 38–41. 17, 30

[37] X. T. Nguyen, H. T. Tran, H. Baraki, and K. Geihs, "FRASAD: A framework for model-driven IoT Application Development," in *Proceedings of the IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 387–392. 17, 30

[38] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for Internet of Things," *Journal of Network and Computer Applications*, vol. 42, pp. 120–134, 2014. 17, 30

[39] S. Babar, P. Mahalle, A. Stango, N. Prasad, and R. Prasad, "Proposed security model and threat taxonomy for the Internet of Things (IoT)," in *Proceedings of the International Conference on Network Security and Applications*. Springer, 2010, pp. 420–429. 17, 30

[40] J. Bau and J. C. Mitchell, "Security Modeling and Analysis," *IEEE Security and Privacy*, vol. 9, no. 3, p. 18, 2011. 18, 30

[41] Massachussets Institute of Technology, "The Alloy Analyzer," https://web.archive.org/web/20070607115817/http://alloy.mit.edu/index.php, accessed: 2019-09-14. 18

[42] M. T. Lazarescu, "Design of a WSN platform for long-term environmental monitoring for IoT applications," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, no. 1, pp. 45–54, 2013. 18, 30

[43] M. Vučinić, B. Tourancheau, F. Rousseau, A. Duda, L. Damon, and R. Guizzetti, "OSCAR: Object security architecture for the Internet of Things," *Ad Hoc Networks*, vol. 32, pp. 3–16, 2015. 18, 30

[44] F. Ye and Y. Qian, "A Security Architecture for Networked Internet of Things Devices," in *Proceedings of the 2017 IEEE Global Communications Conference (GLOBECOM 2017)*, Dec 2017, pp. 1–6. 19, 30

[45] C. M. Sosa-Reyna, E. Tello-Leal, and D. Lara-Alabazares, "Methodology for the model-driven development of service oriented IoT applications," *Journal of Systems Architecture*, vol. 90, pp. 15–22, 2018. 19, 30

[46] H. Ning, H. Liu, and L. T. Yang, "Cyberentity Security in the Internet of Things," *Computer*, vol. 46, no. 4, pp. 46–53, 2013. 19, 30

[47] D. A. Robles-Ramirez, P. J. Escamilla-Ambrosio, and T. Tryfonas, "IoTsec: UML Extension for Internet of Things Systems Security Modelling," in *Proceedings of the 2017 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE)*. IEEE, 2017, pp. 151–156. 20, 30

[48] M. Ge and D. S. Kim, "A Framework for Modeling and Assessing Security of the Internet of Things," in *Proceedings of the 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2015, pp. 776–781. 20, 30

[49] R. Kirichek, A. Vladyko, M. Zakharov, and A. Koucheryavy, "Model Networks for Internet of Things and SDN," in *Proceedings of the 2016 18th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2016, pp. 76–79. 20, 30

[50] M. Dowd, J. McDonald, and J. Schuh, *The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities*. Pearson Education, 2006. 22, 23

[51] M. Kazim and D. Evans, "Threat Modeling for Services in Cloud," in *Proceedings of the 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, March 2016, pp. 66–72. 23

[52] J. A. Ingalsbe, D. Shoemaker, and N. R. Mead, "Threat Modeling the Cloud Computing, Mobile Device Toting, Consumerized Enterprise-an overview of considerations," in *Proceedings of the 2011 Americas Conference on Information Systems (AMCIS)*, 2011. 23

[53] Microsoft, "Microsoft Threat Modeling Tool 2016," https://www.microsoft.com/en-us/download/details.aspx?id=49168, 2019, accessed: 2019-02-04. 23

[54] Continuum Security, "IriusRisk - Threat Modeling Tool," https://continuumsecurity.net/threat-modeling-tool/, 2019, accessed: 2019-01-22. 23

[55] G. Helmer, J. Wong, M. Slagell, V. Honavar, L. Miller, and R. Lutz, "A Software Fault Tree Approach to Requirements Analysis of an Intrusion Detection System," *Requirements Engineering*, vol. 7, no. 4, pp. 207–220, Dec 2002. 24

[56] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements engineering*, vol. 10, no. 1, pp. 34–44, 2005. 24

[57] J. Steffan and M. Schumacher, "Collaborative attack modeling," in *Proceedings of the 2002 ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2002, pp. 253–259. 24

[58] F. Chowdhury, "Modelling Cyber Attacks," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 9, 2017. 24

[59] J. P. McDermott, "Attack Net Penetration Testing," in *Proceedings of the 2000 Workshop on New Security Paradigms (WNSP)*. New York, NY, USA: ACM, 2000, pp. 15–21. 25

[60] E. Fernandez, J. Pelaez, and M. Larrondo-Petrie, "Attack Patterns: A New Forensic and Design Tool," in *Proceedings of the 2007 International Conference on Digital Forensics (IFIP)*. Springer, 2007, pp. 345–357. 25

[61] L. Piètre-Cambacédès and M. Bouissou, "Beyond Attack Trees: Dynamic Security Modeling with Boolean Logic Driven Markov Processes (BDMP)," in *Proceedings of the 2010 European Dependable Computing Conference*. IEEE, April 2010, pp. 199–208. 25

[62] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, "Internet of Things: Security vulnerabilities and challenges," in *Proceedings of the 2015 IEEE Symposium on Computers and Communication (ISCC)*. IEEE, 2015, pp. 180–187. 26, 30, 56, 58, 60, 62

[63] I. Agadakos *et al.*, "Jumping the Air Gap: Modeling Cyber-Physical Attack Paths in the Internet-of-Things," in *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and Privacy*. ACM, 2017, pp. 37–48. 27, 30

[64] Q. M. Ashraf and M. H. Habaebi, "Autonomic schemes for threat mitigation in Internet of Things," *Journal of Network and Computer Applications*, vol. 49, pp. 112–127, 2015. 27, 30

[65] T. Stepanova and D. Zegzhda, "Applying Large-scale Adaptive Graphs to Modeling Internet of Things Security," in *Proceedings of the 7th International Conference on Security of Information and Networks*. ACM, 2014, p. 479. 27, 30

[66] P. Sarigiannidis, E. Karapistoli, and A. A. Economides, "Modeling the Internet of Things Under Attack: A G-network Approach," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1964–1977, 2017. 27, 30

[67] I. Kotenko and A. Chechulin, "A cyber attack modeling and impact assessment framework," in *Proceedings of the 2013 5th International Conference on Cyber Conflict (CyCon)*. IEEE, 2013, pp. 1–24. 27, 30

[68] M. Mohsin, Z. Anwar, G. Husari, E. Al-Shaer, and M. A. Rahman, "IoTSAT: A formal framework for security analysis of the internet of things (IoT)," in *Proceedings of the 2016 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2016, pp. 180–188. 28, 30

[69] A.-M. Vilamovska, E. Hatziandreu, H. R. Schindler, C. van Oranje-Nassau, H. de Vries, and J. Krapels, *Study on the requirements and options for RFID application in healthcare: Identifying areas for Radio Frequency Identification deployment in health care delivery: A review of relevant literature*. RAND Corporation, 2009. 32

[70] D. Niyato, E. Hossain, and S. Camorlinga, "Remote patient monitoring service using heterogeneous wireless access networks: architecture and optimization," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 4, pp. 412–423, 2009. 32

[71] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, and K. Mankodiya, "Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare," *Future Generation Computer Systems*, vol. 78, pp. 659–676, 2018. 32

[72] N. Niknejad, W. B. Ismail, A. Mardani, H. Liao, and I. Ghani, "A comprehensive overview of smart wearables: The state of the art literature, recent advances, and future challenges," *Engineering Applications of Artificial Intelligence*, vol. 90, p. 103529, 2020. 33

[73] T. Wark, P. Corke, P. Sikka, L. Klingbeil, Y. Guo, C. Crossman, P. Valencia, D. Swain, and G. Bishop-Hurley, "Transforming Agriculture through Pervasive Wireless Sensor Networks," *IEEE Pervasive Computing*, vol. 6, no. 2, pp. 50–57, 2007. 34

[74] S. S. Reka and T. Dragicevic, "Future effectual role of energy delivery: A comprehensive review of Internet of Things and smart grid," *Renewable and Sustainable Energy Reviews*, vol. 91, pp. 90–108, 2018. 35

[75] Q. Ou, Y. Zhen, X. Li, Y. Zhang, and L. Zeng, "Application of Internet of Things in Smart Grid Power Transmission," in *Proceedings of the Third FTRA International Conference on Mobile, Ubiquitous, and Intelligent Computing*. IEEE, 2012, pp. 96–100. 35

[76] Y. Ding, M. Jin, S. Li, and D. Feng, "Smart logistics based on the internet of things technology: an overview," *International Journal of Logistics Research and Applications*, vol. 24, no. 4, pp. 323–345, 2021. 36

[77] Y. Song, F. R. Yu, L. Zhou, X. Yang, and Z. He, "Applications of the Internet of Things (IoT) in Smart Logistics: A Comprehensive Survey," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4250–4274, 2020. 36

[78] A. Verma, S. Prakash, V. Srivastava, A. Kumar, and S. C. Mukhopadhyay, "Sensing, Controlling, and IoT Infrastructure in Smart Building: A Review," *IEEE Sensors Journal*, vol. 19, no. 20, pp. 9036–9046, 2019. 36

[79] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017. 36

[80] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (IIoT): An analysis framework," *Computers in industry*, vol. 101, pp. 1–12, 2018. 37

[81] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014. 37

[82] H. Schaffers, N. Komninos, M. Pallot, B. Trousse, M. Nilsson, and A. Oliveira, *Smart cities and the future internet: Towards cooperation frameworks for open innovation*. Springer Berlin Heidelberg, 2011. 37

[83] M. J. Alam, M. R. Hossain, S. Azad, and R. Chugh, "An overview of LTE/LTE-A heterogeneous networks for 5G and beyond," *Transactions on Emerging Telecommunications Technologies*, vol. 34, no. 8, p. 4806, 2023. 50

[84] K. Pahlavan and P. Krishnamurthy, "Evolution and impact of Wi-Fi technology and applications: A historical perspective," *International Journal of Wireless Information Networks*, vol. 28, pp. 3–19, 2021. 51

[85] R. Heydon and N. Hunn, *Bluetooth Low Energy: The Developer's Handbook*, Bluetooth SIG. 51

[86] C. Wang, T. Jiang, and Q. Zhang, *ZigBee® Network Protocols and Applications*. CRC Press, 2014. 51

[87] M. B. Yassein, W. Mardini, and A. Khalil, "Smart homes automation using Z-wave protocol," in *Proceedings of the 2016 International Conference on Engineering & MIS (ICEMIS)*. IEEE, 2016, pp. 1–6. 51

[88] V. Coskun, B. Ozdenizci, and K. Ok, "A Survey on Near Field Communication (NFC) Technology," *Wireless Personal Communications*, vol. 71, no. 3, pp. 2259–2294, Aug 2013. 51

[89] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A study of lora: Long range & low power networks for the internet of things," *Sensors*, vol. 16, no. 9, p. 1466, 2016. 51

[90] X. Jia, Q. Feng, T. Fan, and Q. Lei, "RFID technology and its applications in Internet of Things (IoT)," in *Proceedings of the 2012 2nd international conference on consumer electronics, communications and networks (CECNet)*. IEEE, 2012, pp. 1282–1285. 51

[91] V. Cerf and R. Kahn, "A Protocol for Packet Network Intercommunication," *IEEE Transactions on communications*, vol. 22, no. 5, pp. 637–648, 1974. 51

[92] Internet Engineering Taskforce, "RFC 768 - User Datagram Protocol," https://datatracker.ietf.org/doc/html/rfc768, accessed: 2023-07-27. 52

[93] ——, "RFC 7252 - The Constrained Application Protocol," https://datatracker.ietf.org/doc/html/rfc7252, accessed: 2023-07-28. 52

[94] G. C. Hillar, *MQTT Essentials-A lightweight IoT protocol*. Packt Publishing Ltd, 2017. 52

[95] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of LoRaWAN for IoT: From technology to application," *Sensors*, vol. 18, no. 11, p. 3995, 2018. 52

[96] S. K. Shandilya, S. A. Chun, and S. Shandilya, *Internet of Things Security: Fundamentals, Techniques and Applications*. CRC Press, 2022. 54

[97] J. Valente, M. A. Wynn, and A. A. Cardenas, "Stealing, Spying, and Abusing: Consequences of Attacks on Internet of Things Devices," *IEEE Security & Privacy*, vol. 17, no. 5, pp. 10–21, 2019. 56

[98] O. Shwartz, Y. Mathov, M. Bohadana, Y. Elovici, and Y. Oren, "Reverse engineering IoT devices: Effective techniques and methods," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4965–4976, 2018. 57

[99] M. Pirretti, S. Zhu, N. Vijaykrishnan, P. McDaniel, M. Kandemir, and R. Brooks, "The Sleep Deprivation Attack in Sensor Networks: Analysis and Methods of Defense," *International Journal of Distributed Sensor Networks*, vol. 2, no. 3, pp. 267–287, 2006. 57

[100] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of Things security: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10–28, 2017. 57

[101] F. Ullah, M. Edwards, R. Ramdhany, R. Chitchyan, M. A. Babar, and A. Rashid, "Data exfiltration: A review of external attack vectors and countermeasures," *Journal of Network and Computer Applications*, vol. 101, pp. 18–54, 2018. 59

[102] G. Rajendran, R. R. Nivash, P. P. Parthy, and S. Balamurugan, "Modern security threats in the Internet of Things (IoT): Attacks and Countermeasures," in *Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019, pp. 1–6. 59

[103] S. S. Ullah, I. Ullah, H. Khattak, M. A. Khan, M. Adnan, S. Hussain, N. U. Amin, and M. A. K. Khattak, "A Lightweight Identity-Based Signature Scheme for Mitigation of Content Poisoning Attack in Named Data Networking With Internet of Things," *IEEE Access*, vol. 8, pp. 98 910–98 928, 2020. 59

[104] B. Bostami, M. Ahmed, and S. Choudhury, "False Data Injection Attacks in Internet of Things," *Performability in Internet of Things*, pp. 47–58, 2019. 60

[105] V. Mohindru and A. Garg, "Security attacks in internet of things: A review," *Recent Innovations in Computing: Proceedings of ICRIC 2020*, pp. 679–693, 2021. 60

[106] Y. Zhang, Y. Shen, H. Wang, J. Yong, and X. Jiang, "On Secure Wireless Communications for IoT Under Eavesdropper Collusion," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 3, pp. 1281–1293, 2015. 61

[107] L. Xiao, Y. Li, G. Han, G. Liu, and W. Zhuang, "PHY-Layer Spoofing Detection With Reinforcement Learning in Wireless Networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 10 037–10 047, 2016. 61

[108] S. Khanam, I. B. Ahmedy, M. Y. I. Idris, M. H. Jaward, and A. Q. B. M. Sabri, "A Survey of Security Challenges, Attacks Taxonomy and Advanced Countermeasures in the Internet of Things," *IEEE access*, vol. 8, pp. 219 709–219 743, 2020. 61, 64

[109] M. M. Salim, S. Rathore, and J. H. Park, "Distributed denial of service attacks and its defenses in IoT: a survey," *The Journal of Supercomputing*, vol. 76, pp. 5320–5363, 2020. 62, 64

[110] M. Conti, N. Dragoni, and V. Lesyk, "A Survey of Man In The Middle Attacks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016. 62

[111] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, 2004, pp. 259–268. 62

[112] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, "Advanced social engineering attacks," *Journal of Information Security and Applications*, vol. 22, pp. 113–122, 2015. 63

[113] S. Srinivasa, J. M. Pedersen, and E. Vasilomanolakis, "Open for hire: Attack trends and misconfiguration pitfalls of iot devices," in *Proceedings of the 21st ACM Internet Measurement Conference*, 2021, pp. 195–215. 63

[114] D. Stiawan, M. Y. Idris, R. F. Malik, S. Nurmaini, N. Alsharif, R. Budiarto, and M. Martina, "Investigating Brute Force Attack Patterns in IoT Network," *Journal of Electrical and Computer Engineering*, vol. 2019, jan 2019. 63

[115] S. Gastellier-Prevost and M. Laurent, "Defeating pharming attacks at the client-side," in *Proceedings of the 5th International Conference on Network and System Security*. IEEE, 2011, pp. 33–40. 64

[116] R. Tahir, "A Study on Malware and Malware Detection Technique," *International Journal of Education and Management Engineering*, vol. 8, no. 2, p. 20, 2018. 65

[117] A. Sadeghian, M. Zamani, and S. M. Abdullah, "A Taxonomy of SQL Injection Attacks," in *Proceedings of the 2013 International Conference on Informatics and Creative Multimedia*. IEEE, 2013, pp. 269–273. 65

[118] SECURIoTESIGN Team, "SAM API Documentation," https://securiotesign.github.io/SAM-API/, accessed: 2023-12-05. 68

[119] J. Brooke, "Sus: a "quick and dirty'usability scale," in *Usability Evaluation In Industry*. Taylor & Francis, 1996, vol. 189, no. 3, pp. 189–194. 78, 79

[120] N. Bevan, "Measuring usability as quality of use," *Software Quality Journal*, vol. 4, pp. 115–130, 1995. 78

[121] I. D. F. IxDF, "What is Usefulness?" https://www.interaction-design.org/literature/topics/usefulness, accessed: 2024-02-26. 78

[122] J. Robinson, *Likert Scale*, A. C. Michalos, Ed. Dordrecht: Springer Netherlands, 2014. 79

[123] MITRE, "Common Weakness Enumeration," https://cwe.mitre.org/index.html, 2024, accessed: 2024-01-20. 83

[124] M.-L. Zhang, Y.-K. Li, X.-Y. Liu, and X. Geng, "Binary relevance for multi-label learning: an overview," *Frontiers of Computer Science*, vol. 12, 11 2017. 89

[125] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier Chains for Multi-label Classification," *Machine Learning*, vol. 85, pp. 333–350, 2011. 89

[126] G. Nasierding and A. Z. Kouzani, "Comparative evaluation of multi-label classification methods," in *Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery*. IEEE, 2012, pp. 679–683. 89

[127] J.-Y. Jiang, S.-C. Tsai, and S.-J. Lee, "FSKNN: Multi-label text categorization based on fuzzy similarity and k nearest neighbors," *Expert Systems with Applications*, vol. 39, no. 3, pp. 2813–2821, 2012. 90

[128] H. Zhang, S. Kiranyaz, and M. Gabbouj, "A k-nearest neighbor multilabel ranking algorithm with application to content-based image retrieval," in *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2587–2591. 90

[129] U. Niaz and B. Merialdo, "Improving video concept detection through label space partitioning," in *Proceedings of the 2014 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2014, pp. 1–6. 90

[130] T. G. Dietterich, "Ensemble methods in machine learning," in *Proceedings of the International Workshop on Multiple Classifier Systems*. Springer, 2000, pp. 1–15. 91

[131] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. 91

[132] P. Szymański and T. Kajdanowicz, "A scikit-based Python environment for performing multi-label classification," *arXiv preprint arXiv:1702.01460*, 2017. 91

[133] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997. 91

[134] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, pp. 3–42, 2006. 91, 92

[135] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001. 91

[136] P. H. Swain and H. Hauska, "The decision tree classifier: Design and potential," *IEEE Transactions on Geoscience Electronics*, vol. 15, no. 3, pp. 142–147, 1977. 92

[137] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification and Scene Analysis*. Wiley, 1995. 92

[138] J. L. Bentley, "Survey of techniques for fixed radius near neighbor searching," U.S. Department of Energy Office of Scientific and Technical Information, Tech. Rep., August 1975. 92

[139] L. Bottou, "Stochastic Gradient Descent Tricks," in *Neural Networks: Tricks of the Trade: Second Edition*. Springer Berlin Heidelberg, 2012, pp. 421–436. 92

[140] J. Berkson, "Application of the Logistic Function to Bio-Assay," *Journal of the American Statistical Association*, vol. 39, no. 227, pp. 357–365, 1944. 92

[141] D. Li, Z. Zhang, X. Chen, H. Ling, and K. Huang, "A Richly Annotated Dataset for Pedestrian Attribute Recognition," *arXiv preprint arXiv:abs/1603.07054*, 2016. 93

[142] K. R. Chowdhary, "Natural language processing," *Fundamentals of artificial intelligence*, pp. 603–649, 2020. 95

[143] M. Shanahan, "Talking about Large Language Models," *Commun. ACM*, vol. 67, no. 2, p. 68–79, jan 2024. 97

[144] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-Training," https://api.semanticscholar.org/CorpusID:49313245, 2018, Accessed: 2024-01-24. 97

[145] K. G. A. Ludvigsen, "The carbon footprint of GPT-4," https://towardsdatascience.com/the-carbon-footprint-of-gpt-4-d6c676eb21ae, accessed: 2024-01-23. 98

[146] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713. 98

[147] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," *arXiv preprint arXiv:2305.14314*, 2023. 99

[148] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," *arXiv preprint arXiv:2106.09685*, 2021. 99

[149] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open Foundation and Fine-Tuned Chat Models," *arXiv preprint arXiv:2307.09288*, 2023. 100

[150] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A Tutorial on the Cross-Entropy Method," *Annals of operations research*, vol. 134, pp. 19–67, 2005. 101

Appendix A

ATIoT SUS-Based Questionnaire

Usability of the ATIoT Tool

This form was created to assess the usability and usefulness of the ATIoT tool, a tool to select a set of attacks an IoT system may be exposed to, and outputting a set of corresponding attack trees.

Please classify the categories presented below from 1 to 5, with one being Strongly disagree, and 5 Strongly agree.

Thank you for your time!

* Required

* This form will record your name, please fill your name.

1. I think that I would like to use this tool frequently *

Strongly disagree — Strongly agree

2. I found the tool unnecessarily complex *

Strongly disagree — Strongly agree

3. I thought the tool was easy to use *

Strongly disagree — Strongly agree

4. I think I would need the support of a technical person to be able to use this tool *

Strongly disagree — Strongly agree

5. I found the outputs of the tool very clear to understand *

Strongly disagree — Strongly agree

6. I thought there was too much inconsistency in the tool *

Strongly disagree ○ ○ ○ ○ ○ Strongly agree

7. I would imagine that most people would learn to use this tool very quickly *

Strongly disagree ○ ○ ○ ○ ○ Option 5

8. I found the tool very cumbersome to use *

Strongly disagree ○ ○ ○ ○ ○ Strongly agree

9. I felt very confident using the tool *

Strongly disagree ○ ○ ○ ○ ○ Strongly agree

10. I needed to learn a lot of things before I could get going with this tool *

Strongly disagree ○ ○ ○ ○ ○ Strongly agree

11. I found the tool to be useful in development work *

Strongly disagree ○ ○ ○ ○ ○ Strongly agree

12. I found the questions made by the tool unclear and difficult to answer *

Strongly disagree ○ ○ ○ ○ ○ Strongly agree