Agent-Based Modeling Framework for Adaptive Cyber Defence of the Internet of Things

by

Laura Rafferty

A thesis submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

Faculty of Business and IT

University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

December 2022

Thesis Examination Information

Submitted by: **Laura Rafferty**

Doctor of Philosophy in Computer Science

Thesis title: Agent-Based Modeling Framework for Adaptive Cyber Defence of the Internet of Things

An oral defence of this thesis took place on **November 24, 2022,** in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Stephen Jackson
Research Supervisor	Dr. Patrick Hung
Research Co-supervisor	Dr. Farkhund Iqbal, Zayed University
Examining Committee Member	Dr. Miguel Vargas Martin
Examining Committee Member	Dr. Bill Kapralos
University Examiner	Dr. Stephen Marsh
External Examiner	Dr. Mohammad Zulkernine, Queen’s University

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

The adoption of the Internet of Things (IoT) continues to increase significantly, introducing unique challenges and threats to cybersecurity. In parallel, adaptive and autonomous cyber defence has become an emerging research topic leveraging Artificial Intelligence for cybersecurity solutions that can learn to recognize, mitigate, and respond to cyber attacks, and evolve over time as the threat surface continues to increase in complexity. This paradigm presents an environment strongly conducive to agent-based systems, which offer a model for autonomous, cooperative, goal-oriented behaviours which can be applied to perform adaptive cyber defence activities. This thesis aims to bridge the gap between theoretical multi-agent systems research and cybersecurity domain knowledge by presenting a novel applied framework for adaptive cyber defence that can address a wide range of challenges and provide a foundation for significant future research in systems modeling for cybersecurity. Belief-Desire-Intention (BDI) agent architecture is extended within this work through a novel application of knowledge graphs to provide a scalable data model for agents to understand their environment, infer the context of threats, create goals associated with security requirements, and select plans based on possible actions and expected results. The framework has been implemented to demonstrate the feasibility of the architecture and evaluate the design properties through applied security use cases. While the experimental results have demonstrated the value of the framework applied to IoT systems, the concept can be easily expanded to other domains. This thesis provides the foundation to inspire further research works in this area for continued development, application, and optimization to support the advancement of the industry and bring autonomous, adaptive cyber defence to realization.

Keywords: Multi-Agent Systems (MAS); Security; Belief-Desire-Intention (BDI); Internet of Things (IoT); Adaptive Cyber Defence; Knowledge Graphs

Author's Declaration

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize the University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

_______________________________________ Laura Rafferty

Statement Of Contributions

In the development of this thesis, I have created the architecture, design, and testing of the proposed model, as well as the writing of this manuscript. Standard referencing practices have been used throughout this work to acknowledge ideas, research techniques, or other materials that belong to others.

In this work, I have accomplished the following contributions:

- Development of a multi-agent architecture for adaptive cyber defence with an individual agent reasoning model as well as control and coordination hierarchy.
- A novel extension of the BDI model enabled by knowledge graphs for cybersecurity modeling based on industry knowledge bases which can be leveraged for policy-based, adaptive agent reasoning.
- Implementation and experimental results to prove the design through use cases.

Parts of this work have been published or are pending publication as:

L. Rafferty, P.C.K. Hung, E. Kafeza, "BDI Agents for Adaptive Cyber Defense of IoT Systems," Transactions on Computational Science & Computational Intelligence, Proceedings of the 2021 International Conference on Artificial Intelligence (ICAI21), Springer. *Accepted for publication*.

L. Rafferty, F. Iqbal, S. Aleem, Z. Lu, S. -C. Huang and P. C. K. Hung, "Intelligent Multi-Agent Collaboration Model for Smart Home IoT Security," *2018 IEEE International Congress on Internet of Things (ICIOT)*, 2018, pp. 65-71, DOI: 10.1109/ICIOT.2018.00016.

L. Rafferty, F. Iqbal, P.C.K. Hung (2017). A Security Threat Analysis of Smart Home Network with Vulnerable Dynamic Agents. In: Tang, J., Hung, P. (eds) Computing in Smart Toys. International Series on Computer Entertainment and Media Technology. Springer, Cham. https://doi.org/10.1007/978-3-319-62072-5_8.

Table of Contents

List Of Tables

List Of Figures

List Of Abbreviations and Symbols

AAA	Authentication, Authorization, Accounting
ABC	Agent-Based Computing
ABM	Agent-Based Modeling
AI	Artificial Intelligence
API	Application Programming Interface
ART	Average Response Time
BDI	Belief-Desire-Intention
CAPEC	Common Attack Pattern Enumeration and Classification
CAR	Cyber Analytics Repository
CIA	Confidentiality, Integrity, Availability
CPN	Coloured Petri Net
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DC	Device Capabilities
DoS/DDoS	Denial of Service / Distributed Denial of Service
FNR	False Negative Rate
FPR	False Positive Rate
IoT	Internet of Things
IT	Information Technology
MAC	Multi-Agent Collaboration
MAS	Multi-Agent System
ML	Machine Learning
NIST	National Institute of Standards and Technology
PDP	Policy Decision Point
PEP	Policy Enforcement Point
POMDP	Partially Observable Markov Decision Process
TNR	True Negative Rate
TPR	True Positive Rate

1 Introduction

1.1 Motivation

Connected devices across the Internet of Things (IoT) are becoming more prominent and pervasive within enterprises, industries, consumers, and hybrid environments with varying device capabilities. Particularly within the consumer market, smart home devices introduce a unique environment with sensitive personal data, automation and availability requirements, limited expertise of users, and unique security threats.

Securing IoT devices is becoming progressively more complex due to the expanding attack surface, the volume of data, and environmental complexity. Automation, Artificial Intelligence (AI) and machine learning are becoming increasingly adopted in cybersecurity, where increased data sharing, indexing and organization of knowledge and cybersecurity frameworks can be leveraged. Adaptive cyber defence is an emerging research topic bridging the AI and cybersecurity domains to create semi-autonomous cyber defences that can learn to recognize and respond to cyber attacks, discover and mitigate weaknesses while evolving over time in response to changes in attacker behaviour, system health and readiness, and natural shifts in user behaviour [1]. Some limitations of current works include standalone solutions that do not provide interoperability or are too theoretical or abstract. There is a strong need for a practical framework for the implementation of these capabilities, not only for security but also for regular IoT services [2] [3].

While it has been an evolving subject of research for several decades, agent-based computing is now an emerging research topic within the IoT domain with many applications including cybersecurity. A systems approach can consider the cybersecurity domain from the perspective of an organic system, where intelligent agents that can perform self-healing capabilities in response to evolving threats. Many existing multi-agent approaches have been highly theoretical or maintained limited practical application, but show potential to achieve the necessary capabilities for securing IoT devices if provided a solution to bridge the gap between these research fields [2].

This thesis creates a modular applied framework to leverage data models, domain knowledge, and multi-agent architecture to perform adaptive cyber defence capabilities through contextual policy generation and enforcement. The Belief-Desire-Intention (BDI) model is extended for behavioural modeling of agents to perform practical reasoning and deliberation of actions in pursuit of goals.

By addressing gaps in theoretical and applied research, this framework provides a foundation for applications in further works for simulations for adversarial learning, optimization, scalability for new services, attack path modeling, risk analysis, predictions, probabilistic reasoning, utility engineering and experiments, and behavioural analysis, and can be used for testing and practical applications for advancing cybersecurity controls.

1.2 Research Contributions

This work aims to bridge the gap between theoretical multi-agent systems research and cybersecurity domain knowledge to provide a novel applied framework for adaptive cyber defence that can address a wide range of challenges and provide a foundation for significant future research in systems modeling for cybersecurity.

Cybersecurity can be modelled as a defence control problem, where autonomous defence capabilities can be integrated into adaptive intelligent software agents. Processes are modelled as multi-agent plans and tasks, where agents work together to achieve common goals to defend the network. We define a multi-agent adaptive cyber defence model within IoT smart home environments, using the BDI agents to perform autonomous and adaptive goal-based reasoning for defence actions enabled by cybersecurity domain knowledge graphs.

The key contributions of this thesis are as follows:

- Development of a multi-agent architecture for adaptive cyber defence with an individual agent reasoning model as well as control and coordination hierarchy.
- A novel extension of the BDI model enabled by knowledge graphs for cyber modeling based on industry knowledge bases which can be leveraged for policy-based, adaptive agent reasoning.

The presented framework has been implemented to demonstrate the feasibility of the architecture and evaluate the design properties through applied security use cases. While the experimental results have demonstrated the framework applied to IoT systems, the concept can be easily expanded to other domains. This thesis provides the foundation to inspire future research works on agent-based solutions for continued development, application, and optimization to support the advancement of the industry and bring autonomous, adaptive cyber defence to realization.

1.3 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 provides context into the background and review of existing literature on IoT and smart home technologies and associated security requirements, along with a background on agent-based modeling and multi-agent systems. Chapter 3 presents the multi-agent architecture for adaptive cyber defence with a detailed review of each component and design. Chapter 4 presents the data model and knowledge graphs in detail as a foundation for the following Chapter 5, which illustrates an implementation of the model to simulate agent control and coordination, as well as individual agent examples for specific security use cases. Next, Chapter 6 provides the experimental results and an evaluation of the design properties. Finally, Chapter 7 concludes the thesis and identifies future works.

2 Background and Literature Review

This chapter provides a review of relevant background information and existing research on IoT and smart home security, agent-based modeling, and multi-agent systems. The contributions of this thesis have been inspired by a wide range of research domains and topics, with a considerable review of literature across adjacent fields. While our contribution sits on an intersection of a variety of topics of considerable depth, we have positioned the contribution of our work as a unifying framework to enable the convergence of topics. This chapter will provide the essential background knowledge required as a prerequisite for understanding and appreciating the model described in the following chapters.

We will begin with a brief overview of the key themes of IoT, Smart Homes, Security, Adaptive Cyber Defense, and Agent-Based Modeling. While several of these topics have limited relationships in the existing literature, much of our review will highlight key works from the primary field or relate to a subset of the topics. We will also highlight any related works bridging two or more topics.

The literature review within this chapter has been performed to analyze relevant studies and background information with the below research questions:

- What are the characteristics and security concerns of IoT and smart homes?
- What security solutions have been proposed for smart home/IoT?
- What solutions have been proposed for multi-agent systems, BDI, reasoning, planning, etc.?
- How have agent-based approaches been applied to IoT and security domains?
- What are the challenges and gaps?

2.1 IoT and Smart Home Security

2.1.1 Overview of IoT and Smart Home

The explosion of the Internet of Things (IoT) in recent years has made a prominent impact on almost all areas of modern life [4], introducing a network of physical devices endowed with embedded sensors and networking capabilities to enable a vast array of pervasive services. IoT is “a global infrastructure for the information society, enabling advanced services by interconnecting physical and virtual things based on existing and evolving interoperable information and communication technologies” [5]. The development of IoT and smart home solutions are driven by advances in mobile devices, embedded and ubiquitous communication, cloud computing, and data analytics to enable data collection, sharing and analysis in heterogeneous pervasive networks [6]. IoT is one of the fastest growing sectors in the technology industry, as an enabler of the intersection of numerous technology fields to bridge opportunities into services tightly coupled with the physical world. While insights and predictions vary across industry reports, significant investments and expansion are expected to continue over the next decade. A recent report from IoT Analytics has identified that the market for IoT is expected to grow to 14.4 billion active connections by the end of 2022 and to increase to 27 billion by 2025 [7]. The International Data Corporation forecasted an even more significant increase of 55.7 billion connected IoT devices by 2025 [8]. IoT technologies are becoming widely adopted across many industries and applications, including supply chain, lifestyle, retail, industrial control systems, environment, emergency services, agriculture, transportation, energy, healthcare, smart cities, and buildings [4]. Recent trends have shown the adoption of IoT devices as a response to the COVID-19 pandemic has introduced new opportunities for IoT in the healthcare and home consumer industries, while there has also been an emergence of Smart City initiatives driving the market growth, such as in the Kingdom of Saudi Arabia (KSA) and the United Arab Emirates (UAE) [9].

Smart Home technologies are some of the most widely used and deployed applications for consumer IoT solutions [10], which provide digital services within and outside the home through a range of networked devices. Smart homes introduce an environment where IoT exists in the context of everyday objects in homes, such as fridges, furnaces, televisions and lighting, and allow for greater automation and comfort of daily activities. Users can control devices such as lighting, air conditioning, sound systems and security systems through remote interfaces such as smartphones or virtual assistants. There may be automation, personalized, and contextual services based on preferences or previously observed behaviour, such as dimming the lights and turning on the television when a user sits on the couch at 6 PM. A notification may be sent to the user if a plant needs watering, or a device may automatically water the plant. An alert can be sent to a healthcare provider if an individual is displaying abnormal behaviour symptomatic of a health issue. The possibilities are endless while the overall goals of smart home technologies include increased comfort, reduced costs of energy and resource consumption, and creating new opportunities for services within the home [10].

Smart Home Architecture

While much of the literature envisions an eventual direction for fully autonomous, interconnected and pervasive smart home services, this vision has not yet been prominently adopted across the industry. Hammi et al. define contemporary smart home deployments from an industry standpoint as a network of predominantly independent devices focused on specific tasks triggered by a schedule or controlled through a user interface [11]. In this model, many deployments make use of a centralized control platform such as an Intelligent Virtual Assistant (IVA) in the form of a smart speaker or smartphone application to integrate and control multiple smart devices across the home. Such solutions include capabilities for voice commands and programming frameworks for third-party developers to build applications to interact with the devices [12]. Popular solutions include Amazon Alexa [13], Google Home [14], and Apple Homekit [15], which are compatible with a wide range of consumer IoT devices on the market. These solutions allow for ease of use, centralized control of multiple devices, and opportunities for enhanced services for users.

As smart home and IoT environments are comprised of multiple interacting services and components, many layered architectures have been proposed across the literature with varying granularity and objectives. The three-layer architecture [16] (perception, network, application) is defined below, which is sufficient to provide an appreciation for the overall components of a general IoT deployment for the purpose of this work:

- **Perception Layer:** This layer resides on the physical devices to provide sensory and actuation capabilities to gather information and/or perform physical actions within the environment. This has also been referred to as the Edge layer in some models.
- **Network Layer:** This layer enables the transmission and processing of data between devices across the network, including over the internet. Cloud backend services have been consolidated into this layer. Common communication protocols at this layer include IEEE 802.x, Near-Field Communication (NFC), Zigbee, and Bluetooth [17].
- **Application Layer:** This layer provides applications and services to users and devices based on the application type. The application layer can operate as middleware and commonly interacts through an Application Programming Interface (API) such as REpresentational State Transfer (REST) and Hypertext Transfer Protocol (HTTP).

Key Characteristics and Challenges

IoT and smart home technologies maintain a set of unique characteristics and challenges that differ from traditional Information Technology (IT) systems, motivating appropriate design considerations along with their development, as shown in [4].

Key characteristics of IoT technologies include interconnectivity, heterogeneity, pervasiveness, dynamic environment, and scale [4]. Inherently, a smart home ecosystem can consist of many interconnected heterogeneous devices across different hardware, platforms, and protocols that communicate with each other over a network. The sensor and networking capabilities allow the systems to collect and exchange substantial amounts of data on users and the environment. With this data, inferences can be made about user interactions to provide personalized context-aware services and integrate with other technologies, such as smart phones and smart watches, to improve the user experience further. In addition, smart home technologies often make use of cloud services, where the mass amounts of data collected are processed and stored in the cloud. This creates new opportunities for providing valuable services to users but also requires capabilities for secure and effective management of personal data.

Key challenges for the implementation of IoT and smart home services include resource limitations, interoperability, reliability, data volume and sensitivity [4], [18], [19] due to resource constraints on devices, sensitivity and volume of data collected, and requirements for connectivity. Resource constrained devices must consider lightweight applications due to potential power, storage, bandwidth, and memory limitations. Availability and ease of use are additional key considerations in smart home deployments, aligned with the primary objective of providing convenient services to users. Solutions should be straightforward to users and provide real-time services as needed [10], [20].

The characteristics of IoT technologies affect cybersecurity and privacy risks in unique ways that differ from traditional IT devices as defined by the National Institute of Standards and Technology (NIST) [19] below:

1. Many IoT devices interact with the physical world in ways conventional IT devices usually do not.
2. Unlike conventional IT devices, many IoT devices cannot be accessed, managed, or monitored.
3. The availability, efficiency, and effectiveness of cybersecurity and privacy capabilities are often different for IoT devices than conventional IT devices.

With these characteristics in mind, the following sections elaborate on the cybersecurity challenges, requirements, and proposed solutions in the literature.

2.1.2 Security Requirements and Guidelines

Industry Standards and Regulation

Along with the vast opportunities introduced by IoT, there are increasing concerns across the industry, governments and consumers regarding the privacy and security of these technologies. Limitations associated with the security, integrity, and privacy of connected devices have been identified as an inhibitor to growth and adoption [9]. From an IoT solution provider's perspective, security is often deprioritized due to complexity, time-to-market pressure, or lack of knowledge [21]. A well-defined framework and standard for an end-to-end IoT application are unavailable due to the diversity of protocols, technologies, and devices involved. The industry has been challenged to balance the trade-offs between cost-effectiveness, security, reliability, privacy, and other factors [10].

However, the escalating risk of IoT threats has caught the attention of governments and regulators internationally, with several notable developments in recent years. In September 2015, the Federal Bureau of Investigation (FBI) issued its first public service announcement stating that "the Internet of Things poses opportunities for cyber crime" in the United States [22], indicating that insufficient security capabilities and complications with patching devices open opportunities for attackers to exploit IoT device weaknesses. Following the Mirai botnet attacks in October 2016, concerns about cyber threats to IoT gained increasing attention, when the US Department of Homeland Security in collaboration with NIST released a report on Strategic Principles for Securing the Internet of Things [23] which further identified the need to prioritize the security of IoT devices. Another public service announcement by the FBI in 2018 brought attention to cases where "Cyber Actors Use IoT Devices as Proxies for Anonymity and Pursuit of Malicious Cyber Activities" [24]. Most notably, in 2021, the United States issued an executive order on "Improving the Nation's Cybersecurity," which, among other items, directed NIST to initiate a pilot program for cybersecurity product labeling to educate the public on the security capabilities and requirements for IoT devices [25].

This directive to NIST followed a series of existing works by the organization on the evolution of cybersecurity requirement definitions. In 2019, NIST published NISTIR 8228 Considerations for Managing Internet of Things Cybersecurity Privacy Risks" [19], which provides a reference for organizations to better understand and manage cybersecurity and privacy risks associated with IoT devices. Three high-level risk mitigation goals have been defined as protecting device security, data security, and individuals' privacy. For each of these goals, a set of risk mitigation areas have been proposed in Table 2.1 below. Following this, NISTIR 8259A [26] issued a set of IoT security core baseline capabilities in 2020 which were strongly aligned with the identified risk mitigation goals. The work has continued in response to the executive order, and NIST published a recommended criteria for cybersecurity labeling for consumer IoT Products [19] in February 2022, reflecting the capabilities defined in NISTIR 8259A. Our proposed model references these capabilities and will be further expanded in Chapter 4.

Table 2.1 NIST 8228 IoT Cybersecurity Risk Mitigation Areas [19]

Risk Mitigation Goal	**Risk Mitigation Areas**
Goal 1: Protect Device Security	• Asset Management • Vulnerability Management • Access Management • Device Security Incident Detection
Goal 2: Protect Data Security	• Data Protection • Data Security Incident Detection
Goal 3: Protect Individuals' Privacy	• Information Flow Management • Personal Identifiable Information (PII) Processing Permissions Management • Informed Decision Making • Disassociated Data Management • Privacy Breach Detection

Outside of these initiatives, other notable organizations are contributing to the enablement of security within IoT devices. For example, the IoT Security Foundation [27] is a non-profit organization established to promote security efforts for IoT by providing a mechanism for sharing knowledge, best practices and advice. The IoT Security Foundation's guide entitled "Establishing Principles for Internet of Things Security" outlines several security best practices, including designing with security in mind from the beginning, offering appropriate protection for all potential attack surfaces (i.e., device, network, server, cloud, etc.), managing encryption keys securely, verifying the integrity of software, using a hardware-rooted trust chain, applying authentication and integrity protection to data, identifying and revoking compromised or malfunctioning devices, isolating data where applicable, and ensuring device metadata is trusted and verifiable.

Security Requirements

Balancing the need for security and privacy with the characteristics and challenges of the unique architecture of a smart home proves to be a challenging task. Nevertheless, there have been several common security requirements defined in the literature, primarily aligned to the Confidentiality Integrity and Availability (CIA) or Authentication Authorization and Accounting (AAA) models common to traditional security paradigms [21], [28], [29]. The key security requirements of smart home applications include confidentiality, integrity, availability, privacy, and authentication.

While smart home devices can collect large volumes of personal data, preserving the confidentiality and privacy of this data is critical. Additionally, authentication mechanisms to restrict access to unauthorized users are also required to protect the data and access to devices within the home. Smart home systems themselves are also dependent on the integrity of data received from sensors to provide appropriate services. The requirement for availability and convenience to users is of the utmost importance in the smart home, as countering this with stifling security controls would defeat the purpose. For these reasons, a lightweight solution is required for maintaining the security of masses of data collected from lightweight endpoints while embracing the functionality goals of the smart home by appearing seamless to the user. Although the goal of the smart home is automation and convenience, the management of device security must be also straightforward for users.

Smart home technologies include sensors, monitors, interfaces, appliances, and devices networked together to enable automation as well as localized and remote control of the domestic environment. The volumes of sensor data across a variety of sources, in combination with usage patterns and other inferred information, are growing significantly and introduce new assets that need to be protected. As more data can be collected from the smart home environment, the home is able to provide more customized services. Sensor data can be collected from a variety of inputs such as a microphone, camera, accelerometer, and thermometer. Data available to smart home systems can be of volunteered, observed, or inferred types. Volunteered data is explicitly provided through the user in terms of profile preferences. Observed data is collected through sensors such as microphones or usage data. Finally, inferred data refers to information that has been correlated between volunteered and observed data, such as what time a user is likely to return home based on previous usage patterns. Users may be unaware of observed or inferred data that is collected and stored by the system, and this information can become very personal, such as behaviour and life patterns.

Due to the personal value of the data collected and retained by smart home systems, such data can be a target for attackers for a variety of reasons. As sensors are integrated into “things" within the household, collected data can frequently be equated to physical observations, which can be further correlated with information collected from other sensors and sources. As IoT and smart homes are typically connected, other devices on the network, including smartphones and wearable devices, can interact with each other and share data. This makes it possible for further correlation across devices and for the data to be shared externally. Information collected can become increasingly intimate, such as health information, and can be correlated with data collected from other devices for further context extraction. Therefore, the privacy of all individuals within the home is at risk, including children who may be the primary users of some IoT technologies in the home, such as smart toys and social robots.

The physical nature of smart homes also introduces physical safety risks [30] since compromised home automation systems might be in control of devices such as door locks, health systems or furnaces. Smart home IoT devices may also be movable or located in sensitive locations, which further raises the severity of security concerns beyond the traditional digital model. Furthermore, the technology limitations implicit in the nature of IoT devices in smart homes introduce new vulnerabilities and attack vectors for potential intrusion into the home network. Although the compromise of a smart light bulb may not pose immediate risk aside from turning it on or off, if access to the light bulb allows an attacker to connect and gain control of other devices on the internal network (i.e., lateral movement and escalation of privileges), there are far greater risks.

2.1.3 Common Vulnerabilities and Attacks

Common Vulnerabilities

The technology in IoT and smart homes introduce new challenges to security, differing from traditional computing architectures. These challenges include low processing power and storage available to IoT endpoints leading to a lack of adequate endpoint security and encryption. Further, software loaded on devices is often outdated. Palo Alto's Unit 42 reported that the general security posture of IoT devices is declining, leaving organizations vulnerable to new IoT-targeted malware as well as older attack techniques [31]. The report further indicated that 98% of all IoT traffic is unencrypted, and 57% of IoT devices are vulnerable to medium or high-severity attacks, making them a convenient target for attackers. Low patching rates encourage opportunities for the exploitation of long-known vulnerabilities, while many attacks also focus on attacks on default passwords or legacy protocols. One study found that software components of home routers were often four to five years older than the device [32]. Patching or software upgrades are often not possible or are rarely applied, while many IoT devices do not have mechanisms for automated updates. The data is often most vulnerable at the sensor/collector level and when it is in transit at the edge of the network to the cloud. For this reason, endpoints need to be hardened as much as possible, and network communications should be made secure. Internet-facing devices with insufficient authentication, default passwords or other vulnerabilities such as cross-site scripting or code injection create further opportunities for unauthorized external access [33].

Smart homes in particular consist of an array of appliances that can be static or mobile, each with different security concerns while the large volume of devices creates an increased threat vector. Static devices are often large and are not intended to move around, such as a smart fridge or furnace. Many home appliances exceedingly long lifespans, such as refrigerators and televisions, which are likely to go without firmware updates, exposing them to threats associated with unpatched vulnerabilities. Some devices are more dynamic and likely to be moved around, either independently or with a user, possibly in and out of the home network. Mobile phones, wearable devices, and smart phones fall into this category. These types of devices are exposed to external threats outside of the home and may connect to unsecured external networks and expose the devices to external threats. While these devices take the form of traditional home items and appliances, users tend to have higher levels of trust and are perhaps unaware of the capabilities of these devices if they are abused through security breaches [34]. The complex network of devices from multiple vendors and standards presents further difficulty in achieving a unified approach for security across all devices in the smart home. With the rise in the number of connected devices which may be available in a smart home soon rivaling the number of devices in a mid-sized company, users are faced with the complexity of managing all of these devices without the assistance of sophisticated enterprise security tools or staff to monitor or respond to attacks. Each additional device introduces a new potential threat vector into the home network, which is only as secure as its weakest link.

Often the product development and support structure between third-party manufacturers and suppliers are not conducive to a healthy security posture. With a large number of heterogeneous devices and layers involved in providing end-to-end IoT services, there is usually no one entity responsible for security. Manufacturers may not integrate security into the software development lifecycle, focusing only on functionality. The limited processing and memory resources on the devices also inhibit security solutions from being run on the devices. Third-party manufacturers often do not monitor for vulnerabilities in their old systems or provide updates or support for old models, focusing mainly on the development of future models. Finally, another line of defence is the capabilities of the users themselves to implement security controls and configurations within their own networks. For a common user with limited technical knowledge, this is often overwhelming or not considered.

Across the different layers of an IoT and smart home architecture, different vulnerabilities can exist. Several notable contributions have been to the literature, including surveying existing works and mapping security and privacy issues to layered architectures. Deep et al. comprehensively study security and privacy issues across the 4-layered (perception, network, middleware, application) IoT architectures [28]. Another survey by [10] identified threats across the 5-layered architecture. HaddadPajouh et al. [17] on the 3-layered architecture. Verma et al. present a survey of Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks on IoT devices [29].

A final contribution of note is the Open Web Application Security Project (OWASP) IoT Top 10 [33], which is a commonly referenced listing of the top vulnerabilities found within IoT devices, described in Table 2.2 below. This listing shows the most predominant vulnerabilities within IoT devices, which also apply to smart homes and are commonly exploited by malicious actors.

Table 2.2 OWASP IoT Top 10 Vulnerabilities [33]

Vulnerability	**Description**
1. Weak, Guessable, or Hardcoded Passwords	Use of easily brute forced, publicly available, or unchangeable credentials, including backdoors or client software that grants unauthorized access to deployed systems.
2. Insecure Network Services	Unneeded or insecure network services running on the device itself, especially those exposed to the internet, that compromise the confidentiality, integrity/authenticity, or availability of information or allow unauthorized remote control.
3. Insecure Ecosystem Interfaces	Insecure web, backend Application Programming Interfaces (API), and cloud or mobile interfaces in the ecosystem outside the device allow a compromise of the device or its related components. Common issues include a lack of

	authentication/authorization, lacking or weak encryption, and a lack of input and output filtering.
4. Lack of Secure Update Mechanism	Lack of ability to securely update the device. This includes a lack of firmware validation on a device, a lack of secure delivery (un-encrypted in transit), a lack of anti-rollback mechanisms, and a lack of notifications of security changes due to updates.
5. Use of Insecure or Outdated Components	Use of deprecated or insecure software components/libraries that could allow the device to be compromised. This includes insecure customization of operating system platforms and the use of third-party software or hardware components from a compromised supply chain.
6. Insufficient Privacy Protection	Users' personal information is stored on the device or in the ecosystem that is used insecurely, improperly, or without permission.
7. Insecure Data Transfer and Storage	Lack of encryption or access control of sensitive data anywhere within the ecosystem, including at rest, in transit, or during processing.
8. Lack of Device Management	Lack of security support on devices deployed in production, including asset management, update management, secure decommissioning, systems monitoring, and response capabilities.
9. Insecure Default Settings	Devices or systems shipped with insecure default settings or lack the ability to make the system more secure by restricting operators from modifying configurations.
10. Lack of Physical Hardening	Lack of physical hardening measures, allowing potential attackers to gain sensitive information that can help in a future remote attack or take local control of the device.

Attacks on IoT and Smart Home Devices

Common attacker motivations for targeting smart home devices can vary from targeted attacks on homeowner's physical safety, privacy, or disruption to opportunity-based attacks assimilating vulnerable devices into a botnet aimed toward another target. Threats to users' physical safety can be enabled by vulnerable devices, while home burglaries have increased after the deployment of home automation systems which have been used to determine the behavior and presence of residents [10]. Financial loss, physical damage, or service disruption can be incurred by remotely targeted appliances such as smart washing machines, faucets, or thermostats, and potentially at communities on a larger scale to target the power grid [11]. Further, while IoT devices can be the weakest link in a network, seemingly innocuous devices such as smart light bulbs or door locks can be exploited as an entry point into a user’s network to obtain more lucrative objectives [10].

Current trends indicate a rise in security issues related to IoT over the past few years, with industry predictions showing a continuous upward trend. Mandiant predicts continued growth of the IoT device attack surface in 2022 and beyond, with the potential for serious impact as defenders struggle to keep up with no coordinated security initiative for IoT devices [35]. Publicized attacks and vulnerabilities illustrate the troubling state of security threats to consumer IoT devices over the past few years. Countless examples exist including hacked smart fridges exposing gmail credentials [36], vulnerabilities allowing unauthorized access to baby monitors [37] and smart locks [38] [39], and smart kettles leaking WiFi passwords [40].

The most impactful attacks on IoT devices in recent years have been related to the Mirai botnet and its variants. First active in 2016, the Mirai botnet exploited the widespread use of default credentials across millions of internet-facing IoT devices. The exploited devices were weaponized to perform large-scale DDoS attacks directed at high profile targets. An attack on the major Domain Name System (DNS) provider, Dyn, in October 2016 resulted in extreme service disruption to the majority of the U.S. east coast for several hours, with a downward impact on numerous major services such as Twitter, Netflix, and Reddit [41]. Although the end target of these DDoS attacks is not IoT devices, IoT devices such as Digital Video Recorders (DVRs), webcams, and other appliances are effectively being used as a tool for mass impact on greater targets. IoT botnets are one of the biggest threats to internet stability, and risk is expanding as more devices are created. These implications could be devastating as targets could shift to hospitals or critical systems. With the success of Mirai in achieving attacker objectives, similar variants of this malware have continued to evolve over subsequent years, moving from exploiting credentials to additional vulnerabilities. In 2022, Mirai variants such as beast mode and BotenaGo continue to target millions of routers and IoT devices with various exploits [42], [43].

Based on a survey by Hammi et al. [11], a summary of the major types of attacks on smart home devices is shown below in Table 2.3 according to the layer that is targeted, while some attacks can occur on multiple layers:

Table 2.3 Attacks to Smart Home Devices [11]

Layer	**Attack**
Device	• Physical node compromise
Network	• Scanning attack • Message forging or substitution attack • Message replay attack • Sybil attack • Spoofing attack • Eavesdropping
Application	• Default/hardcoded passwords • Malware/botnet • Compromised or over-privileged applications
Multiple Layers	• Denial of Service (DoS) • Adversarial machine learning

2.1.4 Security Solutions

Along with the prominent threats to IoT devices, contributions to the literature on IoT security solutions have been abundant. This section provides an overview of key themes and notable directions in this area.

To address common concerns with the lack of standardization across IoT architectures, the authors of [21] propose that security patterns could help address security concerns by providing reference architectures. However, current architectures focus on general system issues or specific domains and do not address specific technical concerns, and the majority of the works surveyed focus on a specific capability. Key themes of IoT security solution literature fall into the categories of intrusion detection systems, confidentiality, authentication or authorization systems, and DDoS protection solutions. Further, many works focused on emerging technologies such as blockchain, Artificial Intelligence (AI), Machine Learning (ML), and edge/fog/cloud solutions.

Intrusion detection systems, particularly at the network level, have been a prominent area of focus in the literature to relieve end devices of resource constraining tasks. Much of the literature makes use of AI or ML solutions for statistical analysis or anomaly detection to detect threats within network traffic [11]. GHOST [44] is a notable initiative funded by the European Union Horizon 2020 Research and Innovation Programme, which aims to increase the level and effectiveness of automation of existing cybersecurity services and enhance the self-defence of home IoT environments. The solution focuses on usable and transparent security, and presents a vendor agnostic reference architecture that is embedded in a smart home network gateway. Advanced packet flow analysis with self-learning capabilities is used to generate user and device profiles for automated real-time risk assessment, while users are provided with visualization of data analytics to understand their system's security status as well as mitigation guidelines. This solution as well as others are focused heavily on centralized models, however, they maintain limited view or interaction with devices directly

C**onfidentiality/authentication/authorization systems** must consider key challenges to IoT devices, including bandwidth and low power consumption, complexity, sensing, and the requirement for lightweight solutions, as indicated by Deep et al. [28]. The authors elaborate that there is no common mechanism to apply security to resource-constrained devices, and lightweight solutions are a future research direction for services such as key management, authentication, authorization, and access control. Hardware-based lightweight cryptographic solutions have also been recommended as a solution for the security of data at rest within devices [20].

DDoS solutions for prevention, detection, response and mitigation techniques, including filtering, honeypots, signature and anomaly-based detection, and others, are presented in [29]. The authors also identify open research problems for IoT DDoS protection, including functionality, deployment location, cost of the solution, scalability, specificity, accuracy rate, and false positives/negatives.

Emerging trends in the areas of fog and edge computing, AI and ML, and blockchain technologies have been identified across several works as showing promising solutions both integrated and independently [20]. However, there are still several open challenges and security issues with these technologies as a continued area of research [10].

- **Fog and Edge computing** have been identified in much of the literature as an enabler for processing large volumes of data securely and efficiently not only for regular IoT services but also for security services [28].
- **Artificial Intelligence and Machine Learning** techniques introduce opportunities for the detection of malicious or anomalous behaviour across large datasets. Many solutions have been proposed at the network or cloud level, but some more lightweight solutions can also be applied within edge devices to detect threats at run-time [28].
- **Blockchain** technologies have also been an emerging area for security solutions to IoT based on their decentralized architecture, the ability for pseudonymity, and the security and integrity of transactions [28].

Automation and Integration: With the increasing complexity of the expanding attack surface, trends in enterprise security solutions have been moving towards automation and integration of layered defences to reduce manual workload, detection and response times for protection against security threats. Extended Detection and Response (XDR) is a SaaS-based security threat detection and incident response solution that integrates, correlates, and contextualizes data and alerts from multiple sources [45]. This combines telemetry from other security tools such as endpoint detection and network analysis for more accurate detections and simplified visibility and response. XDR can integrate Security Orchestration, Automation and Response (SOAR) capabilities, which have also been increasingly adopted independently of XDR. Researchers at Johns Hopkins University, in collaboration with the US Department of Homeland Security (DHS) and the National Security Agency (NSA), have developed a framework and strategy for Integrated Adaptive Cyber Defence (IACD) [46]. The IACD provides guidelines, playbooks and workflows for SOAR implementation in combination with automated threat intelligence information sharing communities to improve response time and maintain adaptive defences in response to evolving threats. While these solutions are designed for enterprise environments, the benefits of automation and integrated security capabilities can be considered for a consumer environment where maintaining secure systems with low user interaction is a strong requirement.

As we conclude our coverage of the background defining the characteristics of IoT and smart home technologies, along with the security considerations and current solutions in the literature, the following section will introduce the concepts of multi-agent systems as a foundation of our proposed solution. In addition, it will also build on the case for the unification of these concepts into an adaptive cyber defence system for smart home devices.

2.2 Multi-Agent Systems

Agent-based systems have been a topic of extensive research for several decades. While the focus in this area of research seems to have slowed down, challenges with the practical application of a traditionally academic field, as well as limited tools and knowledge, have limited the barrier to agent applications [47]. However, the field is once again gaining increased interest with the expansion of IoT technologies presenting characteristics that are aligned with agent systems [3]. There have been some notable contributions in recent years to agent research in the fields of IoT, security, and in some cases, an intersection of these topics.

This section will provide a background on the preliminaries of agent-based modelling and multi-agent systems and present a review of the relevant literature in applications to the IoT and cybersecurity domains.

2.2.1 Agent-Based Modeling & Reasoning

Agent-based modeling (ABM) is a method of modeling systems composed of autonomous decision-making entities, known as agents, interacting with each other and their environment [48]. Agents execute actions based on a set of rules and often operate within an environment with other agents, known as Multi-Agent Systems (MAS), as shown in Figure 2.1. ABM has been used to simulate complex decentralized systems of autonomous agents to predict global system outcomes based on local interactions. In the biological world, this can be observed within the flocking behaviour of birds, movements of schools of fish, or waves in water. These concepts have also been applied to social network modeling, economic modeling, and market analysis, while more recently has been extended to the areas of IoT and AI.

An agent-based model consists of three basic elements: agents, relationships, and the environment [49], as defined below. The rules for how agents make individual decisions and interact with each other are formally defined in the model.

- **Agent**: An agent is an autonomous entity that makes decisions and actions based on a set of rules based on independent goals and perceptions.
- **Environment:** The physical or logical environment is shared by all agents in a system and contains artifacts that can be perceived and impacted by the actions of the agents.
- **Relationships**: The rules through which agents interact with each other, work together, or resolve conflicts.

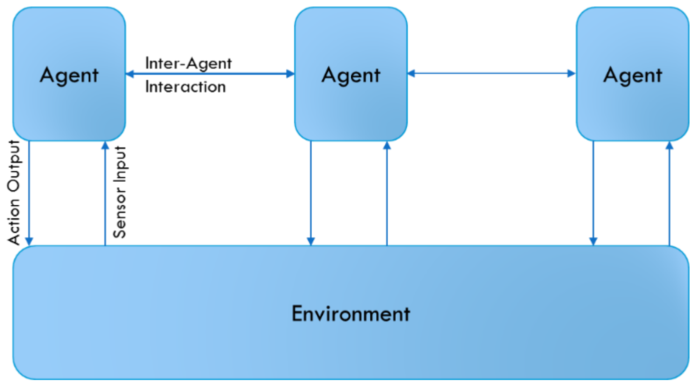


Figure 2.1 Simple Multi-Agent System

Intelligent agents also exhibit additional properties of reactivity, proactiveness, and social ability. A reactive agent is able to sense its environment and perform an action in response. Proactive agents display goal-oriented behaviour and will actively perform actions to reach a pre-defined goal. The social ability of agents allows them to interact with other agents to achieve their goals through cooperation or negotiation. For example, Macal and North [49] identify the following properties exhibited by intelligent agents:

- **Autonomy:** Discrete entity with attributes, behaviors and decision-making capability, and Independent and self-directing functionality.
- **Decision-making ability:** Rules to define agent behavior and decision-making.
- **Sociality:** The agent's ability to interact with other agents in the system through interaction protocols for mechanisms such as collision avoidance, agent recognition, communication and information exchange.
- **Conditionality**: A state consisting of a set or subset of the agent's attributes or behaviors.

Agents may also exhibit the following additional properties:

- **Goal-Oriented:** Possessing explicit goals to drive behaviour.
- **Adaptability**: Ability to learn and adapt behaviors based on past experiences.

The practical reasoning and decision-making ability of an agent is central to the operation of the system. Our research makes use of the BDI model for this purpose.

Further, there are various architectures that agents can follow for control models and shared resources, particularly in the case of collaborating agents, which will be described in the following section.

Belief-Desire-Intention (BDI)

The BDI model, originally developed by Bratman [50], is used for behavioural modeling of agents to perform practical reasoning and the process of deciding what actions to perform to reach a goal. In this architecture, agents receive sensory input through perceptions that influence their beliefs and implement their intended behaviours (intentions) to achieve desired states based on these beliefs, as illustrated in Figure 2.2. The components and functions of BDI can be modeled as per below [51]:

- **Percept** $P_n = \{p_0, p_1, \ldots, p_n\}$ represents a set of perceptions p taken as input by agents.
- **Beliefs** $B_n = \{b_0, b_1, \ldots, b_n\}$ represents a set of the information b maintained by the agent on its internal state and the environment states, updated according to each perception p.
- **Desires** $D_n = \{d_0, d_1, \ldots, d_n\}$ represents a set of the agent's goals to be achieved d, including properties and costs associated with each goal.
- **Intentions** $I_n = \{i_0, i_1, \ldots, i_n\}$ represents an action plan providing a set of states i the agent intends to bring about. The set of intentions must be consistent and not contain any conflicts.
- **Belief Revision Function** $BRF(p_n,B_n) \rightarrow B_m$ takes a perceptual input p_n and the agent's current belief set B_n, and determines a new set of beliefs B_m. Belief revision can include the following [52]:
    - **Expansion**: a new sentence b_{n+1} is added to the belief set B.
    - **Revision**: a new sentence b_{n+1} that is inconsistent with a belief set B is added, but to maintain consistency with the resulting belief set, some old sentences are deleted.
    - **Contraction**: some sentence b is retracted from belief set B without adding any new artifacts.

- **Option Generation Function** OPG(B_n,I_n) → D_n determines the possible alternatives (desires) available to an agent-based on its current beliefs and intentions.
- **Filter Function** FIL(B_n,D_n,I_n) → I_n determines a consistent set of intentions based on the agent's current beliefs, desires and intentions.
- **Action Selection Function** ACT(B_n,I_n,) → {a_0, a_1,…,a_n} implements means-ends reasoning to map the current set of beliefs *B* and intentions to a sequence of actions *a*.

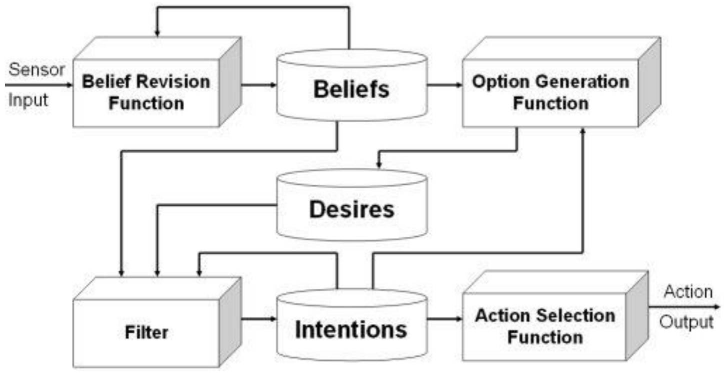


Figure 2.2 BDI Components [53]

The high-level process of the BDI model is described below in Figure 2.3. It is important to note that the BDI model does not account for dynamic plan generation and instead depends on a predefined plan database. While this approach is commonly static and has limitations to scalability and adaptation to evolving collections of knowledge, we expand on this approach by introducing a novel integration with knowledge graphs in Chapter 4.

```
B := $B_0$;
I := $I_0$;
while true do
      get next percept p;
      B := BRF(B,p);
      D := OPG(B,I);
      I := FIL(B,D,I);
      N := PLN(B,I);
      execute( )
end while
```

Figure 2.3 BDI High-Level Process Description [51]

Related Works

There have been many research works in extending the original concept of BDI, particularly to adapt to the concept of dynamic environments. While the limitations of BDI do not allow for dynamic goal deliberation, Pokahr et al. [54] present an enhanced BDI architecture for allowing goal deliberation at any point in time. Males and Ribaric [55] model an extended BDI agent with autonomous entities. Peng et al. [56] extend the BDI model with norms, policies and contracts. Shaw and van der Poel [57] propose genetic algorithms as a mechanism for re-planning in BDI agents. Lastly, Buford et al. [58] extend BDI for situation management, following the steps of event correlation, situation recognition, plan deliberation, plan instantiation and intention execution.

In environments where uncertain or incomplete sensor information is collected, it can be difficult for BDI agents to make appropriate plans. For example, Calderwood et al. [59] present a framework using Dempster-Shafer theory for a contextual merging of data for better informed plan selection in Supervisory Control and Data Acquisition (SCADA) systems. In terms of recent applications, BDI has been used for multi-agent modeling of fire detection in coal mines using wireless sensor networks [60] and pervasive surveillance sensor management architecture [61]. Melgoza-Gutierrez et al. [62] propose a collaborative learning protocol to share decision trees over vertically partitioned data. In a comparison of results with a centralized approach based on Weka, there was not a significant difference in accuracy, although the centralized approach showed faster time due to the reasoning cycle of each agent.

For agent-based development, the Java Agent Development Framework (JADE) has been highlighted as a Java-based platform often used in conjunction with the BDI model (BDI4Jade). Jason is an agent-oriented java-based programming language based on AgentSpeak, which is an agent-oriented programming language for based on logic programming and the BDI architecture for autonomous agents. These frameworks and languages have been researched in development of our implementation, however a custom solution has been developed for the purpose of this work to better fit our use case and extended capabilities.

2.2.2 Multi-Agent Collaboration

Multi-Agent Collaboration (MAC) involves the collaboration between multiple agents through predefined protocols to achieve a common goal. Multi-Agent collaboration goes beyond the concept of agents with independent goals, introducing shared goals across the system. In this model, the distributed nature allows for resiliency if an agent is lost or compromised. Traditional service-oriented architecture does not follow this concept. With a large number of agents all attempting to access the same central resource, this introduces a bottleneck and single point of failure. This collaboration occurs as a system initiative where agents operate in the background to achieve goals, ultimately decided by the user (user initiative) through an interface (i.e., the user is able to configure their security preferences). In addition to BDI, which is relative to an individual agent, a collaboration between agents also requires the establishment of joint intentions, shared plans, and planned team activity. Three theories of multi-agent collaboration as defined by Wilsker [63]:

- **Joint Intentions**: commitment to act in a certain mental state. When an agent adopts a notion not shared by the rest of the team, the agent must communicate the belief to the rest of the team.
- **Shared Plans**: shared responsibility towards other team members and performing individual actions for the achievement of goals and explicit communications requirements.
- **Planned team activity:** forming teams of agents to coordinate actions and tasks.

Organizational Structure

The organizational structure of a multi-agent collaborative system could hold the following:

- **Shared or partially shared perception/resources**: the perception of one agent is shared or partially shared with the other agents for a shared knowledge base.
- **Distributed and shared tasks and results**: complex tasks are broken into smaller parts and coordinated amongst agents. For example, cooperative distributed problem-solving can share resources amongst agents to complete a task quicker or more efficiently.
- **Synthesizing agents**: when a requirement is an input to a task environment, an agent can be automatically generated that will succeed in the environment.
- **Handling inconsistency**: the system must be prepared to deal with inconsistent beliefs or goals of agents and have a method for conflict resolution.

From the perspective of security, agents can have different responsibilities for security tasks in the smart home environment, breaking up large tasks such as scanning into smaller coordinated tasks amongst agents.

Collaboration Mechanisms

Regarding multi-agent collaboration, [63] studies multi-agent collaboration theories, including joint intentions, shared plans and planned team activity. Stergiu et al. [64] propose an XML policy-based framework for managing and modeling social interaction among agents. The framework includes roles, relationships, conversation patterns, and cooperation patterns for agent collaboration and is compatible with BOID (beliefs, obligations, intentions, and desires – an extension to BDI). The framework is implemented using JADE and can be used for peer-to-peer agent communication or agent-management services.

Further, multi-agent systems introduce a problem of decision classification. Xiao et al. [65] explore an effective solution to the multi-agent decision classification problem with a learning processing using Support Vector Machines (SVMs). Uhm et al. [66] propose a multi-agent system architecture for providing context-aware services in a smart home. This architecture uses an ontology-based context model and rule-based reasoning engine to identify the context of an environment and resolve conflicts between entities to provide context-specific services. The performance evaluations found that the ontology-based model offered better query response time than the ontology and rule-based combined to provide faster and more convenient services within the home. MAC architectures have been applied to smart grids and home energy systems by Kang et al. [67].

2.2.3 Applications in IoT

The common characteristics of complex, dynamic autonomous systems between IoT and agents have inspired a recent interest in the convergence of technologies in recent years. Agent-based computing (ABC) has been acknowledged as a comprehensive, effective enabler for cooperating, decentralized, dynamic, and open IoT systems [3], and further supports the vision that "research in the IoT is expected to shift from intelligent objects to objects with a real social consciousness" [68]. Sevaglio et al. present a comprehensive survey of state-of-the-art research in agent-based IoT, which indicates a strong conceptual alignment between IoT development requirements and (multi-)agent systems benefits, which has been exploited to drive and speed up the development of IoT systems. In particular, the survey identifies ABC as a promising paradigm for modeling, programming, and simulations of IoT environments [3] as further described below:

- **IoT Modeling**: the agent model naturally embeds IoT autonomic, proactiveness and situatedness, among other features which can be explicitly described through agent-related concepts.
- **IoT Programming:** agent-oriented programming approaches support uniform interfaces for heterogeneous resources and protocols, providing technical and syntactical interoperability of devices, as well as semantic operability by means of shared ontology and knowledge representation.
- **IoT Simulation**: agent-based simulation can enable verification and validation of individual and system-level emergent behaviours, protocols and performance of complex deployments of IoT ecosystems. These simulations can be further enhanced to integrate evolutionary game theory concepts to analyze cooperative patterns, dynamic processes, and macro emerging actions in the IoT scenario.

A notable approach to agent-based solutions within IoT integrates the use of microservices. Kravari et al. [68] introduce further commonalities of characteristics between IoT, multi-agent systems, and microservice architecture due to their distributed, autonomous, collaborative and goal-oriented nature. The authors apply this approach through a novel reputation-oriented trust model to support the challenge of intelligence and trustworthiness of IoT, using reputation estimation based on social principles, microservices combined with learning and adoption properties, and a distributed locating mechanism based on social graphs and peer-to-peer networks. Further, Rafalimanana et al. [69] adopt a collaborative agent-based approach to create a link between artificial intelligence and services choreography in IoT. The authors pair BDI-agents with Representational State Transfer (REST) service technologies to exploit the agent capabilities as a service.

Agent-based approaches have also been applied to smart home environments, as illustrated by [70], where "In the home environment, computer software that plays the role of an intelligent agent perceives the state of the physical environment and residents using sensors, reasons about this state using AI techniques, and then takes actions to achieve specific goals". There are several approaches to designing IoT architecture in a smart home environment, as defined by Roman et al. [71]:

- **Centralized***:* A centralized architecture connects the service to the user directly.
- **Collaborative***:* The IoT architecture consists of intelligent entities that exchange data.
- **Connected Intranets***:* segregated intranets connect to a central entity, with the possibility of also connecting to each other depending on the configuration.
- **Distributed***:* All entities can retrieve, process, combine and provide information or services to other entities.

Hilal et al. [61] propose an agent-based sensor management architecture for pervasive surveillance to support the coordination of sensor nodes and maintain situational awareness of the environment. The approach combines the advantages of holonic, federated, and market-based coordination architectures and models each node as an intelligent sensor using BDI. The architecture aims to address the design goals for scalability, flexibility, structured control, localized operation, and distributed autonomy within the system. It demonstrates higher effectiveness when compared with a centralized approach.

Holonic architectures have been a notable approach to the organization and collaboration of distributed multi-agent systems to achieve shared objectives. Ye et al. [72] provide a model for multi-agent holonic architecture for wireless sensor networks, including communication models, control and decision-making capabilities, and self-organization. Further, Pazzi et al. [73] adopt a holonic-model for cyber-physical systems, which provides modularity and hierarchy to control physical nodes while addressing the complexity of decomposing feedback loops, maintaining distributed invariants, and maintaining ongoing interactions with controlled entities.

2.2.4 Applications in Security and Adaptive Defence

A notable survey connecting the concepts of cybersecurity, intelligent agents, and IoT is provided by Coulter et al. [2], who identify that "the structure of an IoT environment sees communication and cooperation across many different system levels, while the evolution of computing structures requires adaptive and self-adaptive technologies to maintain affordable security." The authors illustrate the motivation for addressing the outdated integration of these domains for autonomous defence and discuss applications within the intrusion detection domain. A distributed agent model enables higher level reasoning through the network, where defence transcends independent layers and is achieved as a collective effort through knowledge-sharing and coordination. Further, the authors suggest that reflex and state agents can better utilize the benefits of machine learning through a more individualized approach to rule construction and state training. While goal, utility, and learning agents can benefit from natural, nature-inspired approaches.

Recent efforts of the North Atlantic Treaty Organization (NATO) to develop an Autonomous Cyber-Defence Agent (AICA) reference architecture [74] have been the most significant development in the literature toward an autonomous agent system for cyber defence. Although it has been developed independently of our work, we have found that it is very much aligned with our vision at a conceptual level. While not specific to IoT or BDI agents, the focus of NATO's work is to enable future defence actions on largely autonomous military assets where human intervention may not be possible. The authors present a concept of intelligent, autonomous, mobile agents specialized in active cyber defence with capabilities to monitor networks, detect malicious cyber activities, and destroy or degrade adversary malware. The architecture provides capabilities for autonomous planning and execution of multi-step activities, adversarial reasoning in response to intelligent, adaptive malware, and the ability to remain undetected through deception and camouflage capabilities. At this stage, the architecture has been provided only at a very high level with limited detail. A multi-phased roadmap for continued development over the next 9 years as further development in specific approaches for knowledge-based planning of actions, learning and negotiation, and multi-agent collaboration is planned for the future.

Adaptive and autonomous cyber defence is an emerging research topic bridging between AI and cybersecurity domains to create semi-autonomous cyber defences that can learn to recognize and respond to cyber attacks, discover and mitigate weaknesses in cooperation with other cyber operation systems and human experts [1]. Adaptive defence systems are able to evolve over time in response to changes in attacker behaviour, system health and readiness, and natural shifts in user behaviour over time [1]. Current works in this area include machine learning agent-based solutions for autonomous deception systems [75], attack simulation [76], penetration testing [77], and malware detection [78]. While the existing works demonstrate promising directions in applications of agent-based approaches to cybersecurity, they have been mainly focused on specific security capabilities rather than approaching the problem from a holistic point of view.

2.3 Chapter Summary

There are many layers of opportunity for security issues for IoT devices in smart homes. While the attack surface continues to increase as IoT devices become more prominent, security threats continue to increase in severity and frequency. While many have argued that manufacturers should provide security in the development of devices, this is not the current reality. Legislation and regulations attempt to improve security posture. However, enforcement of these policies across such a wide target is an extreme task. Although recent advancements are making some improvements, the complexity of this issue spanning across multiple domains does not have an indication of a near solution. While there have been many prominent contributions in the literature, the majority of the proposed solutions for smart home and IoT security are focused on a single purpose approach (DDoS, encryption, access control, intrusion detection) and do not consider the end-to-end security services for the IoT environment at a holistic level. Further, there is limited attention to embracing IoT technologies distributed, heterogeneous and data-centric architecture toward intelligent autonomous security services.

Agent-based modeling and multi-agent systems approaches are promising research areas for IoT and smart home systems with their ability to operate within complex, dynamic autonomous environments. With the unique architecture and increased need for autonomous reasoning, coordination, sharing, and analysis of data, a multi-agent architecture can be applied to achieve cybersecurity defence goals. While each of the domains of IoT, MAS, and security have rich collections of literature independently, the intersection of these topics provides an underexplored opportunity for enabling autonomous cyber defence within IoT environments.

Security for IoT requires an adaptive approach, where agent-based systems offer a flexible design for autonomous actions and goal-oriented decisioning. While agent-based solutions have previously been applied to the cybersecurity domain, there have been limited applications with barriers of adoption to the industry. The remaining chapters of this thesis will proceed to address these gaps with a proposed architecture for agent-based adaptive cyber defence to enable a modular and accessible framework towards a wide range of cyber defence capabilities. Domain knowledge graphs are leveraged to extend the traditional BDI model through a novel approach to support context-based modeling and agent reasoning of the cybersecurity domain, enabled by domain knowledge and frameworks available to the industry.

3 Multi-Agent System Architecture

This chapter presents our proposed multi-agent system architecture, which integrates autonomous defence capabilities into adaptive intelligent software agents situated to respond to the evolving cybersecurity threat landscape. Processes are modelled as multi-agent plans and tasks, where agents work together through a control and coordination hierarchy to achieve common goals to defend the network according to security requirements. Agents use the Belief-Desire-Intention (BDI) model to perform multi-agent goal-based deliberative reasoning for defence actions which are informed by domain knowledge graphs further described in Chapter 4.

3.1 System Architecture and Design

3.1.1 System Architecture Overview

The Multi-Agent System (MAS) architecture is composed of 3 main components: Security Services, Coordination, and Mission Deployment into the Internet of Things (IoT) environment. The high-level system architecture is shown in Figure 3.1 below, illustrating how the core model interacts with the IoT environment. Within each of these layers, agents perform operations, communicate with each other, and make use of the available resources throughout the system.

The Security Services layer is where high-level security decisions are made by control agents, making use of system observations, security requirements, and domain knowledge graphs to generate defence policies to be actioned by agents throughout the network. The results of these policies are utilized by the Coordination Layer, where coordination agents take the defence policy as input, identify security goals to be achieved and map them to actions with corresponding utility for prioritization. These goals and actions are planned and prioritized through workflow planning and coordination of available resources to generate subsequent missions, which are monitored by mission control.

Each mission consists of an action set, goal(s), prospective utility, and a set of agents with predefined beliefs, desires, intentions, and roles. Agents deployed through missions interact with each other in an agent collaboration environment as well as directly in the IoT environment to perform actions to achieve their mission objectives. An agent can interact with different components within the IoT environment, including devices, applications, and cloud services, either through API or directly hosted within the resource. We will continue to describe each of these components in detail in the remainder of this section.

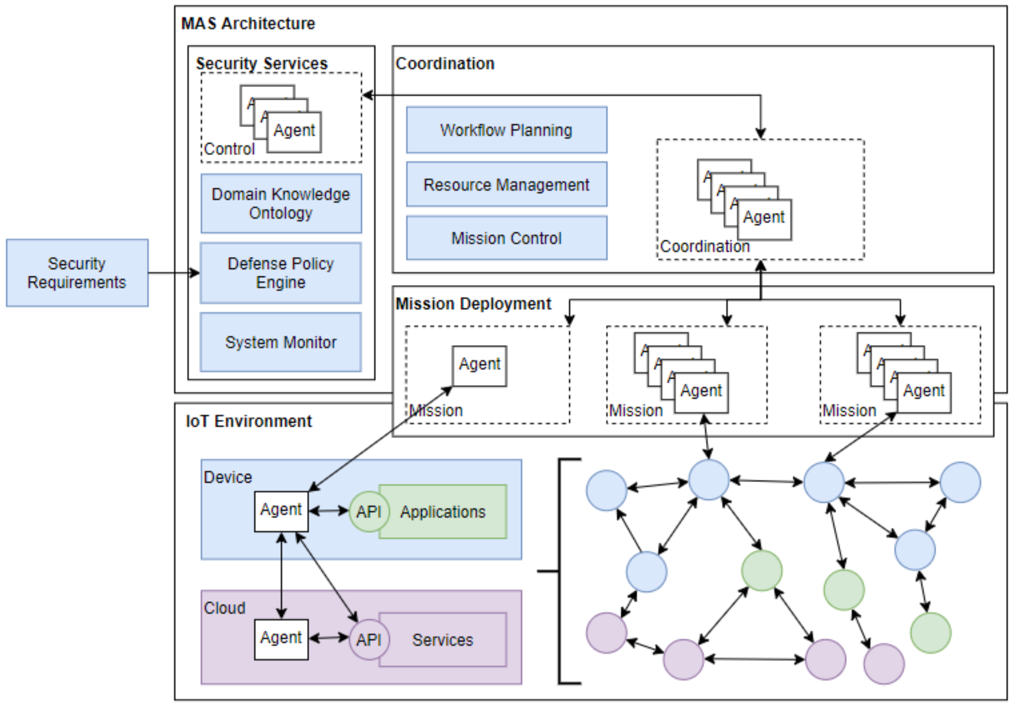


Figure 3.1 High-Level System Architecture

Security Services & Control

Control agents maintain an overall system view, leveraging the system monitor and defence policy engine to provide updated requirements to the coordination agents.

Defence Policy Engine

The Defence Policy Engine takes security requirements as input from industry standards, vendor policies, and user preferences in a common format. These requirements are combined with observations from the system monitor and ontological domain knowledge to generate context-aware policies with respect to availability, coverage, and exposures. Together these form the defence policy used by the MAS control agent(s) to generate security goals and corresponding plans. If any changes are made to the requirements, the defence policy will be updated as necessary. The policy engine is described in detail in Section 3.2.3.

System Monitor & Environment Graph

System observations are shared resources used by the agents to make informed decisions on courses of action based on the system security state. These observations are maintained within the system monitor and are separated into four categories: operational availability, coverage, exposures, and attacks. These are used as parameters in system state calculations and the utility function for generating defence policy rules, missions, and their respective payoff. The exposures and attack monitors maintain an inventory of known vulnerabilities and exposures, suspicious activity, and active attacks observed on the network. These indicate negative system security states that must be remediated through appropriate security coverage. The coverage monitor maintains visibility of countermeasures in place, including access control rules, security controls, active missions, etc. Within coverage, situational awareness is also tracked to ensure the system monitor has appropriate visibility into the network. Lastly, operational availability is also monitored to ensure that IoT services are operational and experiencing minimal impact due to security controls or attacks. The security parameters within the system monitor and applications for system utility are described in detail in Section 3.2.3.

Environment data, including known assets, device capabilities, and associated attributes, are tracked within an environment graph for ongoing situational awareness of the network. The environment graph can be overlayed with the domain knowledge graph for informed decisions and policies. The environment and domain knowledge graphs will be elaborated on in more detail in the following chapter.

Domain Knowledge Graph

To understand security threats and corresponding defence actions, security domain knowledge and intelligence is used in a structured framework for relating entities and mapping relevant threat libraries. The domain knowledge ontology is described in detail in the following chapter on knowledge graphs.

Coordination

Workflow Planning

The workflow planner maps items from the defence policy to options for action selection and corresponding utility. Once these are defined, these options are prioritized and defined into missions.

Resource Manager

The resource manager works with the workflow planner and mission control to identify required resources for a mission, maintains an active inventory of available resources, and supports the organization model templates for agents to be assigned to missions. This includes role descriptions, action sets, as well as initial beliefs, desires, and intentions.

Mission Deployment

Mission Control

The mission controller is created for a specific mission to oversee the deployment and monitoring of the mission to its success. The mission controller is responsible for ongoing communication and re-evaluation of mission requirements if necessary. Once a mission is completed, the mission controller provides an evaluation of the success of the mission and reports back with the results and utility.

Agent Collaboration Environment

When missions are created and agents are deployed, an agent collaboration environment is established where agents can interact and collaborate with each other to achieve their goals. This is also the interface where agents are deployed to each layer of the IoT environment, as well as a proxy to shared agents' resources.

IoT Environment

Agents can interact with the IoT environment through API connection or deployed and directly hosted on the device, application, or cloud resource. From this layer, agents can perform a variety of actions by interacting with the IoT environment for activities such as data gathering, remediation activities, or control actions. The three components of the IoT environment are described below:

Device

The IoT devices exist in the device layer, comprised of the embedded device, sensors, radio communications, software, and firmware. Our model interacts at the firmware level through an open firmware architecture for the agent microservices to execute. Devices can also include edge devices such as IoT hubs or controllers and the network gateway. The IoT hub devices act as a bridge and controller for IoT devices across the network. While most IoT devices have limited processing and storage capability, the hub provides additional capabilities for coordinating these services. The network gateway can provide basic security functionality through firewall and intrusion detection capabilities. We provide an interface for agent interaction with the devices as this layer.

Application

The application layer provides mobile or Web application services to IoT devices. User-facing applications also provide capabilities for personalized configurations and settings. We introduce an Application Programming Interface (API) at this layer for agent interaction.

Cloud

The cloud layer is where many resources and processing capabilities can exist for vendor provided services. This also provides an opportunity for accessing services and analytics. Device or service vendors must also provide internal testing and support for security concerns through our model. Our model can also access these capabilities through an API at this layer.

3.1.2 Agent Hierarchy

A hierarchical agent structure is used for organization-level insights and emergent behavior in agents across the network to achieve holistic security goals. In this section, we continue to expand on our model for the generation of agents to perform the actions selected by the controller. Section 3.3.2 describes the behaviors of the MAS controller (Level 0) as a strategic defensive BDI agent which generates defence policies (intentions) based on beliefs and desires for an ideal security state. As shown in Figure 3.2, this results in a hierarchical BDI agent structure where the coordinator generates a high-level intention which is the template for the creation of sub-agents (Level 1+) with corresponding desires.

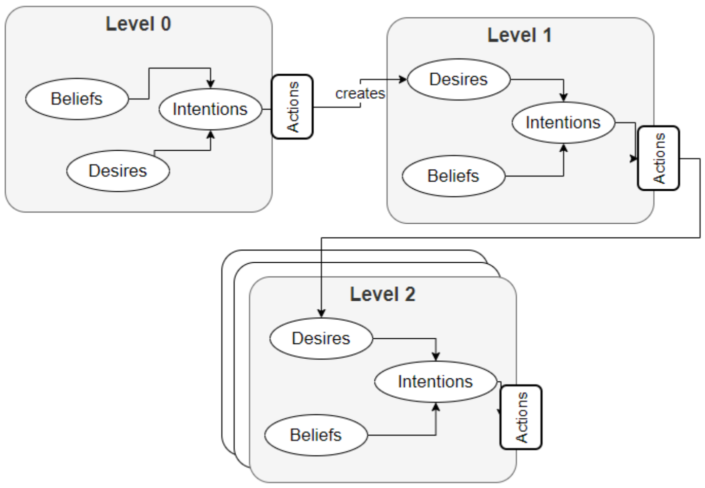


Figure 3.2 BDI Hierarchy

Coordination mechanisms allow coordination between the controllers and coordinators to allocate resources according to the requirements. Taking as input the profile of the defenders θ, attack types a, and resources available k, a coordination mechanism function $\pi : (\theta, k, a) \rightarrow (x, t)$ is generated, which outputs a strategy x for the target t.

3.1.3 High-Level Algorithm and Data Structures

While all decisions and sensory aspects of the system are performed by Belief-Desire-Intention (BDI) agents of different functions, Figure 3.3 below shows how initial security requirements are inherited as desires by downstream agents with the ability to perform required actions accordingly, where the security requirements are first received by the controller to create the defence policy according to the knowledge of the environment from the system monitor. The defence policy translates into agent desires, in which the coordinator performs mission generation through planned workflows and resource management. Missions are created to assign associated desires and functions to capable agents deployed within the environment to achieve the overall desires. Each of these functions will be defined in further detail in the following sections.

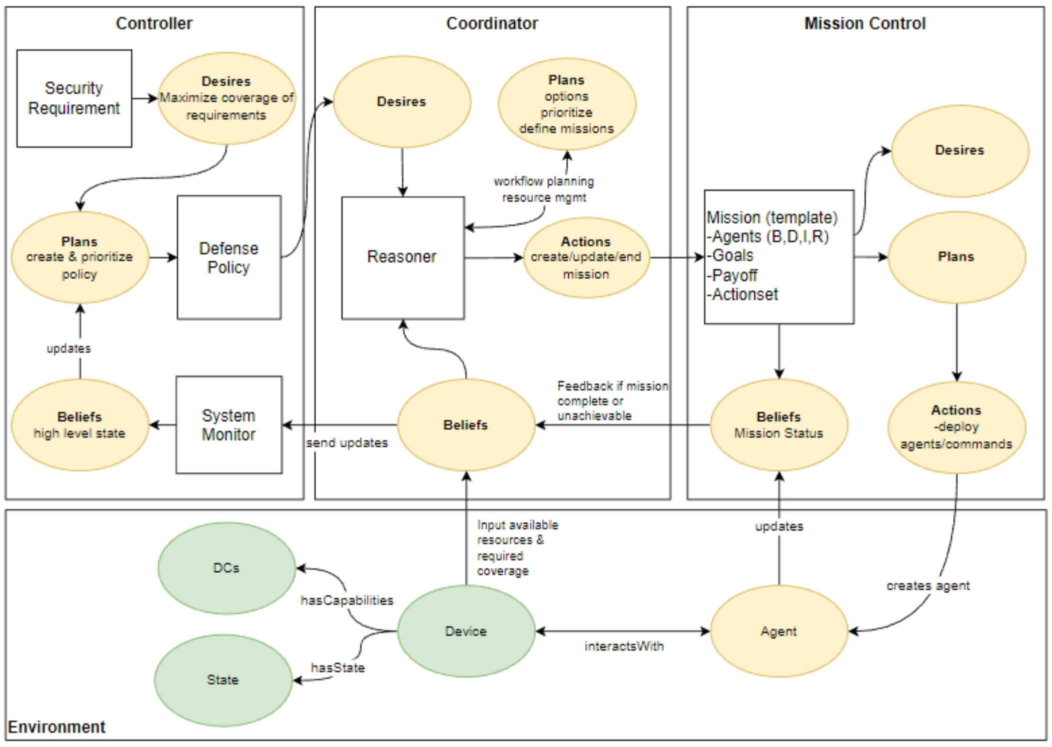

Controller
Security Requirement
Desires Maximize coverage of requirements
Plans create & prioritize policy
Defense Policy
updates
Beliefs high level state
System Monitor
Coordinator
Desires
Plans options prioritize define missions
workflow planning resource mgmt
Reasoner
Actions create/update/end mission
Beliefs
send updates
Feedback if mission complete or unachievable
Mission Control
Desires
Mission (template) -Agents (B,D,I,R) -Goals -Payoff -Actionset
Plans
Beliefs Mission Status
Actions -deploy agents/commands
Input available resources & required coverage
updates
creates agent
DCs
hasCapabilities
Device
interactsWith
Agent
State
hasState
Environment

Figure 3.3 Multi-Agent BDI Inheritance of Policy Desires

Modeling the environment in a way that can be understood and reasoned by the agents is critical to situational awareness and understanding of environment states for the agents to act upon. As the system leverages environmental contextual data, in combination with domain knowledge through industry frameworks, we have provided a data structure diagram below in Figure 3.4 to show the relations between different data elements. This data model will be utilized and further expanded in the following sections and chapters.

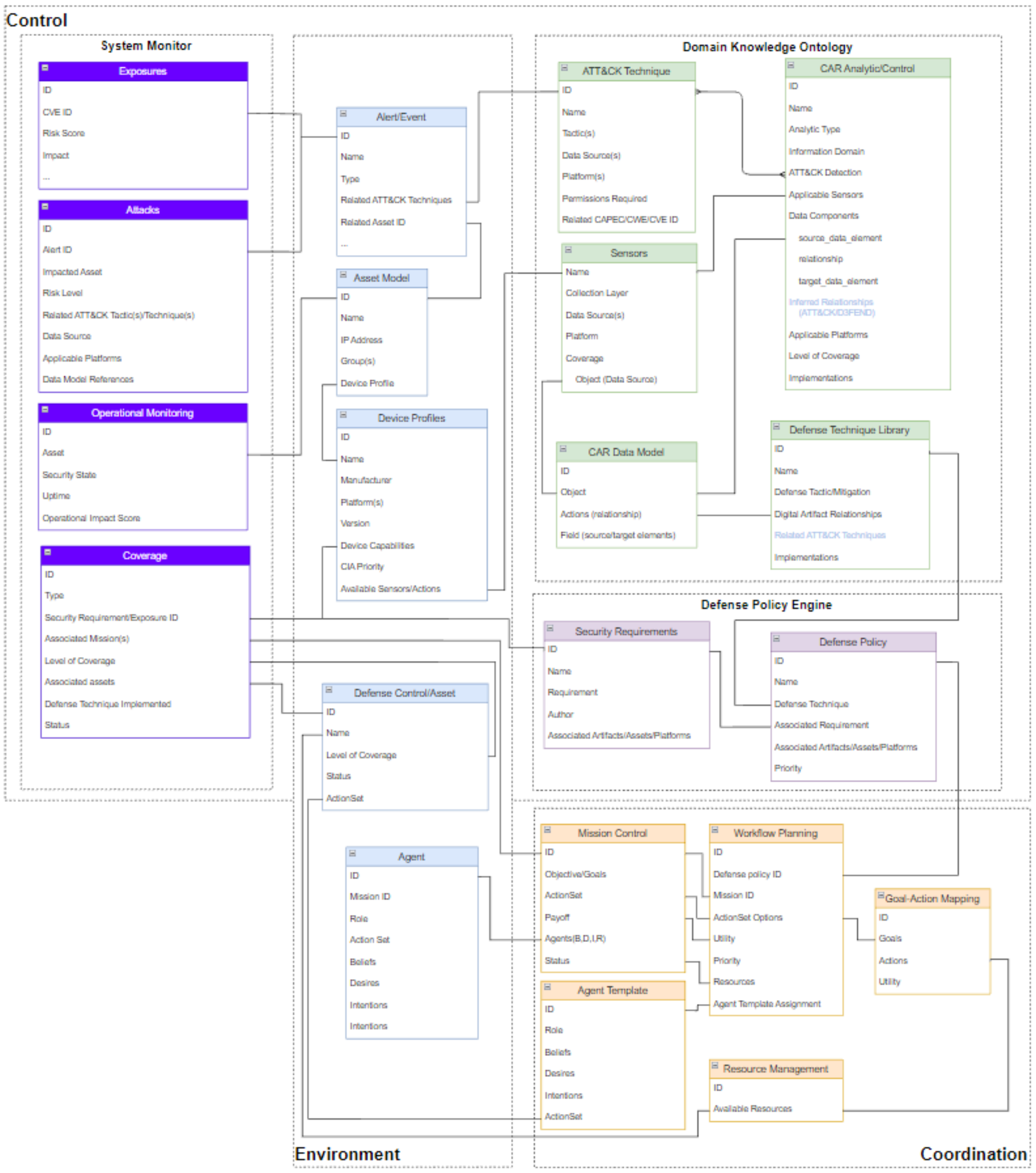


Figure 3.4 Data Structure Diagram

3.2 System Control and Policy Generation

In a common format, security requirements can be defined by industry standards, vendor policies, and user preferences. These requirements are combined with observations from the system monitor and domain knowledge graph to generate context-aware policies with respect to availability, coverage, and exposures. Together these form the defence policy used by the MAS control agent(s) to generate security goals and corresponding plans. If any changes are made to the requirements, the defence policy will be updated as necessary.

Our knowledge graphs provide a data model for security requirements to be interpreted by a policy engine according to the context of the environment and inferences to cybersecurity domain knowledge. The policy engine generates the policies by validating the security requirements provided as input. Through the hierarchical agent model, multiple layers of policy types can be maintained. The first is the device level policy, which applies to each individual device within the network. The second is the domain-level policy, which may apply to grouping devices in different domains. These can include domains of similar devices, individual network segments, roles, or groups of agents participating in a particular mission. The third is the system level policy, which inherits and synthesizes the requirements of each individual device into the context of the entire system. These policies are used by the controller in the prioritization and design of agent plans and missions. The policy engine performs top-down and bottom-up validation to achieve compatibility of policies and negotiate any conflicts. Conflicts will indicate contradictory policies or policies which are unachievable within the current configuration. If a conflict is identified, the user will be notified.

Once the policies are created, they are used to define the goals of the system that the agents work towards achieving. Policies drive the decision-making of the system as enforced through utility, while requirements are mapped to the categories of security state parameters defined in the following section and can include an agent reward or penalty value for compliance or non-compliance, respectively, to influence prioritization and strategy selection. Violations of policies will result in a penalty as well as an appropriate response which may involve user notification or intervention steps.

This section defines the functions of the defence policy engine and how policies are generated to address security requirements based on the domain knowledge graph and situational awareness through the system monitor.

3.2.1 Security Requirements

Baseline Requirements

While our overall architecture can be agnostic to a specific set of requirements, we have selected to reference National Institute of Standards and Technology (NIST) recommendations as a baseline for demonstration. NIST IR 8228 defines a set of IoT Risk Mitigation Goals which we have leveraged as a baseline for initial security requirements. These include Protect Device Security, Protect Data Security, and Protect Individuals' privacy. As the focus of this research is on the security domain, the risk mitigation areas for *Goal 3: Protect Individuals' Privacy* are currently out of scope; however, the model can be extended for privacy applications as a potential future work. A Presidential Executive Order on Improving the Nation's Cybersecurity (14028) [25] published in May 2021 directed NIST to develop two labeling programs on cybersecurity capabilities of IoT consumer devices and software development practices. While these labeling baselines align with the NIST Interagency/Internal Report (NISTIR) 8228 risk mitigation goals, this enables an opportunity to validate security requirements against labeled devices within the network.

We have built on NISTIR 8228 and NIST 8259A [19] [26] to develop a high-level baseline set of security requirements for a smart home IoT network, as defined in Appendix A. The requirements for asset management, device configuration, data protection, access management, vulnerability detection, incident detection, and availability are mapped to a set of corresponding Device Capabilities (DC) defined by NIST 8259A. Through the device, the label can be determined if each device can achieve the defined security requirement. Further, each requirement also maps to a corresponding set of security properties (confidentiality, integrity, availability, authenticity, and non-repudiation).

Device-Specific Requirements

While the above can be used as generic requirements, further granular requirements can be developed specifically for the unique security needs of certain devices according to the prioritization of certain properties of labels or individual user configurations. For example, when considering a smart lock's unique needs for availability and integrity, these corresponding requirements can be given higher priority. Priority-based thresholds can be configured according to user preference and environment-specific needs. Thresholds according to an individual device, type of device, area of a network, or other groupings can be recommended or configured by users. For this initial demonstration, we have added security property prioritization to the device profiles to provide customization according to the type of device.

Requirements Definition

The security requirements will be defined in a common format to allow for processing into the defence policy. Table 3.1 below shows the elements of the baseline security requirement definition format. This concept will be expanded upon in Chapter 4 on BDI and knowledge graphs.

Table 3.1 Baseline Security Requirements Definition Format

Field	**Data**
ID	int
Name	string
Requirement	string
Priority	[1-5]
Associated platforms	Global |Group | Device Profile | Device
Desired state	(“Subject”, “State”)
Associated Security Properties	Confidentiality | Integrity | Availability | Non-Repudiation | Authenticity | All
Associated Device Capabilities	DC […]

The security requirements can be in any common format, such as JavaScript Object Notation (JSON) or eXtensible Markup Language (XML). JSON was selected for demonstrative purposes of this work. For illustration, Figure 3.5 below provides an example of the Vulnerability Management requirement, which is applied globally to all devices on the network. The requirement includes the associated device capabilities which can achieve it, as well as the desired state of "Device Version is Up to Date," which will define the target state for the agents' inherited desires.

```
{
      "ID": 004,
      "Name": "Vulnerability Management",
      "Requirement":   "Identify   and   eliminate   known
vulnerabilities"
      "Priority": 1,
      "Associated Platforms": "Global",
      "Desired State": "Device Version = Up to Date",
      "Associated Security Properties": "All",
      "Associated Device Capabilities": ["DC5.1", "DC5.2",
"DC5.3"]
}
```

Figure 3.5 Security Requirement Example - Vulnerability Management

3.2.2 System Monitor

The system monitor maintains situational awareness of the network and is used by the controller and coordinator agents as a reference for policy updates and prioritization to achieve optimal coverage of the security requirements while maintaining an ongoing view of the overall system's health. The system monitor consists of four components: Operational Monitoring, Coverage, Exposures, and Attacks.

The system monitor components are updated through agent messages and queries within the environment graph and enriched by the domain knowledge graph. While each of the four components maintains an ongoing view into different elements of the health and security state of the network, the data can be leveraged in a granular view to inform contextual decisions regarding individual devices, as well as a high-level summarized view for tracking overall system state. This section defines each

System Monitor component that will be leveraged by the multi-agent system in the following sections.

Operational Monitoring

The operational monitor is used to track services' availability and devices' overall security state. The asset model referenced by the operational monitor is an inventory of assets registered to the network, as well as any that are not registered but have been detected through other agents. The asset model exists within the environment graph and will be further described in Section 4.2

The operational monitor maintains entries for each device with the below fields:

- **Asset ID**: An asset model is maintained separately by the agents and used as input into the operational monitor.
- **Security State**: the security state of the device is updated according to the associated device's coverage, exposures, and attacks. Security states, as defined below, include initialization, disabled/failed, normal, vulnerable/suspicious, and exploited.
- **Availability**: tracks the operational status of a device for availability, if it has been unreachable or disabled

The system can be categorized into the following security states, as shown in Figure 3.6. This can be assigned to each device, network segment, or entire system state. The state of the system will reflect the risk level and type of missions in place.

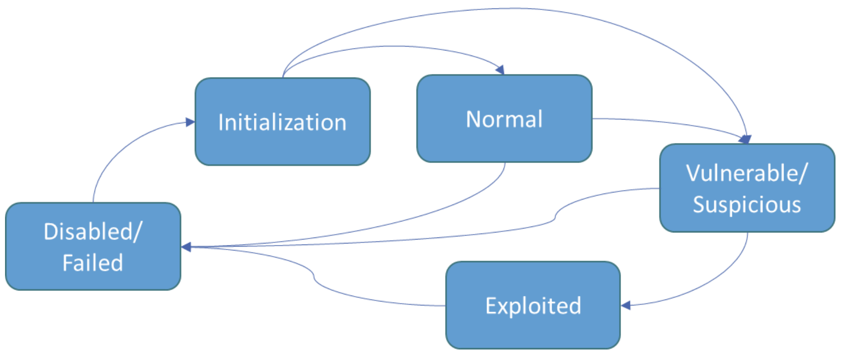


Figure 3.6 System Security States

(0) The *initialization* state has limited knowledge of the true security state of the system. It will be the most active in terms of processing in order to achieve coverage, initial configurations, patching, and coordinating missions. When a new device is added to the network, it will also join in an initialization state.

(1) Once initialization is complete and any security requirements have been satisfied, the system will move to a *normal* state. In this state, monitoring and situational awareness missions will be in place to gather baseline information as well as monitor for suspicious behavior or new devices. If free processing cycles are available, this can also be used to fortify controls and projective scenario strategy calculations.

(2) The system will move to this state if a vulnerability or suspicious event is detected. Missions will be established to patch, project attack paths, countermeasures, and advance monitoring of high-risk devices.

(3) In the event of an attack being detected within the system, it will transition to the exploited state. In this state, remediation missions will be in place to contain the attack and enforce relevant countermeasures. Forensic data may also be collected, and the user will be notified.

(4) A device may also be in a disabled state, where it is offline or unreachable.

Coverage

The Coverage Monitor tracks security controls in place in relation to security requirements and compensating controls for exposures/attacks. This is leveraged by the controller to track overall coverage to inform the policy prioritization. The below fields are tracked by the coverage monitor:

- **Security Requirement / Exposure ID**: a mapping to the security requirement or exposure that the coverage is designed to address.
- **Associated Mission(s):** a list of missions in place.
- **Level of Coverage:** an indicator of the level of coverage for.
- **Associated Assets**: the assets protected by the control.
- **Defence Technique Implemented**: mapping to defence technique or mitigation

- **Status**: current status of the coverage (active, planned, disabled).

While some devices may be required to comply with a certain security requirement without having corresponding device capabilities to achieve it, compensating controls will need to be put in place. Once the controller has situational awareness of the limitations and capabilities of the network, appropriate missions can be deployed to provide coverage, as further described in Section 3.3.1.

Exposures

The exposure monitor tracks known vulnerabilities and configuration risks within the network, as well as an understanding of the risk associated with the exposure. The domain knowledge ontology is used for enriching exposure data based on Common Vulnerability Scoring System (CVSS) [79], which will be further expanded in the following chapter. While exposures are known vulnerabilities that have not been exploited, if an attack is detected targeted in the exposure, it would be listed in the attack monitor. Each exposure should be prioritized by the controller for coverage according to the level of risk. The exposures monitor contains the following fields:

- **Type**: the type of exposure as Common Vulnerabilities and Exposures (CVE) [80] or configuration risk
- **CVE ID**: the CVE associated with the exposure.
- **Risk Score**: as listed in CVE.
- **Impact**: the impact of the exposure as listed in CVE (confidentiality, integrity, or availability).
- **Exploitability**: as listed in CVE (privileges required, attack vector, user interaction, scope).
- **Associated Assets**: the assets affected by the exposure.
- **Status**: tracking of the status to indicate whether the exposure is active or remediated.

Alerts

The alerts monitor tracks alert indicating suspicious or malicious behavior on the network to be investigated and/or remediated. Alerts are created within the monitoring controls and analytics and are further described in Section 4.2.2. The alerts monitor contains the following fields:

- **Alert ID**: unique ID for the alert.
- **Impacted asset(s):** the assets affected by the alert.
- **Risk level**: level of risk determined by the domain knowledge graph.
- **Related attack technique(s):** the MITRE ATT&CK [81] data associated with the alert.
- **Data source**: the data source telemetry that the alert originated from.
- **Applicable platforms**: the type of platform associated with the alert.
- **Data model references**: object, actions, and fields associated with the CAR data model.

3.2.3 Policy Engine

Once the requirements have been defined, a Defence Policy is generated and prioritized by the control agent based on the security requirements and knowledge of the system security state. This policy provides the high-level security objectives to be performed and passed down to the coordination layer in the form of agent desires.

Utility Function

The utility functions are used to guide agent actions by providing rewards and penalties as feedback for reinforced emergent behavior towards the desired system state. In this case, the system utility is used to build an effective strategy profile for the particular environment at a given time step. Utility functions require feedback in order to reinforce the behavior of the agents. This feedback can be collected either from environment artifacts or as a response from another agent. The requirements of an effective utility function are described below:

- **Consistent**: Utility function must be consistent across similar agent types to ensure unified behavior sets.
- **Attributable**: Feedback must provide timely and clear attribution for a particular action to reinforce the intended behavior.
- **Goal-Oriented**: The function must accurately reflect the goals of the system and be evaluated to ensure it will not cause unanticipated agent behaviors that contradict the ultimate security goals in order to receive rewards.
- **Contextually Scalable**: Changes in system states must be taken into consideration. Learned behaviors in a normal state will likely not follow the required risk levels while operating within a system in a vulnerable or exploited state.

We define a system security state utility function in Equation (1), which maintains the overall security posture of the smart home and provides incentive/feedback for mission selection in accordance with the system security goals. This function is maintained at the controller level with visibility into the environment. The utility is calculated as a high-level function of the security policies' adherence within the security parameters described below, taking into consideration the operational availability (OPS) and coverage (COV), with respect to the level of exposures (EXP) and attack detections (ATK).

$$Utility = \sum_{p \in P} [OPS_p + COV_p] - \sum_{p \in P} [EXP_p + ATK_p] \qquad (1)$$

The utility consists of the parameters defined by the System Monitor, which are quantified and mapped to the predefined security requirements to issue individual penalty and reward values for violation or compliance, respectively. These are maintained in the "system monitor" of the logical architecture. The values of the security parameters provide context to translate observations into the security state of the system and each device within it. While the defender aims to maintain a "normal" security state, the parameters also provide additional granularity and quantifiable feedback to determine the utility of defence actions.

Policy Generation

Our model adopts a control-theoretic approach for the defender to maintain a secure state of the system. The controller is a strategic rational BDI agent acting as the “defender,” which aims to maintain the system's preferred security state while considering the observations, defence capabilities, and associated utility. In this model, we represent a control problem with one strategic rational agent as the defender, while adversarial actions are considered non-strategic events (nature). The following section describes the foundations of this model, with the relations further illustrated in Figure 3.7.

The **system state** $s_t \in S$ is used to quantify the security state of the system at time t, where the set of system states $S = \{s_0, s_1, \ldots, s_n\}$ is defined as per Section 3.2.2. A **state transition** is represented as $s_t \rightarrow s_{t+1,}$ which evolves as a function of defender actions and environment events f_t: S × A × E → S. Given an action-event pair of event $e_t \in \mathcal{E}$ and defender actions $a_t \in A$, the state is updated to $S_{t+1} = f_t(s_t, a_t, e_t)$. For this thesis, state transitions are defined as Non-deterministic Finite Automaton (NFA). The limitation of this initial design is that all successor states are considered equally possible as opposed to probabilistic reasoning. This can be enhanced through future improvements to increase the probability of interrelated states based on observations.

The **set of events** is defined as $\mathcal{E} = \{e_1, e_2, \ldots., e_n\}$, where an **Event** $e_t \in \mathcal{E}$ is an observable event from the environment (nature) and is generated from a possible set of events given the system state $\mathcal{E}(s_t) \subseteq \mathcal{E}$. While defenders maintain an incomplete view of system states as all events are not able to be observed, **defender observations** $o_t \in O$ are defined where observation $o_t = O(s_t, e_t)$ is generated as a function of the true underlying system state and the event. Agents are aware of their state at all times and can maintain a **history of observations** based on their actions and observed state as $h_t = (a_0, o_1, \ldots, a_{t\text{-}1}, o_t) \in (A \times O)^t$. The history can then be compressed into **belief state** B_t for making optimal decisions based on historical observations. A new observation will result in **belief update** $b_t(s_t) = p(s_t|b_{t-1}, O_t)$, which is the probability of the system state being s_t given the previous belief and new observations.

In alignment with the system security goals, the defender maintains a set of desires DE $\in S$ for the aspired secure system state, which influences their decisions. **Defence action** $a_t \in A = A^1 \times A^2 \times \ldots \times A^n$ at time t represents the actions that a defender can perform to inhibit an adversary's actions. Each action space A^i consists of a finite set of possible defence actions $A^i = \{a^{i,1}, a^{i,2}, \ldots, a^{i,n}\}$, where each action affects the state s_i of element *i*. Each defence action a_t has an associated **utility** $c(s_t, a_t) = \sum_{i \in N}[OPS_i + COV_i] - \sum_{i \in N}[EXP_i + ATK_i]$, which the defender is provided through a feedback observation. The utility is represented as a function of the value of the security parameters defined in Section 3.2.2.

Defence policy $d = (d_0, d_1, \ldots, d_{T-1})$ contains the action(s) the defenders will take to change system states to the objective system state (desires), where d_t is the function from given belief state B_t to a distribution over defence actions d_t: $S \rightarrow \Delta(A)$. The **optimal defence policy** $d^* = (d^*_0, d^*_1, \ldots, d^*_{T-1})$ can be generated according to the defender’s risk tolerance, for example, using ‘minmax’ criterion to minimize the worst-case cost. Optimizations through approximation, sequential decomposition and dynamic programming have been applied in the literature.

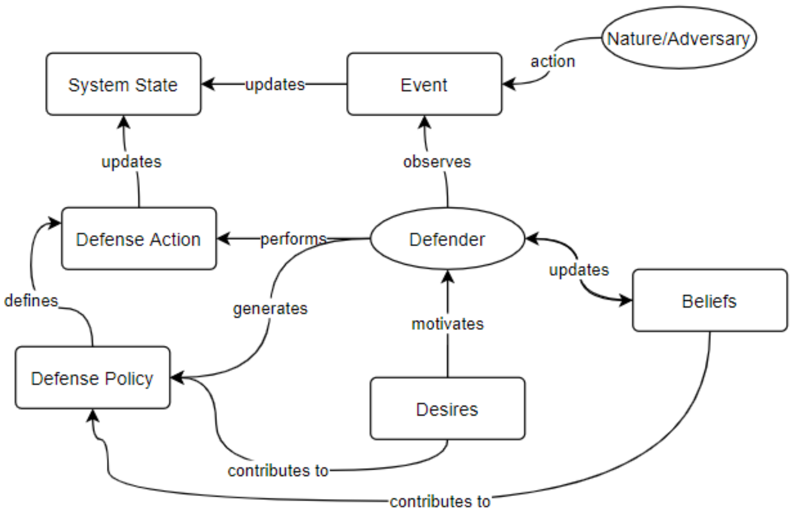


Figure 3.7 Relational System Model

In this scenario, the defender is the controller of a larger system of agents and security resources that can be utilized. While definitions in this section are sufficient for the base model, we will expand on the defence policy and actions in further detail below.

Policy Definition Format

The policy engine maintains a translation of security requirements into agent desires, which are then inherited by the coordination agents for further action. A defence policy entry contains the following fields:

Table 3.2 Policy Definition Format

Field	**Data**
ID	int
Name	string
Defence Technique	string
Priority	[1-5]
Associated artifacts/assets/platforms	Global |Group | Device Profile | Device
Associated Requirement	string

Similar to the security requirements definitions, policies are also maintained in a common format such as JSON. Once a policy is created, new agent "desires" will be created, and the controller agent will send a message to the coordinator(s) to inform them of the policy change for further action.

3.3 Agent Modeling for Cyber Defence

This section expands further on the concept of agent modeling in relation to cyber defence techniques, where a hierarchy of agents is deployed across the network to coordinate actions to perform the security goals defined by the policies. While this section describes the agent structure at a high level, the following chapter will go into further detail on agent behavioural modeling and reasoning through BDI knowledge graphs.

3.3.1 Mission Generation

A mission is a coordinated action set defined in accordance with the security strategy and requirements. Missions are generated and prioritized at the coordinator level, where they can be negotiated with other coordinators that may exist within the system. Each mission has prospective payoff values for increasing/decreasing each utility parameter, mapped to a particular reference ID, which is used to determine the prioritization of plans, and validated on completion. After completing each mission, the controller will be notified, receive the payoff, and cascade it down to the agents involved in the mission.

Once a mission has been selected, the coordinator will deploy agents assigned to specific roles to perform the mission tasks. Missions are composed of one or more action sets assigned to one or more agents. These missions can be categorized into Specifically defined (e.g., patch device *x*), Open ended (e.g., scan network to collect baseline data), Cooperative shared (e.g., agent *x* and y will work together to scan), Cooperative distinct (e.g., analysis agent will continuously send relevant data to response agent), or Hybrid.

3.3.2 Multi-Agent System Preliminary Definitions

Moving on to the detailed components of the multi-agent system model, we proceed to define the below:

- **Multiagent System** = {A, E, O}: is the system comprising all Agents, Environment states, and Organization relationships.
- **Agents** = {ag_0, ag_1, …, ag_N}: is a set of n agents in the system. An agent is defined as an autonomous entity that makes decisions and actions based on rules based on independent goals and perceptions.
- **Environment** = {e_0, e_1, …, e_N}: is a set of n environment states. The physical or logical environment is shared by all agents in a system and contains artifacts that can be perceived and impacted by agent actions.

- **Organization** = {o_0, o_2, …, o_N}: The rules through which agents interact with each other, work together, or resolve conflicts. Organization o = {R, G, N, SP} comprises roles r, groups g, norms n, and social plans sp.

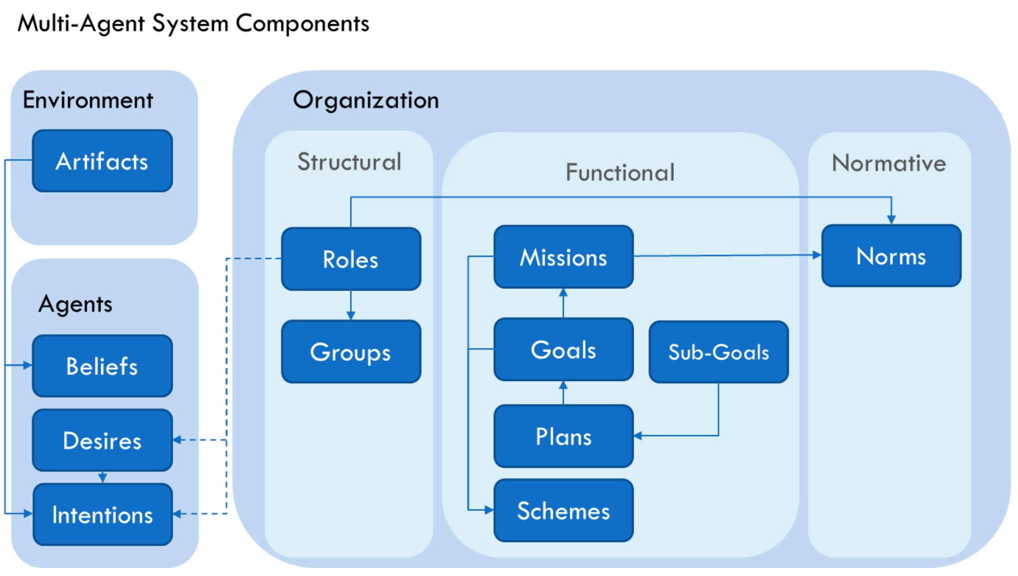


Figure 3.8 Multi-Agent System Components

Roles

Agent roles are defined for the consistent inheritance of objectives and capabilities. Roles commit an agent to specific obligations and may include associated permissions or prohibitions. Roles define the responsibilities of the agents assigned to the role. We define two categories of roles, which can be extended depending on requirements. These roles provide a core model for security capabilities within an environment and are scalable to various use cases.

Control-Based Entities

The second type of role is for the control of the MAS. These entities perform tasks to maintain the health of the system and to ensure effective decision-making. Some example control-based entities are listed below:

- **Controller**: performs high-level rationalization on the security state of the system and manages security policies (as described in Section 3.2).

- **Coordinator**: receives a goal from the controller and gathers necessary resources to create missions and deploy agents.
- **Mission control**: oversees missions involving multiple agents with a common goal.

Security-Based Entities

We define the below agent roles within the system to function with different capabilities to perform distinct security functions. These roles have been developed to enable the range of capabilities defined by the NIST Cyber Security Framework (identity, protect, detect, respond, recover), as will be further described below. The following parent roles provide archetypal functions, which can further be broken down into sub-roles for more specific functions.

- **Sensor**: sensors to collect data across the network, i.e., monitoring, scanning, and sharing.
- **Analyzer**: process data for uses such as anomaly detection, risk assessment, reputation services, and policy generation; Input processed data to detect attacks and malicious activity.
- **Investigator**: request gathering of additional data and perform further analysis on the reported attack. Determine if an activity is malicious; Escalate to a responder.
- **Responder**: perform responsive action in response to malicious activity or trigger, i.e., policy enforcement, patching, access control, network isolation, traffic filtering/limiting, firewall rules, and configuration update.

An example role description of a Sensor agent is shown below, where the objective of the agent to scan environment *env* is fulfilled through sub-objectives utilizing its capabilities such as scan element *e*, and update environment graph *db*.

Table 3.3 Sensor Agent Role Description

Id	**Sensor**
Objectives	Environment_scanned(env)
Sub-Objectives	Scan(e), update(db), …
Capabilities	Scan(), query(), report()

3.3.3 Communication and Policy Enforcement

While agents can interact with all layers of the IoT network, all communications are established through a secure agent channel providing an additional layer of security assurance. As shown in Figure 3.9, all interactions are facilitated through an agent interface, which validates and performs actions on behalf of the requestor. Through this architecture, an agent can be deployed to be hosted on any IoT device, act as a Policy Enforcement Point (PEP), and perform sensory and response actions through the Application Programming Interface (API). This architecture allows for high levels of insight into network and device events to support security monitoring and analytics, as well as an additional layer of control for the enforcement of access control policies. With further validation, this channel can also prevent attacks such as agent spoofing and replay attacks. Requests from a device are facilitated by agents according to trust and risk levels calculated by the security parameters and utility function at the Policy Decision Point (PDP) at the controller resources. For example, if a device has certain attributes, such as active vulnerabilities, only limited access is provided to it, and other devices are accessing it. This allows for the containment of potential exploits until a patch is issued while allowing basic availability requirements as per user needs.

In addition to the requests and response messages proxied through an agent, devices maintain access control policies to allow access to resources according to the role of the agent. This ensures that the least privilege is assigned according to the required permissions of the agent accessing the resources. For example, agents assigned to security and administrator level roles will have certain permissions granted, while operational activities will maintain a lower level of permissions to achieve their requirements.

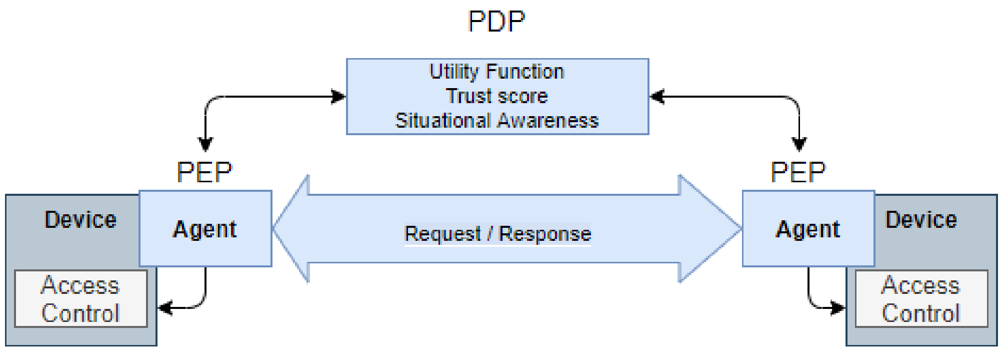


Figure 3.9 Communication and Policy Enforcement

3.4 Chapter Summary

Agent-based technologies provide autonomous, adaptive, cooperative goal-oriented behaviours which can be leveraged to address the unique cybersecurity challenges of an IoT environment. The multi-agent architecture presented in this chapter provides an extendible framework for enabling the coordination of agents deployed across an IoT environment to achieve security goals. The hierarchical structure allows for shared objectives aligned to security requirements, which can be coordinated across the multi-agent system based on environment states, resource capabilities and ongoing requirements where autonomous and coordinated actions are informed by partial and full views of the environment at different layers.

While the architecture shows the overall system capabilities and design, there is a requirement for a strong data model to inform agent reasoning and deliberation based on an understanding of the environment and inferences within the cybersecurity domain context. The following chapter will introduce knowledge graphs as a solution for modeling the system environment along with cybersecurity domain knowledge and BDI agent reasoning, which will be implemented by each agent within our architecture for contextual and goal-based decisioning.

4 Knowledge Graphs for BDI Agent Reasoning

The system architecture and services described in the previous chapter require a comprehensive solution to model and analyze the large volume of environmental data telemetry as well as the agent rationalization and rules with context to cybersecurity domain knowledge. In this chapter, we introduce knowledge graphs as a solution for the below functions:

- To model the various entities and states within the environment.
- To inform decisions with cybersecurity domain knowledge.
- To model BDI agent behaviours and inference rules for rationalization.

Graphs are used to model and analyze data interconnected through complex relationships. A graph contains a set of entities as *nodes* and the *relationships* that connect them, as illustrated through a simple example in Figure 4.1 below. Compared to a traditional relational database model, graph databases provide the benefits of increased performance with larger datasets, as well as increased flexibility to add new components to an evolving data model according to ongoing requirements without the confines of a restrictive schema [82]. With this highly flexible and high-level structure, graphs can be used to model all kinds of systems and have countless use cases across many industries and applications.

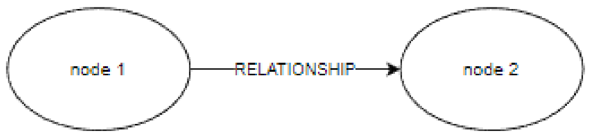


Figure 4.1 Basic graph with a relationship between two nodes

Graphs can be further extended to more descriptive models, such as the commonly used *property graph* model, which allows nodes and relationships to contain key-value pairs as *properties*. As described by Barrasa et al. [83], a property graph model consists of the following characteristics:

1. ***Nodes* representing entities in the domain:**
    - Nodes can contain zero or more *properties*, which are key-value pairs representing entity data.
    - Nodes can have zero or more *labels*, which declare the node's purpose in the graph.
2. ***Relationships* representing how entities interrelate**:
    - Relationships have a *type*.
    - Relationships have a *direction*, going from one node to another.
    - Relationships can contain zero or more *properties*, which are key-value pairs representing some characteristic of the link.
    - Relationships never dangle – there is always a start and end node.

With a basis on the property graph model, *knowledge graphs* can provide further emphasis on contextual understanding. Knowledge graphs provide a contextualized understanding of data, where interlinked sets of properties describe real-world entities, events, or things and their interrelations in a human and machine-readable format [83]. The modeling rules of a knowledge graph are defined using an *organizing principle*, or *semantics*, which provide a layer of organizing metadata to connect context for reasoning and knowledge discovery.

4.1 Graph Architecture and Model

The flexibility and rich ability to model complex entity relationships motivate the adoption of a graph-based solution for the modeling of cybersecurity domain knowledge and agent behaviours in relation to the environment. Our model integrates knowledge graphs to model the data and inform system workflows as a foundation for multi-agent system intelligence. Our graph architecture integrates three separate layers for context into the network environment, cybersecurity domain knowledge and BDI agent knowledge, as shown in Figure 4.2 below:

- **The Environment layer** is used to model the devices and entities within the network to provide ongoing context and state awareness.
- **The Cybersecurity Domain Knowledge layer** integrates industry frameworks into a common model for identifying vulnerabilities and exposures, inferring security risks, attack detections and analysis, and informing applicable defence techniques based on policies and environmental awareness.
- **The BDI Agent layer** is used to model agent planning, actions and workflows based on knowledge of device capabilities, security requirements and context from the other two layers.

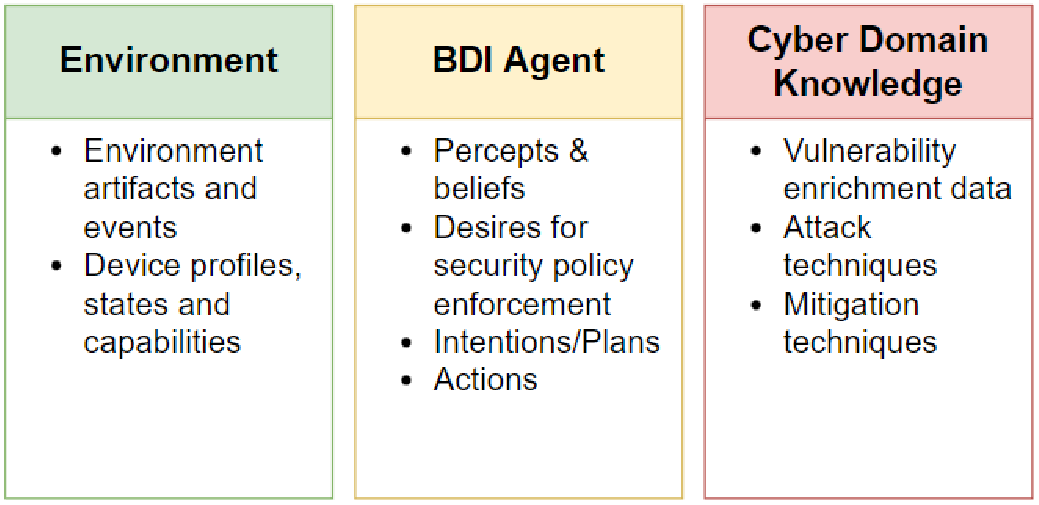

Environment
Environment artifacts and events
Device profiles, states and capabilities
BDI Agent
Percepts & beliefs
Desires for security policy enforcement
Intentions/Plans
Actions
Cyber Domain Knowledge
Vulnerability enrichment data
Attack techniques
Mitigation techniques

Figure 4.2 High-level Graph Layer Interactions

Each of these layers is highly related to inform agent decisions based on knowledge of the environment states and capabilities and cybersecurity domain knowledge to support courses of action based on risk profiles of the known environment and security requirements. Figure 4.3 below shows an overall view of the relationships between each graph component of the environment, agent, and domain knowledge, which will be further described in detail in the remaining sections of this chapter.

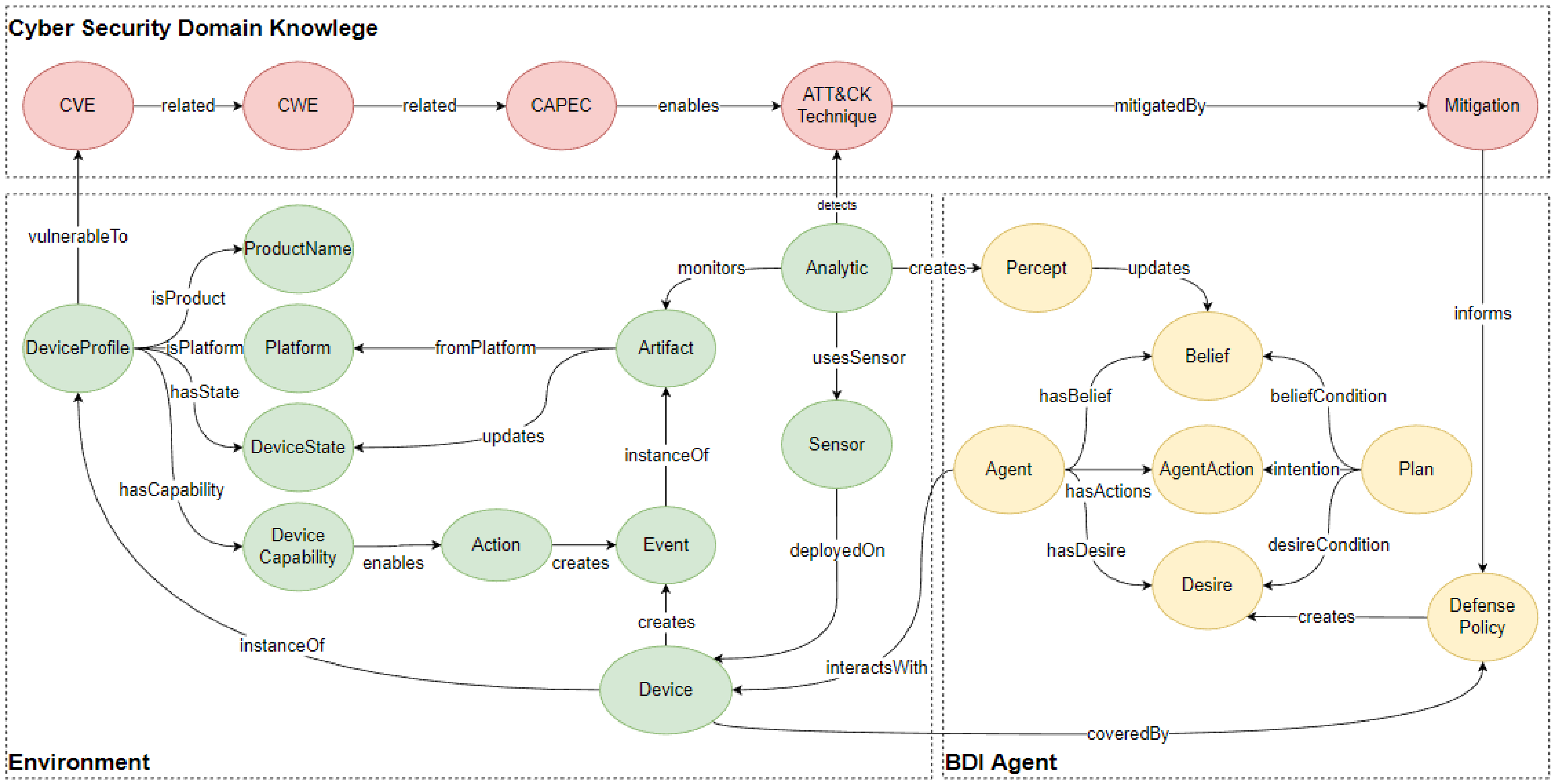


Figure 4.3 Knowledge Graph Meta Model

4.2 Environment Graph

Maintaining ongoing knowledge of the environment is a critical function for rational agents to interact and receive timely feedback on their environment states. In the field of cybersecurity, the same holds true for defenders to understand the environment they aim to secure. Knowledge of assets, behaviours, capabilities, and network topologies allows the defender to know where security vulnerabilities may exist, what controls are or are not deployed, and knowledge of security threats and possible attack paths.

The environment graph is the basis of the agents' situational awareness and is further augmented by domain knowledge enrichment data to infer the security implications. Using a graph model, key components of the environment can be defined, categorized, labelled, and related using a data model that allows for interconnectivity with domain knowledge graphs and agent planning.

The environment graph has been designed with the following key requirements and integrations in mind:

1. Model the environment for agents to interpret devices, attributes, states, capabilities, and possible actions;
2. Attributes to be mapped to security domain knowledge for understanding vulnerabilities, risk analysis, and relation to security requirements and policies; and
3. Flexible reference data profiles and maintenance.

Figure 4.4 illustrates a detailed illustration of the nodes and relationships within the environment graph, which we have divided into two major categories: device profiles and instances and events and analytics. This section will detail the environment graph layer, our design approach, and usage. The node labels within the environment graph are further described in this section.

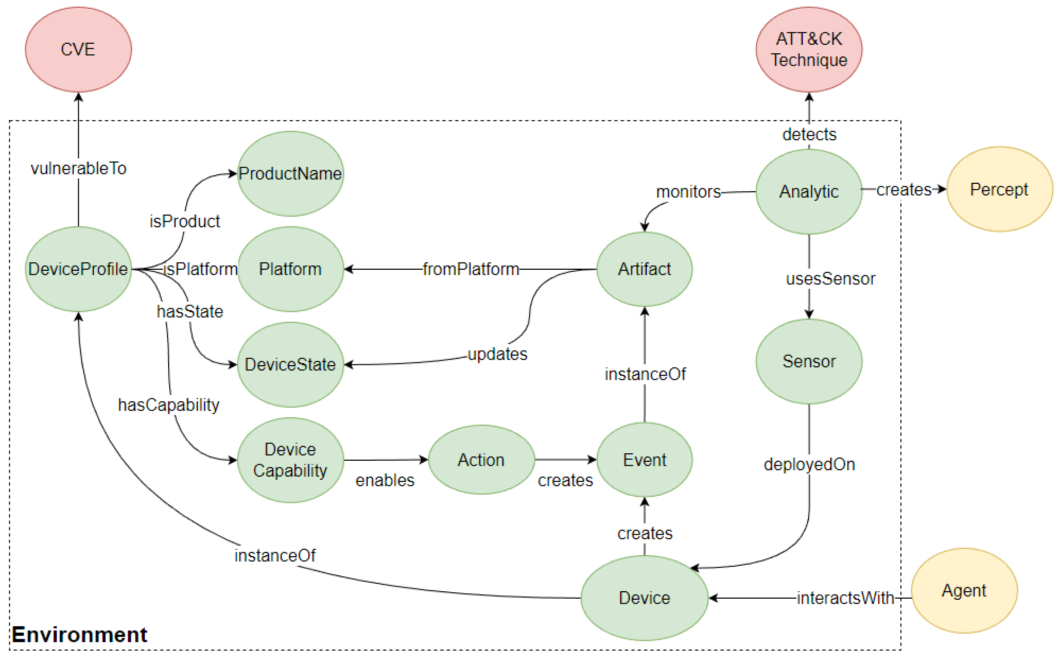


Figure 4.4 Environment Graph Layer

4.2.1 Device Profiles and Instances

The first major category of nodes within the environment graph is to model device profiles and instances. In support of the first and third requirements listed above, we implement the concept of "Device Profiles" to introduce a scalable and repeatable design for instances of devices to inherit the applicable properties and relationships of their parent Device Profile. This is ideal for consistency in asset management, classification, and modeling capabilities for possible agent actions and expected effects.

Table 4.1 Device Profile Node Descriptions

Node Label	Properties	Relationships
Device Profile	Name Version	ProductName Platform DeviceState DeviceCapability Device
Device	Name Internet Protocol (IP) Address	DeviceProfile Event Sensor Agent

	Media Access Control (MAC) Address Ports Configs Access Control Current States	Security Policy
Device Capability	Name Description	DeviceProfile Action
Action	Function	DeviceCapability Event

The common attributes related to device profiles, such as product name and platform, have been adopted for integration with MITRE framework fields to be described further in Section 4.4 .

The Device Capabilities in our model are based on NISTIR 8259A Internet of Things (IoT) Product Cybersecurity Capabilities [26] which defines a set of device capabilities that directly correlate to the NISTIR 8228 Security Requirements [19] as described in Chapter 3. Using the IoT security product labeling standard proposed in this document, all compliant IoT devices would be accompanied by a product label in this format which can be imported into our graph when a device is registered to the network.

Device capabilities are associated with corresponding actions which can be performed by agents. Action sets have been developed as abstract functions which can be called by agents through Application Programming Interface (API) requests to devices to perform actions related to the device's capabilities. While the focus of this work at this time is on the overall framework and agent reasoning, a simulated environment with simple function calls has been sufficient for initial demonstration. It will be further detailed in the following chapter.

4.2.2 Events and Analytics

The second category of nodes within the environment graph pertains to the ability to model system events, or messages, in a consistent way that can be interpreted by agents. While this provides a model for agents to communicate and understand their environment, the data model also integrates with the domain knowledge ontology to be used for security monitoring and analytics. The below table provides an overview of the node relationships and definitions:

Table 4.2 Events and Analytics Node Descriptions

Node Label	**Properties**	**Relationships**
Event	ID Data	Action Artifact Device
Artifact	Object Action Field	Event DeviceState Platform Analytic
Analytic	Name	Artifact Sensor ATT&CK Technique
Sensor	Type	Analytic Device

Message formats are defined for events as a result of device activities or agent actions. Agents receive messages as percepts in this format and can interpret them using the agent graph. Events are an instance of an *artifact*, which models the event data in a way that can be easily parsed and analyzed by *analytics*. Our data model for artifacts and analytics has been heavily influenced by the MITRE Cyber Analytics Repository (CAR) data model [84]. In this model, artifacts are modeled as a tuple of (object, action, field) which details the properties and state changes for an event.

The events and analytics components are illustrated in greater detail in Figure 4.5 below.

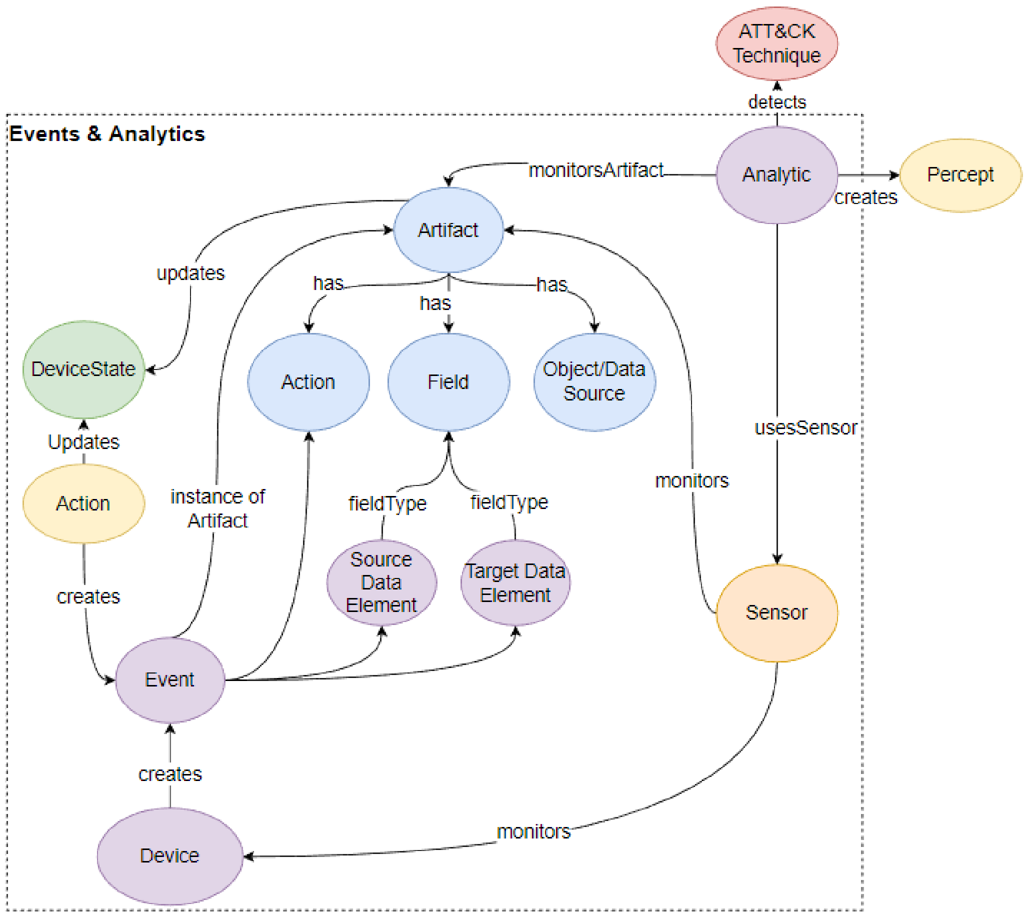


Figure 4.5 Events and Analytics Detail

Although CAR provides a great data model for event modeling and analytics, the dataset is limited in terms of integration with other frameworks for further enrichment. We found that the model could be mapped more directly to existing fields in MITRE ATT&CK for direct correlation to attack techniques which would provide greater contextual enrichment through our domain knowledge graph. Based on this finding, we leveraged the below MITRE ATT&CK fields to the same effect and imported this data set into our graph:

- “Data source” field in place of “object.”
- “Relationship” in place of “actions.”
- “Source element” and “target element” as appropriate in place of “fields.”

Below are some examples from our filtered dataset:

Table 4.3 Object-Action-Field Mapping for Analytics

Object (data_source)	Actions (relationship)	Fields (Src/Target Elements)
command	executed	User Process Command
drive	created	Process Drive
	accessed	
	modified	
driver	loaded	Process Host Driver
	retrieved information about	
file	accessed	User Process File Filestream
	requested access to	
	created	
	retrieved information about	
	modified	
	deleted	

4.3 BDI Agent Knowledge Graph

The reasoning capabilities of the BDI agents within the system are driven through the BDI agent knowledge graph. This knowledge graph contains all relevant data to support agents' rationalization and decisive actions while supported by additional context within the environment and domain knowledge graphs. The objectives of the BDI agent knowledge graph are as follows:

- Interpret environment perceptions into agent beliefs.
- Model desires based on security policies and requirements.
- Capabilities to interact with the environment through available actions.
- Create a plan based on beliefs and desires.

Figure 4.6 below shows the nodes and relationships within the BDI agent graph to be described in further detail within this section.

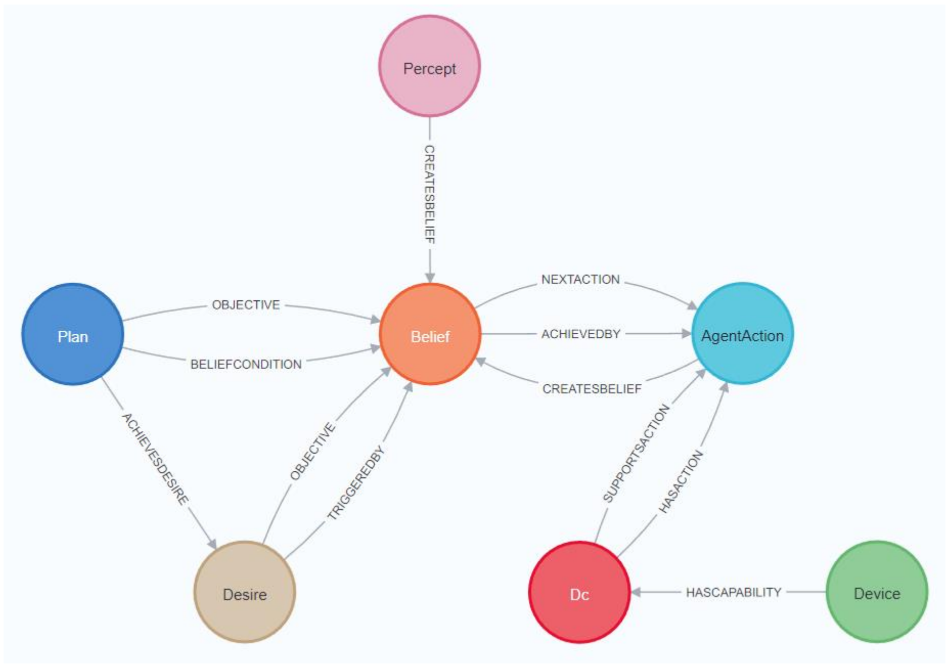


Figure 4.6 BDI Agent Graph

4.3.1 Node and Relationship Definitions

The BDI model within the agent graph is consistent with the BDI definitions as described in earlier chapters. In relation to the rest of the graph, the design is not able to enable the agents to interact with their environment through percepts (events) and actions based on the device capabilities defined in the environment graph. An agent deployed to a particular device with the appropriate role and permissions will be able to perform the available actions on that device.

The foundation of agent reasoning and intelligence is also within the graph to support rational decisions and planning based on contextual awareness and security goals. Desires generated from the security requirements and policies are placed on the graph and related to a belief state. Modeling available actions based on device capabilities and belief states allows an agent to query the graph for a plan of action to achieve a path to a target desired state.

Table 4.4 BDI Graph Node Descriptions

Node Label	Properties	Relationships
Percept	value	Belief
Belief	value	Percept AgentAction
Desire	value	State/Belief Plan
AgentAction	value	DeviceCapability Belief

4.3.2 Cypher Queries for Agent Functions

BDI agents can leverage the graph to support their belief revision, plan selection, and action selection functions, as further described in this section, using Neo4j cypher queries.

Belief Revision

Belief revision is triggered by an agent receiving a new percept. The agent graph supports the agent's belief revision function by providing relationships between types of percepts and the beliefs to be inferred. The below cypher query is used to return the beliefs obtained in response to an observed percept:

```
MATCH (:Percept {Value: "$percept"}) -[:CreatesBelief]-> (beliefs)
RETURN beliefs
```

Plan Selection

When a new desire is obtained, such as through a new security policy or mission, the agent must retrieve an appropriate plan from the graph to pursue the desired state. Below is the process of plan selection along with the applicable cypher query, which will return a plan in the form of belief-action pairs:

1. Find a state/belief that is the objective of the active desire
2. Find *a* actions that the target state is "achieved by" (i.e., download update, install update)
3. Find *b* beliefs that precede these as "next action" (i.e., update available, update downloaded)
4. Build belief/action pairs based on *b* and *a*

```
MATCH (:Desire {value: "$desire"}) -[:OBJECTIVE]-> (targetbelief)
MATCH (targetbelief) -[:ACHIEVEDBY]-> (selectedActions)
MATCH (preBeliefs) -[:NEXTACTION]-> (selectedActions)
RETURN targetbelief, selectedActions, preBeliefs
```

Action Selection

Based on a plan to achieve a particular desired state, an agent must select an appropriate next action. The below query is used to return the next action based on the agent's current beliefs:

```
MATCH (:Desire {value:"$desire"}) -[:OBJECTIVE]-> (targetbelief)
MATCH (targetbelief) -[:ACHIEVEDBY]-> (selectedAction)
MATCH (preBeliefs:Belief{value:"$belief"})-[:NEXTACTION]->
(selectedAction)
RETURN selectedAction
```

4.4 Cybersecurity Domain Knowledge Ontology

While the other sections of this chapter have covered the environment and BDI agent models, another key element of the system is the cybersecurity domain knowledge ontology, which is required to support the contextual understanding of situational awareness as well as direct agent reasoning toward effective security decisions. In addition, interpreting the security implications of observed events and selecting appropriate plans to achieve specific goals requires a structured framework for relating entities and mapping relevant threat libraries.

Fortunately, there has been much attention over the past few years in the cybersecurity industry to develop and maintain many frameworks and knowledge bases to provide classifications, relationships, and descriptions of key information such as vulnerabilities, weaknesses, attack techniques, threat intelligence, defensive controls, and more. Many of these frameworks are leveraged prominently across the industry as a reference for understanding and prioritizing security controls and responses and provide a solid foundation of reference data for the purposes of this work.

However, some current limitations with this reference data must be augmented to support our purposes. While these frameworks are prominent and well utilized within the industry, the approach is often standalone and without interoperability between frameworks for different stages. While there have notably been some recent works to combine a knowledge base of frameworks, there currently does not exist a full solution to the end-to-end entirety that our system requires. A notable work is the OdTM Base Threat Model Ontology framework [85] which provides a base threat model ontology using Web Ontology Language (OWL). The ontology framework can be used to build mappings of domain relevant threats and countermeasures. Based on our observations, it appears that it is likely that relationships between frameworks will soon be bridged, as there currently already exist many similar fields across these frameworks that can be easily related. With the emerging prevalence of automation and machine learning in the security industry, a common and relatable data model for leveraging reference data would provide a strong foundational contribution. However, since no current solution yet publicly exists according to our knowledge, we have proceeded to design our own model to relate the available knowledge bases, according to our requirements.

4.4.1 Reference Knowledge Bases

The cybersecurity domain knowledge ontology has been generated based on several industry frameworks shown below, which can be used for mapping relationships between elements of the environment from vulnerability data into recommendations for mitigations and controls.

Table 4.5 Reference Knowledge Bases

Reference Framework/Database	Description
Common Vulnerability Enumeration (CVE) [80]	Identifies, defines and catalog publicly disclosed cybersecurity vulnerabilities. The Common Vulnerability Scoring System (CVSS) [79] is

	also used for scoring and characteristics of vulnerabilities
Common Weakness Enumeration (CWE) [86]	A community-developed list and software and hardware weakness types in a common language as a baseline for weakness identification, mitigation and prevention efforts.
Common Attack Pattern Enumeration and Classification (CAPEC) [87]	Provides a dictionary of known attack patterns employed by adversaries to exploit known weaknesses in cyber-enabled capabilities.
ATT&CK [81]	A knowledge base of adversary tactics and techniques based on real-world observations
D3FEND [88]	A knowledge graph of cybersecurity countermeasures
Cyber Analytics Repository (CAR) [84]	A knowledge base of analytics based on the MITRE ATT&CK adversary model. The CAR analytics have not been used directly in our graph, however the data model has been used as a reference as a framework for modeling analytics based on data relationships.

We then proceed to map these to associated attack patterns from the MITRE library, which provides a catalog of common attack patterns, attributes, prerequisites, and mitigations. This data is then used to build a domain-specific ontology for IoT devices to be used by the agent plan library at an abstract level. While outside of the implementation scope for this paper, this can be further expanded to more detailed technical-level capabilities as future work.

4.4.2 Node and Relationship Definitions

The meta-graph shown in Figure 4.7 below illustrates the node and relationships within the cybersecurity domain knowledge graph pulled from the knowledge bases described in the previous section. Each of these knowledge bases provide a data model that can be easily related to each other and has been further related within our graph to provide end to end relationships between security weaknesses to mitigation techniques along with their associated properties for risk analysis and prioritization. The node labels, properties, and associated relationships are listed in Table 4.6 below.

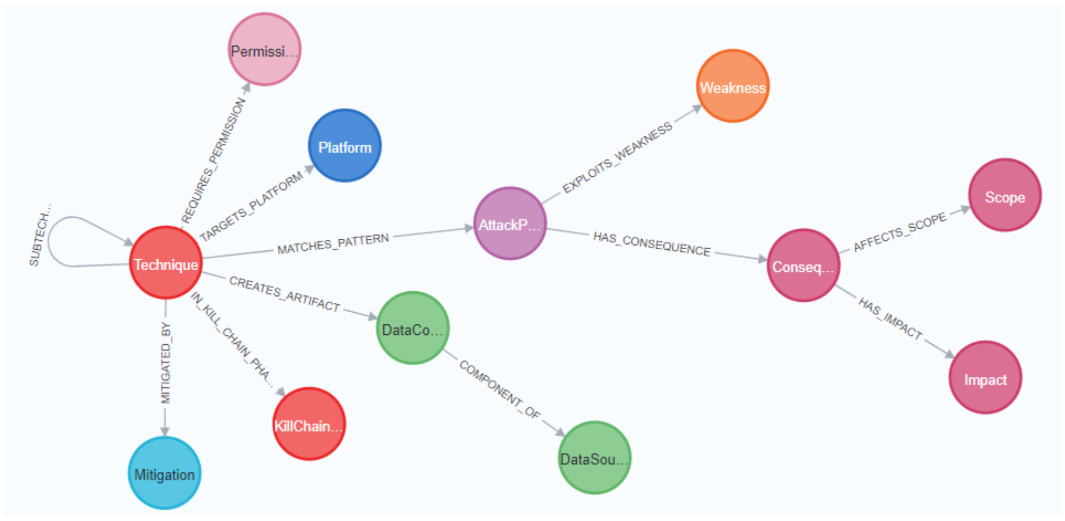


Figure 4.7 Cybersecurity Domain Knowledge Graph

Table 4.6 Cybersecurity Domain Knowledge Graph Node Definitions

Node Label	**Properties**	**Relationships**
Technique	id name description	Permission Platform AttackPattern DataComponent KillChainPhase Mitigation Technique
Platform	name	Technique
Mitigation	id name description	Technique
AttackPattern	id name description likelihood severity	Technique Weakness Consequence
Weakness	id name description likelihood	AttackPattern
Consequence	-	Scope Impact
Scope	name	Consequence
Impact	name	Consequence

4.4.3 Cypher Queries

An important function of the domain knowledge graph is to enrich agents' understanding of the security implications of their environment, including assessing the security implications and level of risk related to known or suspected vulnerabilities and weaknesses. This is the first step towards establishing an appropriate mitigation plan aligned with the priorities identified within the policy.

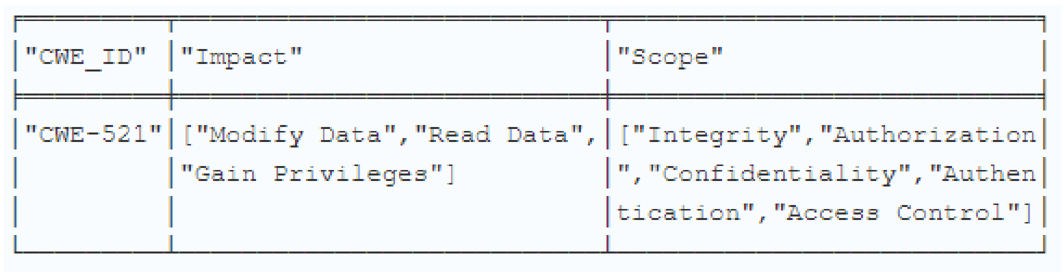
This section provides an overview of the types of queries that can be used to gather information in support of agent situational awareness and defence control planning.

Summary of Impact and Scope of a Known CWE

The below query shows an example Cypher query which returns the associated impact and scope for a particular CWE-521: "Weak Password Requirements" on a Linux system:

```
MATCH (t:Technique)-[:TARGETS_PLATFORM]-> (p:Platform { name:
"Linux"})
MATCH (w:Weakness {id:"CWE-521"}) <- [:EXPLOITS_WEAKNESS] -
(atp:AttackPattern)<-[:MATCHES_PATTERN]-(t)-[:MITIGATED_BY]->
(m:Mitigation)
MATCH (atp) - [:HAS_CONSEQUENCE] -> (c:Consequence) -
[:AFFECTS_SCOPE] ->(scope:Scope)
MATCH (c)-[:HAS_IMPACT]->(impact:Impact)
RETURN w.id AS CWE_ID, collect(distinct impact.name) AS Impact,
collect(distinct scope.name) AS Scope
```

The results of the query are shown below, where a summary of the impact and scope of the CWE are listed. This provides useful information to be stored within the Security Monitor exploits to be used for prioritizing remediation of CWEs based on the priorities of the security policy.

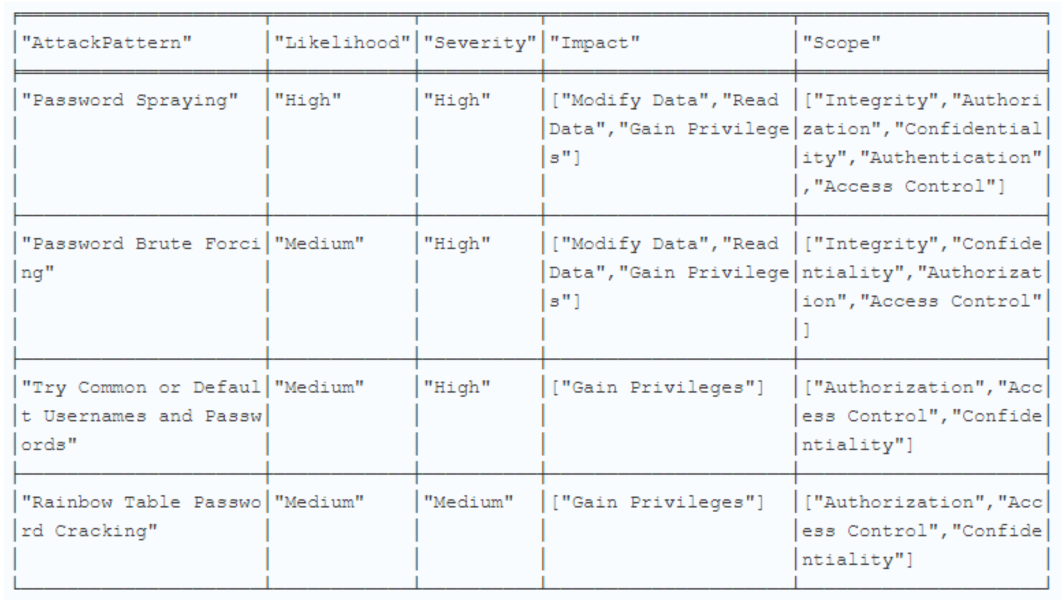


```
╒═════════╤════════════════════════════════╤═════════════════════════════╕
│"CWE_ID" │"Impact"                        │"Scope"                      │
╞═════════╪════════════════════════════════╪═════════════════════════════╡
│"CWE-521"│["Modify Data","Read Data",     │["Integrity","Authorization  │
│         │"Gain Privileges"]              │","Confidentiality","Authen  │
│         │                                │tication","Access Control"]  │
└─────────┴────────────────────────────────┴─────────────────────────────┘
```

Enriched Data for Attack Patterns Associated with a Known CWE

Further, there may be multiple attack patterns associated with a CWE, raising the requirement for a more granular view of the different types of CAPEC attack patterns and their associated consequences. In addition to this, an understanding of the likelihood and severity of a potential attack pattern will be useful for prioritizing specific remediation plans. The below query is used to retrieve this data from the graph:

```
MATCH (t:Technique)-[:TARGETS_PLATFORM]-> (p:Platform
{name:"Linux"})
MATCH (w:Weakness {id:"CWE-521"}) <- [:EXPLOITS_WEAKNESS] -
(atp:AttackPattern) <- [:MATCHES_PATTERN]- (t) -[:MITIGATED_BY] ->
(m:Mitigation)
MATCH (atp) - [:HAS_CONSEQUENCE] -> (c:Consequence) -
[:AFFECTS_SCOPE] ->(scope:Scope)
MATCH (c)-[:HAS_IMPACT]->(impact:Impact)
RETURN atp.name AS AttackPattern, atp.likelihood AS Likelihood, at
p.severity AS Severity, collect(distinct impact.name) AS Impact,
collect(distinct scope.name) AS Scope
```



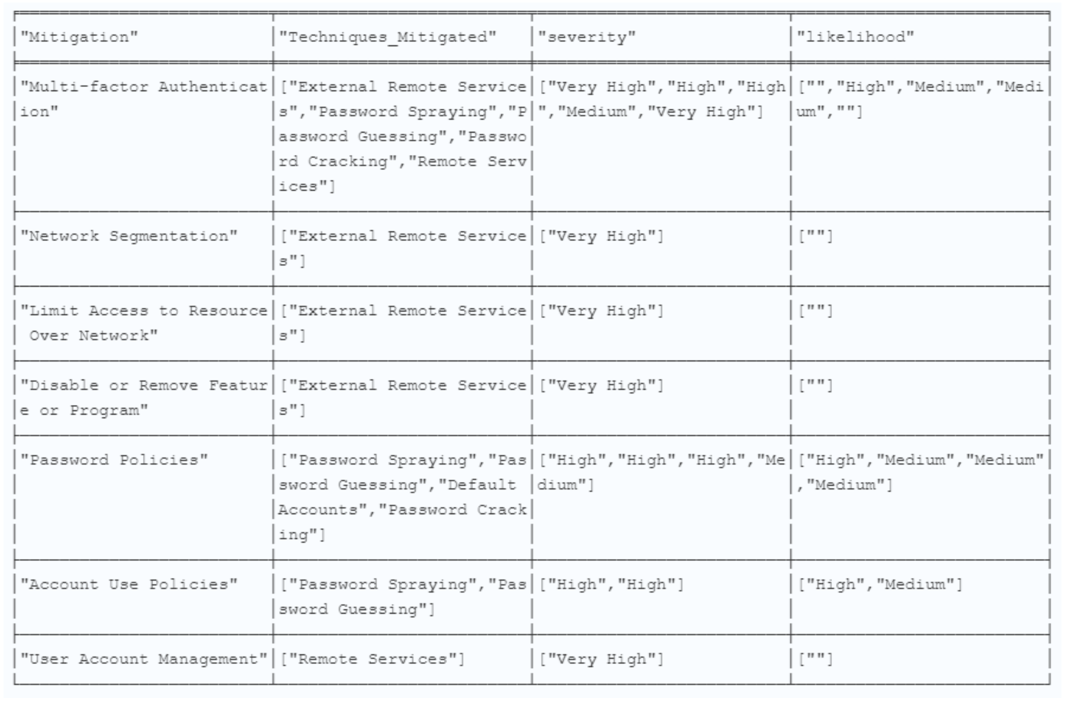
```
│"AttackPattern"      │"Likelihood"│"Severity"│"Impact"           │"Scope"                │
│"Password Spraying"  │"High"      │"High"    │["Modify Data","Read│["Integrity","Authori│
│                     │            │          │Data","Gain Privilege│zation","Confidential│
│                     │            │          │s"]                 │ity","Authentication"│
│                     │            │          │                    │,"Access Control"]   │
│"Password Brute Forci│"Medium"    │"High"    │["Modify Data","Read│["Integrity","Confide│
│ng"                  │            │          │Data","Gain Privilege│ntiality","Authorizat│
│                     │            │          │s"]                 │ion","Access Control"│
│                     │            │          │                    │]                    │
│"Try Common or Defaul│"Medium"    │"High"    │["Gain Privileges"] │["Authorization","Acc│
│t Usernames and Passw│            │          │                    │ess Control","Confide│
│ords"                │            │          │                    │ntiality"]           │
│"Rainbow Table Passwo│"Medium"    │"Medium"  │["Gain Privileges"] │["Authorization","Acc│
│rd Cracking"         │            │          │                    │ess Control","Confide│
│                     │            │          │                    │ntiality"]           │
```

Mitigation Techniques and Prioritization

While the results of the previous query can support the prioritization of selecting mitigations mapped to attack patterns, it is also important to consider that there are mitigation techniques that can protect against multiple types of attacks. The following query can be used to retrieve the list of related mitigations for the CWE with the associated attack techniques, severity, and likelihood.

```
MATCH (t:Technique)-[TARGETS_PLATFORM]-> (p:Platform {name:
"Linux"})
MATCH (w:Weakness {id:"CWE-521"}) <- [EXPLOITS_WEAKNESS] -
(atp:AttackPattern)  <- [MATCHES_PATTERN]- (t) - [MITIGATED_BY] -
> (m:Mitigation)
RETURN m.name AS Mitigation, collect(t.name) AS Techniques_Mitigat
ed, collect(atp.severity) AS severity, collect(atp.likelihood) AS
likelihood
```

Based on the below results, multi-factor authentication and password policies would protect against the largest number of techniques related to CWE-521:



```
╒═════════════════════════╤═════════════════════════╤═════════════════════════╤═════════════════════════╕
│"Mitigation"             │"Techniques_Mitigated"   │"severity"               │"likelihood"             │
╞═════════════════════════╪═════════════════════════╪═════════════════════════╪═════════════════════════╡
│"Multi-factor Authenticat│["External Remote Service│["Very High","High","High│["","High","Medium","Medi│
│ion"                     │s","Password Spraying","P│","Medium","Very High"]   │um",""]                  │
│                         │assword Guessing","Passwo│                         │                         │
│                         │rd Cracking","Remote Serv│                         │                         │
│                         │ices"]                   │                         │                         │
├─────────────────────────┼─────────────────────────┼─────────────────────────┼─────────────────────────┤
│"Network Segmentation"   │["External Remote Service│["Very High"]            │[""]                     │
│                         │s"]                      │                         │                         │
├─────────────────────────┼─────────────────────────┼─────────────────────────┼─────────────────────────┤
│"Limit Access to Resource│["External Remote Service│["Very High"]            │[""]                     │
│ Over Network"           │s"]                      │                         │                         │
├─────────────────────────┼─────────────────────────┼─────────────────────────┼─────────────────────────┤
│"Disable or Remove Featur│["External Remote Service│["Very High"]            │[""]                     │
│e or Program"            │s"]                      │                         │                         │
├─────────────────────────┼─────────────────────────┼─────────────────────────┼─────────────────────────┤
│"Password Policies"      │["Password Spraying","Pas│["High","High","High","Me│["High","Medium","Medium"│
│                         │sword Guessing","Default │dium"]                   │,"Medium"]               │
│                         │Accounts","Password Crack│                         │                         │
│                         │ing"]                    │                         │                         │
├─────────────────────────┼─────────────────────────┼─────────────────────────┼─────────────────────────┤
│"Account Use Policies"   │["Password Spraying","Pas│["High","High"]          │["High","Medium"]        │
│                         │sword Guessing"]         │                         │                         │
├─────────────────────────┼─────────────────────────┼─────────────────────────┼─────────────────────────┤
│"User Account Management"│["Remote Services"]      │["Very High"]            │[""]                     │
└─────────────────────────┴─────────────────────────┴─────────────────────────┴─────────────────────────┘
```

Mitigations are related to Device Capabilities (DCs) described in the previous sections. The prioritization and selection process take all of the above criteria into consideration as well as the available DCs that the agents can leverage to select the most effective available mitigation. If no mitigations are possible, the system monitor will still maintain a view of the possible risks and can provide notifications to the users.

4.5 Chapter Summary

While recent years have observed increasing attention towards data modeling and intelligence sharing within the cybersecurity industry, this data provides a foundation of knowledge that can be applied for structured agent reasoning. Existing BDI research and applications have utilized ontologies in other forms, however to our knowledge there has been no previous work integrating BDI agents with knowledge graphs. The flexibility and ability to model complex relationships motivate the adoption of our presented graph-based solution for modeling cybersecurity domain knowledge and agent behaviours in relation to the environment as a foundation for multi-agent intelligence.

This chapter introduced our model and examples for the environment graph, cybersecurity domain knowledge graph, and BDI agent knowledge graphs to inform agent behaviours and inference rules for rationalization. With the foundations defined, the following chapter will proceed to discuss our implementation which brings the knowledge graph model together with the multi-agent system architecture described in Chapter 3.

5 Implementation

This chapter presents an implementation of our multi-agent system architecture for adaptive cyber defence in a smart home network. Our implementation architecture consists of three components: coloured petri nets, knowledge graph database, and the simulation engine. We will proceed to describe the implementation for the control and coordination functions and associated policy generation, followed by two agent use cases for vulnerability management and access management. The implementation described in this chapter will provide a foundation for the experimental evaluation and results discussed in Chapter 6.

5.1 Model Smart Home Scenario

The implementation simulates a fictional smart home environment, as shown in Figure 5.1 as an illustrative example of a realistic use case. The scenario illustrates a single-bedroom apartment that contains a variety of IoT devices for physical security, lighting, temperature control, entertainment, personal devices, and network devices. In addition, the home contains a single user with moderate adoption of consumer IoT devices for the primary purposes of home automation and comfort.

5.1.1 Devices and Network

The model smart home environment contains the following devices shown in Table 5.1 below, which have been added to the Neo4j [89] environment graph and basic functions created within the simulation engine. We have also generated a set of device capabilities for each device according to the National Institute of Standards and Technology (NIST) device capabilities[19]. Some devices have been configured with limited capabilities to show how the system would respond to constrained resources.

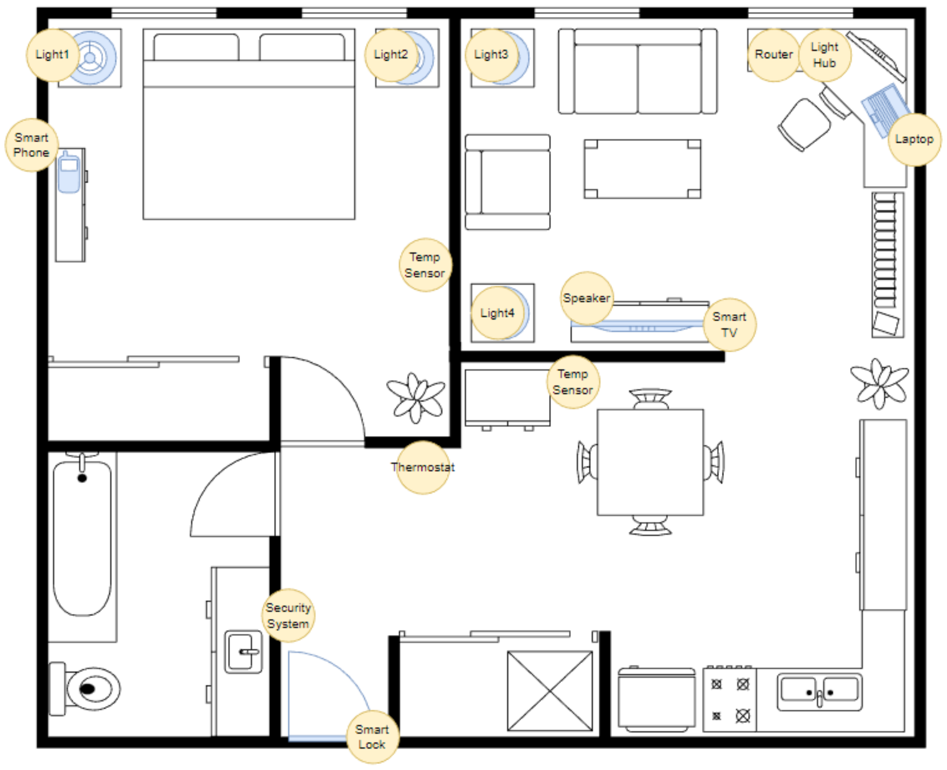


Figure 5.1 Model Smart Home Environment

Table 5.1 Smart Home Device Listing

Type	**ProductName**
Physical Security	Security System
	Smart Lock
Lighting	Light Hub
	Light1
	Light2
	Light3
	Light4
Temperature Control	Thermostat
	Temp Sensor 1
	Temp Sensor 2
Audio/Video	Smart TV
	Smart Speaker
Personal Devices	Laptop
	Smart Phone
Network & Control	Router
	IoT Hub

5.1.2 Security Requirements

The security requirements that need to be addressed through our model have been defined using the NIST framework discussed in Chapter 3. Each requirement maps to a corresponding agent Desire within the knowledge graph:

- Asset Management
- Device Configuration
- Data Protection
- Access Management
- Vulnerability Management
- Incident Detection
- Availability

5.2 Implementation Architecture

The architecture for our implementation is shown in Figure 5.2 below, with the following three major components to be described in further detail throughout the rest of this section:

- **Coloured Petri Nets (CPN)**: the message exchanges between agents and devices are simulated within CPN. CPN Tools [90] is used to visualize and facilitate the network environment and agent BDI through each time step of the simulation.
- **Simulation Engine**: a set of scripts developed in Go [91] to simulate basic device and agent instances and the interface between CPN Tools and Neo4j. The device and agent instances are called from CPN tools to execute appropriate actions when they receive messages.
- **Neo4j Graph Database** [89]: contains the knowledge graphs for agent reasoning, including the domain knowledge ontology, BDI knowledge graph, and knowledge graph. A dashboard has also been created for the visualization of system states and summarized data within the graph.

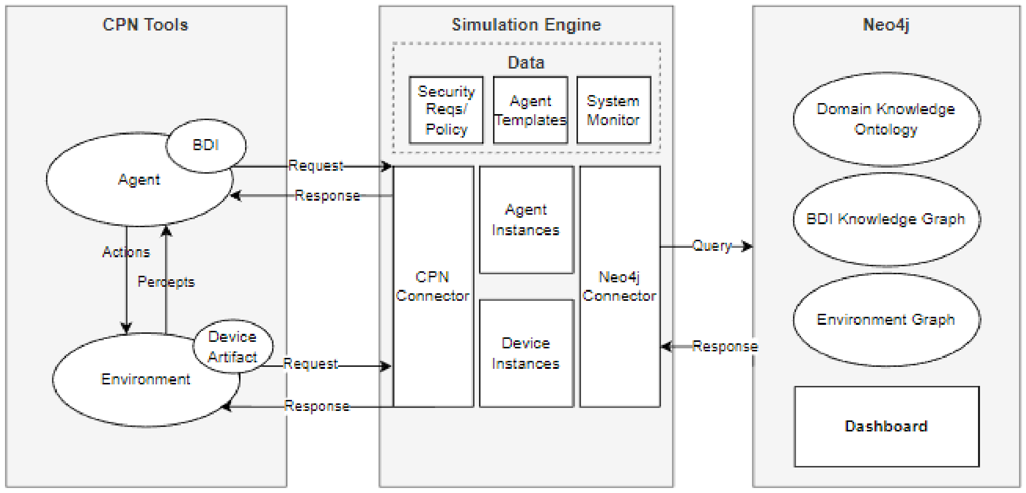


Figure 5.2 Implementation Architecture

This architecture enables a demonstration of each major component of our model, from the controller, coordinator and missions, and device agents acting in different use cases.

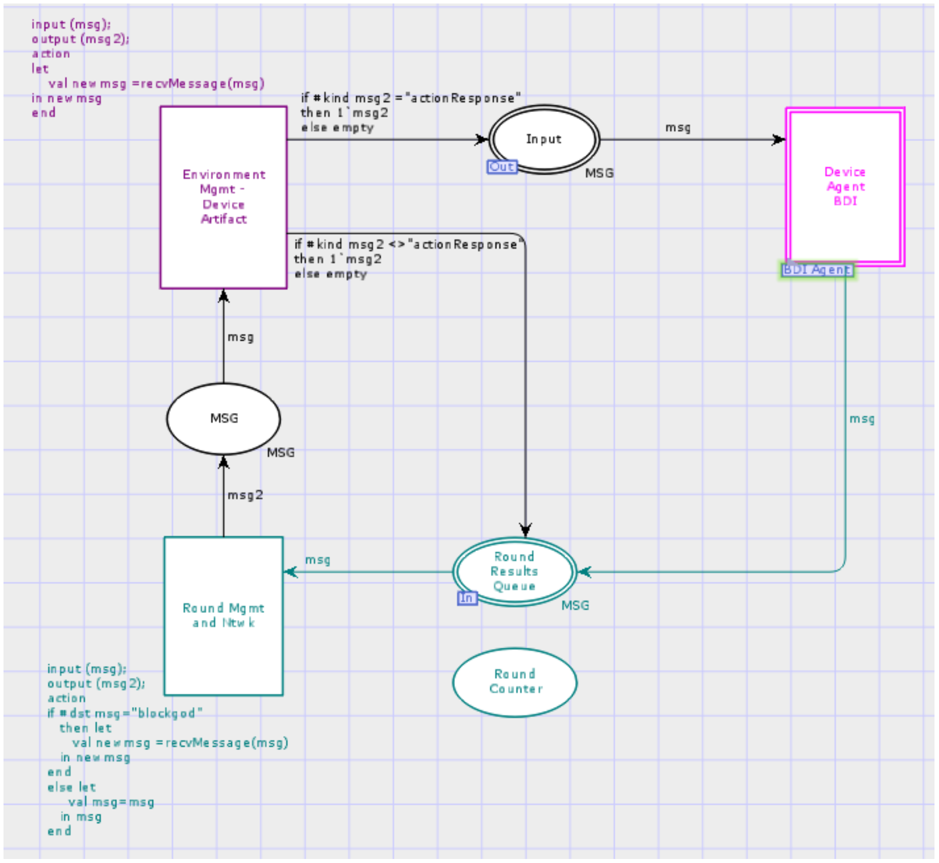
5.2.1 Coloured Petri Nets

Colored Petri Nets (CPN) are a graphical oriented language for the design, specification, simulation and verification of systems [92]. CPN can be used for modeling and simulating behaviours of systems where concurrency and communication are key characteristics, such as business processes and workflows, manufacturing systems, and agent systems [93]. For example, Petri Nets have been used to implement BDI agents by Jimenez-Ochoa et al. [94] to model Interpreted Petri Nets (IPN) to represent agent beliefs and beliefs revision transitions for flexible manufacturing systems.

Our simulation makes use of CPN Tools [90], a tool for editing, simulating and analyzing our model through colored Petri Nets. A CPN model represents states of the system (places), and events that can change states (transitions). Through this model, it is possible to walk through and execute simulated systems to better understand system behaviour and design. CPN ML is based on Standard ML, a functional programming language that provides the definition of data types.

We simulate our environment and agent instances in CPN Tools to visualize the states and transitions as messages are sent through the network. While early versions of our model were designed using CPN Tools for validation of basic agent BDI reasoning and communication, our implementation has now expanded to leverage our external integrations with Neo4j and Simulation engine as the basis for larger scale intelligence and knowledge reasoning.

We have modeled the environment and BDI agent nets, as shown in Figure 5.3 and Figure 5.4 respectively. The environment net provides a model for simulating the environment, including device instances, messages, and network communications while the agent net models the rationality of the BDI agents that interact with it. We proceed to describe each component in more detail within the rest of this section.



```
input (msg);
output (msg2);
action
let
   val newmsg =recvMessage(msg)
in newmsg
end
```

```
if #kind msg2 ="actionResponse"
then 1`msg2
else empty
```

```
if #kind msg2 <>"actionResponse"
then 1`msg2
else empty
```

```
input (msg);
output (msg2);
action
if #dst msg="blockgod"
   then let
      val newmsg =recvMessage(msg)
   in newmsg
end
else let
      val msg=msg
   in msg
end
```

Figure 5.3 Environment CPN

Message Format

Within the simulation, we define a common message format for simple simulated network communications within the system. A colorset of “MSG” has been defined consisting of the following fields: msg = { src=””, dst=””, kind=””, data=[] }.

- **src**: the source of the message.
- **dst**: the destination of the message.
- **kind**: identifies the type of message to handle data fields.
- **data**: a string list is consisting of the actual data of the message within the appropriate fields.

The below “kinds” of messages described in Table 5.2 have been defined to pass messages across the network as well as for internal messages between the agent and host device.

Table 5.2 Message "Kind" Definitions

Kind	**Data Format**	**Usage**
notify	[("value","UpdateNotification"), ("ver", "$ver")]	For update notifications
request	Same as action	Request msg to another device
response	[("Function name", "id"),("status", "Success/Failed"),(result/error))]	Response msg to another device
action	(Function name, ID),(Function parameters) [("update", "id"), ("ver", "$ver")]	Agent action on host device
actionResponse	[("Function name", "id"),("status", "Success/Failed"),(result/error))]	Host device response to action successful/failed

While the actions and functions are not performed within CPN tools directly, the simulation engine parses the message format to perform the appropriate functions and update states accordingly.

Network Simulation and Round Management

The design of our system models concurrent behaviours through discrete time events in a “turn-based” model, where each time interval runs through a queue of concurrent events in a “round.” Rather than hard-coding each device and agent into the net, the environment and agent nets have been reduced to a simple shell to represent the basic structure of any device or agent receiving, processing, and acting on an event. The device and agent transitions load the state and attributes of the appropriate device and agent when their turn begins. This allows for the scalability of testing scenarios as the device and agent instances also sit outside of CPN tools in the simulation engine.

A major component for managing the message flows and timing of the simulation is the “Round Management and Network” transition. This is a transition to simulate the routing of network communications to the appropriate devices while enforcing the coordination of event processing for each round. When a device or agent sends a response message, it will append to the “Round Results Queue,” which will collect all messages from the round. All actions during a round will leverage the same state space for the round time step, which will be released after the completion of the round. If any conflicts occur during a round, a random number will determine which event happened “first” and send a failure response to the unsuccessful requestor to be evaluated in the following round. Based on our simulations, there has been a negligible operational impact with this design. However, it would need to be revisited in the future.

Another key capability within this transition is to act as an interface to inject events for simulation as well as respond to Internet requests through simulated functions within the simulation engine. For example, events such as update notifications or malicious activity can be triggered here for simulation activity.

Device Artifact

The device artifact is a transition that takes a message as input and processes it as the recipient device. When a message is received, the below action calls the recvMessage() function, which outputs the message to the simulation engine to interpret as the appropriate device.

```
input (msg);
output (msg2);
action
let
    val newmsg = recvMessage(msg)
in newmsg
end
```

The output message for this transition can be directed either to an agent percept or the round results queue for processing in the next round.

Agent BDI

The Agent BDI net is a useful tool to model and visualize the BDI reasoning for an agent. Following the BDI architecture described in previous chapters, the net shown in the figure below demonstrates the agent's BDI capabilities in action with percepts, beliefs, desires, intentions, and actions represented as places that contain tokens for relevant data at any given time. The belief revision function and reasoner are represented as transitions that implement the functions through interfacing with the BDI knowledge graph in Neo4j.

Similar to the device artifact described above, the agent net is a general model that initiates the BDI of the appropriate agent that is acting at any given time. When a message is sent to an agent, the related agent attributes are loaded into the corresponding beliefs, desires, and intentions placed in the net.

```
input (dst);
output(b,d,p);
action
let
  val b = loadB(dst);
  val d = loadD(dst);
  val p = loadP(dst);
in b,d,p
end
```

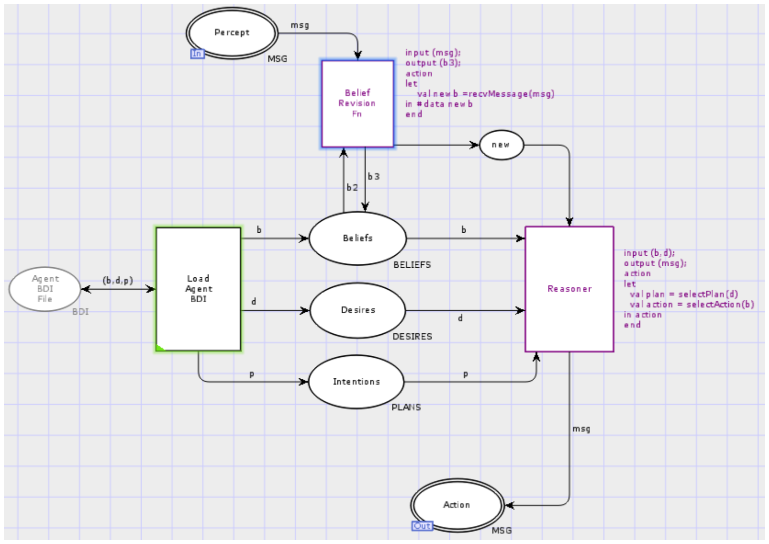
A percept is received in the form of a message and input into the belief revision transition, which performs the function below to update the new beliefs:

```
input (msg);
output (b3);
action
let
    val newb =recvMessage(msg)
in #data newb
end
```

Next, the reasoner transition inputs the new belief, desires, and intentions (plans) to generate a resulting plan and action, as shown in the function below:

```
input (b,d);
output (msg);
action
let
  val plan = selectPlan(d)
  val action = selectAction(b)
in action
end
```

While the CPN walks through all of the logical steps of the environment and agent algorithm and visualizes the data at each step within the corresponding place and transition, the actual intelligence and processing take place outside of the net through the functions called to the simulation engine integration to execute the corresponding functions and execute cypher queries to the Neo4j graphs accordingly. The simulation engine and Neo4j components will be discussed further in the following sections.



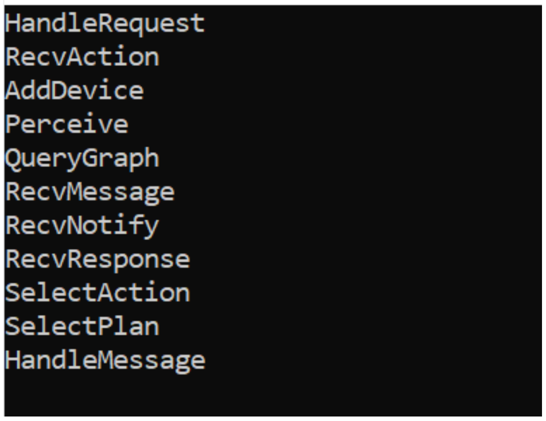
```
input (msg);
output (b3);
action
let
    val newb =recvMessage(msg)
in #data newb
end
```

```
input (b,d);
output (msg);
action
let
  val plan = selectPlan(d)
  val action = selectAction(b)
in action
end
```

Figure 5.4 Agent CPN

5.2.2 Simulation Engine

The simulation engine was developed for the purposes of simulating the device and agent instances for demonstrating our model. While the network environment and round management are orchestrated within CPN Tools, this component hosts the actual instances of the devices loaded into CPN tools as needed. The simulation engine has been developed in Go and performs a set of functions shown in Figure 5.5. The engine sends and receives messages to the appropriate device and agent as the simulation is executed and executes its functions supported by Neo4j graph integration.



```
HandleRequest
RecvAction
AddDevice
Perceive
QueryGraph
RecvMessage
RecvNotify
RecvResponse
SelectAction
SelectPlan
HandleMessage
```

Figure 5.5 Simulation Functions

The simulation engine consists of the below services:

Integration Connectors

The simulation engine integrates CPN tools and Neo4j using the respective connectors to send and receive messages between the platforms.

Device Simulation

This acts in place of a real device for the purposes of our simulation. The device simulation contains the functions to receive and process messages to perform basic actions within the action sets defined in the graph.

Agent Simulation

An instance of each agent is created to maintain an ongoing memory and reasoning of the agent’s beliefs, desires, and intentions throughout the simulation. The agent instance is called by CPN tools as described in the previous section, which is called through the transition for belief revision and reasoning. The agent has the ability to query the BDI Knowledge Graph in Neo4j for belief inferences and applicable plans.

The basic functions within the simulated device and agent instances are sufficient to model a responsive demonstration for the defined actions within our initial action set for the simulation. While this is sufficient to demonstrate the overall architecture implementation of the system, future work would be to integrate with actual devices which can host the agents and execute actions directly. This would require another level of integration beyond the scope of this work.

5.2.3 Neo4j Graph Database

As described in the previous chapter, we make use of Neo4j graph databases for domain knowledge, agent BDI, and environment graphs. Neo4j [89] is a widely used graph data platform that has widespread applications across industries and uses cases such as fraud detection, financial services, life science, data science and knowledge graphs. Neo4j uses Cypher query language, a declarative graph-optimized language for expressive and efficient queries for nodes and relationships in property graphs. For our implementation, we have populated the agent BDI, environment, and domain knowledge graphs within Neo4j to be leveraged for agent reasoning.

Figure 5.6 below shows the merged agent BDI and environment graphs visualized in Neo4j Bloom. As shown in the node label index on the right side of the Figure, agent desires have been created, corresponding to security requirements in the defence policy. Our knowledge graphs have been created to represent BDI relationships for each of the security requirements and corresponding device capabilities related to each device as applicable within the environment graph.

Agents can leverage the graph for belief revision and planning to select appropriate actions based on their beliefs and desires.

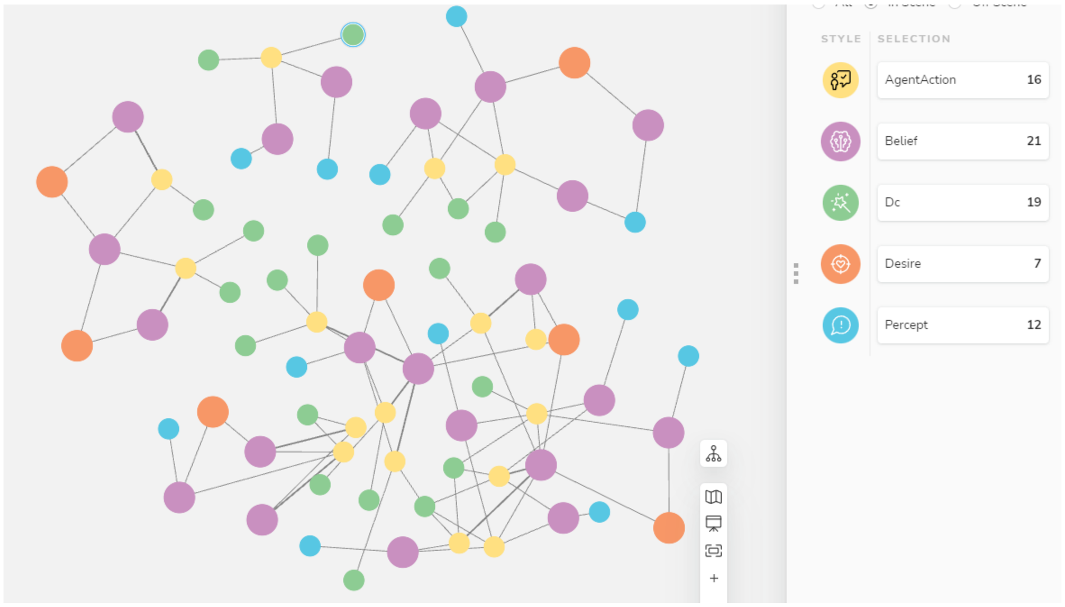


Figure 5.6 Environment and BDI Graph in Neo4j

5.3 Implementation of Control and Coordination

The first component is the control and coordination functions, which maintain a high-level view of the network and security states to direct the overall goals for the rest of the multi-agent system. As discussed in Chapter 3 and further visualized in Figure 5.7 below, the controller begins by taking the security requirements and creating a defence policy to be passed on to the coordinator. Next, the coordinator manages the resources and workflow planning to coordinate and deploy missions for the device agents to act on the environment.

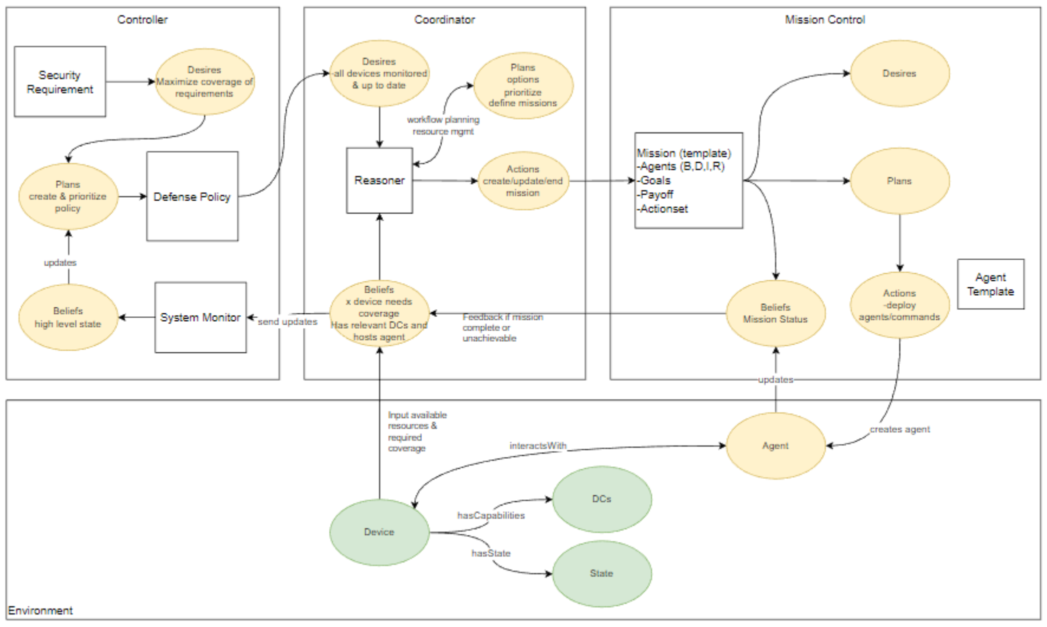

Controller
Security Requirement
Desires Maximize coverage of requirements
Plans create & prioritize policy
Defense Policy
updates
Beliefs high level state
System Monitor
send updates
Coordinator
Desires -all devices monitored & up to date
Plans options prioritize define missions
workflow planning resource mgmt
Reasoner
Actions create/update/end mission
Beliefs x device needs coverage Has relevant DCs and hosts agent
Feedback if mission complete or unachievable
Mission Control
Desires
Mission (template) -Agents (B,D,I,R) -Goals -Payoff -Actionset
Plans
Agent Template
Beliefs Mission Status
Actions -deploy agents/commands
updates
Input available resources & required coverage
interactsWith
Agent
creates agent
Device
hasCapabilities
DCs
hasState
State
Environment

Figure 5.7 Model Components

5.3.1 Defence Policy Generation

As described in Section 5.1.2, we have defined a set of security requirements based on NIST 8259A to be enforced by the system. Based on the security requirements, a defence policy is created by the controller according to the knowledge of the environment, which defines the high-level agent's desires for the system.

The elements of the defence policy are input into the knowledge graph, where the goal of each policy is identified as an agent "desire" associated with the desired system state. Each desire is related to each device or group of devices to which the policy is to be applied.

5.3.2 Coordination

After the defence policy is defined and passed on to the coordinator, it must take the appropriate actions to ensure the policy is enforced. The agent profile and BDI graph for the coordination agent are shown below, where the agent has an action set of CoordinatePolicy() and DeployMission(). The coordination agent maintains an ongoing desire to "CoordinatePolicy," where the objective desired state of each policy is "up to date."

```
Role: "coordinator"
ActionSet: CoordinatePolicy(), DeployMission()
Beliefs: ""
Desires: "CoordinatePolicy"
Intentions: ""
```

As shown in Figure 5.8 below, when a new policy is received without any associated coverage, it is in a state of “new.” Therefore, the coordination agent proceeds to perform actions in pursuit of the desired state of “covered” for each policy by coordinating and deploying the required missions according to the resources available.

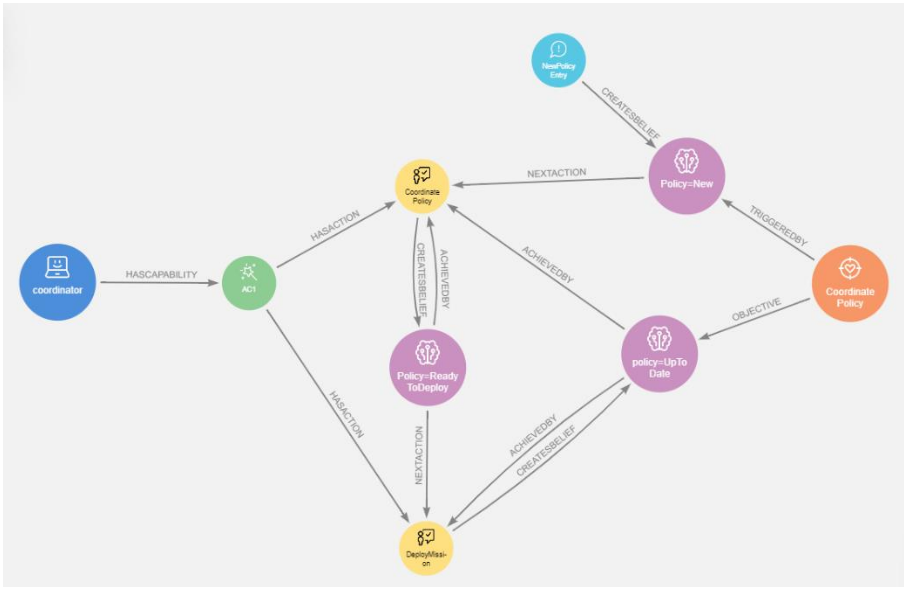


Figure 5.8 Coordinator BDI Graph

5.4 Implementation of Agent Use Cases

This section demonstrates the implementation of two use cases to demonstrate how an agent can deliberate upon different security-focused desires and enforce policies embedded into the BDI knowledge graphs. The first use case demonstrates vulnerability management and patching capabilities, and the second demonstrates access management capabilities.

5.4.1 Use Case 1: Vulnerability Management and Patching

This section demonstrates how a device agent acts within the device to maintain the most up-to-date version of the software. While patch management is a significant concern for the security posture of a home or enterprise network, this example may appear simplistic. However, we have presented it here to demonstrate the basic functions of an agent and how it would receive percepts from the environment, rationalize using the BDI model, and perform actions in pursuit of a goal. The foundation shown through this example can be expanded to more complex capabilities.

Agent Profile and Reasoning

The profile of a basic patching agent's initial BDI is shown below, where the agent has a set of actions available on the host device to installUpdate() and DownloadUpdate(). The agent does not have any initial beliefs or intentions until it receives an initial percept, and it has inherited a desire to “update” based on the defence policy.

```
Role: “deviceAgent”
ActionSet: InstallUpdate(), DownloadUpdate()
Beliefs: “”
Desires: “Update”
Intentions: “”
```

The agent leverages the below subset of the BDI graph for belief revision and planning related to the “Update” desire defined in the policy and mission. For example, the graph in Figure 5.9 shows the relationships between percepts and beliefs, a series of actions based on the possible belief set, available actions based on device capabilities, and expected state outcomes of performing the actions.

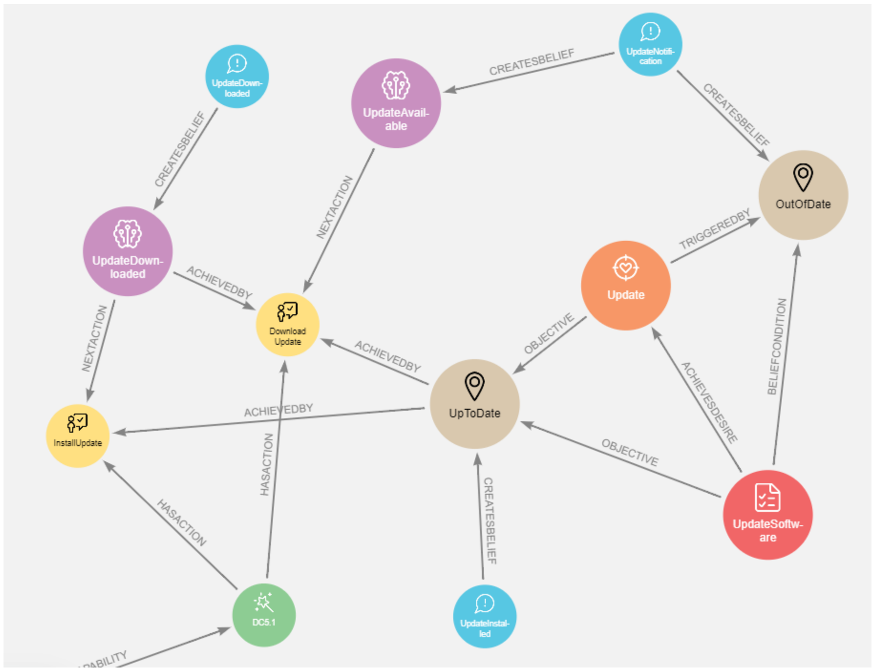


Figure 5.9 Agent BDI Graph for "Update" Desire

Detailed Simulation Walkthrough

Figure 5.10 shows the message format and flow of the scenario, where the agent receives a notification that there is a new software version available. This triggers a series of actions by the agent to pursue its desired state of “up-to-date” software. The agent leverages the plan obtained through the BDI knowledge graph to select an appropriate action based on each percept and updated belief of the state of the environment. Each step will be described further with screenshots as we walk through the implementation in the remainder of this section.

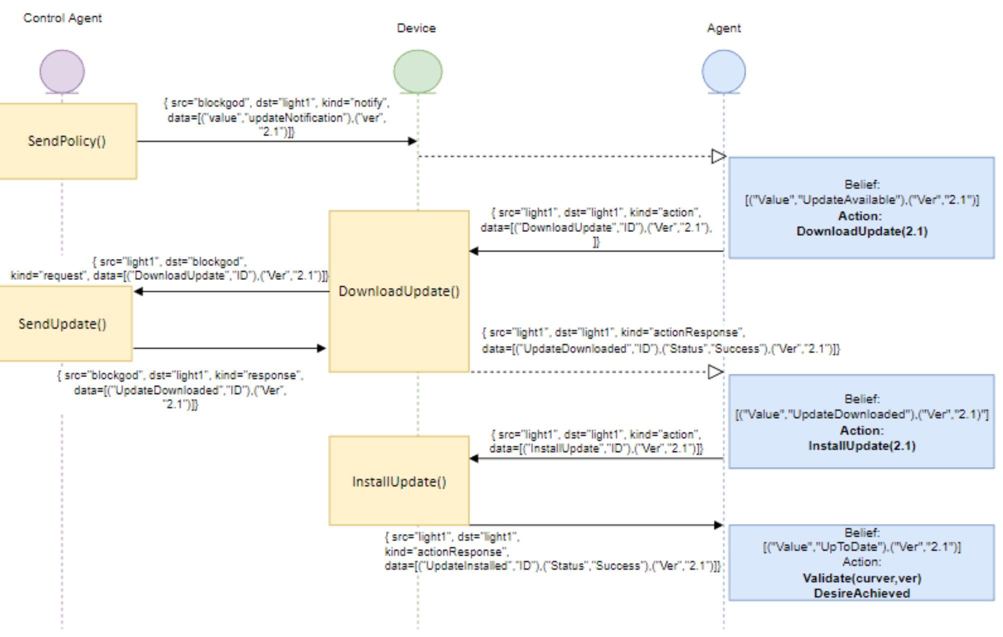

Control Agent
Device
Agent
SendPolicy()
{ src="blockgod", dst="light1", kind="notify", data=[("value","updateNotification"),("ver", "2.1")]}
Belief: [("Value","UpdateAvailable"),("Ver","2.1")] Action: DownloadUpdate(2.1)
{ src="light1", dst="light1", kind="action", data=[("DownloadUpdate","ID"),("Ver","2.1"),]}
{ src="light1", dst="blockgod", kind="request", data=[("DownloadUpdate","ID"),("Ver","2.1")]}
DownloadUpdate()
SendUpdate()
{ src="light1", dst="light1", kind="actionResponse", data=[("UpdateDownloaded","ID"),("Status","Success"),("Ver","2.1")]}
{ src="blockgod", dst="light1", kind="response", data=[("UpdateDownloaded","ID"),("Ver", "2.1")]}
Belief: [("Value","UpdateDownloaded"),("Ver","2.1)"] Action: InstallUpdate(2.1)
{ src="light1", dst="light1", kind="action", data=[("InstallUpdate","ID"),("Ver","2.1")]}
InstallUpdate()
{ src="light1", dst="light1", kind="actionResponse", data=[("UpdateInstalled","ID"),("Status","Success"),("Ver","2.1")]}
Belief: [("Value","UpToDate"),("Ver","2.1")] Action: Validate(curver,ver) DesireAchieved

Figure 5.10 Patching Agent Sequence Diagram

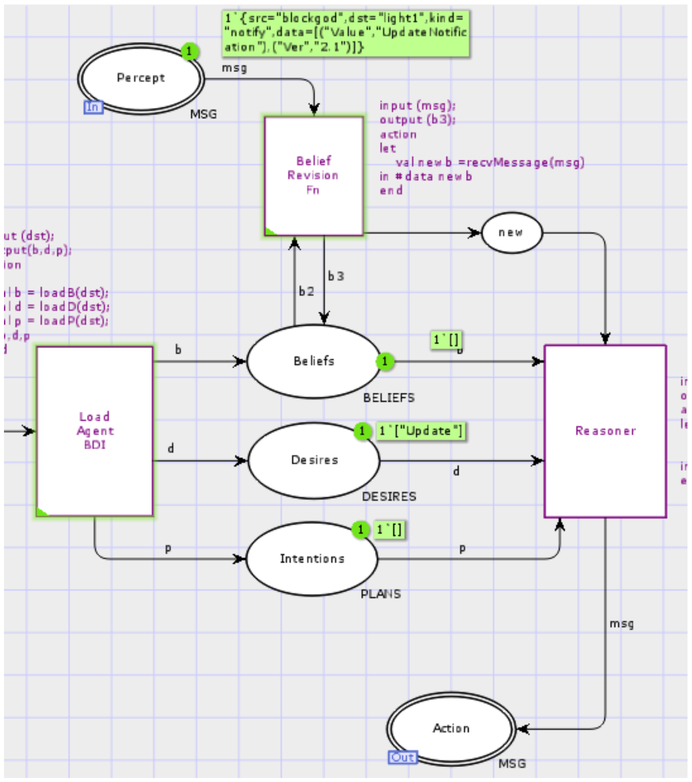
First, a notification is sent to the agent on the device, which is passed to the agent and received as a percept as follows. (a) The agent receives a percept in the form of an update notification message which (b) triggers the belief revision transition to update the agent's belief that the device version is out-of-date, and an updated version is available. The cypher query request and response for the belief revision are shown below:

Query:

```
MATCH (:Percept { Value: "UpdateNotification"} ) -
[:CREATESBELIEF]-> ( beliefs ) RETURN beliefs
```

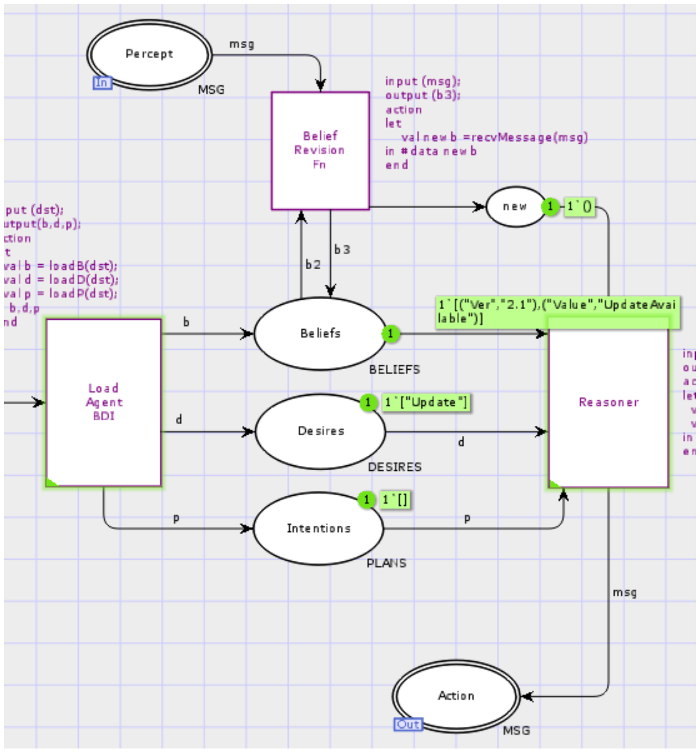
Response:

"beliefs"
{"Value":"OutOfDate"}
{"Ver":"string","Value":"UpdateAvailable"}



```
input (msg);
output (b3);
action
let
    val newb =recvMessage(msg)
in #data newb
end
```

Figure 5.11 (a) update notification received as percept



```
input (msg);
output (b3);
action
let
    val newb =recvMessage(msg)
in #data newb
end
```

Figure 5.12 (b) new belief revision

Next, the reasoner transition is triggered with the input of the new belief and current desire and references to the plans (intentions) stored in the agent's memory. Finally, an action is selected using a cypher query based on the plan and current belief, as shown below:

Plan:

"targetbelief"	"selectedActions"	"preBeliefs"
{"Value":"UpToDate"}	{"Ver":"string","Value":"InstallUpdate"}	{"Ver":"string","Value":"UpdateDownloaded"}
{"Value":"UpToDate"}	{"Ver":"string","Value":"DownloadUpdate"}	{"Ver":"string","Value":"UpdateAvailable"}

Cypher Query for Action Selection:

```
MATCH (:Desire { Value: "Update"} ) -[:OBJECTIVE]-
> ( targetbelief )
MATCH ( targetbelief ) -[:ACHIEVEDBY]-> (selectedAction)
MATCH ( preBeliefs:Belief { Value: "UpdateAvailable"} ) -
[:NEXTACTION]-> (selectedAction)
RETURN selectedAction
```

Response:

"selectedAction"
{"Ver":"string","Value":"DownloadUpdate"}

Through the belief-action pairs retrieved through the query, (c) the agent is able to select an action of "DownloadUpdate(2.1)" which is (d) performed through a request to the host device. Once the next round begins, the agent's request is sent to the host device, which then proceeds to (e) download the update from the internet. Once downloaded, the device (f) sends a successful action response to the agent to inform that the download has been completed.

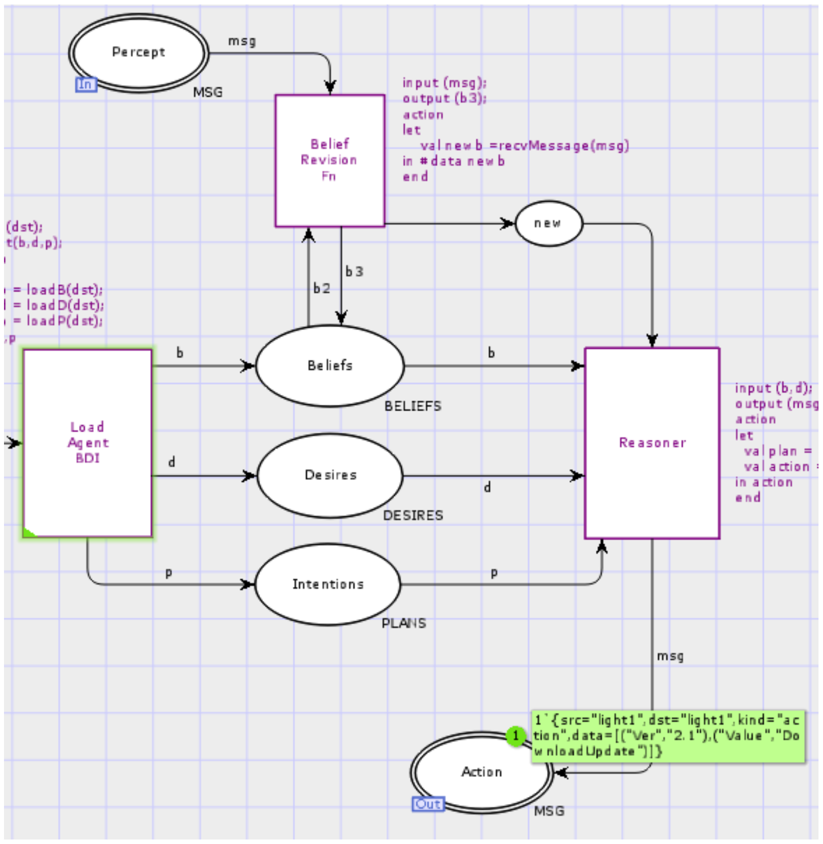


Figure 5.13 (c) action is selected and performed by the agent

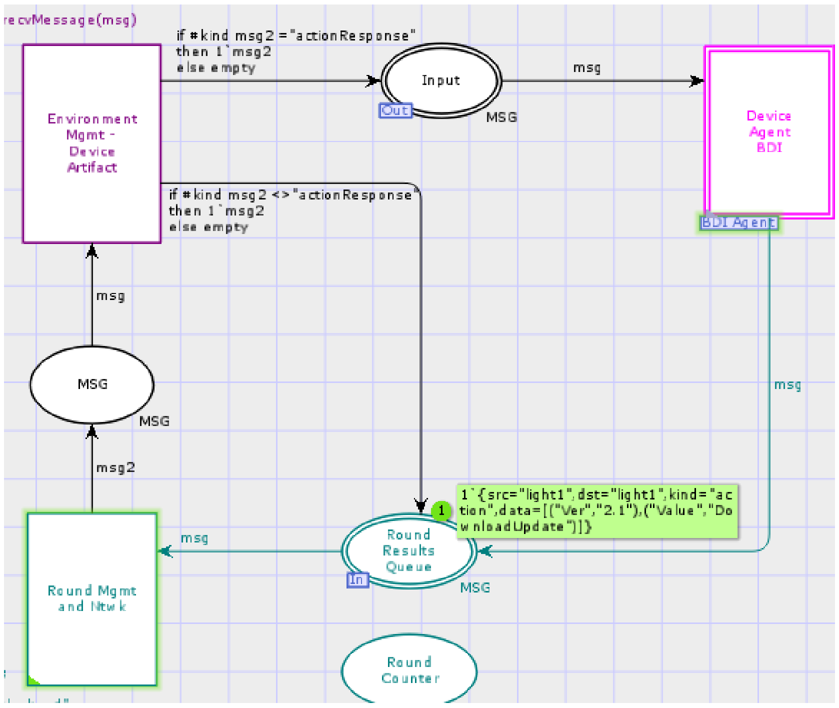

recvMessage(msg)
if #kind msg2 ="actionResponse"
then 1`msg2
else empty
Input
Out
MSG
msg
Device Agent BDI
BDI Agent
Environment Mgmt - Device Artifact
if #kind msg2 <>"actionResponse"
then 1`msg2
else empty
msg
MSG
MSG
msg2
msg
1`{src="light1",dst="light1",kind="action",data=[("Ver","2.1"),("Value","DownloadUpdate")]}
Round Results Queue
In
MSG
msg
Round Mgmt and Ntwk
Round Counter

Figure 5.14 (d) request is sent to the environment round results queue

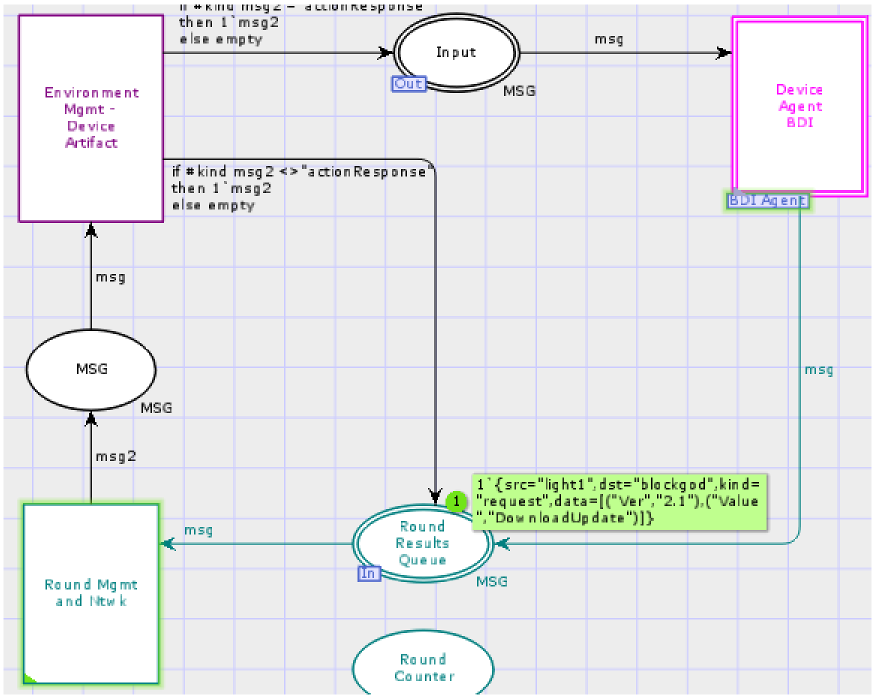

then 1`msg2
else empty
Input
Out
MSG
msg
Device Agent BDI
BDI Agent
Environment Mgmt - Device Artifact
if #kind msg2 <> "actionResponse"
then 1`msg2
else empty
msg
MSG
MSG
msg2
msg
1`{src="light1",dst="blockgod",kind="request",data=[("Ver","2.1"),("Value","DownloadUpdate")]}
1
Round Results Queue
In
MSG
Round Mgmt and Ntwk
Round Counter

Figure 5.15 (e) device downloads the update

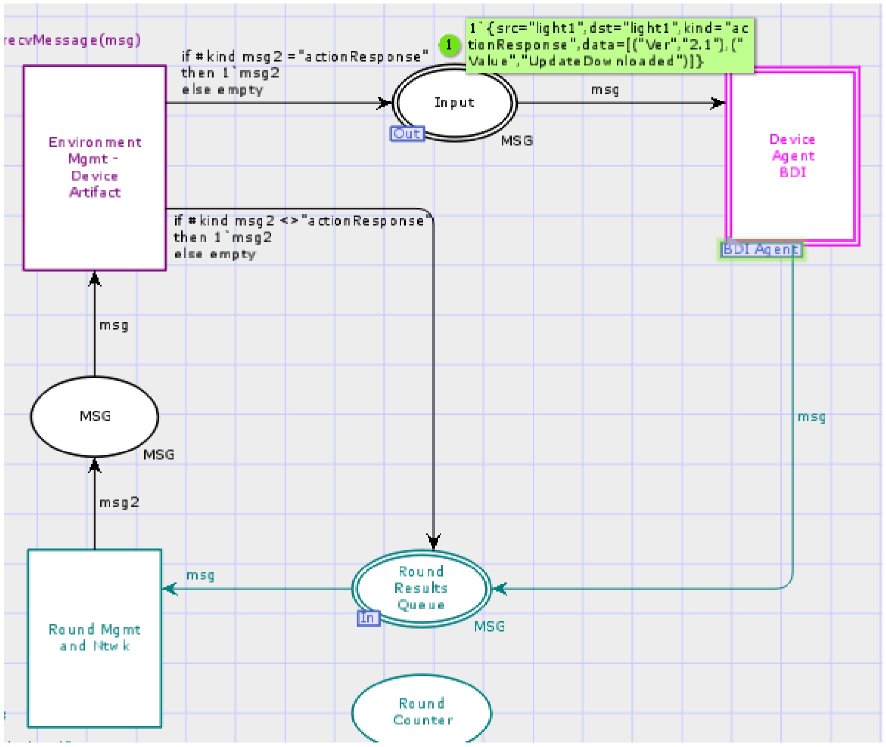
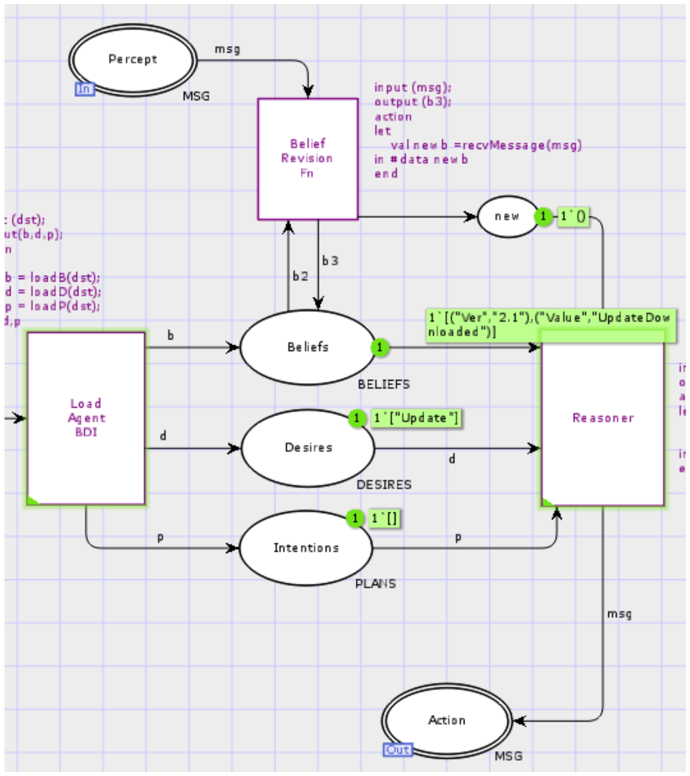

recvMessage(msg)
1`{src="light1",dst="light1",kind="actionResponse",data=[("Ver","2.1"),("Value","UpdateDownloaded")]}
1
if #kind msg2 = "actionResponse"
then 1`msg2
else empty
Input
Out
MSG
msg
Device Agent BDI
BDI Agent
Environment Mgmt - Device Artifact
if #kind msg2 <> "actionResponse"
then 1`msg2
else empty
msg
MSG
MSG
msg2
msg
Round Results Queue
In
MSG
Round Mgmt and Ntwk
Round Counter

Figure 5.16 (f) action response is sent to the agent

The agent receives the response as a new percept and performs its next belief revision accordingly. With the new (g) belief that the update has been downloaded, (h) the agent responds with a subsequent action to install the update, sent again as a request to the host device. Once receiving the request, the host device proceeds to install the update and (i) sends a successful response to the agent. (j) The agent's belief is now updated to reflect that the most current version is installed.



```
input (msg);
output (b3);
action
let
   val newb =recvMessage(msg)
in #data newb
end
```

```
: (dst);
ut(b,d,p);
n

b = loadB(dst);
d = loadD(dst);
p = loadP(dst);
d,p
```

Figure 5.17 (g) belief revision for downloaded update

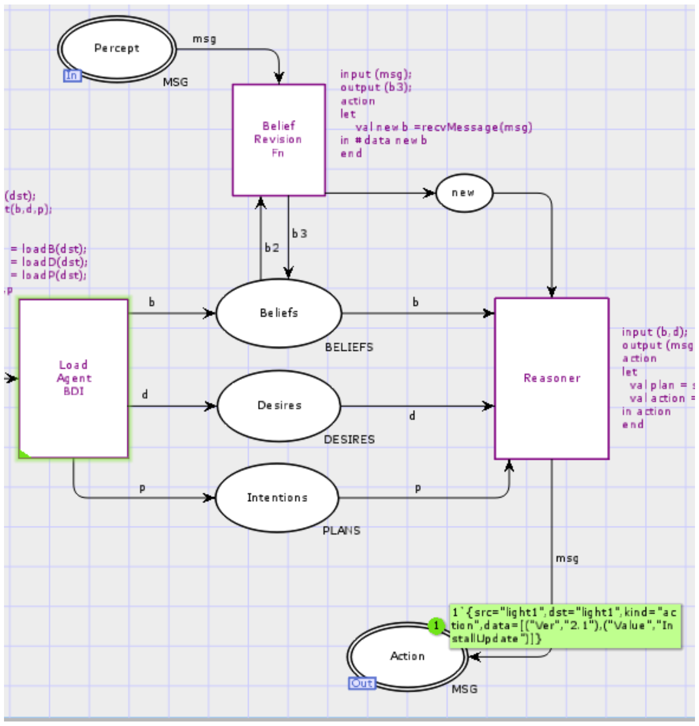


Figure 5.18 (h) the action selected to install an update

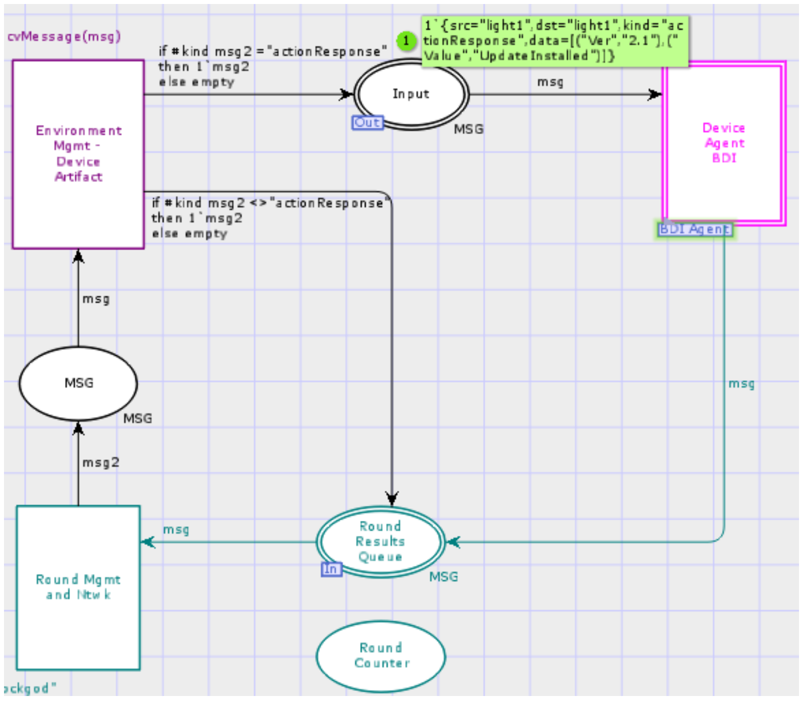
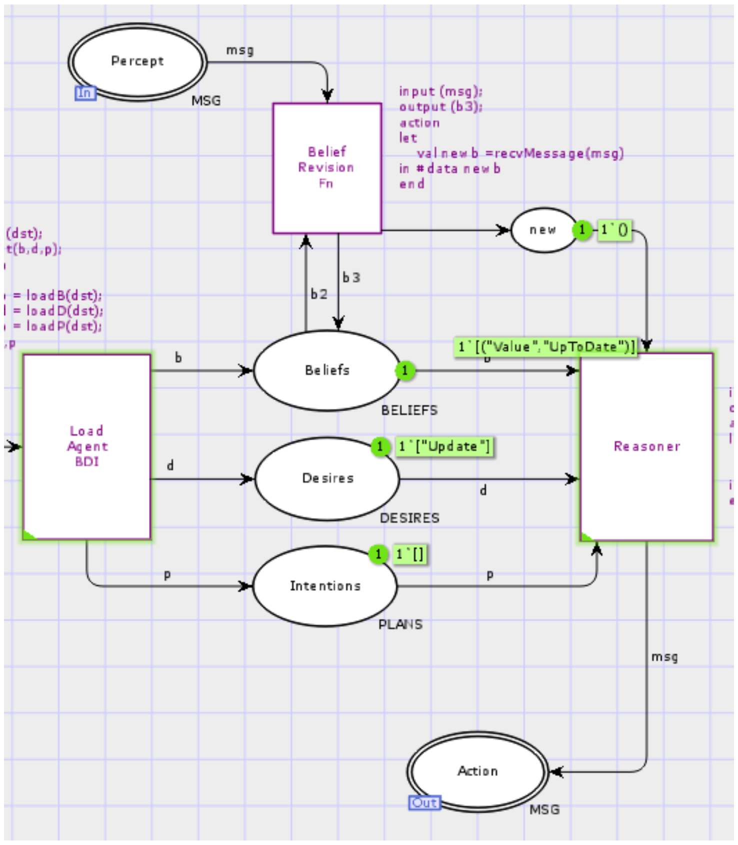

Environment Mgmt - Device Artifact
if #kind msg2 ="actionResponse"
then 1`msg2
else empty
1`{src="light1",dst="light1",kind="actionResponse",data=[("Ver","2.1"),("Value","UpdateInstalled")]}
Input
Out
MSG
msg
Device Agent BDI
BDI Agent
if #kind msg2 <>"actionResponse"
then 1`msg2
else empty
msg
MSG
msg2
Round Mgmt and Ntwk
Round Results Queue
In
Round Counter

Figure 5.19 (i) action response to the agent that update is installed



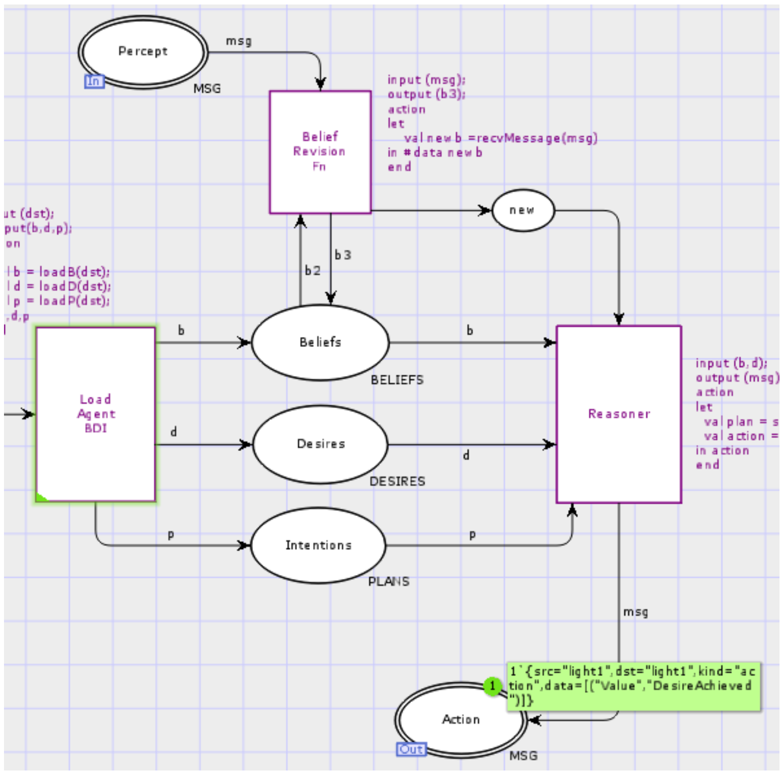
```
input (msg);
output (b3);
action
let
   val newb =recvMessage(msg)
in #data newb
end
```

```
(dst);
t(b,d,p);

 = loadB(dst);
 = loadD(dst);
 = loadP(dst);
,p
```

Figure 5.20 (j) belief revision that software is up to date

Finally, the new belief and active desire are input into the reasoner transition and (k) confirms that its desired state has now been achieved. With no further actions required in pursuit of this desire, the agent can send an update back to the coordinator to update the status of the mission. This data will be tracked within the system monitor for awareness of the security state of the device.



```
input (msg);
output (b3);
action
let
   val newb =recvMessage(msg)
in #data newb
end
```

```
ut (dst);
put(b,d,p);
on

l b = loadB(dst);
l d = loadD(dst);
l p = loadP(dst);
,d,p
l
```

```
input (b,d);
output (msg)
action
let
  val plan = s
  val action =
in action
end
```

```
1`{src="light1",dst="light1",kind="ac
tion",data=[("Value","DesireAchieved
")]}
```

Figure 5.21 (k) "Update" desire has been achieved

As the agent is aware that it has now achieved its current goal and no other actions are required at this time, if a new percept is received to indicate that a new update is available, the process will begin again to continue to maintain the most recent version available. While this demonstration may appear simplistic, it provides a simple example demonstrating the deliberation of just one of many possible agent desires. An agent can maintain multiple desires simultaneously and prioritize the appropriate actions according to its ongoing belief set.

5.4.2 Use Case 2: Access Management

The following use case to be demonstrated is an access control scenario. Access management is one of the core foundations of information security to restrict access to systems and data based on authentication and authorization procedures. Furthermore, with agents deployed throughout the network with access to contextual environment data, access control capabilities provide many opportunities for the enforcement of policies within the multi-agent system.

Access management is central to many functions within security, and our BDI graph contains many relationships to access-related capabilities. It is possible to code access management policies further into the agent BDI graph by creating additional relationships as a prerequisite to other agent actions. This illustrates the effectiveness of the graph database as a platform for modeling complex relationships between actions. Depending on the device capabilities available on each device, agents can perform queries to traverse the graph to find the optimal plan of action based on available resources and security requirements for a particular environment. The graph is also modular to allow additional functions to be added based on evolving capabilities. Figure 5.22 shows the basic BDI graph for access management functions used within our implementation. The access management subset of the graph is related to other functions, such as input validation.

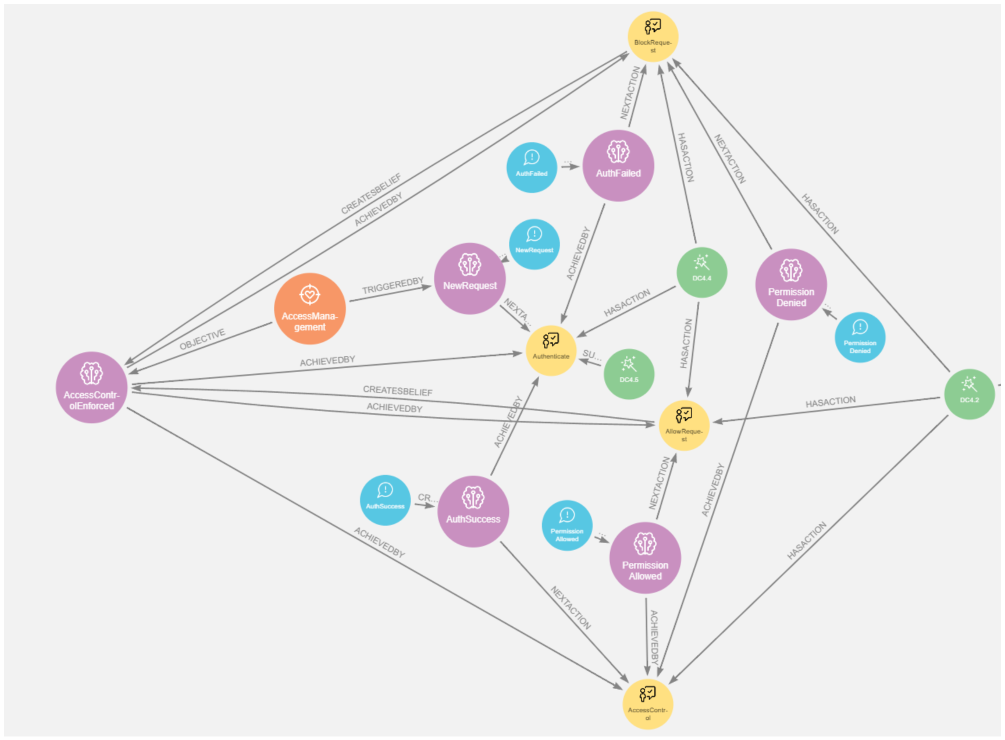


Figure 5.22 Access Management BDI Graph

Access Management can be performed by a device agent as defined below.

```
Role: “deviceAgent”
```

```
ActionSet: AccessControl(), Authenticate(),
AllowRequest(), BlockRequest()
Beliefs: “”
Desires: “AccessManagement”
Intentions: “”
```

While IoT devices may have different capabilities available for access control, an agent can leverage the knowledge graph to determine the best course of action based on the available resources, shared knowledge, and collaboration with other agents or compensating controls. As each agent hosted on a device will evaluate all requests to and from the device, the agent can act as a Policy Enforcement Point (PEP) to enforce policy decisions. Policy decisions are made at the Policy Decision Point (PDP), which can be defined by the controller or coordinated through a mission in response to an incident or compensating control. Alternatively, the PDP can be expanded through additional security services, which could be added in future works.

In this scenario, we will demonstrate how an agent can enforce access control policies based on the available device capabilities. We will show two examples of successful and unsuccessful access requests.

Successful Login Request

This scenario will show a simple example of a successful access request to illustrate the basic access control capabilities where DeviceA sends a login request to DeviceB. As shown in Figure 5.23, DeviceA sends a login request to DeviceB, which processes the request and passes it to the agent hosted on the device. The agent proceeds with its BDI reasoning first to authenticate the request, perform access control, and finally provide an access control decision to successfully allow the request.

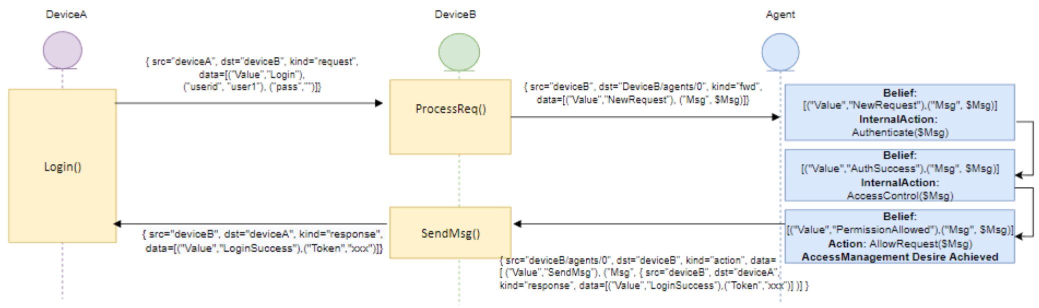

DeviceA
DeviceB
Agent
{ src="deviceA", dst="deviceB", kind="request", data=[("Value","Login"), ("userid", "user1"), ("pass","")]}
Login()
ProcessReq()
SendMsg()
{ src="deviceB", dst="deviceA", kind="response", data=[("Value","LoginSuccess"),("Token","xxx")]}
{ src="deviceB", dst="DeviceB/agents/0", kind="fwd", data=[("Value","NewRequest"), ("Msg", $Msg)]}
Belief: [("Value","NewRequest"),("Msg", $Msg)] InternalAction: Authenticate($Msg)
Belief: [("Value","AuthSuccess"),("Msg", $Msg)] InternalAction: AccessControl($Msg)
Belief: [("Value","PermissionAllowed"),("Msg", $Msg)] Action: AllowRequest($Msg) AccessManagement Desire Achieved
{ src="deviceB/agents/0", dst="deviceB", kind="action", data= [("Value","SendMsg"), ("Msg", { src="deviceB", dst="deviceA", kind="response", data=[("Value","LoginSuccess"),("Token","xxx")])] }

Figure 5.23 Agent Processing Login Request

Denied Command Execution Request

In this scenario, a potentially compromised DeviceA has made a suspicious command execution request to DeviceB on the network. As illustrated in Figure 5.24, DeviceA has obtained a legitimate token according to the previous scenario and has successfully authenticated to DeviceB. However, the device is now attempting to send a command to download a malicious file and is subsequently denied permission to execute the request as per the access control policy. The agent subsequently responds with a deny message back to the host device, which proceeds to deny the requested action.

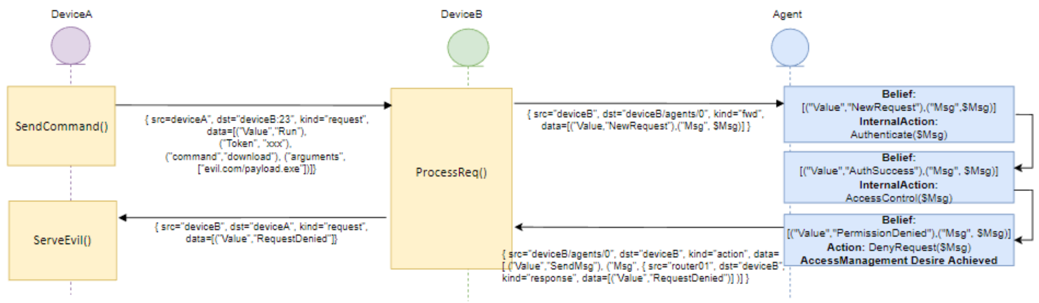

DeviceA
DeviceB
Agent
SendCommand()
ServeEvil()
ProcessReq()
{ src=deviceA", dst="deviceB:23", kind="request", data=[("Value","Run"), ("Token", "xxx"), ("command","download"), ("arguments", ["evil.com/payload.exe"])]}
{ src="deviceB", dst="deviceA", kind="request", data=[("Value","RequestDenied"]}
{ src="deviceB", dst="deviceB/agents/0", kind="fwd", data=[("Value,"NewRequest"),("Msg", $Msg)] }
Belief: [("Value","NewRequest"),("Msg",$Msg)] InternalAction: Authenticate($Msg)
Belief: [("Value","AuthSuccess"),("Msg", $Msg)] InternalAction: AccessControl($Msg)
Belief: [("Value","PermissionDenied"),("Msg", $Msg)] Action: DenyRequest($Msg) AccessManagement Desire Achieved
{ src="deviceB/agents/0", dst="deviceB", kind="action", data= [("Value","SendMsg"), ("Msg", { src="router01", dst="deviceB", kind="response", data=[("Value","RequestDenied")])] }

Figure 5.24 Denied Command Execution Request

While we are showing this as a standalone scenario, access control policies can be adapted in response to situational awareness of security states within the network and can be used as a response to a detected threat.

5.5 Chapter Summary

The results of our implementation described in this chapter demonstrate our model's feasibility for intelligent agent defence capabilities leveraging BDI agents and knowledge graphs. Using our knowledge graphs to model the relationships across environment artifacts and their properties and linking to security domain knowledge provides a rich contextual dataset that can be leveraged by BDI agents to maintain a strong level of situational awareness about the environment state and perform intelligent actions and reasoning for autonomous cyber defence capabilities.

The hierarchical agent model has been designed to limit unnecessary communications between agents while allowing autonomous agent behavior with distributed control and knowledge. Leveraging reusable agent templates and control hierarchies, we can limit unnecessary agent calls to the neo4j graph by retaining appropriate plans within agent memory after they are deployed. Through this design, the control and coordination agents will perform most of the Neo4j requests to the environment and BDI graphs. In contrast, the majority of the decisions and queries would take place during the initialization of the system when the defence policy is being created and missions are coordinated.

Through the demonstration of the current implementation in this chapter, we have shown a smart home environment with 16 devices and 8 security requirements to be enforced by the multi-agent system. While the simulation environment contains synthetic devices and data, our implementation can provide initial results to infer how the model would scale to perform in a real-world scenario.

While Chapters 6 and 7 will provide a more in-depth evaluation of our model and implementation in line with the potential implications and future works, we will summarize some initial limitations and assumptions with the implementation described in this chapter. Our implementation proves the initial design and feasibility of the architecture, and future work would be to implement with real data and devices to evaluate and expand the model for real-life scenarios and events. Due to the simulated environment, device functions have been simplified and limited to basic capabilities for demonstration. Hard coded functions and actions have been modeled, which would be more complex in a real-life environment.

Another significant assumption was that device profiles are available for all devices according to the NIST labeling framework, which has been heavily referenced as a foundation of the BDI graphs. While the labeling system is not yet prominent in the industry as a standard, our model demonstrates a useful use case for leveraging this framework for automation. While this limitation had been strongly considered before making the design choice, several workarounds for a real-world implementation have been considered and will be discussed further in the following chapter, along with other considerations for the potential of the model.

6 Evaluation and Results

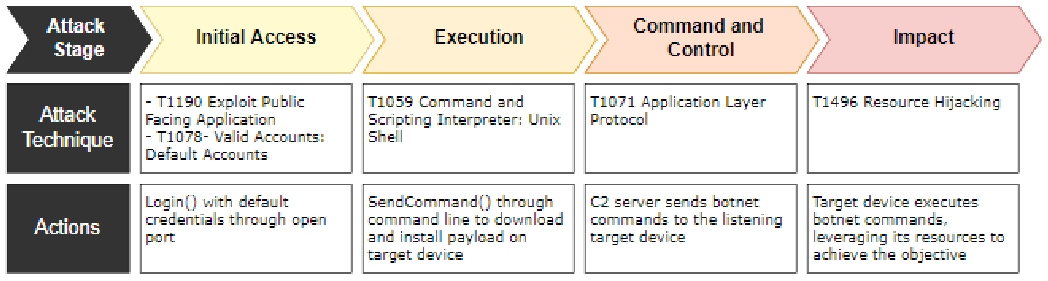
To evaluate the implemented design, this chapter provides an experimental evaluation to demonstrate operational and defence capability performance of the implemented system. We then proceed to evaluate the design against the IoT design challenges and security requirements identified earlier in this work. Lastly, additional considerations are discussed in support of implementation recommendations for future works.

6.1 Experiment Design

A set of experiments have been conducted to evaluate the performance of the proposed architecture in response to an active botnet attack under different environment scenarios with varying device capabilities. Different environment states can exist depending on which capabilities are enabled on the devices and different actions are available to the agents. We proceed to analyze the results to evaluate the operational performance and defence capability performance across each scenario:

- **Operational Performance** – Results evaluating the average response time and memory utilization for the BDI agents.
- **Defence Capability Performance** – Results evaluating the performance of classification of attacks (True Positive Rate, True Negative Rate, False Positive Rate, False Negative Rate, Error Rate, Recall, Precision, F1-Score).

The botnet attack follows a four-stage process targeting the network gateway, as illustrated in Figure 6.1. The attacker first attempts initial access using default credentials on a public-facing port. Upon successful login, a command is sent to download and execute the malicious payload on the target device. Once executed, the device runs a service to listen for commands from the command and control server. Once a command is received, the device resources are used to execute the botnet commands sending malicious traffic to another external target.



Attack Stage	Initial Access	Execution	Command and Control	Impact
Attack Technique	- T1190 Exploit Public Facing Application - T1078- Valid Accounts: Default Accounts	T1059 Command and Scripting Interpreter: Unix Shell	T1071 Application Layer Protocol	T1496 Resource Hijacking
Actions	Login() with default credentials through open port	SendCommand() through command line to download and install payload on target device	C2 server sends botnet commands to the listening target device	Target device executes botnet commands, leveraging its resources to achieve the objective

Figure 6.1 Botnet Attack Stages

Our experiment runs through 256 scenarios implementing different combinations of device capabilities enabled within the environment. Based on the device capabilities that are utilized within the above attack scenario from an exploitation or protection perspective, we have chosen a set of relevant capabilities for our experiment, as described in Table 6.1.

Table 6.1 Capability Descriptions

Type	**Description**	**Capabilities**
Base Capabilities	Base access control-related capabilities	DC4.1 ability to disable interfaces
		DC4.2 ability to restrict access to interfaces
		DC4.4 ability to authenticate
Secure Configuration	Proactive controls	DC4.5 secure auth (no default passwords)
		DC2.3 secure default settings
Enhanced Agent Capabilities	Active agent capabilities to respond to vulnerabilities and threats	AC1 Update Port Settings - review open ports and disable unnecessary ones
		AC2 Endpoint Controls - block weird access and execution
		AC3 Network Controls - block weird traffic

To demonstrate the impact of different device capabilities, the following four sets of scenarios were evaluated, each with different combinations of device capabilities enabled within the environment, as shown in Table 6.2. Set A includes combinations of only base device capabilities for access management. Set B simulates an environment with additional proactive controls for secure default settings, including open ports and secure authentication practices such as changing default passwords. Next, Set C introduces three other capabilities for enhanced agent actions for detection and response, which can make use of the previous capabilities. Finally, Set D iterates through the entire set of remaining combinations. The total number of scenarios run in our experiment is 256, for each combination of the 8 capabilities relevant to this attack scenario (2^8).

Table 6.2 Overview of Experiment Scenarios

Set	**# of Scenarios**	**Base DCs [4.1, 4.2, 4.4]**	**Configuration DCs [4.5, 2.3]**	**Enhanced ACs [AC1, AC2, AC3]**
Set A	8	X		
Set B	24	X	X	
Set C	56	X		X
Set D	168	X	X	X
TOTAL	256			

The simulation environment has been executed on two devices: a Linux virtual machine hosting the Neo4j graph database, and a Windows laptop running the CPN Tools application and simulation engine. The technical specifications of the simulation environment are shown below:

Table 6.3 Specifications of Devices Used in Simulation Experiment

Device	**OS**	**Processor**	**RAM**	**Application Info**
Neo4j Server – Virtual Machine	Burmilla 4.14.248	AMD Ryzen 9 3900X, 4267 Mhz, 12-Core (1 allocated)	4GB	Neo4j Enterprise 4.4.4 • Database size: 1.02 MiB • Nodes: 190 • Properties: 125 • Relationships: 175
Laptop	Windows 10.0.19044	AMD Ryzen 9 5900HS Radeon Graphics, 3301 Mhz, 8 Cores, 16 logical processors	16GB	CPN Tools 4.0.1

6.2 Operational Performance Evaluation

This section evaluates the operational performance of our simulation based on memory utilization and the average response time of agents. The following results have been generated from 179 measured agent queries that had been executed for access control and authentication capabilities within the simulations of the botnet scenario described in the previous section.

6.2.1 Average Response Time

As agent behaviors are directed through the BDI knowledge graph, Neo4j cypher queries are performed to retrieve the appropriate rational inferences and corresponding actions. We evaluate the performance of the average query time for agent plan selection, belief revision, and action selection, shown in Table 6.4 and illustrated in Figure 6.2 and Figure 6.3 below, from a sample of request types.

Table 6.4 Response Time Thresholds (ms)

Request Type	**Avg**	**Min**	**Max**	**Count**
Plan Selection	2.8	0.96	5.67	37
Belief Revision	1.7	0.7	4.33	71
Action Selection	1.2	0.84	3	71

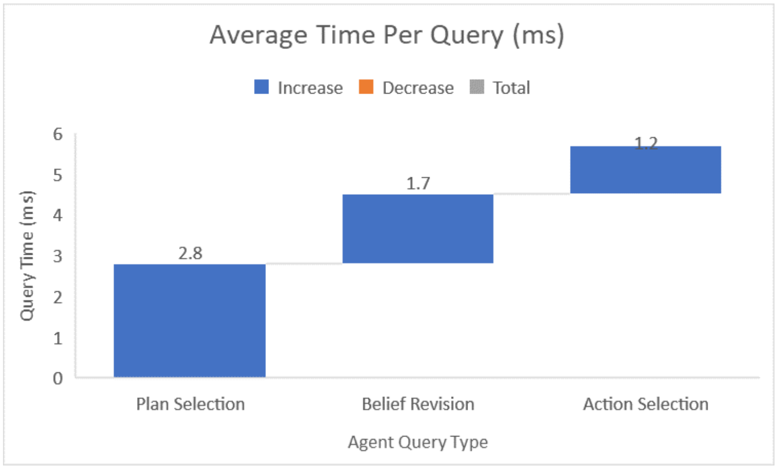
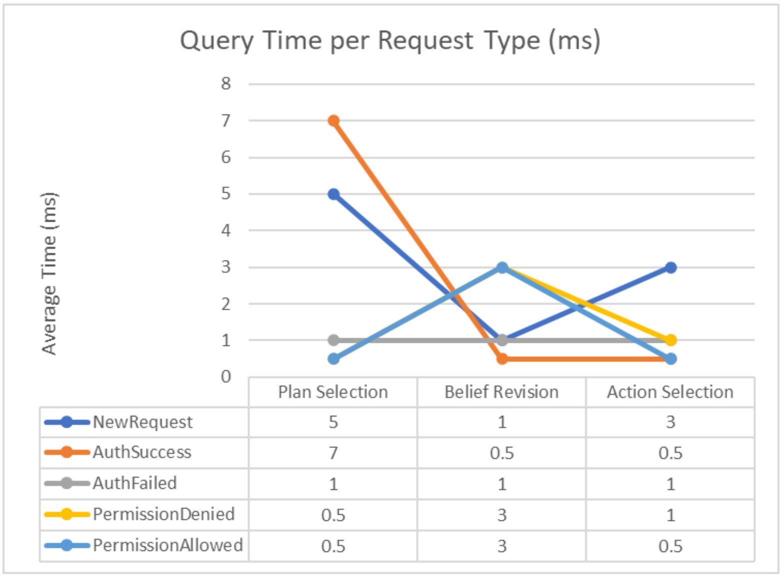


Figure 6.2 Average Time Per Query (ms)



	Plan Selection	Belief Revision	Action Selection
NewRequest	5	1	3
AuthSuccess	7	0.5	0.5
AuthFailed	1	1	1
PermissionDenied	0.5	3	1
PermissionAllowed	0.5	3	0.5

Figure 6.3 Query Time Per Request Type (ms)

While plan selection has the highest average query time of 2.8 ms, our implementation has been designed to optimize the performance by limiting the number of plan selection queries to the agent initialization stage. With this design, plan selection will only need to be performed when the agent is initialized or if there are any updates to the capabilities or desires. The graph for the plan is stored within the agent's memory going forward, so subsequent decisions will only need to be performed through local queries for belief revision or action selection within this graph.

Agent actions leverage the device resources and are executed through messages. Due to this, the agent response time is very lightweight at 0.25ms to receive and send messages from the host device. After the command is sent to the device to execute, the operational performance of the implementation of specific actions is highly variable, depending on the implementation of external functions and device resources. However, this is out of the scope of the direct agent response time measurement.

The Average Response Time (ART) in Equation (2) [95] evaluates the performance of a BDI agent's average response time from when an event occurs to when the agent's response is completed. The below formula is used to calculate the ART, where S_1 is the average detection time for the agent's beliefs are revised based on a received percept, S_2 is the average deliberation time where an intention is created for a selected action, and S_3 is the average time needed to execute the action. WT_1, WT_2, and WT_3 represent the average waiting times for the detection, deliberation, and intention queues, respectively.

$$ART = WT_1 + S_1 + WT_2 + S_2 + WT_3 + S_3 \qquad (2)$$

As described above, the values of S1, and S2 have been measured at 1.7ms and 1.2ms, respectively. S_3 and WT_1 are each 0.25ms based on the average time to receive an event message and execute an action. While the current implementation manages the event queue sequentially through CPN tools, the waiting times W2 and W3 are 0ms as the agent does not currently receive enough messages to accumulate into a queue.

We have calculated the agent ART in our simulation as 3.9ms, as shown in Table 6.5 below.

Table 6.5 Average Response Time for Agents (ms)

S_1	**S_2**	**S_3**	**WT_1**	**WT_2**	**WT_3**	**ART**
1.7	1.2	0.25	0.25	0	0	3.9ms

6.2.2 Memory Utilization

Measurements of memory size for agents have been captured through our experiments to show the impact of agent resources on an IoT device. The agent design has been optimized to reduce plan selection time and network overhead by storing the agent plans in memory after initialization. The average size of an agent independent of its plans is 158 bytes. Memory requirements increase based on the number of plans, as shown in Figure 6.4, where the average plan size is 153 bytes. An agent with all plans currently in the 190-node database reaches a maximum memory size of 1.21KB. While an agent will likely only have 0-2 plans active at a given time according to its active desires, this maintains a fairly low memory utilization rate of 311 bytes for an agent with one active plan. Figure 6.5 shows the average expected memory size by a number of plans.

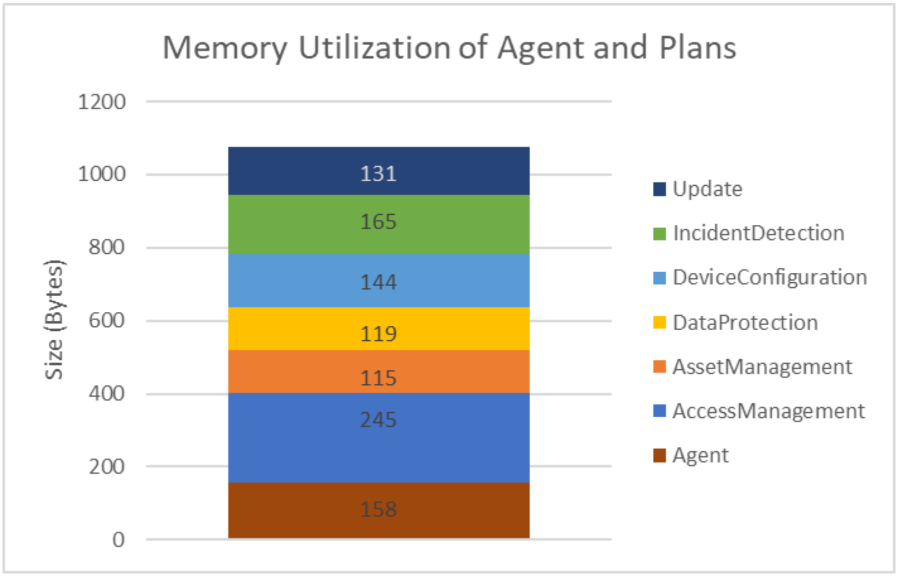


Figure 6.4 Memory Utilization of Agent and Plans

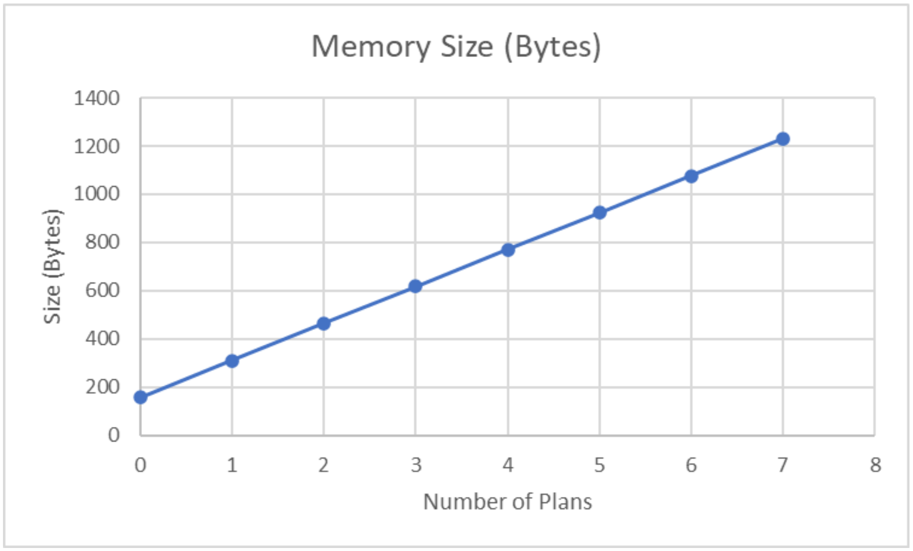


Figure 6.5 Memory Size By Number of Plans

6.2.3 Baseline Comparison

The memory utilization results can be compared to a baseline of memory specifications to understand the resource impact expected on a set of IoT devices that can be found in a smart home. The devices identified in Table 6.6 range from

256 MB to 4 GB of memory, with the exception of the Raspberry Pi Pico at the lowest with 264 KB, and Raspberry Pi 4B at the highest with 8 GB of memory. The maximum current memory size of 1.21 KB would operate on less than 0.0005% of a device with 256 MB memory and less than 0.5% on the 264KB Pico.

Table 6.6 Memory Specifications for Sample IoT Devices

Device	**Model**	**Memory**
Google ChromeCast [96]	1	512 MB
	2	512 MB
Raspberry Pi [97]	B	256 MB/512MB
	A	256 MB
	B+/A+	512 MB
	2 B	1 GB
	Zero/W/WH/2W	512 MB
	3 B/B+	1 GB
	3 A+	512 MB
	4 B	1/2/4/8 GB
	4 400	4 GB
	Pico/W	264 KB
Amazon Echo Plus [98]	2nd Gen	1 GB
Ubiquiti Dream Machine Router [99]	UDM	2 GB DDR RAM / 16GB flash
Asus RT Router [100]	RT-AC88U	516 MB RAM / 128 MB flash

These results indicate a very low expected resource impact to memory utilization on many IoT devices that can be found within a smart home network. Meanwhile, for lower capacity devices such as sensors and lightbulbs, coverage can be distributed accordingly to devices with higher capabilities as needed.

6.3 Defence Capability Evaluation

This section evaluates the performance of each scenario in protecting against the botnet attack from a security perspective. We analyze how different capabilities impact susceptibility to attack scenarios and highlight how agents can create plans to introduce proactive or reactive controls to improve security posture.

6.3.1 Evaluation Metrics

Confusion matrix performance measurements are used to evaluate the results of each scenario's ability to protect against the botnet attack. These metrics are commonly used in machine learning and cybersecurity domains to measure classifier performance of detections/predictions and their accuracy. The confusion matrix classifications are defined as follows [101]: True Positive (TP) is an instance that is positive and is classified as positive, while False Positive (FP) is a negative instance that is incorrectly classified as positive. True Negative (TN) is a negative instance that is correctly classified as negative, while False Negative (FN) is a positive instance that is incorrectly classified as negative. The True Positive Rate (TPR) shows how many detections were actual attacks, while the False Positive Rate (FPR) indicates detections that were not. Further, the False Negative Rate (FNR) is significant to show the attacks that were not detected, and True Negative Rate (TNR) indicates benign activity that was accurately categorized. The formulas for each are defined below in Equations (3) through (6) [101]:

$$True\ Positive\ Rate\ (TPR) = \frac{TP}{TP + FN} \tag{3}$$

$$False\ Positive\ Rate\ (FPR) = \frac{FP}{TN + FP} \tag{4}$$

$$True\ Negative\ Rate\ (TNR) = \frac{TN}{TN + FP} \tag{5}$$

$$False\ Negative\ Rate\ (FNR) = \frac{FN}{TP + FN} \tag{6}$$

Next, we calculate Recall and Precision. Recall indicates the percentage of total positive rates that were predicted as positive. Precision indicates the percentage of true positive rates out of the total positive predicted values. The formulas for Recall and Precision are defined below [101]:

$$Recall\ (R) = \frac{TP}{TP + FN} \quad (7) \qquad Precision\ (P) = \frac{TP}{TP + FP} \quad (8)$$

Finally, the F1 Score can be calculated based on Precision and Recall. F1 Score is used when FN and FP are most important as evaluation criteria, as is the case in most security scenarios [101]:

$$F1\ Score\ \frac{2 * (P * R)}{P + R} \quad (9)$$

Further, we also consider the maximum attack stage achieved, as the risk is more significant as the attack is allowed to progress to advanced stages.

6.3.2 Evaluation Results Per Scenario

We proceed to generate the confusion metrics for each set based on the results. The table below shows the summary of each set and is further illustrated per scenario in the subsequent figure.

Table 6.7 Confusion Metrics Per Set

Set	**FNR**	**TNR**	**TPR**	**FPR**	**Recall**	**Precision**	**F1 Score**	**AVG Attack Stage**
Set A	100%	0%	0%	0%	0%	0%	0%	100%
Set B	0%	100%	100%	0%	100%	100%	100%	0%
Set C	25%	75%	93%	0%	82%	93%	86%	25%
Set D	0%	100%	100%	0%	100%	100%	100%	0%

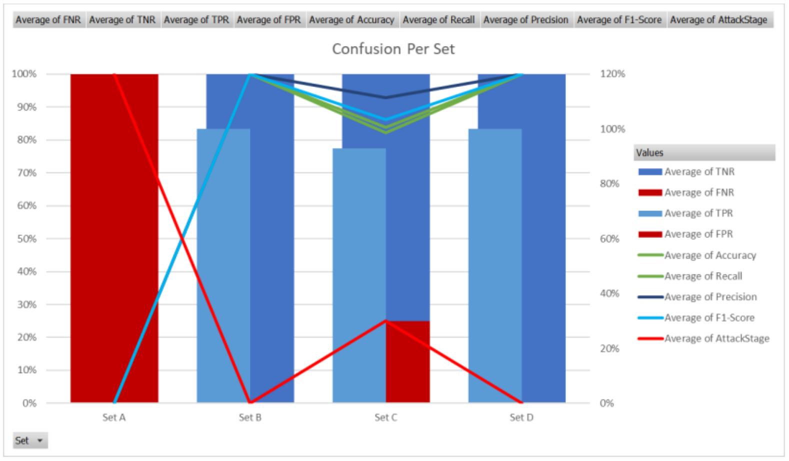


Figure 6.6 Confusion Metrics Per Set

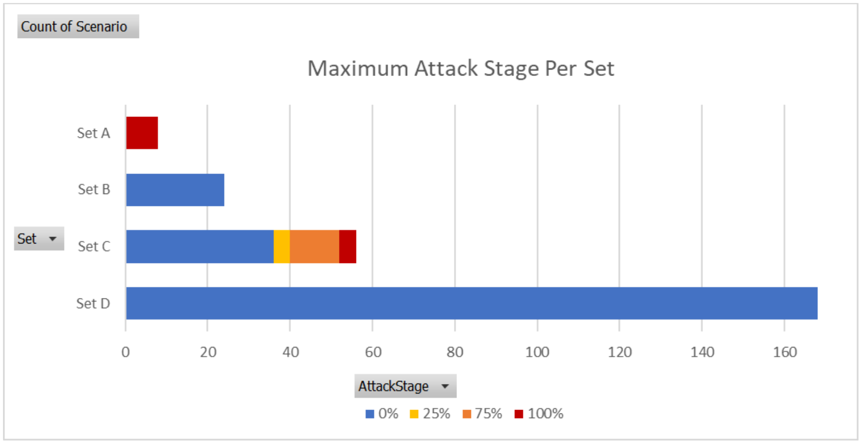


Figure 6.7 Maximum Attack Stage Per Set

We can see that the capabilities tested within Set A are insufficient for protection against the botnet attack, as expected, due to the nature of the attack making use of default passwords that will bypass authentication and authorization mechanisms. The false negative rate is 100% as the attack had succeeded in all scenarios, resulting in a maximum attack stage of 100 (full impact).

Clearly, additional capabilities will be required to better protect the environment. We introduce two additional capabilities (DC 4.5 and 2.3) in Set B, which represent proactive controls. Secure default configurations ensure that unnecessary interfaces are disabled and secure passwords are enforced by not allowing default credentials to be used. We can see the strong impact of these capabilities clearly, with an F1-Score of 100%, and no attack stages are achieved.

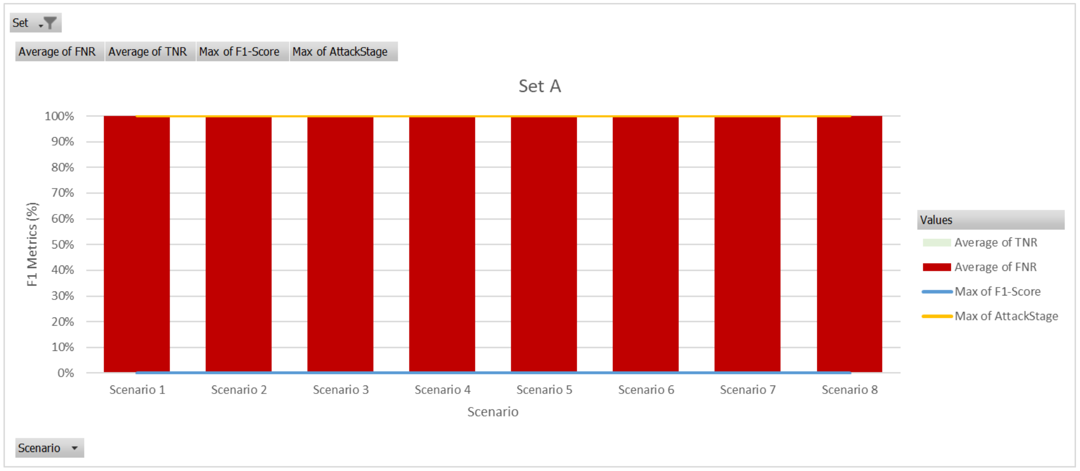


Figure 6.8 Results for Set A

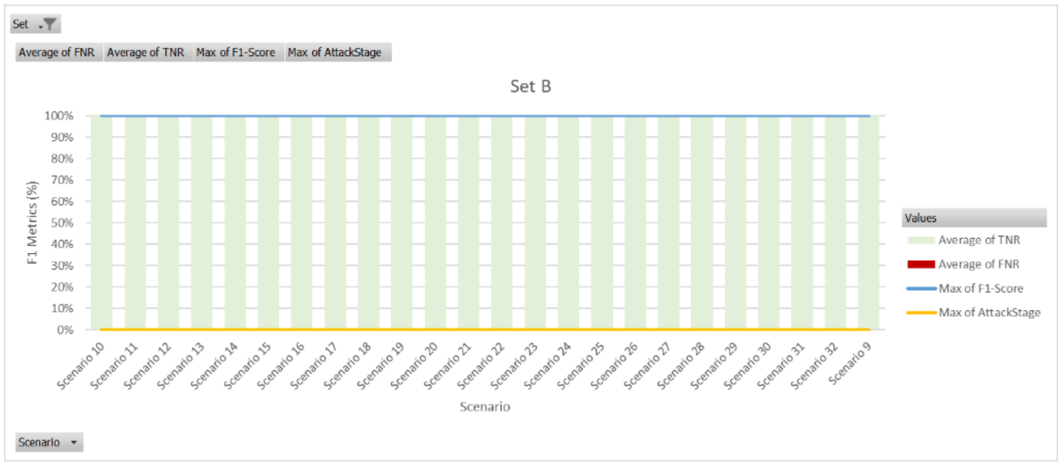


Figure 6.9 Results for Set B

While the capabilities introduced in Set B show such a strong success rate, we chose to remove them from the following set of scenarios to better observe the impact of the next set of capabilities. Set C investigates another approach, making use of responsive agent controls (AC1, AC2, AC3), which support real-time monitoring and response to detected security vulnerabilities and threats. We can see the most interesting results from this set, where there is a more significant variance in F1-Scores, and attack stages achieved depending on the different combinations of enabled capabilities. From the 20 out of 56 attacks that were successful, we can see a distribution of different attack stages achieved, where only 4 had achieved full impact. The remaining 16 attempts had bypassed the initial attack stages. However, it had been blocked after the initial impact or command and control. We will provide additional analysis of these findings by capability in the following section.

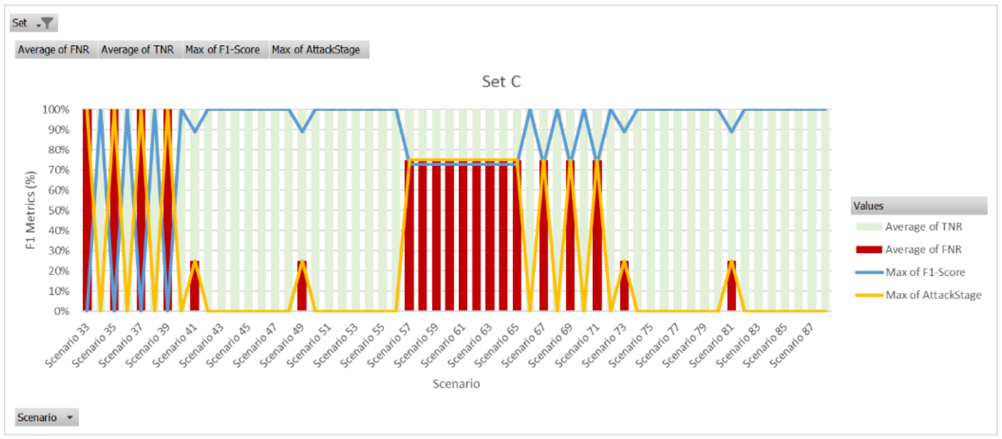


Figure 6.10 Results for Set C

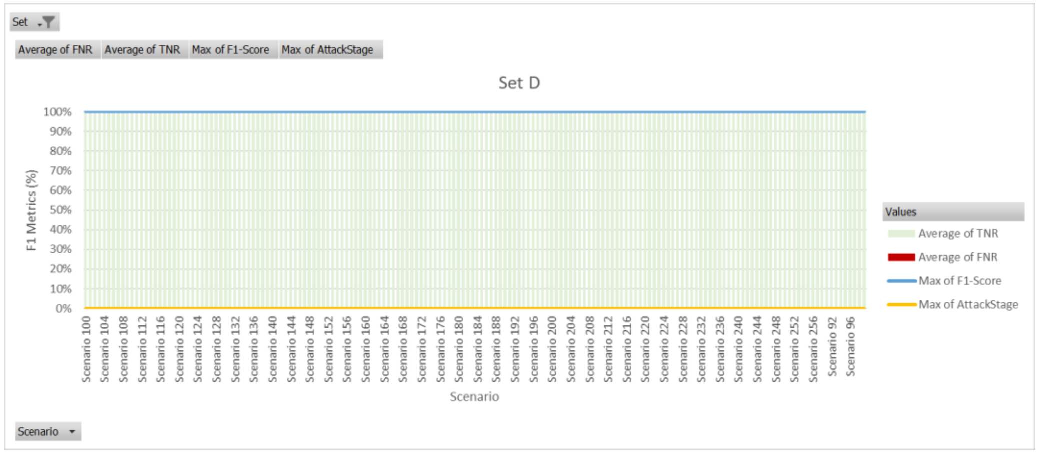


Figure 6.11 Results for Set D

Finally, Set D introduced the remaining combinations of combinations between Set B and Set C. As shown, the proactive controls from Set B prove to be effective in compensating for the limitations of the Set B capabilities and have once again achieved a 100% F1-Score with no successful attacks.

6.3.3 Evaluation Results per Capability

While the previous section provided a staged approach to evaluation per set, we now proceed with a statistical analysis of the individual impact per capability. As we observed from Set B, there are some capabilities that have been able to perform 100% successfully independently of other capabilities being enabled. Figure 6.12 illustrates a view of the enabled capabilities and corresponding F1-Score for each scenario.

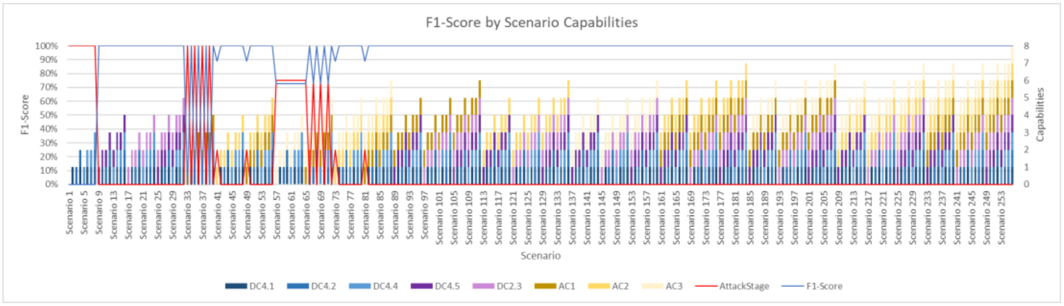


Figure 6.12 F1-Score and Capabilities per Scenario

The summarized results per capability are shown in Table 6.8 and illustrated in Figure 6.13, where it is clear that DC4.5 and DC2.3 have the highest independent success rates. Other capabilities are dependent on others to support effective protective controls.

Table 6.8 Results Per Capability

	FNR	TNR	TPR	FPR	Max Attack Stage	Recall	Precision	F1-Score
DC4.1	5%	95%	97%	0%	100%	95%	100%	97%
DC4.2	8%	92%	95%	0%	100%	92%	100%	96%
DC4.4	8%	92%	95%	0%	100%	92%	100%	96%
DC4.5	0%	100%	100%	0%	0%	100%	100%	100%
DC2.3	0%	100%	100%	0%	0%	100%	100%	100%
AC1	6%	94%	97%	0%	25%	94%	100%	97%
AC2	1%	99%	100%	0%	25%	99%	100%	100%
AC3	1%	93%	100%	0%	75%	99%	100%	100%

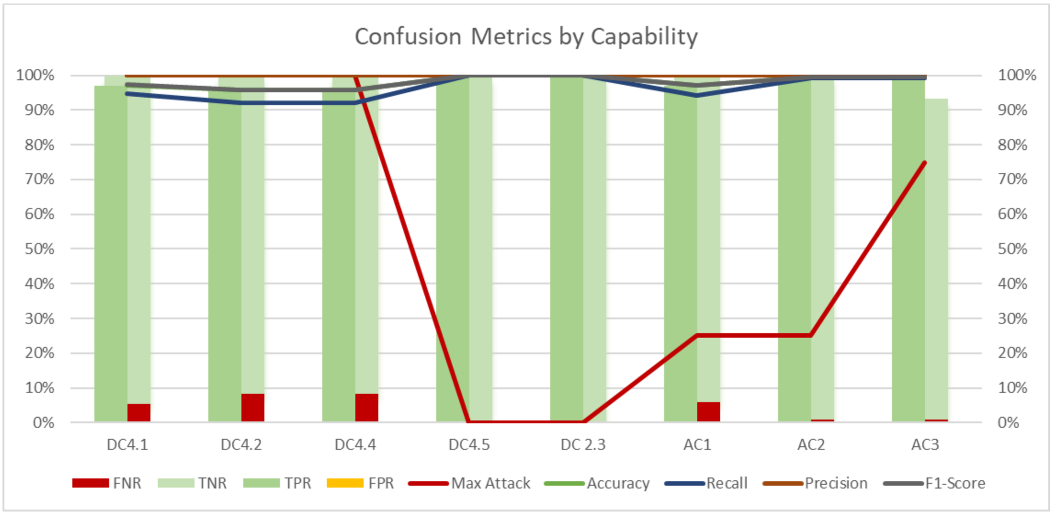


Figure 6.13 Confusion Metrics by Capability

Despite the lower scores for ACs 1-3, these capabilities provide effective protection against advanced attack phases, as shown in Figure 6.14, where many of the scenarios with these capabilities enabled did not progress further than Initial Access or CNC.

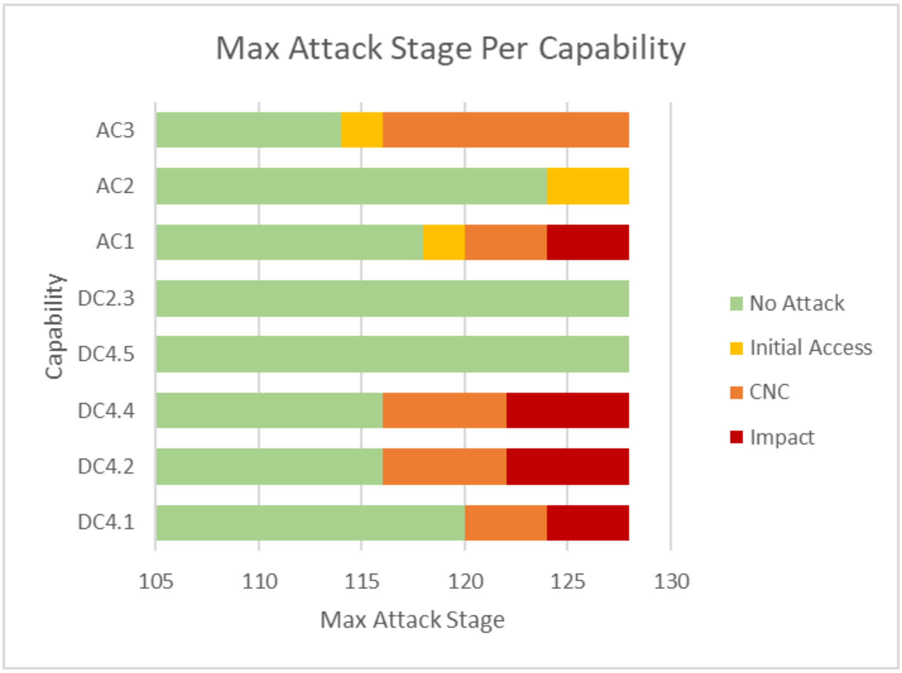


Figure 6.14 Maximum Attack Stage Per Capability

Finally, an analysis of results by a number of enabled capabilities shows a correlation between the number of capabilities and performance. This is aligned with the probability distribution of capability performance rates evaluated earlier in this section.

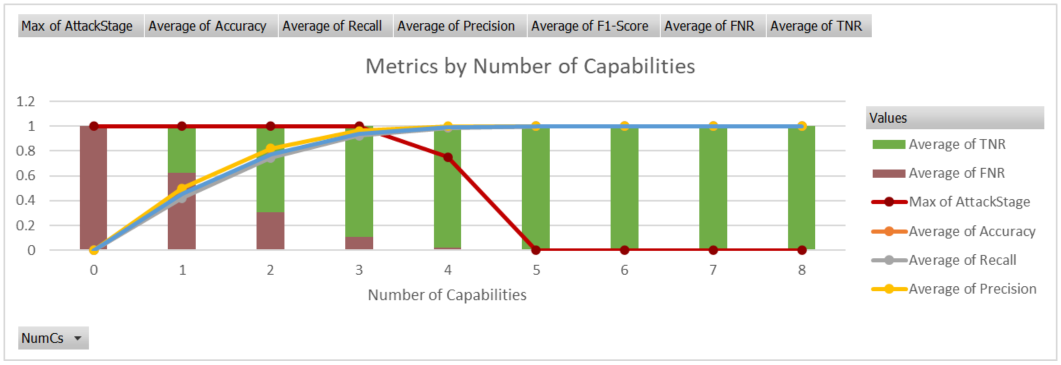


Figure 6.15 Results by Number of Capabilities

6.4 Evaluation of Design Requirements

To evaluate our model, we will proceed to reflect on how its design effectively addresses the requirements outlined in Chapter 2 to support (1) the architectural challenges for IoT security and (2) the security requirements for smart home environments.

6.4.1 Addressing Design Challenges

The proposed architecture has been developed with a vital consideration for the design challenges for IoT systems outlined in Chapter 2: resource limitations, interoperability, reliability and error handling, data volume and sensitivity, and ease of use. We describe each of the challenges below in relation to our solution:

- **Resource Limitations:** The multi-agent architecture allows for adaptive hierarchies according to the environment and resource capabilities. Controller and coordinator agents can be generated as needed for a centralized, distributed, or partially distributed model according to the requirements of the system. While some IoT devices on the network may have limited resources with limited ability to host agent capabilities, these limitations would be known to the system, and an appropriate control structure would be deployed as needed. Further, these limitations would be known by the control agent according to the environment graph, where implications to the situational awareness and security capabilities would be inferred to identify appropriate compensating controls.

- **Interoperability:** The agent model is device agnostic, as the majority of the capabilities operate on a higher layer with a common language for agents. We operate with an assumption of a framework that device vendors can integrate the agent technology through middleware on the device. For devices that do not host an agent, agents on other devices or network devices can perform compensating actions.
- **Reliability:** The system monitor maintains a view of the health of the system from the control level and can respond accordingly. The highly flexible distributed architecture can adapt to resource limitations or outages and deploy agents optimally across the network as available.
- **Data Volume:** The agent hierarchy allows for effective management of data by maintaining a semi-distributed knowledge base where the controller and coordinators have a full view of the network, and device-level agents with lower capabilities are responsible only for a limited portion of the network according to their capacity. This model can be scalable and adapt to different deployments as necessary.
- **Ease of Use:** The autonomous and context-driven design of our framework reduces the level of effort required from the user to maintain the security of their network, as agents can adapt to different situations and prepare and respond to security threats largely autonomously while balancing the availability of services to the user. While there may be cases where user notifications and/or intervention may be required, the communication mechanism can also be designed for ease of use.

6.4.2 Addressing Security Requirements

Our framework provides visibility and coverage of security requirements through modeling the environment, cybersecurity domain knowledge, and BDI reasoning into knowledge graphs. Through this method, we establish security policies as code into agent behaviours, which are subsequently embedded into the environment genome as social agents are deployed across the network. The requirements are identified as BDI agent desires with relationships to appropriate belief states and actions to establish contextual plans. The security requirements are enforced in a scalable, pervasive, autonomous and context-based way by the multi-agent system BDI architecture.

As outlined in Chapter 2, the key security requirements for smart home IoT systems are confidentiality, integrity, availability, authentication, and privacy. While the framework is scalable to address any security requirement that can be input into the knowledge graph, we leveraged the Baseline Security Requirements for IoT Devices defined by NIST [26] to identify actionable security requirements to achieve these objectives. Each of these requirements is then mapped to capabilities that are related in the graph to the environment and assets for a view of coverage. In combination with the cybersecurity domain knowledge base, we can further model each of these requirements and capabilities in relation to the environmental context.

While this model is highly data-driven and dependent on a data model and data inputs through, there are several emerging opportunities with standardized data mapping initiatives that support this architecture:

- **The NIST cybersecurity labeling framework**, while primarily intended for user informational purposes, introduces a common standard for labeling consumer IoT device capabilities which can be further applied to a codified data model. This labeling standard applied at scale to consumer IoT devices will provide standard data input for device capabilities. It can be easily integrated into the environment graph to model each device and its corresponding relationships for situational awareness and available actions. Further, the security requirements are easily correlated with the device capabilities to understand the extent of coverage and inform the need for compensating controls. While our model remains flexible to adapt to a variety of frameworks, this is a promising direction that would accelerate potential adoption and effectiveness.

- **Cybersecurity Domain Knowledge** databases, ontologies, and libraries have been an emerging focus within the industry as defending organizations aim to better understand the implications of vulnerabilities, assessment of risks, attack tactics and techniques, and defence techniques. In particular, the knowledge frameworks developed by MITRE provide comprehensive cybersecurity domain knowledge bases which are prominently referenced across the industry.

The above data mapping initiatives provide a strong foundation for the required agent reference knowledge and environment modeling. Furthermore, within the multi-agent architecture, the control hierarchy allows for visibility and coordinated actions to be enforced according to the defined security requirements and generated policies.

6.5 Additional Considerations for Implementation

6.5.1 Securing the Multi-Agent System

While we have presented a security solution based on multi-agent systems, it is essential to establish a secure design for implementing the agents themselves as potential threat vectors. As any system has the potential to introduce new security vulnerabilities, it is important to incorporate security during the design phase. Although specific implementation details have been so far outside of the scope of this work, we will identify some potential threats to multi-agent systems and discuss design considerations for secure implementation.

Threats to Multi-Agent Systems

Hedin and Moradian [102] identify security threats to multi-agent systems and present a model for secure design. The authors provide a list of potential threats at the system level and at the agent level, as shown in Table 6.9 below, while highlighting the areas of agent identification and authentication, secure communication, and preventing unauthorized access to agents as key requirements for a secure multi-agent system.

Table 6.9 Threats to Multi-Agent Systems [102]

Threats at System Level	**Threats at Agent Level**
• Threats from mobile agents to hosts • Threats from the Internet: DoS, damage, event-triggered, compound, or user attacks • Altering the event logging system of a MAS • Altering the agent code, data, and configuration • Fake agent • Fake service • Delegation of services • Insecure communication channels • Insecure agent delegation • Lack of accountabilities • Agent authorization • Reputation attack	• Threats from hosts to agents • Threats from agents to agents • Agent authentication • Verification of information that agents collect from the internet • Threats from users to agents • Threats to communication among agents: identification and authentication, unauthorized access to agents, ontology attack, active probing attack, message injection, modification of agents' interaction by altering the transferring information, fake message

Based on these key requirements, the authors propose an agent ID code and associated permissions to be included in message headers for secure communication, agent identification and authentication. Further, the concept of a Gate Agent is introduced to handle communications between the host system and components to protect the system level.

With our specific multi-agent BDI architecture, we anticipate that key targets for security threats would fall on the controller or coordinator agents, the graph database, or within agents themselves. New devices joining the network will require special consideration and secure registration processes. Within agents, threats to data leakage or manipulation of BDI data, compromised agents, or potential disruption of services are also potential concerns. The following section presents secure design considerations to address these challenges.

Secure Design Considerations

While the above model presents security solutions to some aspects of our threat model, we expand on further secure design considerations when designing the multi-agent system implementation. There have been some existing works on secure design solutions for mobile agents, with a particular interest in protecting the confidentiality and integrity of data. For example, Sabir et al. propose a blockchain-based solution to secure migration of smart home mobile agents. To ensure agent integrity, a security agent is responsible for managing agent migrations by registering and validating hashes of transactions stored on the Ethereum blockchain [103].

Securing agents follows the same principles as any piece of software, where the host systems and overall threat model must be considered. Depending on the available resources of the host devices, implementation models can vary. However, we list the below considerations:

- **Host System Hardening and Security Controls:** As agents can be deployed to and interact with various devices within the network, the security of the host systems is covered by our proposed model and can be correlated with potential implications for the multi-agent system.
- **Self-Healing System Capabilities:** While appropriate security components for other aspects of the network are modeled within the knowledge graphs, our framework can also model the security of the multi-agent system as another layer of this. The system monitor provides a holistic view of the security states and relevant data of the system to maintain situational awareness of the system's health, including security telemetry of the agents. Based on this, security requirements, policies, capabilities and defensive actions for the multi-agent system can be generated and actioned in a similar way. The self-awareness and self-healing properties of the system apply to both the security of the IoT devices themselves and the supporting multi-agent system.

- **Secure Communication:** Communications between agents and between agents and devices can introduce a key vector for interception or tampering. Data security and encryption mechanisms should be in place to preserve confidentiality and integrity. The process for registering a new device joining the network must be developed with strong security considerations, as this is where the devices will be registered and considered for future communications with the system. Secure key management schemes can be used to provide private keys and identifiers to devices and agents to support authentication and identification. Depending on the capabilities of the host device, there may be varying capabilities for encryption and other security protocols. Devices with limited security capabilities should be treated with compensating controls and perhaps lower trust levels.
- **Agent Authentication & Identification**: Aligned to the join process, agents can be given a unique identifier and private key. This can be used for secure communications and access control decisions.
- **Access Control**: Access to data and systems should be provisioned with the principle of least privilege. Access control mechanisms should be applied to prevent unauthorized access to agents, as well as to limit access of agents themselves.
- **Logging and Auditing**: Agent actions should be logged for auditing purposes and can also be monitored for suspicious or malicious behaviour. Further, as the system design is intended to be largely autonomous, logs that can provide artifacts to explain decisions made by the agents will be valuable for a trustworthy and auditable system [104].
- **Trust and Reputation of Agents:** Trust and Reputation Management (TRM) systems for mobile agent systems have been of particular interest in the literature. For example, Geetha and Jayakumar [105] propose a TRM model for mobile agent security through trust-based secure routing tables and cryptographic algorithms to preserve the integrity and confidentiality of data and secure the execution of agents. For example, Xu et al. [106] present a hardware-based autonomic agent trust model for IoT systems where a

Trustworthy Agent Execution Chip (TAEC) is installed on each sensor node to provide a trusted execution environment for agents.

6.5.2 Potential Economic Motivations for Adopting the Model

We can define an "accredited," or "compliant" device vendor as one which supports the functionality of our framework and has been validated as a participant. Accredited vendor products can be considered to have a higher level of trust as they have been validated to comply with a set of industry accepted security standards. However, as vulnerabilities remain an eternal possibility, the system maintains a cautious skepticism and ensures that layers of controls are always in effect. This model ensures that participants maintain a baseline level of security configuration, as well as a foundation for effective monitoring, analysis, and remediation of potential vulnerabilities or incidents. Smart home device vendors can opt into security accreditation with the intention of proving and enriching security capabilities to maintain customer trust. A coalition of smart home vendors and device owners are provided the platform for the security assurance of their homes and products. Given the economic factors and overall complications with enforcing a single industry standard, it is assumed that not all vendors will adopt accreditation for various reasons, including costs and/or inability to support functionality going forward. Our framework accounts for the inevitable diversion from the standard and has been developed with this in mind. These vendor products can still maintain some level of assurance if they exist in a smart home environment with other accredited vendors to offset the risk. In this case, the incentives for smaller vendors may not be enough to adapt the framework if they are able to continue to provide low-cost products and services with minimal security. The framework enables consumers to make their own informed product choices for the security of their homes while maintaining some resilience if they introduce any non-compliant products. While the consumer IoT industry is also moving towards greater interoperability, this framework allows for a common interface across multi-vendor platforms. While we have developed this model with the purpose of security services, it is also possible that the foundation could be extended to other areas, such as cross-vendor coordination for cooperative smart home services. However, this is out of the scope of this work at this time.

6.6 Chapter Summary

This chapter has provided an evaluation and experimental results of our implemented architecture. As highlighted in earlier chapters, two major concerns for IoT-based solutions are availability and resource utilization. Our experimental results show that an agent's average response time is 3.9 ms with an average memory utilization of 311 bytes. The fast response time and low memory utilization prove to be good performance metrics that have a relatively low impact on the resource constraints of an IoT device. Further, the defence capability results have shown that the strategic application of compensating controls strategized by agents' situational awareness can effectively result in a better understanding of security risks and the implementation of defence mechanisms through layered controls and coordination of agents.

Our architecture has been evaluated against the design requirements for IoT security devices defined in earlier chapters, providing an effective solution for resource limitations, interoperability and reliability requirements, high data volumes, and ease of use as is necessary for an effective security solution to exist within an IoT and smart home environment. Security requirements are addressed through policy as code within agent BDI knowledge and can adapt to evolving threats and environment scenarios with ongoing domain knowledge inputs.

While the initial implementation has provided a strong proof of concept, we have provided some additional considerations for future works including security considerations for the multi-agent system itself, as well as potential economic motivations for industry adoption of this model. Additional limitations and future works will be discussed in the final chapter in conclusion of this thesis.

7 Conclusion and Future Works

7.1 Thesis Summary

The combined domains of IoT, agent-based modeling, and cybersecurity present many opportunities that have largely been underexplored in the literature to date, while the intersection of these topics show strong potential to address contemporary and emerging cybersecurity challenges. Based on the increasing demand for intelligent, distributed cyber defence capabilities within IoT systems, this work presents a framework and proof of concept for bridging the capabilities of agent-based technologies into an adaptive cyber defence model for IoT smart home networks. While our framework has set the groundwork for the future application of key use cases in autonomous security, simulations, optimization of strategies, modeling and experimentation of theoretical frameworks, broader implications of the model can be applied to general distributed planning capabilities for enterprise security as well as non-security use cases. Our novel approach to BDI agent reasoning based on knowledge graphs introduces opportunities for distributed intelligence with shared knowledge and can be applied to broader agent-based use cases outside of the cybersecurity domain.

The key contributions of this thesis have been summarized below:

- **Multi-agent Architecture for Adaptive Cyber Defence:** Our architecture leverages agent-based technologies to provide autonomous, adaptive, cooperative goal-oriented behaviours in software agents deployed across an IoT network to achieve security goals. We presented our architecture in Chapter 3 with an individual agent model as well as a control and coordination hierarchy. This design has been evaluated to demonstrate how it addresses the unique requirements and design challenges for smart home IoT environments including resource limitations, interoperability requirements, reliability, data volume, and ease of use.

- **Knowledge Graphs for BDI Agent Reasoning:** Chapter 4 introduced a design for knowledge graphs for cybersecurity modeling based on industry knowledge bases which can be leveraged for agent reasoning. Our novel approach to agent BDI reasoning powered by knowledge graphs introduces a data-driven and adaptable model for distributed contextual intelligence based on an evolving environment.

Our implementation described in Chapter 5 provided the details of our implementation architecture developed to simulate the multi-agent capabilities for security policy generation and mission deployment, as well as BDI-agent reasoning with Neo4j knowledge graphs. Initial security use cases were modeled to illustrate the capabilities for vulnerability and access management.

The experimental results in Chapter 6 demonstrated the practical feasibility of our model tested through simulation of a botnet scenario under 256 different environment configurations. The operational performance results showed an average response time and memory utilization with low impact to resource utilization of baseline smart home IoT devices. Further, the impact of different environment configurations in relation to available agent capabilities for defence mechanisms has shown how situational awareness enabled by our knowledge graph model can inform agent actions towards defending numerous environment types, as well as perform risk analysis and coordinated actions for compensating controls.

7.2 Limitations

While this thesis has presented a framework for very wide scope of capabilities, the key contributions have set the foundation for significant future work in refinement and optimization of each component in detail, with flexibility for the addition of modular components. The main limitations of the current work are highlighted below, where future works for each of these areas are elaborated in the following section.

- **Reference to NIST Labeling framework**: the initial implementation is strongly aligned to the NIST labeling framework and assumes that devices maintain “device profiles” or self-claim their capabilities when joining the network. While this demonstrates a strong use case for creating an industry labeling standard that can inform automated decisioning, the labeling framework is currently still in development and is not currently deployed across the industry. The labeling framework provided a reference for our data model to map security policies with device capabilities and associated agent actions. With this data model in mind, future works could investigate alternative methods of gathering similar data, such as through network/device analytics or community intelligence sharing.

- **Limitations with Datasets and Knowledge Graph:** the cybersecurity domain knowledge graphs have been generated using industry datasets such as MITRE ATT&CK, CVE, and CAPEC. While this provides a strong initial reference point, we discovered some limitations and inconsistencies within the data which could cause challenges for automated decisioning in its current form. These datasets are constantly evolving with the industry, and we expect data quality improve over time, especially as the industry shifts towards greater automation rather than manual interpretations. Further, the BDI agent knowledge graphs have been manually developed for this work based on the NIST framework, and we believe this can also be expanded and refined to follow other standards and requirements. There are also opportunities for development of agents’ capabilities to continually refine the knowledge graph based on new observations and experiences which could provide enhancements to the datasets for greater context awareness and inferences.
- **Policy Inheritance and Conflicts:** while our framework and implementation provide an initial model for policy generation and planning, the capabilities for managing conflicts and prioritization according to utility have not yet been fully built out for scale. This is highlighted as a future work, where there are many works in the literature that can be tested and applied within the BDI knowledge graph model and simulated with our model to find optimal methods.
- **Agent coordination and Learning capabilities:** similar to the above, our model provides a foundation for expanded works in agent coordination and

learning capabilities to enhance situational awareness, risk analysis, action recommendations and predicted outcomes.

- **Implementation is Limited to Simulated Environment**: our implementation has been deployed in a purpose-built simulated environment using abstractions and custom message formats. This has been effective for the development and evaluation of the initial design, however it will need to be expanded and integrated with larger datasets and real-life devices as a future work, which should also include provisions for securing the agents themselves as discussed in the previous chapter.

7.3 Future Works

This thesis has established a foundation for many future works as an emerging research area in agent-based applications for cybersecurity, for continued development on defining utility and modeling optimal behaviors for multi-agent defence systems and leveraging the existing model for enhanced simulations. While we have focused on applications to smart home and IoT environments specifically, the framework can be easily extended to broader domains of security or other applications of BDI agents where large collections of domain knowledge can be recorded in knowledge graphs. We will further elaborate on areas of future work in this section.

7.3.1 Expansion of Simulation and Real-World Implementation

The implementation discussed in Chapter 5 provides a foundation for further simulated experiments, which can reveal system behaviours and refine security strategies and optimization. Running large scale simulation experiments with different environment configurations will be useful for testing at scale. Additional capabilities for validation and auditing of agent actions and behaviours will provide relevant data for further enhancements to agent learning capabilities and optimization, as further described in the following section. Interactive simulations of IoT environments and security scenarios can also support the education and training of security professionals, provide modeling capabilities for threat modeling and design of secure systems, and better understand the implications of various security capability deployments to inform the prioritization of controls.

Our simulations thus far have been based on limited simulation data for proof of concept. Further integration with real life datasets will be valuable to further refine the model. An additional layer for parsing data will be required. Lastly, while we have demonstrated the feasibility of our solution from a modeling perspective, the next step after testing with larger datasets would be to further integrate with real life IoT devices with the considerations described in Section 6.5 .

7.3.2 Agent Coordination and Learning Capabilities

While our architecture and knowledge graph model provide a foundation for modeling agent knowledge, there are many opportunities for expansion to support future research on agent learning capabilities. The knowledge graphs in the present state are relatively static for demonstration. However, they provide flexibility for enhancement and extended applications of multi-agent learning, utility refinement, and other opportunities for the application and testing of various learning capabilities within a cybersecurity context.

Coordination and Social Functions

As multiple agents can exist in an environment, an important step is to negotiate missions and tasks between them to support the entire system. A coordination mechanism must be further defined to determine effective strategies. While each agent or mission may maintain independent security goals to achieve, they may also have common goals, in which case coalition formation is beneficial for the collective utility. Negotiation incentives include increased resources to performing certain tasks, greater visibility and information sharing, and shared utility. There may be cases where higher payoffs are awarded to missions completed in collaboration between two or more agents. Conversely, agents may also have conflicting priorities or missions. In this case, negotiation is required to achieve the highest possible utility for both. Coordination mechanisms exist to allow coordination between the controllers to achieve the highest utility for the overall system state.

The following social functions can be further defined for the collaboration and coordination of agents: Resource allocation; coalition formation; distributed cognitive abilities such as multi-agent planning, control and execution; conflict resolution functions through mechanism design using auction, voting, or negotiation protocols; and organizational evolution to enable changes and adaptive behaviour over time to meet new requirements and changes in the environment.

Macro-level coordination can also be possible from a vendor perspective at a larger scale across smart home deployments. From the perspective of the global system, there are two major components: (1) the set of all vendor products $v_i = \{p_i, c_i\}$, and (2) the set of all smart homes $c_j = \{v_j, p_j\}$. Each vendor v_i has a set of products p and clients c. Likewise, each client's smart home consists of a set of vendors and products. The primary goal of each vendor is to ensure the operation and security of their products in their client's homes. Each vendor does not explicitly regard the security of other vendor devices. However, if they exist within a client's home, the vendor may be able to protect the other devices if it improves the security of the total system.

Further, the primary goal of each client is to maintain the security of their own smart home and devices. While each client c_i does not explicitly regard the state of other clients' smart homes $c_{'i}$, the collective coordination between clients works to achieve a global social optimum in the case of widespread attacks. This is demonstrated through the basic economics of cybersecurity, where the attacker will aim for a target with the greatest reward, requiring the least cost/effort. Often the most profitable form of attack is to exploit large numbers of vulnerable devices, for example, to join a botnet. By increasing the collective baseline for consumer IoT devices, the attacker will either adapt their strategy to invest more for each attack, choose a different target, or receive a generally lower reward.

Trust Models

In a multi-agent system as well as any environment with potential for cybersecurity concerns, the concept of trust is foundational. While this topic has been largely out of scope of this initial work, our framework provides a foundation for future research in applying trust models for further enhancement. Potential directions could include extending the knowledge graphs with trust-based relationship or node properties, which can be dynamically enriched through environment observations. Trust scores could be developed for new or suspicious devices with associated controls to be implemented such as access control or network isolation. Additionally, relationships between agents can also be informed through trust models which can direct coordinated actions.

Multi-agent Learning

Multi-agent learning is defined as the problem of devising learning algorithms for agents that are capable of learning (sub)optimal solutions in the presence of other (learning) agents or algorithms – facing the difficulties of incomplete information, large state spaces, credit assignment, cooperative and/or competitive settings, and reward shaping [107] [108]. Reinforcement learning is based on behavioral change to reward desirable behavior and discourage undesirable behavior from obtaining maximum utility. Reinforcement learning capabilities can be adopted into our model for enhanced learning and planning capabilities.

It is important to note that while baselines can be gathered, there will often be limited learning data available for attacks within an individual environment. Additionally, due to the environment's operational sensitivity, there is low risk tolerance for experimentation with negative reinforcement when there is an attack. This raises the question of how we can proactively gather data to learn the best responses to attacks. We suggest that to supplement the baseline anomaly detection, this can be done through a combination of knowledge base templates gathered by federated learning models across home environments, as well as data gathered through simulations of attack scenarios.

Federated Learning and Intelligence

It is useful to look at the bigger picture for large scale patterns and trends across different environments. Participating systems can provide sampling data at occasional intervals to develop collective strategies across environments. This data can be provided anonymously and in privacy through federated learning algorithms.

Types of data that can be shared and analyzed include baseline behaviour thresholds, recommended security requirements and user preferences from similar environments, feedback on ineffective and effective strategies, threat intelligence and countermeasures for attacks, and validation of similar behaviour across other environments. To preserve privacy and security in federated learning model, the system will require assurances for privacy preservation and anonymity, resilience to poisoning attacks or adversarial AI, data leak protection, data integrity, timing and accuracy.

In a distributed system for collective data aggregation and computation, no single entity has visibility into the entire global system. Product vendors can only see data related to their products, and home users can only see data related to their own homes and products owned. Privacy is preserved within home local networks, with respective models federated at the cloud layer by each vendor for their products. Through federated learning, client models and alerts aggregate to the vendor for assessment and model refinement. Within each home network, the edge coordinators negotiate policies with each other, taking into consideration the recommendations from each vendor, in alignment with the risk posture of the local network. Once negotiation has been completed, the updated model will be sent back to each vendor for consideration.

Vendor claims can be validated by consensus across other participating nodes in the global system. This can take the form of other vendors or MAS controllers, with validation functions predefined at nodes, or further validation agents (for example, through a bug bounty model) can be adapted. Smart contracts may be employed to ensure Service Level Agreements (SLAs) with regard to security posture. Vendors are incentivized to validate others as this resilience capability increases their security utility. Participation allows for their respective controls to be validated. Collective security concerns can be communicated across partner nodes, validated, and collective countermeasure strategies can be generated and redirected back to client sites (secure multiparty computation). Assurance, privacy, and trust through federated learning and secure multi-party communication across multi-agent systems is a future research area.

7.3.3 Refinement of Knowledge Graphs

Enhancements to Data and Model

While we have presented a preliminary proof of concept of knowledge graphs, continued expansion and modeling of different security considerations is a future research area to be refined. As this is the first work of our knowledge to define BDI agent plans within a knowledge graph correlated with cybersecurity domain knowledge, a focused research effort in further refinement would support practical extended implementation. Further, libraries of domain knowledge are continually evolving, and a mechanism for updating intelligence on an automated basis would be beneficial to ensure up to date awareness of the security domain.

Enhancements for Agent Learning & Reasoning

Enhancement of the knowledge graphs to support agent learning and decisioning can be possible by introducing additional properties within nodes, such as utility scores, to support the optimization of plan selection. With the continuously evolving and incomplete information available (i.e., partially observable environment), a probabilistic method for agent decision-making under uncertainty can be applied using the Partially Observable Markov Decision Process (POMDP). Probabilistic uncertainty involves the use of probability distributions for state transitions and observations, where agents can determine the transition to the next state according to a fixed conditional probability. An example by Rens et al. implements a BDI agent architecture for Partially Observable Markov Decision Process (POMDP) Planner [109].

7.3.4 Applied Game Theory and Control Theory

In the literature, there have been several recent works in modeling adversarial situations in cybersecurity, mainly with the goals of extracting optimal defence strategies and/or creating the foundation for autonomous adaptive defences. Two possible approaches include the applications of game theory and control theory.

Using the control-theoretic approach, the actions of the adversary are not assumed to be strategic and are not taken into consideration in defender strategies. In contrast, through a game theoretic model, adversarial behavior is modeled with its own utility associated with attack patterns. One major struggle is the complexity of the decision set due to the attributes of dynamic and partially observed information.

Musman [110] presents a Cyber Security Game (CSG) to quantitatively identify cyber risks and determine the optimal deployment of security controls for the level of resources available in the environment. This method uses a calculated risk score based on mission impact models of the likelihood and impact of cyber incidents, aligned with applying threat modeling to the system topology and defender model. Musman’s approach uses attack path modeling to determine chain rules and probabilities. The Cyber Mission Impact Assessment (CMIA) tool [111] provides granular impact assessment according to levels of impact per effect of each type of asset. The attacker model can be determined using a chain rule of probabilities of success:

$$P(A_1, \ldots, A_n) = P(A_1|A_2,\ldots A_n)\, P(A_2|A_3,\ldots A_n)\, \ddot{P}(A_{n-1}|A_n)\, P(A_n)$$

With visibility into the topology of the system, we can overlay the attacker's behavior and impact on generating plausible attack trees for pathways into the system. The data obtained from the attack tree provides an expected value for loss for each branch of the attack option. This expected value can then be plugged into the risk calculation:

$$(1)\ \text{Risk} = \sum_{CI=1}^{N} P_{CI} L_{CI} \qquad \text{or} \qquad (2)\ \text{Max}_{CI} = 1{,}N(P_{CI}L_{CI})$$

Where risk is calculated as either (1) the sum of all probabilities and losses associated with all possible cyber incidents or (2) the maximum probability and loss for the worst potential incident, depending on risk appetite. Once attack paths are generated and provided with associated risks, we can determine defence policies that can be employed as countermeasures. Each method will also have an associated expected success rate for reducing the probability of attack.

While many works exist to model cybersecurity scenarios for adaptive cyber defence, many are single purpose solutions or highly abstracted and have not been applied at a practical level. Our model provides a framework for further experimentation and application of agent-based game theory and control theory research applied to the cybersecurity domain in a modeled simulated environment to bridge this gap and further the industry. It is important to note that most of the time, it can be expected that the network will be operating in a normal operational state with no active security threats. Game-theoretic models often assume an active adversarial opponent with highly focused data inputs. However, this is highly unrealistic in an applied scenario. Control-theoretic models provide a single player control-based approach to responding to environmental observations more aligned to an applied context. We suggest a primary focus on control-based approaches, with potential opportunities for a hybrid response model where game theoretic strategies can be applied in detected adversarial scenarios.

Utility and Incentivization for Agents

Aligned to the above, further refinement to the utility calculation and incentivization for agent behaviours have also been noted for future work. There are several layers in which the utility is directed towards specific goals, which can form a utility hierarchy at each layer of the system used by different types of agents:

Table 7.1 Utility Goals at each Layer

Layer	Goals
Control Layer	Minimize risk, achieve security goals, and operational availability.
Coordination Layer	Health of agents, optimize agent deployment.
Vendor Layer	Negotiate/generate strategies and missions to improve and/or maintain system state.
Device Agent Layer	Contribute to the completion of mission/task (i.e., sensor, detection, analysis, response). Beliefs, Desires, Intentions model.

This hierarchy is defined to coordinate utility across strategic risks with impact to the entire home, along with tactical risks associated with individual components.

Each mission has prospective payoff values for increasing/decreasing each utility parameter, mapped to a particular reference ID, used to determine plans' prioritization, and validated on completion. The controller will be notified after the completion of each mission, receive the payoff, and cascade it down to the agents involved in the mission. A schema can be provided to classify each mission according to its benefit to the micro and macro level utility.

7.4 Conclusion

The contributions of this work have provided an agent-based modeling framework for adaptive cyber defense, addressing key requirements for adaptive and autonomous cybersecurity capabilities. While the implementation and experimental results demonstrate the feasibility of our design in smart home IoT systems, this model can also be easily expanded to other domains. It is our intention to provide the foundation to inspire further research works in this area for continued development, application, and optimization of this paradigm to support the advancement of the industry and bring autonomous cyber defence to realization.

References

[1] D. Marriott, K. Ferguson-Walter, S. Fugate, and M. Carvalho, “Proceedings of the 1st International Workshop on Adaptive Cyber Defense,” 2021, doi: 10.48550/ARXIV.2108.08476.

[2] R. Coulter and L. Pan, “Intelligent agents defending for an IoT world: A review,” *Computers & Security*, vol. 73, pp. 439–458, Mar. 2018, doi: 10.1016/j.cose.2017.11.014.

[3] C. Savaglio, M. Ganzha, M. Paprzycki, C. Bădică, M. Ivanović, and G. Fortino, “Agent-based Internet of Things: State-of-the-art and research challenges,” *Future Generation Computer Systems*, vol. 102, pp. 1038–1053, Jan. 2020, doi: 10.1016/j.future.2019.09.016.

[4] K. K. Patel, S. M. Patel, and P. Scholar, “Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges,” p. 10, 2016.

[5] International Telecommunication Union, “Overview of the Internet of things,” International Telecommunication Union, ITU-T Y.2060, Jun. 2012. [Online]. Available: https://handle.itu.int/11.1002/1000/11559

[6] K. Gafurov and T.-M. Chung, “Comprehensive Survey on Internet of Things, Architecture, Security Aspects, Applications, Related Technologies, Economic Perspective, and Future Directions,” *Journal of Information Processing Systems*, vol. 15, no. 4, pp. 797–819, Aug. 2019, doi: 10.3745/JIPS.03.0125.

[7] M. Hasan, “State of IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally,” *IoT Analytics: Market Insights for the Internet of Things*, May 18, 2022. https://iot-analytics.com/number-connected-iot-devices/

[8] C. MacGillivray and D. Reinsel, “Worldwide Global DataSphere IoT Device and Data Forecast, 2021–2025,” International Data Corporation, Market Forecast US48087621, Jul. 2021. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=US48087621

[9] Fortune Business Insights, “Internet of Things (IoT) Market Size, Share & COVID-19 Impact Analysis, By Component (Platform, Solution & Services), By End-Use Industry (BFSI, Retail, Government, Healthcare, Manufacturing, Agriculture, Sustainable Energy, Transportation, IT & Telecom, and Others), and Regional Forecase, 2022-2029,” Fortune Business Insights, Market Research Report FBI100307, Mar. 2022. [Online]. Available: https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307

[10] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, “A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures,” *IEEE Access*, vol. 7, pp. 82721–82743, 2019, doi: 10.1109/ACCESS.2019.2924045.

[11] B. Hammi, S. Zeadally, R. Khatoun, and J. Nebhen, “Survey on smart homes: Vulnerabilities, risks, and countermeasures,” *Computers & Security*, vol. 117, p. 102677, Jun. 2022, doi: 10.1016/j.cose.2022.102677.

[12] E. Fernandes, J. Jung, and A. Prakash, “Security Analysis of Emerging Smart Home Applications,” in *2016 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, May 2016, pp. 636–654. doi: 10.1109/SP.2016.44.

[13] Amazon, “Alexa,” 2022. https://developer.amazon.com/en-US/alexa

[14] Google, “Google Home,” *Google Home*, 2022. https://home.google.com/welcome/

[15] Apple, “Apple Homekit,” *Apple*, 2022. https://www.apple.com/ca/home-app/

[16] N. M. Kumar and P. K. Mallick, “The Internet of Things: Insights into the building blocks, component interactions, and architecture layers,” *Procedia Computer Science*, vol. 132, pp. 109–117, 2018, doi: 10.1016/j.procs.2018.05.170.

[17] H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, M. Aledhari, and H. Karimipour, “A survey on internet of things security: Requirements, challenges, and solutions,” *Internet of Things*, vol. 14, p. 100129, Jun. 2021, doi: 10.1016/j.iot.2019.100129.

[18] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, “Internet of things security: A top-down survey,” *Computer Networks*, vol. 141, pp. 199–221, Aug. 2018, doi: 10.1016/j.comnet.2018.03.012.

[19] K. Boeckl *et al.*, “Considerations for managing Internet of Things (IoT) cybersecurity and privacy risks,” National Institute of Standards and Technology, Gaithersburg, MD, NIST IR 8228, Jun. 2019. doi: 10.6028/NIST.IR.8228.

[20] P. Williams, I. K. Dutta, H. Daoud, and M. Bayoumi, “A survey on security in internet of things with a focus on the impact of emerging technologies,” *Internet of Things*, vol. 19, p. 100564, Aug. 2022, doi: 10.1016/j.iot.2022.100564.

[21] T. Rajmohan, P. H. Nguyen, and N. Ferry, “A decade of research on patterns and architectures for IoT security,” *Cybersecurity*, vol. 5, no. 1, p. 2, Dec. 2022, doi: 10.1186/s42400-021-00104-7.

[22] Federal Bureau of Investigation, “INTERNET OF THINGS POSES OPPORTUNITIES FOR CYBER CRIME,” Federal Bureau of Investigation, United States, Public Service Announcement I-091015-PSA, Sep. 2015. [Online]. Available: https://www.ic3.gov/Media/PDF/Y2015/PSA150910.pdf

[23] US Department of Homeland Security, “Strategic Principles for Securing the Internet of Things (IoT),” US Department of Homeland Security, Nov. 2016.

[24] US Federal Bureau of Investigation, “Cyber Actors Use Internet of Things Devices as Proxies for Anonymity and Pursuit of Malicious Cyber Activities,” US Federal Bureau of Investigation, Public Service Announcement I-080218-PSA, Aug. 2018. [Online]. Available: https://www.ic3.gov/Media/Y2018/PSA180802

[25] Executive Office of the President, “Executive Order 14028: Improving the Nation’s Cybersecurity,” *Presidential Document: Executive Order*, vol. 86, no. 93, May 2021.

[26] M. Fagan, K. N. Megas, K. Scarfone, and M. Smith, “IoT device cybersecurity capability core baseline,” National Institute of Standards and Technology, Gaithersburg, MD, NIST IR 8259A, May 2020. doi: 10.6028/NIST.IR.8259a.

[27] IoT Security Foundation, “IoT Security Foundation,” *Make it Safe to Connect*. https://www.iotsecurityfoundation.org/about-us/ (accessed Aug. 06, 2022).

[28] S. Deep, X. Zheng, A. Jolfaei, D. Yu, P. Ostovari, and A. Kashif Bashir, “A survey of security and privacy issues in the Internet of Things from the layered context,” *Trans Emerging Tel Tech*, vol. 33, no. 6, Jun. 2022, doi: 10.1002/ett.3935.

[29] A. Verma, R. Saha, N. Kumar, G. Kumar, and Tai-Hoon-Kim, “A detailed survey of denial of service for IoT and multimedia systems: Past, present and futuristic development,” *Multimed Tools Appl*, vol. 81, no. 14, pp. 19879–19944, Jun. 2022, doi: 10.1007/s11042-021-11859-z.

[30] M. Pelino and T. Shields, “Secure IoT As It Advances Through Maturity Phases: Predict And Prevent Attacks Targeting The Internet Of Things,” Forrester, Trend Report, Jan. 2016.

[31] Unit 42, “2020 Unit 42 IoT Threat Report,” Palo Alto Networks, 2020.

[32] B. Schneier, “The Internet of Things Is Wildly Insecure—And Often Unpatchable,” *Schneier on Security*, Jan. 06, 2014. https://www.schneier.com/essays/archives/2014/01/the_internet_of_thin.htm l

[33] OWASP, “OWASP IoT Top 10,” Open Web Application Security Project (OWASP), Dec. 2018.

[34] Canonical, “Taking Charge of the IoT’s Security Vulnerabilities,” Canonical, Whitepaper, Jan. 2017.

[35] Mandiant, “14 Cyber Security Predictions for 2022 and Beyond,” Mandiant, 2022.

[36] J. Leyden, “Samsung Smart Fridge Leaves Gmail Logins Open to Attack,” *The Register*, Aug. 24, 2015. https://www.theregister.com/2015/08/24/smart_fridge_security_fubar/

[37] A. Hashim, “Zero-Day Bugs Spotted in Nooie Baby Monitors,” *Latest Hacking News*, Feb. 14, 2022. https://latesthackingnews.com/2022/02/14/zero-day-bugs-spotted-in-nooie-baby-monitors/

[38] I. Arghire, “Nuki Smart Lock Vulnerabilities Allow Hackers to Open Doors,” *SecurityWeek*, Jul. 27, 2022. https://www.securityweek.com/nuki-smart-lock-vulnerabilities-allow-hackers-open-doors

[39] J. Hollington, “Bluetooth hack compromises Teslas, digital locks, and more,” *Digital Trends*, May 16, 2022. https://www.digitaltrends.com/mobile/bluetooth-hack-compromises-teslas-digital-locks-and-more/

[40] M. Kumar, “Cracking WiFi Passwords by Hacking Smart Kettles,” *The Hacker News*, Oct. 21, 2015. https://thehackernews.com/2015/10/hacking-wifi-password.html

[41] J. A. Jerkins, “Motivating a market or regulatory solution to IoT insecurity with the Mirai botnet code,” in *2017 IEEE 7th Annual Computing and*

Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, Jan. 2017, pp. 1–5. doi: 10.1109/CCWC.2017.7868464.

[42] Nozomi Networks Labs, "New BotenaGo Variant Discovered by Nozomi Networks," *Nozomi Networks Blog*, Apr. 18, 2022. https://www.nozominetworks.com/blog/new-botenago-variant-discovered-by-nozomi-networks-labs/

[43] J. Salvio and R. Tay, "Fresh TOTOLINK Vulnerabilities Picked up by Beastmode Mirai Campaign," *Fortinet Threat Research Blog*, Apr. 01, 2022. https://www.fortinet.com/blog/threat-research/totolink-vulnerabilities-beastmode-mirai-campaign

[44] A. Collen *et al.*, "GHOST - Safe-Guarding Home IoT Environments with Personalised Real-Time Risk Control," in *Security in Computer and Information Sciences*, vol. 821, E. Gelenbe, P. Campegiani, T. Czachórski, S. K. Katsikas, I. Komnios, L. Romano, and D. Tzovaras, Eds. Cham: Springer International Publishing, 2018, pp. 68–78. doi: 10.1007/978-3-319-95189-8_7.

[45] C. Lawson, P. Firstbrook, and P. Webber, "Market Guide for Extended Detection and Response," Gartner, G00747261, Nov. 2021.

[46] Johns Hopkins University Applied Physics Laboratory, "Integrated Adaptive Cyber Defense," *INTEGRATED ADAPTIVE CYBER DEFENSE*. https://www.iacdautomate.org/

[47] H. Yu, Z. Shen, and C. Leung, "From Internet of Things to Internet of Agents," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, Beijing, China, Aug. 2013, pp. 1054–1057. doi: 10.1109/GreenCom-iThings-CPSCom.2013.179.

[48] E. Bonabeau, "Agent-based modeling: Methods and techniques for simulating human systems," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 99, no. suppl_3, pp. 7280–7287, May 2002, doi: 10.1073/pnas.082080899.

[49] C. Macal and M. North, "INTRODUCTORY TUTORIAL: AGENT-BASED MODELING AND SIMULATION," p. 15.

[50] M. E. Bratman, *Intentions, Plans, and Practical Reason*. Cambridge: Harvard University Press, 1987.

[51] G. I. Simari and S. D. Parsons, *Markov Decision Processes and the Belief-Desire-Intention Model*. New York, NY: Springer New York, 2011. doi: 10.1007/978-1-4614-1472-8.

[52] P. Gärdenfors, "Belief revision: An introduction," in *Belief Revision*, 1st ed., P. Gärdenfors, Ed. Cambridge University Press, 1992, pp. 1–28. doi: 10.1017/CBO9780511526664.001.

[53] I. Nunes, "BDI4JADE: a BDI layer on top of JADE," p. 16.

[54] A. Pokahr, L. Braubach, and W. Lamersdorf, "A Goal Deliberation Strategy for BDI Agent Systems," in *Multiagent System Technologies*, vol. 3550, T. Eymann, F. Klügl, W. Lamersdorf, M. Klusch, and M. N. Huhns, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 82–93. doi: 10.1007/11550648_8.

[55] L. Males and S. Ribaric, "A model of extended BDI agent with autonomous entities (integrating autonomous entities within BDI agent)," in *2016 IEEE*

8th International Conference on Intelligent Systems (IS), Sofia, Bulgaria, Sep. 2016, pp. 205–214. doi: 10.1109/IS.2016.7737422.
[56] Y.-B. Peng, J. Gao, J.-Q. Ai, C.-H. Wang, and H. Guo, "An Extended Agent BDI Model with Norms, Policies and Contracts," in *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, Dalian, China, Oct. 2008, pp. 1–4. doi: 10.1109/WiCom.2008.1197.
[57] G. Shaw and E. van der Poel, "Genetic Algorithms as a feasible re-planning mechanism for Belief-Desire-Intention Agents," in *Proceedings of the 2015 Annual Research Conference on South African Institute of Computer Scientists and Information Technologists - SAICSIT '15*, Stellenbosch, South Africa, 2015, pp. 1–9. doi: 10.1145/2815782.2815817.
[58] J. Buford, G. Jakobson, and L. Lewis, "Extending BDI Multi-Agent Systems with Situation Management," in *2006 9th International Conference on Information Fusion*, Florence, Jul. 2006, pp. 1–7. doi: 10.1109/ICIF.2006.301781.
[59] S. Calderwood, K. McAreavey, W. Liu, and J. Hong, "Contextual merging of uncertain information for better informed plan selection in BDI systems," in *2015 World Congress on Industrial Control Systems Security (WCICSS)*, London, United Kingdom, Dec. 2015, pp. 64–65. doi: 10.1109/WCICSS.2015.7420326.
[60] Z. A. Khan, E. Pignaton de Freitas, T. Larsson, and H. Abbas, "A Multi-agent Model for Fire Detection in Coal Mines Using Wireless Sensor Networks," in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, Melbourne, Australia, Jul. 2013, pp. 1754–1761. doi: 10.1109/TrustCom.2013.275.
[61] A. R. Hilal and O. A. Basir, "A Scalable Sensor Management Architecture Using BDI Model for Pervasive Surveillance," *IEEE Systems Journal*, vol. 9, no. 2, pp. 529–541, Jun. 2015, doi: 10.1109/JSYST.2014.2334071.
[62] J. Melgoza-Gutierrez, A. Guerra-Hernandez, and N. Cruz-Ramirez, "Collaborative Data Mining on a BDI Multi-agent System over Vertically Partitioned Data," in *2014 13th Mexican International Conference on Artificial Intelligence*, Tuxtla Gutierrez, Mexico, Nov. 2014, pp. 215–220. doi: 10.1109/MICAI.2014.39.
[63] B. Wilsker, "A Study of Multi-Agent Collaboration Theories," p. 26, 1996.
[64] C. Stergiou, G. Arys, and M. Wooldridge, "A Policy Based Framework for Agents: On the Specification of an Agent Policy Language including Roles, Relationships, Conversation Patterns and Co-operation Patterns," p. 2.
[65] Y. Xiao, F. Deng, B. Liu, S. Liu, D. Luo, and G. Liang, "A Learning Process Using SVMs for Multi-agents Decision Classification," in *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, Sydney, Australia, Dec. 2008, pp. 583–586. doi: 10.1109/WIIAT.2008.430.
[66] Y. Uhm, Z. Hwang, M. Lee, Y. Kim, G. Kim, and S. Park, "A Context-Aware Multi-Agent System for Building Intelligent Services by the Classification of Rule and Ontology in a Smart Home," in *32nd IEEE Conference on Local*

Computer Networks (LCN 2007), Dublin, Ireland, Oct. 2007, pp. 203–204. doi: 10.1109/LCN.2007.28.

[67] D. J. Kang and S. Park, "MAS based Approach to HEMS modeling: Application of social interaction mechanism to demand-side dynamics," in *2016 11th System of Systems Engineering Conference (SoSE)*, Kongsberg, Norway, Jun. 2016, pp. 1–6. doi: 10.1109/SYSOSE.2016.7542894.

[68] K. Kravari and N. Bassiliades, "StoRM: A social agent-based trust model for the internet of things adopting microservice architecture," *Simulation Modelling Practice and Theory*, vol. 94, pp. 286–302, Jul. 2019, doi: 10.1016/j.simpat.2019.03.008.

[69] H. F. Rafalimanana, J. L. Razafindramintsa, S. Cherrier, T. Mahatody, L. George, and V. Manantsoa, "Jason-RS, A Collaboration Between Agents and an IoT Platform," in *Machine Learning for Networking*, vol. 12081, S. Boumerdassi, É. Renault, and P. Mühlethaler, Eds. Cham: Springer International Publishing, 2020, pp. 403–413. doi: 10.1007/978-3-030-45778-5_28.

[70] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan, "CASAS: A Smart Home in a Box," *Computer*, vol. 46, no. 7, pp. 62–69, Jul. 2013, doi: 10.1109/MC.2012.328.

[71] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, no. 10, pp. 2266–2279, Jul. 2013, doi: 10.1016/j.comnet.2012.12.018.

[72] Y. Ye, V. Hilaire, A. Koukam, and C. Wandong, "A Holonic Model in Wireless Sensor Networks," in *2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Harbin, China, Aug. 2008, pp. 491–495. doi: 10.1109/IIH-MSP.2008.37.

[73] L. Pazzi and M. Pellicciari, "From the Internet of Things to Cyber-Physical Systems: The Holonic Perspective," *Procedia Manufacturing*, vol. 11, pp. 989–995, 2017, doi: 10.1016/j.promfg.2017.07.204.

[74] A. Kott *et al.*, "Autonomous Intelligent Cyber-defense Agent (AICA) Reference Architecture, Release 2.0," p. 154.

[75] M. Major, B. Souza, J. DiVita, and K. Ferguson-Walter, "Informing Autonomous Deception Systems with Cyber Expert Performance Data." arXiv, Aug. 31, 2021. Accessed: Oct. 15, 2022. [Online]. Available: http://arxiv.org/abs/2109.00066

[76] M. Standen, M. Lucas, D. Bowman, T. J. Richer, J. Kim, and D. Marriott, "CybORG: A Gym for the Development of Autonomous Cyber Agents." arXiv, Aug. 20, 2021. Accessed: Oct. 15, 2022. [Online]. Available: http://arxiv.org/abs/2108.09118

[77] K. Tran *et al.*, "Deep hierarchical reinforcement agents for automated penetration testing." arXiv, Sep. 14, 2021. Accessed: Oct. 15, 2022. [Online]. Available: http://arxiv.org/abs/2109.06449

[78] A. T. Nguyen, E. Raff, C. Nicholas, and J. Holt, "Leveraging Uncertainty for Improved Static Malware Detection Under Extreme False Positive Constraints." arXiv, Aug. 09, 2021. Accessed: Oct. 15, 2022. [Online]. Available: http://arxiv.org/abs/2108.04081

[79] Forum of Incident Response and Security Teams (FIRST), “Common Vulnerability Scoring System (CVSS),” *First.org,* 2022. https://www.first.org/cvss/

[80] The MITRE Corporation, “CVE Program Mission,” *CVE,* 2022. https://cve.mitre.org/

[81] The MITRE Corporation, “MITRE ATT&CK Framework,” *MITRE ATT&CK*, 2022. https://attack.mitre.org/

[82] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases*. O’Reilly Media, 2013.

[83] J. Barrasa, A. E. Hodler, and J. Webber, “Knowledge Graphs: Data in Context for Responsive Businesses,” O’Reilly Meida, USA, 2021.

[84] The MITRE Corporation, “CAR Data Model,” *MITRE Cyber Analytics Repository*, 2022. https://car.mitre.org/data_model/

[85] A. Brazhuk, “Security patterns based approach to automatically select mitigations in ontology-driven threat modelling,” p. 6.

[86] The MITRE Corporation, “Common Weakness Enumeration: A Community Developed List of Software and Hardware Weakness Types,” *CWE*, 2022. https://cwe.mitre.org/

[87] MITRE, “Common Attack Pattern Enumeration and Classification,” *Common Attack Pattern Enumeration and Classification (CAPEC)*. https://capec.mitre.org/

[88] The MITRE Corporation, “MITRE D3FEND: A Knowledge Graph of Cybersecurity Countermeasures,” *MITRE D3FEND,* 2022. https://d3fend.mitre.org/about/

[89] Neo4j Inc., “Neo4j Graph Database,” *Neo4j Graph Database*. https://neo4j.com/product/neo4j-graph-database

[90] M. Westergaard and H. M. W. Verbeek, “CPN Tools.” Eindhoven University of Technology. [Online]. Available: https://cpntools.org/

[91] Google, “Go Language,” Oct. 01, 2022. https://go.dev/doc/

[92] K. Jensen, “A brief introduction to coloured Petri Nets,” in *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 1217, E. Brinksma, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 203–208. doi: 10.1007/BFb0035389.

[93] K. Jensen and L. M. Kristensen, *Coloured Petri Nets*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. doi: 10.1007/b95112.

[94] I. Jimenez-Ochoa, O. Begovich, A. Ramirez-Trevino, and L. I. Aguirre-Salas, “Implementing BDI agents using petri nets,” in *SMC’03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483),* Washington, DC, USA, 2003, vol. 1, pp. 286–291. doi: 10.1109/ICSMC.2003.1243830.

[95] H. Zhang, Z. Shen, S. Y. Huang, and C. Miao, “Predicting Responsiveness of BDI Agent,” p. 6.

[96] R. Smith, “Google’s Chromecast 2 is Powered by Marvell’s ARMADA 1500 Mini Plus - Dual-Core Cortex-A7,” *AnandTech*, Oct. 05, 2015. https://www.anandtech.com/show/9688/googles-chromecast-2-is-powered-by-marvells-armada-1500-mini-plus-dual-cortexa7

[97] Raspberry Pi (Trading) Ltd, “Raspberry Pi Products,” *raspberrypi.com*, 2022. https://datasheets.raspberrypi.com/
[98] Canada Computers & Electronics, “AMAZON Echo Plus (2nd gen),” *Canada Computers*, 2022. https://www.canadacomputers.com/product_info.php?cPath=1578&item_id=137733
[99] Ubiquiti Inc., “UniFi Dream Machine Datasheet,” Ubiquiti Inc., United States, Product Datasheet JL121819, 2019. [Online]. Available: https://dl.ui.com/ds/udm_ds
[100] ASUSTeK Computer Inc., “RT-AC88U Tech Specs,” *Asus.com*, 2022. https://www.asus.com/ca-en/networking-iot-servers/wifi-routers/asus-wifi-routers/rt-ac88u/techspec/
[101] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, Jun. 2006, doi: 10.1016/j.patrec.2005.10.010.
[102] Y. Hedin and E. Moradian, “Security in Multi-Agent Systems,” *Procedia Computer Science*, vol. 60, pp. 1604–1612, 2015, doi: 10.1016/j.procs.2015.08.270.
[103] B. E. Sabir, M. Youssfi, O. Bouattane, and H. Allali, “Towards a New Model to Secure IoT-based Smart Home Mobile Agents using Blockchain Technology,” *Eng. Technol. Appl. Sci. Res.*, vol. 10, no. 2, pp. 5441–5447, Apr. 2020, doi: 10.48084/etasr.3394.
[104] H. Hagras, “Towards Human Understandable Explainable AI,” *Computer*, vol. 51, no. 9, pp. 28–36, Sep. 2018, doi: 10.1109/MC.2018.3620965.
[105] G. Geetha and C. Jayakumar, “Implementation of Trust and Reputation Management for Free-Roaming Mobile Agent Security,” *IEEE Systems Journal*, vol. 9, no. 2, pp. 556–566, Jun. 2015, doi: 10.1109/JSYST.2013.2292192.
[106] X. Xu, N. Bessis, and J. Cao, “An Autonomic Agent Trust Model for IoT systems,” *Procedia Computer Science*, vol. 21, pp. 107–113, 2013, doi: 10.1016/j.procs.2013.09.016.
[107] K. Zhang, Z. Yang, and T. Başar, “Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms.” arXiv, Apr. 28, 2021. Accessed: Jul. 17, 2022. [Online]. Available: http://arxiv.org/abs/1911.10635
[108] L. Canese *et al.*, “Multi-Agent Reinforcement Learning: A Review of Challenges and Applications,” *Applied Sciences*, vol. 11, no. 11, p. 4948, May 2021, doi: 10.3390/app11114948.
[109] G. Rens, “A BDI Agent Architecture for a POMDP Planner,” in *9th International Symposium on Logical Formalization of Commonsense Reasoning: Commonsense 2009*, Toronto, Canada, Jun. 2009, p. 6.
[110] S. Musman and A. Turner, “A game theoretic approach to cyber security risk management,” *Journal of Defense Modeling & Simulation*, vol. 15, no. 2, pp. 127–146, Apr. 2018, doi: 10.1177/1548512917699724.
[111] S. Musman and A. Temin, “A Cyber Mission Impact Assessment Tool,” presented at the 2015 IEEE International Symposium on Technologies for Homeland Security (HST), Apr. 2015. doi: 10.1109/THS.2015.7225283.

Appendix

IoT Device Baseline Security Requirements [19], [26]

Security Requirement	Description	Device Capabilities (NIST 8259A)
Asset Management	Maintain a current, accurate inventory of all IoT devices and their relevant characteristics throughout the devices' lifecycles in order to use that information for cybersecurity and privacy risk management purposes.	• DC1.1 A unique logical identifier. • DC1.2 A unique physical. The identifier at an external or internal location on the device accessible to the consumer.
Device Configuration	Identify and eliminate known vulnerabilities in IoT device software and firmware in order to reduce the likelihood and ease of exploitation and compromise.	• DC2.1 The ability to change the product component's software configuration settings, including disabling unwanted features. • DC2.2 The ability to restrict configuration changes to unauthorized individuals and other IoT product components only. • DC2.3 A default setting for the initial configuration which makes the product component secure for unexpected use cases. Any security features should be enabled by default. • DC2.4 The ability for authorized individuals and other IoT product components to restore the product component to the default security configuration.
Data Protection	Prevent access to and tampering with data at rest or in transit that might expose sensitive information or allow manipulation or disruption of IoT device operations.	• DC3.1 The ability to use demonstrably secure cryptography (e.g., modules consistent with FIPS 140-3) for cryptographic algorithms (e.g., encryption with authentication, cryptographic hashes, digital signature validation) to protect the confidentiality and integrity of all the product component's stored (e.g., collected and received data, internal software) and transmitted data. Note: The product component host may depend on or limit available cryptographic modules. • DC3.2 The ability to protect the product component's stored data from unauthorized change (e.g., protect against

		injected code or data manipulation attacks). • DC3.3 The ability for authorized persons to render all data on the product component that is not the initial default configuration (see Device Configuration) and any initial software included on the device (including updates) inaccessible to anyone, whether previously authorized or not. Note: for components implemented in a shared environment (e.g., auxiliary backend), and this may be limited to data and configurations associated with the IoT product customer. • DC3.4 The ability for authorized individuals, other IoT product components, and/or systems to delete data at rest from the product component. Note: Components are implemented in a shared environment (e.g., auxiliary backend), and this may be limited to data associated with the IoT product customer.
Access Management	Prevent unauthorized and improper physical and logical access to, usage of, and administration of IoT devices by people, processes, and other computing devices.	• DC4.1 The ability to logically or physically disable any local and network interfaces that are not necessary for the core functionality of the product component. • DC4.2 The ability to logically restrict access to each network interface to only authorized persons or devices. • DC4.3 The ability of the product component to validate that the input received through its interfaces matches specified definitions of format and content. • DC4.4 The ability to authenticate individuals and other IoT product components using appropriate mechanisms to technology, risk and use case. Authenticators could be biometrics, passwords, etc. • DC4.5 The ability to support secure use of authenticators (e.g., passwords) including a. if necessary, the ability to locally manage authenticators b. ability to ensure a strong, non-default authenticator is used (e.g., not delivering the product with any single default password or enforcing a change to a default password before the product component is deployed for use). • DC4.6 Configuration settings for use with the Device Configuration capability, including the ability to enable, disable, and adjust thresholds for any ability the

		device might have to lock or disable an account or to delay additional authentication attempts after too many failed attempts (*NIST8259A only).
Vulnerability Management	Identify and eliminate known vulnerabilities in IoT device software and firmware in order to reduce the likelihood and ease of exploitation and compromise.	• DC5.1 The ability to update the product component's software remote (e.g., network download). • DC5.2 The ability for the product component to verify and authenticate any update before installing it. • DC5.3 The ability to enable or disable notifications about updates.
Incident Detection	Monitor and analyze IoT device activity for signs of incidents involving device security and data security.	• DC6.1 The ability to log cybersecurity-related state information (e.g., software update installations, failed login attempts, configuration changes). • DC6.2 The ability to restrict access to the state information so only authorized individuals and IoT product components can view it. • DC6.3 The ability to prevent any unauthorized edits of state information by any entity.
Availability	The ability for the device to perform its service operations as defined in the non-technical capability label.	• DC7.1 The ability for the device to continue operating (possibly with limited digital functionality) in the case of a network outage or other connectivity disruption. Operational features of the device should continue to function without connectivity.