POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

In collaboration with University of Calgary

User-friendly Security Automation for Domotic Networks

Supervisors

Prof. Riccardo SISTO

Prof. Lorenzo DE CARLI

Prof. Fulvio VALENZA

Dott. Daniele BRINGHENTI

Candidate

Vincenzo MARINO

Academic Year 2022/2023

Summary

The Network Functions Virtualization (NFV) paradigm revolutionizes networking technology by enabling the installation of software processes as service functions on general-purpose servers. Through NFV, a Service Graph is created, offering improved flexibility and performance by enabling multiple paths between endpoints.

However, the manual creation and configuration of Service Graphs, especially when it involves security functions, present challenges. Security functions, such as firewalls and parental controls, play a crucial role in ensuring network safety. Misconfigurations and delays in updating security defenses to meet evolving security requirements are common risks associated with manual operations.

To address these obstacles, this thesis contributes to the development of VEREFOO (VErified REFinement and Optimized Orchestration). VEREFOO is a framework that aims to automate Security and Network Management by automatically allocating and configuring Network Security Functions on a Service Graph. It achieves this by addressing a set of Network Security Requirements that can be formulated using a high-level security language and a refinement process. The proposed approach involves the formulation of a MaxSMT problem, with the objective of maximizing the sum of weights assigned to satisfied soft clauses while adhering to hard constraints, that always require to be satisfied. This formulation provides a formal verification of the solution's correctness.

The significant contribution of this thesis work lies in extending the functionality of VEREFOO's ADP module to enable automatic allocation and/or configuration of Parental Control Systems. These systems can be implemented either within a device with full internet capabilities or within an Allocation Place along the path between the source and destination of the requirement. This thesis has introduced two types of Parental Control Systems. The first type is a simpler system that allows traffic customization for each user based on predefined filter levels. The second type is a more advanced system that enables device-level constraints, such as setting a daily time limit for device usage.

Extensive testing of the implementation has been conducted in various network scenarios, demonstrating that the proposed approach provides a viable alternative to manual allocation and configuration of Parental Control Systems. Furthermore, the approach has been designed to be extensible, allowing for future enhancements such as defining requirements for specific categories of websites and incorporating other features commonly found in current Parental Control solutions.

Acknowledgements

I would like to express my sincere gratitude and appreciation to my supervisors, Sisto, De Carli, Valenza, and Bringhenti, for their unwavering support and guidance throughout the entire process of conducting this research and writing my thesis. Their prompt and insightful feedback played a crucial role in shaping the quality and depth of my work. I am truly fortunate to have had their constant availability and willingness to address any questions or concerns that arose during my research journey.

I am also profoundly grateful for the incredible opportunity they provided me to undertake research in Canada. This experience has not only broadened my horizons and enriched my academic journey but has also allowed me to meet extraordinary individuals and form lasting friendships. I am forever grateful to those who accompanied me on this incredible journey and for the memorable experiences we shared.

Furthermore, I want to express my heartfelt appreciation to my family for their unwavering support, love, and encouragement throughout my academic journey. Their belief in my abilities and their constant encouragement have been my source of strength, inspiring me to persevere through challenges and pursue my goals.

I would also like to extend my gratitude to my friends and my girlfriend for their understanding, encouragement, and unwavering support during this challenging yet fulfilling journey. Their presence and uplifting words have provided me with the necessary motivation to overcome obstacles and maintain a positive mindset.

Finally, I would like to acknowledge the contributions of all those who have played a part in this journey. Your unwavering support and belief in my abilities have been instrumental in transforming this thesis into a reality. I am deeply grateful for your presence and the positive impact you have had on my academic and personal growth.

Thank you all from the bottom of my heart.

Contents

List of Figures

Listings

Chapter 1

Introduction

The field of networking has experienced a revolutionary transformation with the introduction of SDN (Software-Defined Networking) and NFV (Network Function Virtualization). SDN enables the utilization of software processes to establish and manage packet paths within the physical network, whereas NFV allows for the execution of diverse network functions on standard hardware platforms. These innovative technologies provide an innovative solution to the conventional implementation of Service Function Chains (SFC), addressing significant limitations such as limited resource sharing and costly modifications.

In the traditional approach, the provisioning of services necessary for creating a Service Graph (a generalized form of SFC) such as NAT or load balancing heavily relied on dedicated hardware devices. However, the emergence of SDN and NFV has significantly reduced the dependence on dedicated hardware. These technologies enable the virtualization of network functions, resulting in more efficient and flexible deployment of services within a Service Graph. By sharing physical resources among multiple virtual functions, not only service functions but also Network Security Functions (NSFs) like Intrusion Detection Systems (IDS) and Packet Filter can benefit from this approach.

However, the process of manually creating and configuring Service Graphs, particularly when security functions are involved, can be challenging. Security functions, such as firewalls and parental controls, are essential for maintaining network safety. Manual operations pose risks such as misconfigurations and delays in updating security defenses to keep up with evolving security requirements. By embracing Network Automation, it becomes possible to overcome these challenges. Network Automation allows for seamless adaptation to diverse scenarios without the need for human intervention. It enables automatic management of configuration changes, ensuring efficient and timely updates to security defenses.

Verefoo is a comprehensive solution designed to allocate and configure selected Network Security Functions (NSFs) in an automatic way. Its primary objective is to fulfill a predefined set of Network Security Requirements (NSRs), which can be expressed using a high-level language. In addition to satisfying the NSRs, the framework ensures the optimal placement of virtual functions present in the Service Graph onto the servers of a physical network.

The main objective of this thesis is to extend the functionality of Verefoo by addressing the unique requirements of Smart Home environments. While Verefoo was initially designed for automating the configuration aspects of enterprise networks, this project focuses on adapting it to address the security challenges specific to home networks.

The security of home networks is paramount due to several reasons. Firstly, these networks store personal and sensitive information, including financial data and personal documents. Securing these networks is essential for safeguarding individuals' privacy and preventing unauthorized access to their personal information.

Secondly, with the proliferation of smart devices in homes, including smartphones, tablets, smart TVs, and IoT devices, the security of these devices becomes crucial. Home networks serve as the backbone for connecting and controlling these devices, making them potential targets for cyber attacks. By ensuring the security of home networks, individuals can protect their devices from vulnerabilities and potential attacks that could compromise their functionality or expose them to risks.

Additionally, home networks often connect multiple devices used by different family members, including children. Implementing robust security measures, such as parental controls, is crucial for protecting children from online threats and inappropriate content. Parental control mechanisms enable parents to regulate access rights and privileges, creating a safe online environment for their children.

The need for parental controls arises from the increasing internet usage among children, exposing them to various risks like cyberbullying, inappropriate content, and online predators. Parental control features, such as content filtering, time restrictions, and activity monitoring, help parents ensure a secure online experience for their children.

The primary objective of this project is to modify Verefoo to enable the automatic allocation and configuration of Parental Control Systems, addressing a crucial issue in home networks. The complexity and technical expertise required for configuring Parental Control Systems pose significant challenges. Traditional methods involve manual configuration on individual devices or network routers, which can be time-consuming, prone to errors, and difficult for non-technical users to navigate. As a result, parents may encounter difficulties in understanding and implementing appropriate controls and filters to safeguard their children from potential online threats.

Furthermore, the dynamic nature of the online environment requires continuous monitoring and adjustments to Parental Control settings. Manual updates and modifications can be cumbersome, leading to delays in adapting to evolving online risks. Automating the allocation and configuration of Parental Control Systems streamlines this process, ensuring efficient and up-to-date protection.

The extension of Verefoo's ADP module empowers it to automatically allocate and configure Parental Control Systems as Network Security Functions (NSFs) within the home network. This automation simplifies the management of parental controls, providing parents with peace of mind knowing that their children are accessing the internet safely.

Overall, the automatic allocation and configuration of Parental Control Systems in home networks address a crucial problem faced by parents. By leveraging the capabilities of Verefoo and the ADP module, this project offers an efficient and effective approach to securing home networks, enhancing online safety for children, and simplifying parental control management.

1.1 Thesis objectives

The objectives of this thesis can be summarized as follows:

1. Creation of realistic scenarios: The first objective is to create scenarios that reflect real home network configurations, including commercial IoT devices. These scenarios will help identify the problems and available solutions in the field of Smart Home Networks, providing a foundation for further research

2. Address the challenges of Parental Control configuration: The second objective is to highlight the specific problem of configuring Parental Controls in Smart Home networks. This task can be cumbersome for parents due to their lack of technical skills and the diverse range of devices, as well as the dynamic and complex nature of the environment. The aim is to demonstrate how the automatic allocation and configuration of Parental Controls can provide an effective solution to support parents in managing online safety for their children.

3. Leveraging Verefoo framework: The third objective is to showcase the value of the Verefoo framework in enabling automatic allocation and configuration of Parental Controls. This includes modifications to the ADP module of the framework to accommodate Parental Control Requirements, new EndPoints, and the introduction of a new Network Security Function, the Parental Control System.

4. Provide practical examples and costructive discussion: The final objective is to provide concrete examples of the modified framework in action. These examples serve as use case scenarios, illustrating the practical application of the framework in addressing Parental Control configuration challenges.

Additionally, the thesis discusses the limitations and technical difficulties encountered during the thesis work, highlighting the potential for further advancements to create a comprehensive solution.

By achieving these objectives, the thesis aims to contribute to the understanding and advancement of Smart Home Networks, particularly in the area of Parental Control configuration. The ultimate goal is to provide a complete and practical solution that empowers parents to ensure a safe and secure online environment for their children within the home network.

1.2 Thesis description

The organization of this thesis is outlined below, providing a brief description of each chapter:

- Chapter 2 provides a comprehensive analysis of previous research and studies in the field. It begins by addressing works related to home network security in general, followed by a focus on the security issues that arise when multiple users share a smart home environment. The chapter also explores the user perception of IoT security and emphasizes the importance of understanding the capabilities of IoT devices to effectively protect the system against potential security threats. Furthermore, it delves into relevant literature concerning the Verefoo framework, which is particularly relevant as this thesis aims to extend Verefoo's functionality to address some of the aforementioned challenges. By reviewing the existing knowledge, this thesis establishes a solid groundwork for tackling the unique context of Smart Home Network security, which diverges from the conventional topics addressed by Verefoo.

- Chapter 3 aims to thoroughly examine the diverse scenarios present within smart home environments, with the objective of identifying potential threats that may compromise the security and privacy of the occupants. A particular focus will be placed on addressing the urgent issue of automatic allocation and configuration of parental controls. To provide a solid foundation for this research, the concept of Software Networking has been discussed, in a way to introduce the Verefoo framework. To facilitate this, a concise overview of Software Defined Networks (SDN) and Network Function Virtualization (NFV) has been provided, highlighting their contribution to the development of Network Automation. Finally, the Verefoo framework has been introduced, encompassing a detailed description of its key components, including the ADP module and z3.

- Chapter 4 focuses on the design and implementation of the extension for the ADP module. The chapter begins by describing the design and implementation of the newly developed Network Security Requirements, the Parental Control Requirements. The endpoints introduced in this thesis work are then discussed, along with their implementation in the XML Schema. The chapter proceeds to explain the model and implementation of the Parental Control Systems. To validate and provide a concrete example of the framework's functionality, some use cases are presented, showcasing the results obtained when running the framework on the Input Allocation Graph containing the newly developed elements and utilizing the new requirements as constraints. Finally, a Discussion section offers insights into the achieved objectives and addresses the current limitations of the framework.

- Chapter 5 serves as the concluding chapter of the thesis, encompassing the final remarks and discussing future prospects. In this chapter, the main findings and contributions of the research conducted throughout the thesis are summarized and discussed. Furthermore, it explores potential directions for future work, such as extending the developed Parental Control System model or delving deeper into the various challenges inherent to the Smart Home environment. By contemplating these possibilities, the chapter sets the stage for further advancements and exploration in the field.

Chapter 2

Related Work

This chapter presents an overview of some of the related works already present on this topic. The first section presents some works related to the home network security problems in general. The second section describes the concept of multiple users sharing a smart home environment and the security problems that can arise. The third section discusses the user perception of IoT Security and the importance of being aware of the capabilities of an IoT device, in a way to protect the system from possible security issues. Finally, the last section highlights the Verefoo framework, developed at Politecnico di Torino, as a key reference. This is particularly relevant as the thesis work primarily focuses on extending the functionality of Verefoo to address some of the aforementioned challenges.

2.1 Home network security

The rapid growth of the Internet of Things (IoT) has brought numerous benefits to users, but it has also introduced new security challenges. Home IoT devices, in particular, are subject to poor security due to both unsecure programming and a lack of user awareness. Successful attacks on these devices can be very serious and difficult to detect, making traditional security techniques ineffective. This section provides an overview of related studies that have addressed the issue of home network security. While these studies extend beyond the specific focus of this thesis, they contribute to a better understanding of the main challenges faced by home networks. This understanding is crucial for this thesis, as its main objective is to adapt Verefoo, a framework designed for the automatic generation of various configuration aspects in enterprise networks, to address the unique challenges of the smart home environment. The exploration of various scenarios and solutions within this context has helped identify the specific problem to be addressed in this thesis, refining its focus.

2.1.1 User/action identification in home networks

Bhosale et al. [1] proposed a technique for detecting abnormal user behaviors by analyzing the network traffic of each IoT device.

Here, a random-forest classifier is used to map the flow of packets to user activities. Subsequently, an unsupervised K-means clustering algorithm is employed to differentiate between malicious and benign users. One notable advantage of this method is its focus on user activities rather than the network level, enabling the discovery of abnormal user behaviors. Moreover, it remains effective even in the presence of encrypted traffic, where traditional Deep Packet Inspection techniques may fail. However, an attacker could potentially evade detection by mimicking a user's behavior, which poses a challenge to this approach.

One aspect that stands out about this technique is its ability to identify privacy violations, including unauthorized normal activities. By basing the analysis on activities rather than packets, the method detects deviations from the typical usage patterns of individual IoT devices, thus building user-specific usage models. Additionally, for devices that use unencrypted packets and generate minimal traffic, where machine learning classification may not be effective, the technique leverages a comparison of packet payloads with the training data for classification purposes.

However, a potential limitation of this technique is its difficulty in handling users who exhibit varying patterns of IoT device usage, possibly due to a hectic lifestyle. In such cases, abnormal behavior might be incorrectly classified as normal, potentially impacting the accuracy of the framework.

Overall, this framework can be highly valuable, particularly in contexts where users have limited awareness of network security issues. It provides assistance in detecting malicious usage of IoT devices and offers a means of uncovering privacy violations. However, it is important to address the potential challenges associated with diverse user behaviors to ensure the accuracy and effectiveness of the framework

2.1.2 Device type identification for Iot networks

Miettinen et al. [2] addressed the vulnerabilities commonly found in IoT devices and the need for mechanisms to protect other devices within the same network.

IoT Sentinel presents a system able to identify the types of devices connected to an IoT network in an automatic way. It performs a vulnerability assessment based on these device types and restricts the communication of vulnerable devices. The identification process is performed in an automatic, accurate, and scalable manner, relying on the behavioral characteristics of the devices without prior knowledge of their message syntax. By conducting vulnerability assessments, the system identifies devices with security flaws and limits their communication. This prevents attackers from compromising other devices within the network or accessing data from vulnerable ones. In cases where vulnerable IoT devices are equipped with external communication channels, network isolation and traffic filtering alone are insufficient. Therefore, IoT Sentinel offers a mechanism to help users identify the specific devices in question and ensure their removal from the network.

The primary advantage of this approach is its scalability during the identification phase. Each time the fingerprint of a new device is captured, there is no need to modify the existing classifiers. Instead, a new classifier is trained, avoiding the need to repeat the entire learning process. Another noteworthy aspect is that the framework can protect other devices in the same network without requiring software installation. This capability enables it to work effectively even in the presence of legacy devices.

However, one potential limitation is the level of granularity used for classifying devices, which may be too coarse-grained. Consequently, there is a possibility that a particular device may not be affected by the vulnerability despite being assigned to a vulnerable device type.

In conclusion, this approach is highly valuable, especially for protecting devices in the same network as legacy IoT devices that often lack firmware update mechanisms. Given the difficulty of updating such devices, attackers can easily exploit the most vulnerable devices to compromise others. Thus, IoT Sentinel provides an effective solution to enhance the security of IoT networks and safeguard against such risks.

2.1.3 SDN-based redirection for home networks

Taylor et al. [3] addressed the emerging security challenges in residential networks caused by the increasing number of devices. They proposed the outsourcing of network management and security tasks to Cloud-based Security Services utilizing Software-Defined Networking (SDN). However, this approach may introduce performance implications that need to be evaluated.

The authors presented a technique for measuring the performance impact of a Cloud-based OpenFlow controller and the influence of controller latency on user-perceived performance. The study considers residential network connectivity in the continental US and utilizes 13 measurement servers hosted within Virtual Machines by four major providers, geographically distributed across the US. The paper focuses on the OpenFlow Controller placement problem for residential SDNs and reveals that 90% of users were within a 50ms cloud distance. This latency has an impact on the web browsing experience, although optimization strategies can mitigate this effect.

One favorable aspect of this measurement technique is its conservative approach. The controller only installs a uni-directional rule, resulting in an additional round of latency when a response reaches the controller. Another noteworthy aspect is that the study quantifies the performance impact using Page Load Time, which represents the worst-case scenario for controller latency as it consists of numerous short flows that cannot be amortized.

Regarding the comparison between enterprise routers and consumer routers, it would have been beneficial to conduct more experiments to measure the overhead associated with Transport Layer Security (TLS). Since enterprise routers do not support TLS, the authors were unable to evaluate this aspect.

In conclusion, considering that end-users often lack the expertise to administer their networks and that residential consumer routers are not frequently updated, the utilization of a Cloud-based SDN Controller in residential networks emerges as a promising option. However, further investigation is necessary to delve deeper into this topic and explore its potential implications and benefits.

2.1.4 Capability extraction for home networks

Dolan et al. [4] proposed a technique aimed at addressing the limited visibility of IoT device capabilities, which can pose privacy and security risks. As IoT devices become increasingly prevalent, users often lack a comprehensive understanding of their functionality and associated threats due to manufacturers abstracting design complexities.

The proposed technique takes a proactive approach by extracting device capabilities from informational specifications, enabling the verification of potential threats even prior to device installation. The paper also evaluates the accuracy and efficiency of the technique. Notably, unlike other approaches reliant on environmental data, this method offers distinct advantages. It avoids data interpretation challenges and prevents the disclosure of sensitive information. However, it relies solely on vendor-provided materials, which may be incomplete.

The primary advantage of this approach lies in its proactive nature. By analyzing device capabilities, it can identify potential privacy and security threats, leveraging publicly available specifications without the need for device deployment. Additionally, the evaluation process encompasses devices from various vendors, ensuring its applicability across different device classes.

However, the technique has some limitations. Its effectiveness is dependent solely on vendor specifications, which may lack crucial information. To enhance effectiveness, consideration should be given to incorporating other sources, such as device configuration or software specifications. Moreover, the technique primarily focuses on threats related to device transducers, indicating that it alone cannot address all security aspects.

In conclusion, this innovative solution enables the early detection of threats before IoT device installation. However, to have a stronger security, it is advisable to combine this technique with other approaches to strengthen overall protection.

2.1.5 Policy-based networking for home networks

Nobakht et al. [5] addressed the growing security challenges associated with the increasing adoption of IoT devices. Given the limitations in addressing vulnerabilities across all IoT devices, particularly legacy or unsupported ones, a common solution is to implement additional security measures at the network level.

This paper introduces IoT-NetSec, a framework that enables policy-based and fine-grained traffic monitoring within the network. The authors describe a prototype implementation and its integration with an SDN controller. With this framework, users can register new IoT devices and define security policies that govern communication with them. Additionally, the framework monitors network traffic for these devices, enforcing predefined security requirements specified in the policy and triggering an alarm if the data traffic rate exceeds a specified threshold.

The main advantage of this approach lies in its focus on network-level security within IoT device segments, distinguishing it from previous SDN-based solutions. Unlike previous approaches that assumed a network composed solely of IoT devices or proposed general-purpose security applications for both IoT and conventional computer networks, this paper's solution specifically targets network-level security monitoring for IoT device segments. This strategic choice significantly reduces the burden on security applications in SDN environments. Furthermore, it leverages existing mature solutions designed for conventional computer network systems, which enhances efficiency and effectiveness.

However, one potential drawback is the detection capability of the framework, which could be improved by incorporating the ability to identify and address other common attacks prevalent in the IoT domain.

In conclusion, IoT-NetSec offers a robust solution for implementing security measures in IoT devices, particularly when applying network-level security becomes the most viable option. This is particularly relevant when dealing with legacy devices or when updating or patching the firmware is challenging.

2.2 Multi User Control in Home Devices

The increasing adoption of IoT devices brings convenience to our lives, but it also introduces security threats, both from external sources and legitimate users within the home. Conventional access control mechanisms implemented in common IoT platforms often fail to address insider threats effectively. This section explores three papers that propose innovative solutions for enhancing security and access control in the context of IoT devices.

2.2.1 "SoftAuthZ: A Context-Aware, Behavior-Based Authorization Framework for Home IoT"

Ghosh et al. [6] proposed SoftAuthZ, a novel security framework that addresses the security threats posed by legitimate users within a home IoT environment. Common IoT platforms often lack sufficient security measures to handle insider threats resulting from device sharing, social relationships, and trust among users.

SoftAuthZ introduces a context-sensitive and behavior-based security framework that utilizes soft security mechanisms to support authorization decisions. It constructs a linear regression model based on multiple IoT environment-specific attributes and computes confidence scores for access requests. The authorization decisions are then made based on these confidence scores. The framework's efficacy is evaluated through analysis and simulation-based evaluation, and it demonstrates the ability to classify users based on their usage patterns.

One key advantage of this technique is its consideration of the context of access requests, such as the time of the request, the age of the requester, and the presence of other individuals in the home. This contextual information enhances the authorization process, moving beyond relying solely on past request patterns. Additionally, the framework takes into account device categorization, giving weightage to devices belonging to specific categories like 'Safety and Security', thereby influencing the computation of confidence scores.

However, there are a few limitations worth noting. The predefined hierarchical membership classes used by the framework might be overly specific and may not accommodate situations where individuals are not part of a family, such as roommates. Furthermore, the framework primarily focuses on supporting authorization decisions based on confidence scores, lacking the ability to define specific permissions for different actions.

In conclusion, SoftAuthZ is a promising technique for supporting authorization decisions in home IoT environments. However, considering the focus of this thesis on the automatic allocation and configuration of Parental Control Systems, the scope of SoftAuthZ extends beyond the specific objectives of this research. While both SoftAuthZ and Parental Control Systems involve making decisions regarding access and permissions, such as granting or denying access to specific resources, Parental Control Systems tackle a more specific problem within the broader domain of access control.

2.2.2 "Enabling Multi-user Controls in Smart Home Devices"

Jang et al. [7] discussed the increasing usage of IoT devices and the associated security concerns, particularly the problem of unintended user access within the home, which becomes more pronounced in multi-user environments. The authors highlighted that most devices currently employ a coarse-grained access control mechanism, where a user is either granted full access or no access at all.

The paper presents several scenarios that emphasize the significance of flexible authorization control and seamless authentication in multi-user environments. Additionally, it identifies the challenges faced in meeting the expectations of all users within a home and provides design suggestions for IoT device manufacturers to incorporate precise access control and authentication mechanisms.

A notable aspect of the paper is the classification of user roles assumed in IoT devices, including the primary user, alternate primary users, secondary users, and guests. Each role is associated with varying levels of risk. This classification effectively addresses the diverse situations that may arise in a multi-user environment, without overly specifying the relationships among users.

While the paper thoroughly analyzes the pros and cons of different techniques for enabling effortless multi-user authentication, it would have benefitted from conducting interviews and user studies. Such research methods would have provided a clearer understanding of the real issues and identified the most suitable solutions for multi-user environments.

In conclusion, this paper has significant practical implications, not only for IoT device manufacturers but also for researchers to enhance their understanding of the primary problems in multi-user environments. It provides valuable insights into the primary issues that arise in such settings and contributes to further analysis of the problem. However, it should be noted that while this broader perspective aids in obtaining a comprehensive understanding of the challenges that may affect multi-user environments, the scope of this paper extends beyond the focus of this thesis.

2.2.3 "Who's Controlling My Device? Multi-User Multi-Device-Aware Access Control System for Shared Smart Home Environment"

In the context of a multi-user multi-device smart home, various challenges arise, such as managing conflicting, complex, and dynamically changing demands from users, as well as handling the presence of IoT devices from different vendors that share the same environment. Traditional access control techniques, designed for single-user scenarios, are inadequate in addressing these challenges independently.

Sikder et al. [8] presented Kratos+, a multi-user multi-device-aware access control system that enables users of IoT devices to specify their access control demands. Kratos+ implements a priority-based policy negotiation technique that automatically resolves conflicting user demands in shared environments with multiple users and devices. It enforces policies without imposing significant overhead.

The system was tested against five different threats associated with inadequate access control, achieving a 100% success rate in threat detection. Additionally, a usability study was conducted, resulting in a 4.6 rating out of 5, indicating its effectiveness.

One notable advantage of Kratos+ is its platform-independent and fine-grained access control implementation in multi-user multi-device environments. Unlike current access control systems in smart home platforms that operate in a binary manner, allowing users to either have full control or no control, Kratos+ introduces user priority to define the roles and permissions for each user within the system. Another favorable aspect is its support for both app-level and access point-level access control implementation, enabling multi-platform smart environments. Kratos+ can be integrated into vendor-provided mobile apps, providing a single user interface for managing users, assigning policies to connected devices, and offering a useful notification system.

However, a potential drawback of this solution is that, when executing policies at the access point, Kratos+ cannot identify value-based device conditions specified by users within policies due to encrypted payloads. Executing the policies in the access point is mostly required for the presence of vendor-specific smartphone apps which cannot be modified to accommodate Kratos+.

In conclusion, the presented solution offers a robust approach to access control management in multi-user multi-device environments, being able to effectively resolve conflicting user requests and enforcing policies without imposing significant overhead. However, while Kratos+ offers a comprehensive solution for access control, this thesis work specifically focuses on a particular type of access control: parental control. Parental control systems aim to regulate and supervise the access rights and privileges of children within digital environments. Specifically, the primary objective of this thesis is to develop an automated approach for allocating and configuring parental controls within a home network.

2.3 "User perceptions of IoT security and IoT security survey"

In the recent years, the proliferation of IoT devices has raised concerns about privacy and security. To offer a deeper understanding of these critical aspects, researchers have conducted studies to explore user perceptions, knowledge, and attitudes towards privacy and security in the context of smart homes and IoT environments. This section of the thesis discusses a selection of papers that delve into these topics and provide valuable insights.

Emami-Naein et al. [9] and Zheng et al. [10] employed semi-structured interviews to investigate the awareness and understanding of privacy and security among smart IoT device owners. In their study, Emami-Naein et al. even developed a prototype privacy and security label for IoT devices, while Zheng et al. generated recommendations to be used in future to better understand users' evolving attitudes towards privacy in smart homes.

In a similar way, Haney et al. [11] conducted qualitative interview studies with residents of smart homes. The objective was to gain a deeper comprehension of their perceptions regarding privacy and security within smart home environments and educate users about their responsibilities in these domains.

Kulyk et al. [12] conducted a survey to better understand the extent to which the users perceive the ability and the motivation of system providers to ensure the necessary protection, and some recommendations were proposed from the results.

To gain insights into people's concerns about device tracking in IoT environments, Lee et al. [13] employed cluster analysis to examine how contextual parameters influence individuals' privacy concerns related to device tracking.

Lastly, Fagan et al. [14] presented a comprehensive technical review of the security features implemented in various IoT devices. The primary goal was to assess the security capabilities of these devices and offer general considerations for manufacturers to enhance their security measures.

While the papers discussed delve into important aspects of privacy and security in IoT environments, it is important to acknowledge that their scope extends beyond the specific focus of this thesis. The primary objective of this research is to address the unique challenges that arise in shared smart home environments, specifically focusing on the allocation and configuration of Parental Controls within a home network. However, these papers serve as valuable resources as they provide a broader understanding of the privacy and security concerns in the context of Smart Home Networks, which differs from the traditional focus of Verefoo on enterprise networks. These studies shed light on the inherent vulnerabilities of IoT devices and the lack of awareness regarding security issues, emphasizing the significance of addressing the problems specific to home networks.

2.4 Verefoo and network policy

The ever-increasing prevalence of cybersecurity attacks has brought significant attention to the network security functions (NSFs) misconfiguration such as VPN terminators and firewalls. Manual and suboptimal configuration approaches have contributed to the rise of this critical exploit. In response, researchers are exploring the potential of automated policy-based network security management tools to reduce human errors and enhance security services. By leveraging the advancements in networking technologies, such as Network Functions Virtualization (NFV) and Software-Defined Networking (SDN), these automated tools can optimize the allocation and configuration of security functions within virtualized network services.

Bringhenti et al. [15] proposed a novel framework called VEREFOO (Verified Refinement and Optimized Orchestration), which aims to automate and optimize the allocation and configuration of NSFs in a formal and verified manner. The framework integrates directly with cloud orchestrators, providing agility and efficiency in security management. By transforming high-level security requirements into an optimal allocation scheme and configuration on a Service Graph (SG), VEREFOO ensures correctness and optimality while minimizing resource utilization. The framework is designed to work seamlessly with popular orchestrators like Open Baton and Kubernetes, enabling the deployment and configuration of virtual functions without manual intervention.

In the VEREFOO framework section, the paper explains the approach taken in designing the architecture. VEREFOO follows a modular approach and focuses on optimized allocation and configuration of NSFs based on input Network Security Requirements (NSRs) expressed using a high-level language. The framework automatically transforms a given Service Graph into an Allocation Graph and formulates a Maximum Satisfiability Modulo Theories (MaxSMT) problem to determine the optimal allocation scheme. It also incorporates a correctness-by-construction approach, ensuring formal assurance of correctness, and optimizes the allocation and configuration of NSFs based on various cost functions.

By incorporating the VEREFOO framework into security automation processes, organizations can achieve enhanced network security management with reduced human errors, optimal resource utilization, and improved agility in dynamic virtualized environments.

In this thesis work the Verefoo framework holds particular significance as the main objective of the thesis is given by the extension of the ADP module of Verefoo in a way to allocate and configure Parental Control Systems as Network Security Functions (NSFs) in an automatic and optimal manner, ensuring the fulfillment of the associated Parental Control Requirements.

2.4.1 Verefoo selection of functions

Here are presented two papers related to the optimization of security configuration workflows in network virtualization environments.

In the first paper, Bringhenti et al. [16] presented a new approach to security configuration workflow. This approach is centered around abstractions called projections, which define the security operations needed to be performed by Virtual Network Functions (VNFs) to enforce security policies. This new workflow allows for the postponement of VNF selection until their deployment in the physical network, resulting in optimized results in terms of the number of selected VNFs and their deployment cost.

In the second paper, Bringhenti et al. [17] presented an innovative method that incorporates functionality abstraction to synthesize virtual security services. This approach aims to optimize the selection of virtual security functions from a large pool of available options by working at the virtual level without considering the different function implementations. The function selection is postponed until after the configuration of the virtual graph, enabling more efficient selection when dealing with a large number of available functions.

Both papers address the challenges faced by network administrators in selecting and configuring the most suitable security functions in virtualized networks. By proposing new workflows and abstractions, these papers contribute to improving the efficiency and effectiveness of security configuration processes.

While both papers make valuable contributions to the field by addressing challenges in security configuration for virtualized networks, they do not directly align with the specific objectives of this thesis.

2.4.2 Application to VPN

The growing complexity and dynamism of virtualized networks, particularly in the context of Software-Defined Networking (SDN) and Network Functions Virtualization (NFV), have introduced new challenges in ensuring data confidentiality, integrity, and authentication. The configuration of secure channels, such as Virtual Private Networks (VPNs), over unreliable network infrastructures has traditionally been performed manually. However, manual operations are becoming unfeasible and error-prone, leading to vulnerabilities and cyber attacks. This section presents a summary and analysis of the paper authored by Bringhenti et al. [18].

The paper proposes an innovative methodology for automating the allocation and configuration of channel protection systems (CPSs) in virtualized networks. The authors emphasize the need for a trade-off between flexibility and complexity by automating the configuration process through policy refinement. They introduce the concept of Channel Protection Policies (CPPs) that describe the security requirements for channel protection.

The paper highlights the limitations of manual configuration in virtualized environments, including the difficulty of choosing the appropriate technology and protocol for channel protection and the challenges posed by the dynamic nature of virtual networks. The increased complexity and size of virtual networks make manual operations error-prone and time-consuming.

To address these challenges, the authors propose an automated methodology based on a MaxSMT (Maximum Satisfiability Modulo Theories) problem formulation. The methodology aims to combine automation, formal verification, and optimality in a single technique. It involves refining the security requirements expressed in CPPs into configuration rules for each CPS, ensuring correctness and minimizing the number of allocated functions and configured rules.

The main contribution of this paper lies in its novel combination of automation, formal verification, and optimality criteria within a single technique for channel protection in virtualized networks. By addressing the challenges specific to virtualized environments, the proposed methodology offers a promising solution to automate the allocation and configuration of CPSs. Furthermore, this research direction represents a significant advancement compared to existing techniques, which either lack formal verification and optimality criteria or are not explicitly designed for virtualized networks.

In conclusion, this paper presents a preliminary study on the automated configuration of Channel Protection Systems in virtualized networks. The proposed methodology offers a promising solution to overcome the challenges associated with manual configuration, enabling improved security and efficiency in dynamic network environments. Future research can focus on further refining and validating the proposed technique, exploring additional optimization objectives, and extending the methodology to cover other security properties and function types in virtualized networks. Moreover, this paper serves as an introductory example for the present thesis work, highlighting the issues related to manual configuration in virtual networks. It demonstrates how the automated configuration of a security function can be addressed, setting the stage for the thesis's primary focus on the automatic allocation and configuration of Parental Controls.

2.4.3 Application to firewalls

Bringhenti et al. [19] addressed the challenges of manual configuring security functions in computer networks, which may result in security breaches and lengthy re-configuration periods. The study focuses on the allocation and configuration of packet filter firewalls in the logical topology of a virtual network, particularly in the context of network virtualization.

Traditionally, the configuration of network security functions has been performed manually, which is not only time-consuming but also prone to human errors and misconfigurations. With the advent of network virtualization and software-defined networking, the complexity and dynamics of modern networks make manual configuration impractical. Therefore, automation is necessary to ensure correct and efficient configuration of security functions. By leveraging advanced solvers, the proposed method addresses this challenge by formulating and solving a well-designed partial weighted Maximum Satisfiability Modulo Theories (MaxSMT) problem.

A notable advantage of this methodology is its ability to ensure solution correctness by satisfying all security requirements while minimizing the number of required firewalls and firewall rules. The authors conducted comprehensive evaluations of their methodology, employing various metrics and tests on both synthetic and real-world use cases. These evaluations consistently demonstrated the superiority of the proposed approach over existing state-of-the-art solutions.

Another paper worth to be mentioned is the one authored by Bringhenti et al. [20]. The rise of cyber threats in modern networking environments, driven by virtualization paradigms and automation, necessitates effective reaction mechanisms. This paper presents cutting-edge framework that utilizes software-defined infrastructures and the programmability of cloud to enhance the efficiency and effectiveness of security mechanisms. The proposed framework eliminates the need for manual reaction to events and security task definition by developers of cybersecurity appliances. Moreover, it automates firewall ruleset generation, reducing human intervention, minimizing execution time, and mitigating the likelihood of errors. The focus of this paper is on the technical challenges associated with defining common actions at the policy level and translating them into configurations for diverse security functions within a Kubernetes context.

In conclusion, both papers provide valuable insights into the field of network security and introduce novel approaches and techniques that contribute to the advancement of virtual firewall management. While these papers focus on the advantages of reducing human intervention in security function configuration and employ advanced solvers to solve MaxSMT problems, it is important to note that this thesis work addresses a different security function, Parental Controls, while sharing the same approach of solving a distinct problem.

2.4.4 Application to IoT

Bringhenti et al. [21] proposed a novel methodology to address the security challenges in Software-Defined Networking (SDN)-aware Internet of Things (IoT) networks. The authors argued that the manual configuration of security measures in IoT networks is impractical and error-prone due to the increasing complexity and heterogeneity of these systems. To address this issue, they proposed an automated approach that utilizes Maximum Satisfiability Modulo Theories (MaxSMT) to calculate an allocation scheme and configuration of SDN switches that is both formally correct and optimized.

The proposed approach aims to dynamically change the security configuration of SDN switches based on user-defined security policies or in response to detected attacks. The automation is achieved by formulating the configuration problem as a MaxSMT problem, which provides a cost-effective formal approach and optimization. The authors extended the formal models used in previous research to accommodate the complexity of virtualized IoT networks and defined new optimization goals tailored to IoT networks' specific needs.

The paper introduces a framework that implements the proposed approach and interacts with the SDN orchestrator and monitoring agents. The framework has been validated in a realistic use case, demonstrating its effectiveness and scalability in terms of automatic configuration and SDN enforcement.

Overall, this paper presents a valuable contribution to the field of IoT network security management. By combining SDN, MaxSMT, and automation techniques, it provides a formal and optimized approach for allocating and configuring SDN switches in IoT networks. The proposed methodology addresses the challenges of security configuration in IoT deployments, enhances the correctness assurance of security management operations, and enables self-healing and self-protection capabilities.

While this paper addresses the same context as the thesis work (IoT networks), its primary focus lies in the automatic allocation and configuration of virtual SDN switches. In contrast, the thesis work centers specifically on Parental Control Systems. Therefore, although both works contribute to the field of IoT network security, they differ in their specific objectives and do not align directly with each other.

2.4.5 Smart Home/Smart Systems

Valenza et al. [22] introduced a novel threat model for cyber-physical systems (CPS) that incorporates the cyber, physical, and human aspects of these systems. The authors recognized the challenges in identifying potential threats and selecting adequate protection measures in CPS due to the complex interplay between human, physical, and cyber components. Existing threat models often focus on only one or two aspects and fail to capture the vulnerabilities and interactions between all three.

To address this gap, the paper proposes a hybrid threat model that considers the relations and security properties of CPS components while accounting for their cyber, physical, and human characteristics. This comprehensive model enables a more holistic analysis of the security state of CPS components and facilitates the understanding of vulnerabilities and potential attacks. The model takes into account multi-step attacks, referred to as hybrid attacks, which exploit vulnerabilities in different aspects of the system.

Furthermore, the paper presents a threat analysis method based on the proposed threat model. This method employs derivation rules to gather facts about attack graphs and evaluates the security properties of system components. The authors have developed an automatic tool called TAMELESS that implements the threat model and analysis method. The tool analyzes threats, verifies security properties, and generates graphical representations to assist security architects in identifying suitable prevention and mitigation solutions.

The paper demonstrates the applicability of the threat model and analysis method through case studies in different sectors, including smart buildings and wind farm scenarios. The results highlight the effectiveness of the approach in identifying vulnerabilities and assessing the security state of CPS components. The automatic generation of attack graphs provides valuable insights for security analysts and system administrators.

Overall, this work contributes to the advancement of threat modeling for CPS by considering the interdependencies and interactions between cyber, physical, and human aspects. The proposed hybrid threat model and analysis method offer a comprehensive approach to understanding and addressing security challenges in CPS.

However, it is important to note that while this paper offers valuable insights by identifying potential threats and selecting appropriate protection measures for CPS, its focus does not directly align with the specific objectives of this thesis.

Bringhenti et al. [23] presented a user-friendly approach for automating the configuration of home security solutions to enhance security usability and minimize human interventions. The authors addressed the increasing need for security personalization in smart homes, highlighting the limitations of current approaches in meeting user-centered security requirements.

The authors emphasized the growing risks and threats faced by individuals in domestic environments, particularly in terms of privacy and well-being. They cited recent investigations, such as the Data Breach Investigations Report by Verizon, which identifies social engineering as the most common method for breaching networks, with unintentional human factors contributing to 8% of all breaches. The authors attributed these alarming statistics to the rising use of complex Internet-connected devices by inexperienced users, as well as the heterogeneous nature of smart homes.

While various off-the-shelf security solutions, such as home gateways and routers, are readily available in the market, the authors argued that their adoption by end users is limited. Despite offering comprehensive security functionality, these solutions often lack user personalization options and require technical expertise to configure effectively. This discrepancy between the complexity of security concepts and the usability of available solutions poses a challenge for ensuring user-centered security in smart homes.

To address this challenge, the authors proposed leveraging policy-based management (PBM), a well-established paradigm used in business networks, for security enforcement and personalization in smart homes. They presented a three-pillar workflow for automatic cybersecurity personalization: a user security language, automated generation of configuration, and policy harmonization.

The user security language aims to bridge the gap between nontechnical users and security configurations by providing a simple and intuitive language for expressing security requirements. The authors introduced a high-level security policy language

(HSPL) that allows users to define sophisticated policies using natural language-like statements. The HSPL captures user security intents and serves as a link between human users and automated processes for security configuration.

The automated generation of configuration transforms user-defined policies into a structured representation that contains all the necessary information for configuring security products and services. This process identifies the required security functionality and generates the corresponding configurations, addressing issues of policy conciseness and heterogeneity of security solutions. External knowledge bases are accessed to retrieve relevant information for determining the security configuration, ensuring the accuracy and effectiveness of the automated process.

Policy harmonization focuses on identifying and resolving inconsistencies among the derived policies to prevent incorrect security behavior. As multiple users with different needs coexist in smart homes, the approach aims to support multiple security levels without interfering with one another.

Overall, the paper advocates for a user-centered design approach to security in smart homes, emphasizing the importance of usability, personalization, and user understanding of security principles. The proposed PBM-based approach offers a promising solution to address the complexity of security configuration and customization, providing users with a simplified and intuitive method to enhance security in their smart homes. This is particularly important as it empowers users with the language that has been used to define the network policies within the scope of this thesis work.

Chapter 3

Threat Analysis and Mitigation Strategies in Smart Home Environments

In the rapidly evolving landscape of smart home environments, the protection of users' personal data and devices has become a paramount concern. This is particularly important considering that home networks consisting of IoT devices can become significantly complex and are typically managed by users with limited networking and security expertise. To ensure that these users can utilize and, in some cases, rely on their IoT devices, it is essential to proactively identify and mitigate potential risks and vulnerabilities. This entails developing techniques that can automate the configuration and enhance the security of these networks.

This chapter aims to explore the various scenarios within smart home environments in order to identify potential threats that could compromise the security and privacy of its occupants. Furthermore, it will specifically concentrate on a pressing issue that requires urgent attention: the automatic allocation and configuration of parental controls.

As smart homes become more pervasive, parents face the challenge of safeguarding their children from inappropriate content and online dangers. By automating the allocation and configuration of parental control settings, it is possible to offer a valuable solution that simplifies the process and guarantees consistent protection across multiple devices and platforms.

To address this challenge, it has been adapted and extended the Verefoo framework. Verefoo is a framework for the automatic generation of various aspects of the configuration in an enterprise network, supporting the placement and configuration of virtual appliances with firewall and VPN functionality. However, home networks present very different challenges compared to enterprise networks. For this reason, the primary objective of this project is to characterize the critical aspects of configuring a home network for its security and to adapt Verefoo to support these aspects.

By the end of this chapter, readers will gain a comprehensive understanding of the risks inherent in smart home environments, the specific problem of parental control automatic allocation and configuration, and the extended Verefoo framework as a solution to these challenges. The objective of this exploration is to contribute towards the development of secure and privacy-conscious smart home systems that ensure the well-being of their residents.

3.1 Scenarios

In this section, a series of scenarios are going to be presented to exemplify the potential threats and vulnerabilities that can arise within a smart home environment. These scenarios aim to provide a comprehensive understanding of the risks involved and highlight the need for robust security measures in this rapidly evolving landscape.

To facilitate a better understanding of the scenarios, the smart home environment depicted in Figure 3.1 is introduced.

The depicted environment shows a range of IoT devices, including the Dyson Air Purifier, Atomi Smart Coffee Maker, Philips Hue Light Bulb, Amazon Doorbell, Samsung Smart Refrigerator, Arlo Pro 4 Security Camera, Samsung Smart TV, Nest x Yale Smart Lock, Nest Thermostat, Alexa Echo Dot, Eero Mesh System, as well as several smartphones, a tablet, and a laptop. Four individuals interact with these devices through a smartphone/tablet application: John and Ellen, the owners of the IoT devices, James, their child, and Liam, a temporary resident in the house.

Figure 3.2 provides a more detailed configuration of the devices within the home network, highlighting their connections and arrangement across three rooms: the Entrance, the Kitchen, and James's room.

In the Entrance, the Amazon doorbell and the Arlo Pro 4 Security Camera are found, which can communicate with the tablet, smartphones, and Echo Dot via Wi-Fi. Additionally, the Nest x Yale Smart Lock utilizes OpenWeave to connect to a Nest Connect, enabling it to establish a Wi-Fi connection with the vendor application installed on the tablet and smartphones. Moving to James's room, the Philips Hue Light Bulb is encountered, capable of connecting directly to the smartphones via Bluetooth. While it could potentially connect to the Echo Dot within Bluetooth range, it is assumed that such a connection does not exist in this context. In addition, the Light Bulb can be connected to a Hue Bridge via Zigbee,

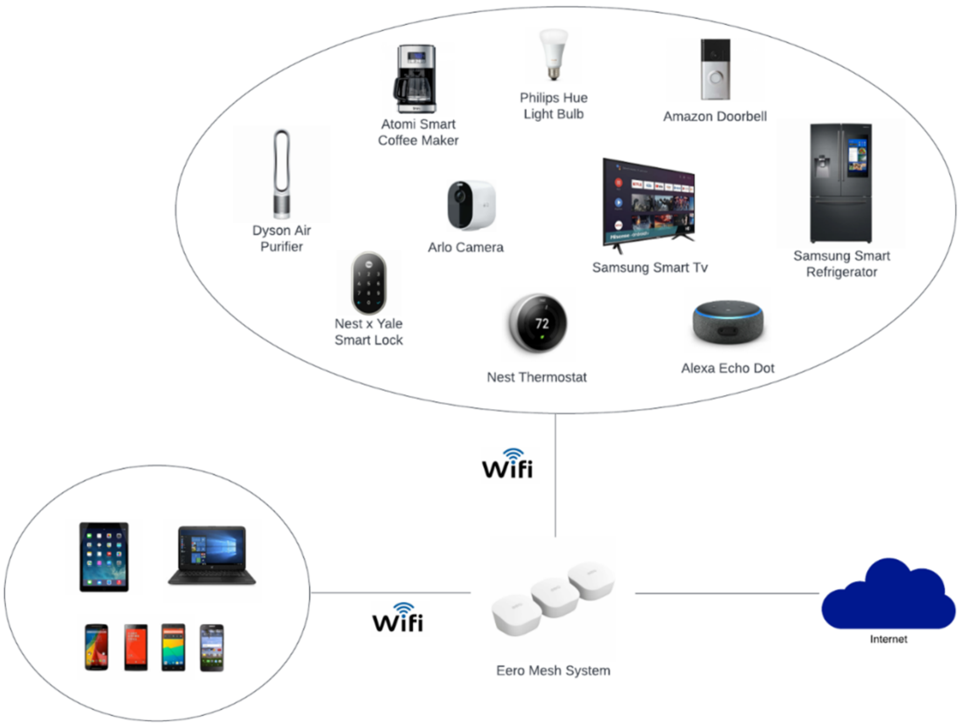

Figure 3.1: Smart home environment with multiple user that can interact with different IoT devices by means of smartphones/tablets

enabling control of the Smart Light from anywhere through a Wi-Fi connection. However, this specific scenario is not considered in the current context. The Dyson Air Purifier in James's room communicates with the smartphones through Wi-Fi. In the Kitchen, five IoT devices contribute to the smart home ecosystem: the Nest Thermostat and the Echo Dot, which establish communication with the vendor application installed on smartphones using Wi-Fi connection and Bluetooth, the Samsung Smart TV, Samsung Smart Refrigerator and Atomi Smart Coffee Maker, that communicate exclusively through Wi-Fi connection with the vendor application installed on smartphones.

Within this intricate smart home environment, four distinct scenarios are explored, each highlighting specific security concerns and the need for effective mitigation strategies. These scenarios encompass a wide range of potential risks, offering valuable insights into the challenges faced by smart home owners and users.

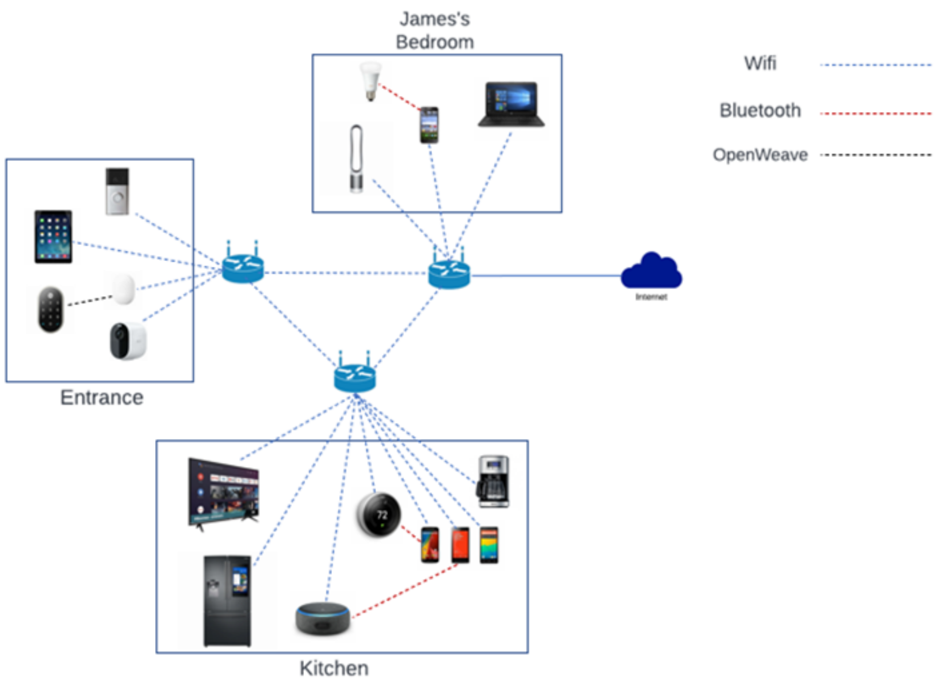

Figure 3.2: Overview of the different connections among the devices inside the smart home environment

3.1.1 Scenario 1: Privacy and Security Problems caused by an insider attacker

In the house, there is a guest room available, which John offers for vacation rentals to Airbnb users. A potential guest, Liam, expresses interest in renting it for a couple of months. Liam would find it convenient to have access to certain IoT devices within the home, such as the thermostat or the doorbell.

However, granting Liam access to these IoT devices by creating separate accounts for him on each device poses potential security and operational risks to the smart home environment. If Liam were to have malicious intentions, he could manipulate system settings without authorization, add unauthorized users to the system, or even remove John as a system user, gaining full control over the smart home environment. Furthermore, once Liam's stay is over, his access to the devices should be revoked.

Traditional security mechanisms, which primarily focus on preventing external attacks, may not detect these internal risks. To mitigate the threats associated with insider attacks, it is necessary to restrict Liam's functionality, providing him with limited access to the devices that allows legitimate usage.

One potential approach to accomplish this goal is to implement an access control framework that provides user-level control. This can be achieved either through manual configuration, which requires extending the functionality of the vendor application to define user roles and their associated permissions within the IoT environment, if possible, or through automatic provisioning that grants users only basic functionality. In the case of automatic provisioning, the device's MAC address is utilized to identify the user's traffic, while machine learning techniques are employed to classify operations and block non-basic functionality.

A similar solution to this access control framework is exemplified by Kratos+ [8], a sophisticated system that supports multiple users and devices. Kratos+ utilizes a priority-based policy negotiation technique to intelligently address conflicting user demands, ensuring efficient management of access control. While Kratos+ encompasses access control in a broader sense, the focus of this thesis work is on a specific aspect of access control: Parental Control. Parental control involves regulating and managing the access rights and privileges of children within digital environments. The primary focus of this thesis is to develop an automated method for allocating and configuring parental controls within a home network.

3.1.2 Scenario 2: Challenges in Ensuring Digital Safety with Limited Parental Controls Customization

James is used to spending a lot of time watching television, which has become a concern for his parents, John and Ellen. They want to limit the amount of time James spends using the Smart TV, ensuring a healthy balance between screen time and other activities. Specifically, they don't want James to exceed 2 hours of TV per day, except on weekends when he can have up to 3 hours.

In addition to the Smart TV, James also has a laptop that his parents gave him as a birthday present. He frequently uses the laptop for playing games, which has become a distraction affecting his academic performance. Recognizing this issue, John and Ellen want to implement a solution that reduces the time James spends on the laptop by setting a daily time limit.

Furthermore, John and Ellen are concerned about James being exposed to inappropriate content on the internet. They want to protect him from accessing adult content that is not appropriate for his age. Creating a secure online environment is their top priority.

Lastly, Ellen wants to restrict James from using the Smart Coffee Maker, but unfortunately, the current application does not provide a way to control or limit its usage.

To address these specific challenges, an automatic allocation and configuration system for parental controls would be highly beneficial. Such a system could monitor and regulate James's screen time on the Smart TV, enforcing the desired limit of 2 hours per day (except on weekends). It could also set a daily time limit for laptop usage, helping James strike a better balance between schoolwork and recreational activities. Moreover, the automatic allocation and configuration system could provide content filtering capabilities, blocking access to adult content and ensuring James's online safety. Additionally, if integrated with the Smart Coffee Maker, the system could provide a way for Ellen to control and restrict James's access to it, addressing her concern about his usage of that particular device.

By implementing this automated system, John and Ellen can effectively manage James's screen time, prioritize his educational commitments, safeguard him from inappropriate content, and address their concerns regarding the Smart Coffee Maker. The system would offer a user-friendly interface or security language that enables John and Ellen to define their desired parental control settings easily. This automation would relieve the burden of manually configuring and enforcing these restrictions, ensuring consistent and reliable application.

3.1.3 Scenario 3: Exploiting vulnerabilities for attacking the system

John, unaware of the security risks associated with not changing default passwords on IoT devices, neglects to update them after purchase. This allows an attacker, Oscar, to exploit these unchanged passwords easily and successfully incorporate John's devices into a botnet for launching a Denial of Service (DoS) attack (1).

Furthermore, one of the IoT devices in John's network has known vulnerabilities for which a patch has been released. However, since John is unaware of the importance of updating his devices, the vulnerable device remains unpatched, enabling Oscar to exploit it and launch attacks on other devices within the network. These attacks may include distributing malware to other devices, gaining unauthorized control over them, stealing sensitive information, or even incorporating them into a botnet (2).

Another consequence of neglecting default password changes and device updates is the potential for data exfiltration or injection of tampered information, resulting in privacy and security issues (3).

To address problem (1), a possible solution is to conduct a password inspection across the network, identifying devices still using default passwords, for example, by comparing them with the default credentials found in the Mirai botnet source code [24]. Once identified, the passwords can be randomly generated and updated, with the user promptly notified of the new credentials.

For problem (2), performing a vulnerability assessment on the network can help isolate vulnerable devices and notify users about necessary actions to mitigate risks. This proactive approach enables users like John to update their devices promptly and protect against potential attacks.

Furthermore, to mitigate the risks of data exfiltration and tampered information

(3), deploying an intrusion detection and prevention system within the network is recommended. This system would monitor network traffic, analyze patterns, and raise alerts in case of suspicious activities. It would also actively block malicious traffic to prevent potential privacy and security breaches.

Another effective measure is implementing network segmentation. By separating IoT devices into different network segments, any compromise or unauthorized access to one device would not automatically grant access to the entire network. This approach restricts an attacker's ability to exfiltrate data or tamper with information on other devices even if one device is compromised.

In conclusion, the context of smart home networks represents a distinct and novel environment compared to the traditional enterprise networks addressed by Verefoo. This scenario is significant as it contributes to the expansion of knowledge regarding smart home networks and sheds light on the existing challenges that can impact such networks, along with the available solutions. However, it is important to note that this particular scenario falls outside the scope of this thesis work and will not be further explored here.

3.1.4 Scenario 4: Possible leak of sensitive information

Despite Ellen's appreciation for the functionality provided by her IoT devices, she has concerns about her family's privacy. Specifically, she desires to minimize the amount of sensitive information being exchanged over the internet by devices such as the Arlo Camera and the Echo Dot, which are equipped with microphones and camera sensors (1). Ellen aims to achieve this objective without compromising the devices' functionality.

The presence of non-essential traffic in the network raises various concerns, including the potential for unauthorized viewing of camera sensor streams, detection of people's presence inside the home through analysis of information transmitted (e.g., light bulb on/off), or analysis of content watched on the Smart TV for advertising purposes. This can occur due to normal traffic or the presence of an attacker attempting to intercept the network traffic.

To address the issue of non-essential data transfer (1), one approach is to analyze the network traffic and identify which communications are vital for the devices to perform their intended functions. By implementing a firewall, unnecessary communication can be blocked without disrupting the devices' functionality [25]. This method effectively reduces the risk of eavesdropping attacks while preserving the devices' core capabilities.

Overall, similar to the previous scenario, this case holds significance as it contributes to the understanding of smart home networks and highlights the important issue of privacy within such networks, along with potential solutions. However, it is worth noting that this particular scenario is beyond the scope of this thesis work and will not be further explored in this context.

3.2 Parental Controls Configuration

The attention will now be shifted towards the second scenario previously presented, emphasizing the challenges associated with configuring parental controls within a smart home environment. The objective is to delve into the different approaches used for setting up parental controls. This focus is crucial as it addresses the methods employed to customize and manage parental controls, considering the growing importance of protecting individuals, particularly children, from online threats and inappropriate content.

With the ongoing proliferation of smart devices and the Internet of Things (IoT), children are increasingly vulnerable to cyberbullying, exposure to unsuitable content, and various other online risks. Effectively safeguarding children in this digital era necessitates the implementation of parental control measures that can regulate their online activities and mitigate potential harms.

3.2.1 Policy-based Parental Control Configuration

Security is becoming an essential consideration in the configuration of network functions in distributed systems. However, without employing a policy-based system for configuring security functions, such as Parental Control or other services, the task is solely reliant on the security administrator. Consequently, the likelihood of errors increases due to the complexity of considering multiple aspects simultaneously, including efficiency, performance, and security. By adopting a policy-based approach, abstract rules can be defined to describe system behavior, enabling security administrators or even users without specific expertise to verify that the network functions configuration meets the necessary security requirements.

A policy rule involves two primary actors: the subject, which is the entity authorized to request the fulfillment of a set of constraints, and the target, which includes the elements affected by the policy, such as service functions or network devices that require allocation and configuration.

Furthermore, a policy rule establishes a connection between a set of actions and a set of conditions:

- A policy condition represents the required state and prerequisites that determine whether the actions defined in a policy rule should be performed. It is considered a match only when all attribute values evaluate as true.

- A policy action formalizes the operations that need to be executed to enforce a policy rule once its conditions are satisfied.

The specification of policies is a critical task in network security. If policies are incorrectly specified or in conflict with each other, the resulting configuration can be vulnerable to security attacks or may block legitimate traffic. Therefore, it is essential to define a high-level language that users can utilize to express network policies.

In this thesis, the approach presented Valenza and Lioy [26] was followed, which introduces two network security policy languages: the High Security Policy Language (HSPL) for non-technical users and the Medium Security Policy Language (MSPL) for technical individuals. These languages aim to provide an abstraction layer from the low-level configurations of network functions.

HSPL is designed for high-level policy specification, adopting a subject-verb-object paradigm. Each expression in HSPL consists of a subject (representing the entity enforcing the network security requirement), a verb (indicating the action to be performed), an object (the target of the action). HSPL focuses on simplicity, allowing end users without specialized knowledge to understand and formulate policies. Flexibility and extensibility are also key aspects of HSPL, enabling the introduction of new policies and support for various conditions and actions.

On the other hand, MSPL is a language targeting technical personnel, such as security managers. It requires extensibility and flexibility, but its main goal is to provide all the necessary information for the subsequent configuration of network functions. MSPL rules should be expressed in an abstract form, independent of specific implementations, to ensure versatility and avoid being tied to particular systems or technologies.

The availability of the High Security Policy Language and the Medium Security Policy Language enables the creation of security policies for both individuals with technical expertise and users without specialized knowledge. However, when expressing requirements using the high-level policy language, an additional refinement step is necessary to derive the corresponding set of policies in the medium-level language. These medium-level policies are then translated into low-level policies, which are tailored to the specific implementation of the network functions used for the service. Since MSPL expressions already encompass all the essential information, the translation phase is relatively simple, involving mainly a syntactical transformation. Figure 3.3 depicts the 2 processes.

3.2.2 Automatic Parental Control Configuration

The process of manually configuring parental controls can be overwhelming for users, particularly those who lack experience or familiarity with the technical aspects of security. Commercial security solutions often offer preconfigured parental control features with limited options for customization, which may not align with the specific needs and preferences of individual users or families.

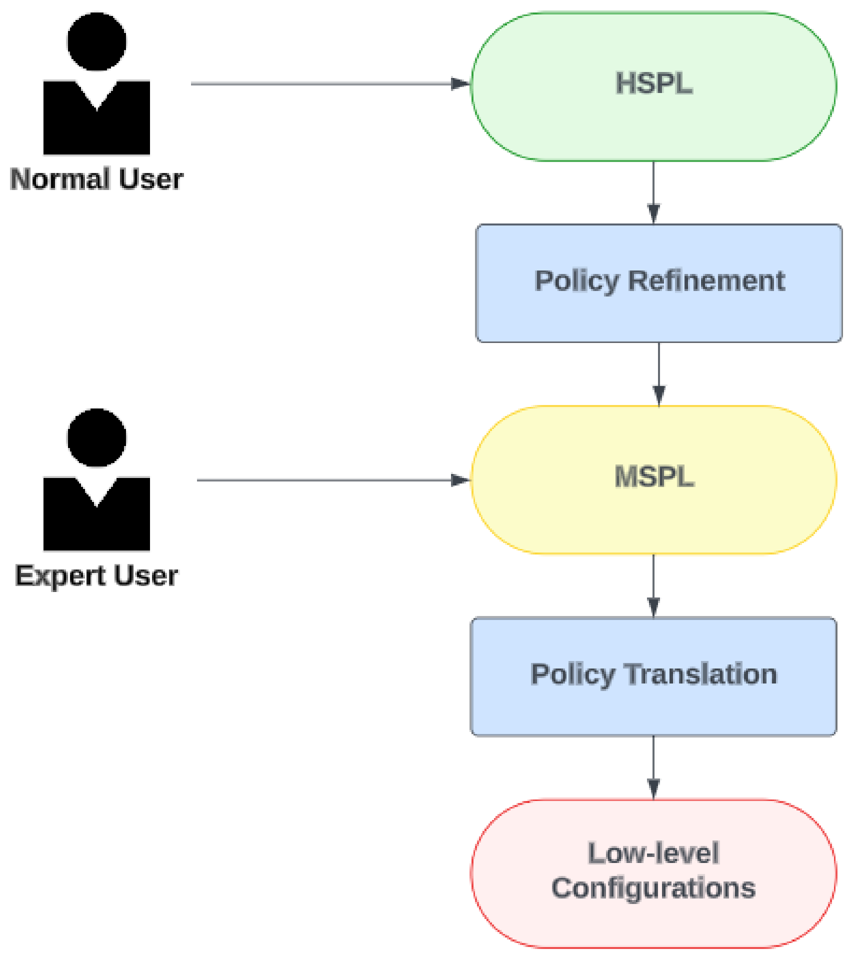

Figure 3.3: Policy Refinement and Translation Model

To address these challenges, an automatic allocation and configuration system for parental controls becomes necessary. Such a system would enable users to define their desired security requirements and preferences through a user-friendly interface or security language. Subsequently, the system would automatically generate and apply the appropriate settings, minimizing the need for human intervention.

By implementing an automated approach to parental controls, users can benefit from enhanced usability and customization. The system simplifies the process of setting up and maintaining parental control settings, ensuring consistent and reliable enforcement of security measures. This automation overcomes the challenge of users not utilizing available security features due to complexity or lack of understanding.

Additionally, an automated system can adapt to changing needs and evolving online risks. As new threats emerge or individual circumstances change, the system can dynamically adjust the security settings to maintain optimal protection. This flexibility and adaptability are crucial in ensuring the ongoing safety and well-being of children in today's digital landscape.

In this research, the proposal aims to extend the Verefoo framework to achieve the goal of automating the allocation and configuration of parental control settings. The Verefoo framework, originally developed at Politecnico di Torino, provides a comprehensive solution to allocate and configure selected Network Security Functions (NSFs) in an automatic way. The goal is the satisfaction of a set of Network Security Requirements (NSRs) that can be expressed using an high-level language. Additionally, the framework ensures the placement of the virtual functions present in the Service Graph on the servers of a physical network.

By leveraging the capabilities of the Verefoo framework, the objective is to provide an effective solution to the challenges specific to parental control in smart home environments. The extended framework will enable the automated allocation and configuration of parental control settings, guaranteeing consistent protection across multiple devices and platforms. This approach simplifies the process for users, enhances customization options, and ensures the ongoing safety of children by adapting to evolving online threats.

Subsequent section will explore the concept of Software Networking, serving as an introduction to the Verefoo framework. Then, the workflow of the framework will be described along with its components, followed by a detailed examination of the proposed extension and its implementation in the next chapter. This progression will provide an effective solution to the specific challenges of parental control in smart home environments, ensuring automated allocation and configuration of settings, consistent protection across devices, enhanced customization options, and adaptation to evolving online threats.

3.3 Software Networking

In this section, the focus will be directed towards the concept of Service Function Chain (SFC), that consists of a chaining of network functions to fulfill specific service requirements. The SFC concept serves as the foundational theory for constructing a Service Graph, which can be considered as a generalization of a linear SFC. Once the operational aspects of the system have been explained, attention is shifted towards its limitations arising from the conventional method of deploying Service Function Chains (SFCs) on hardware appliances.

Consequently, two significant advancements in the field of computer networks that effectively address these constraints are introduced: Software Defined Networking (SDN) and Network Function Virtualization (NFV). SDN emphasizes the agility and ease of re-programmability through software-based packet forwarding, allowing dynamic path decisions within the SFC. NFV, on the other hand, leverages standard hardware to run network function images, enhancing the flexibility of SFCs.

Additionally, this section will emphasize the advantages of Network Automation, including increased flexibility and responsiveness. Specifically, in terms of configuring a Service Function Chain, Network Automation leverages events generated within the network itself, enabling dynamic and adaptive configuration.

Finally, the Verefoo framework, which implements the previously mentioned approach, will be introduced as a practical demonstration of the discussed concepts.

3.3.1 The Concept of Service Function Chain

Typically, to establish a comprehensive end-to-end service, a specific order of implementation and application of a series of functions is required for the traffic. This ensures that the desired outcome, as defined by the service designer or user, is achieved. The RFC 7665 [27] introduces the following important definitions:

- Service Function Chain (SFC): A service function chain defines an ordered set of abstract service functions and ordering constraints that must be applied to packets and/or frames and/or flows selected as a result of classification.
- Service Function (SF): A function that is responsible for specific treatment of received packets. A Service Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers). As a logical component, a service function can be realized as a virtual element or be embedded in a physical network element. One or more Service Functions can be embedded in the same network element. Multiple occurrences of the service function can exist in the same administrative domain.

In Figure 3.4, an illustration showcases the sequential arrangement of elementary components in an end-to-end service connecting a web client and a web server. These components play distinct roles in enhancing the overall security and functionality of the service.

The first component is a firewall, responsible for filtering out undesirable network traffic and ensuring that only legitimate data reaches the subsequent stages. Following the firewall is an Intrusion Detection System (IDS), which employs pattern detection or data analysis from a prior monitoring phase to identify potential attacks. Finally, the third component is a reverse proxy, which serves multiple purposes such as concealing the true characteristics of the server, web caching, and spoon-feeding.

The significance of these functions lies in their specific order of application to packets traveling from the client to the server. This precise sequence ensures the compensation of relative vulnerabilities and strengthens the overall security of the system.

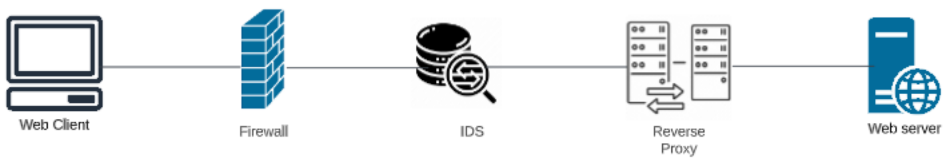

Figure 3.4: Example of Service Function Chain

Historically, network functions primarily existed as physical boxes dedicated to specific purposes. These boxes were designed and built to fulfill their intended functions. As a result, network function chains composed of such functions were subject to significant limitations.

- Limited Resource Sharing: The dedicated hardware nature of these physical network function boxes made it challenging to share resources between different functions. As a result, an underutilized function could not allocate its computational resources to another function that was experiencing a higher workload.
- Lack of Traffic Control: It was difficult to control and direct traffic through specific elements of the Service Function Chain. In traditional IP networks, traffic was simply forwarded based on network addresses, making it challenging to enforce a specific path through the chain.
- Limited Personalization: Customizing the Service Function Chain for individual users was not a straightforward process. The fixed links between the physical appliances made it challenging to adapt the chain's structure to meet the specific requirements of different users.
- Costly Modifications: Introducing a new function or modifying the structure of the Service Function Chain required purchasing dedicated hardware appliances.
- Maintenance and Management Overhead: The constant monitoring and maintenance of the physical appliances in the Service Function Chain imposed significant workload and responsibilities on the provider.

Recognizing these limitations, there has been a shift towards software-based approaches that aim to improve deployment flexibility and performance by virtualizing network functions and decoupling them from dedicated hardware.

3.3.2 Introduction to Software Defined Networks

Software-Defined Networks (SDN) offer a promising solution to address the limitations associated with traditional hardware implementations of Service Function Chains. SDN introduces a novel approach where the creation and management of packet paths within the physical network are performed through software processes.

The key principles of SDN can be summarized as follows:

1. Decoupling of Data Plane and Control Plane: The data plane is responsible for handling the forwarding and processing of network packets, while the control plane manages the network's behavior, policies, and routing decisions.

2. Centralization of Control Plane Functions: SDN centralizes all control plane functions in a single module known as the SDN Controller. This controller serves as the central intelligence of the SDN network. While the centralization can be logical with distributed collaboration among nodes, most current implementations opt for physical centralization.

3. Southbound and Northbound Interfaces: The southbound interface enables communication between the SDN controller and the forwarding infrastructure while the northbound interface empowers the SDN controller to engage with user-level applications or higher-level controllers.

Figure 3.5 illustrates a standard SDN architecture that aligns with the principles discussed earlier.

At the lowest level, the network infrastructure comprises data forwarding elements, which no longer need to be complex routers or vendor-specific devices. Instead, they can be white-label switches, simple devices dedicated solely to forwarding incoming packets to the appropriate output ports, efficiently ensuring their delivery to their intended destinations. Each switch is characterized by its forwarding table, which contains rules categorized into match fields and action fields. Match fields identify subsets of packets, while action fields specify operations to be performed on those packets, such as forwarding to a specific port, forwarding to the controller platform, forwarding at a specific rate, dropping the packet, pushing or popping a field, or modifying a header field.

The separation of the data plane and control plane has significantly simplified the complexity of network devices. As a result, the control plane, which includes tasks like forwarding table computation, has been centralized in a controller. To facilitate

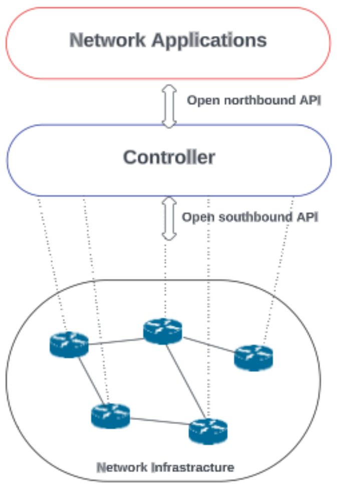

Figure 3.5: SDN Architecture

interaction between the controller and the network infrastructure, various network protocols are utilized through an open southbound interface. Initially, the OpenFlow protocol was employed, but its widespread adoption was limited as it required specialized network devices called OpenFlow switches, necessitating the replacement of existing devices. OpenFlow enabled operations such as data collection for monitoring and statistical purposes, as well as the injection of computed forwarding rules into network devices. However, to overcome this limitation, other protocols such as Network Configuration Protocol (NETCONF), Simple Network Management Protocol (SNMP) and RESTCONF have been utilized. These protocols were already supported by the existing network architecture and offered alternative solutions.

Furthermore, the interaction between the controller and various software modules is facilitated through an accessible northbound interface, providing a range of functionality. This collection of applications allows the addition of new functionality without the constant expansion of the SDN controller's operating system. It should be emphasized that a failure of an application within the controller can have a significant impact on the overall functionality of the controller, highlighting the need for robustness and stability in the upper layer.

3.3.3 Applying SDN Technology to Service Function Chain

Assuming that each hardware appliance corresponds to a service function in a Service Function Chain, these appliances can be interconnected using an SDN switch network, where the controller platform governs the forwarding behavior completely.

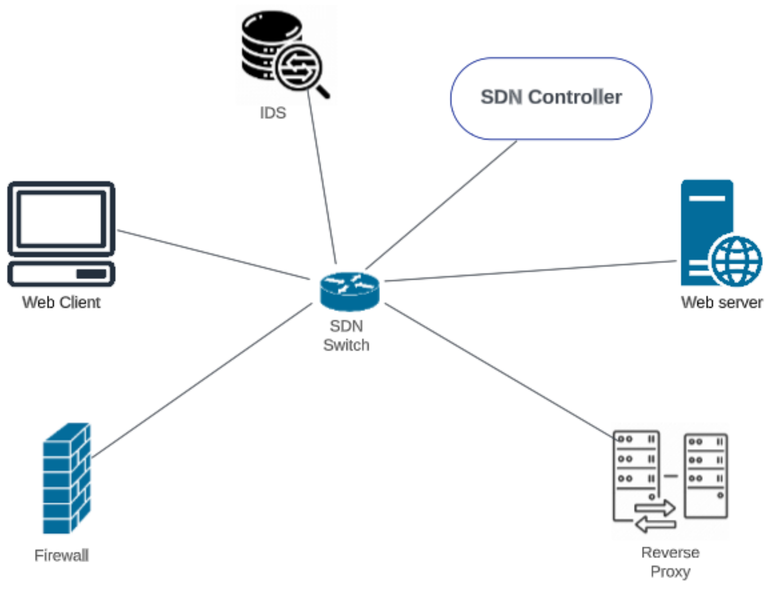

Figure 3.6: Example of SDN applied to a Service Function Chain

Figure 3.6 demonstrates how the SDN architecture can model the Service Function Chain depicted in Figure 3.4. In this example, the firewall, IDS, and reverse proxy are connected to the switch through a single connection. The controller platform determines the packet flow and the sequence in which the middleboxes are traversed.

This solution offers several notable advantages:

- The controller can proactively install forwarding rules on the SDN switches, or reactively in response to network events, making it straightforward to direct IP traffic through specific appliances.

- Flexibility and rapid deployment of new services. The traditional limitations imposed by physical wiring are eliminated, as the routing process is now carried out through software.

- Different Service Function Chains can be personalized according to users' requirements, allowing them to selectively utilize specific network functions.

However, one limitation that remains unaddressed in an SDN-based infrastructure is that the appliances themselves are still hardware-based. Consequently, sharing computational resources between functions is not feasible, and the procurement and installation of new functions are not swift operations.

3.3.4 Introduction to Network Function Virtualization

Network Function Virtualization (NFV) refers to the ability to execute various network functions on standard hardware, leveraging computing virtualization techniques to optimize resource utilization. With NFV, these functions can be performed on a general-purpose server, eliminating the need for specialized hardware. While Software-Defined Networking (SDN) concentrates on software-based path creation, NFV specifically aims to virtualize computing resources, enabling efficient and flexible deployment of network functions.

An essential aspect of NFV is its ability to automatically scale Virtualized Network Functions (VNFs) when additional resources are required. Two approaches can be adopted in this context:

- Scale up involves vertically adding more resources, such as CPU, memory, or storage, to a single instance of a VNF. Although this approach provides advantages, such as increased computational capabilities, it is limited by the maximum capacity of the server hosting the VNF. Furthermore, the benefits gained from scaling up may not always be optimal for certain implementations, such as mono-threaded applications.
- Scale out involves horizontally adding more instances of a VNF to distribute the workload across multiple instances. Each VNF instance operates independently and handles a portion of the overall traffic or workload. Scale out allows for increased scalability and redundancy, as the workload can be efficiently distributed among multiple instances, providing better performance and fault tolerance. However, it may require a load balancer to determine the appropriate instance responsible for handling a particular traffic flow.

The choice between scale up and scale out depends on the specific requirements of the network and the VNFs being used. Both approaches offer advantages and trade-offs in terms of resource utilization, performance, and cost-effectiveness.

3.3.5 Applying NFV Technology to Service Function Chain

The introduction of Network Functions Virtualization (NFV) technology revolutionizes the traditional Service Function Chain (SFC) model by replacing dedicated hardware boxes with software-based functions installed on servers. This shift allows for a more flexible representation of the SFC, as depicted in Figure 3.7, where functions are now virtual machines that can potentially reside on the same server. Complementing this approach is the utilization of Software-Defined Networking (SDN) as a tool to control packet flow between SFC elements.

By implementing network functions using the NFV approach, several advantages can be realized:

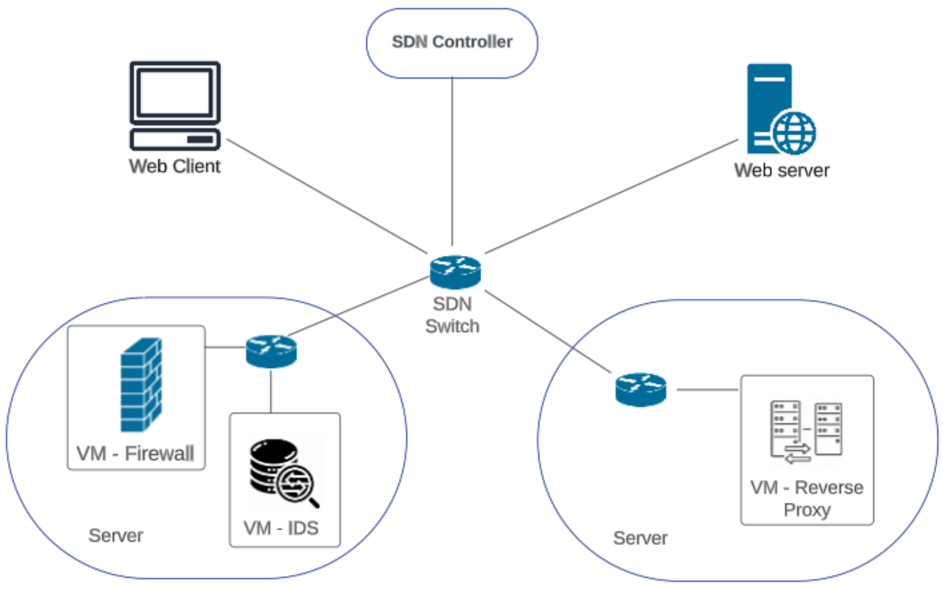

Figure 3.7: Example of a Service Function Chain Implemented with NFV and SDN technologies

- Efficient utilization of hardware resources: Multiple service functions can be deployed on a single server, enabling the sharing of physical resources and optimizing resource utilization.
- Cost-effective addition of new functions: Rather than purchasing expensive dedicated hardware, new functions can be easily installed on existing general-purpose servers, reducing costs and simplifying the deployment process.
- Flexible service partitioning: NFV allows the creation of different instances of Virtualized Network Functions (VNFs), enabling the partitioning of services among different tenants. Each tenant can have their own dedicated instance of a VNF, ensuring isolation and customization as per their specific requirements.

3.3.6 Network Automation

The progress in NFV and SDN technologies has facilitated the development of Network Automation, which involves leveraging software to streamline and automate network and security provisioning and management. This iterative process focuses on continuously enhancing network efficiency and functionality by minimizing manual intervention and introducing automated workflows. It emphasizes the importance of adaptable responses to configuration changes, driven by user needs or network events, while considering the current state and behavior of the underlying physical infrastructure.

Network Automation has significant implications in the realm of security, where rapid response times are crucial in addressing the ever-growing range of cybersecurity threats that cannot always be predicted in advance. In an automated context, security configurations can be adjusted promptly based on alerts from systems like Intrusion Detection Systems (IDS), reducing latency compared to manual intervention by a security manager.

The ultimate goal is to minimize human interaction, allowing software to autonomously close the loop of reactions by gathering network information and utilizing it to enhance overall performance. However, this automated process relies on well-configured algorithms to determine the necessary configuration changes triggered by network events. Therefore, it is imperative that the automated system possesses a comprehensive understanding of the entire infrastructure, leveraging sensors to monitor traffic and establish a baseline of normal behavior.

The key advantage of Network Automation lies in its ability to adapt to diverse scenarios without human intervention, provided that the aforementioned challenge of maintaining a holistic view of the infrastructure is effectively addressed. By enabling self-adaptation, Network Automation empowers the system to continuously improve performance based on real-time insights, eliminating the need for human decision-making based solely on monitoring results.

3.4 Verefoo

To address the limitation often encountered in traditional Service Function Chains [15], the VEREFOO (VErified REfinement and Optimized Orchestration) framework can be used. By utilizing the Verefoo framework, it becomes possible to allocate Network Security Functions on a Service Graph in an optimal manner and automate their configuration. This approach aims to fulfill a predefined set of network security requirements, which can be expressed using a high-level security language. The formulated requirements then undergo a refinement process. Moreover, the framework simplifies the placement of these functions on the servers within the physical network. In a cloud environment, it leverages the z3Opt engine to address the Maximum Satisfiability Modulo Theories (MaxSMT) problem. Working in collaboration with an NFV orchestrator, it ensures the accurate deployment of the chosen Network Security Functions with the appropriate implementation.

3.4.1 Verefoo Workflow

Figure 3.8 highlights the key components of VEREFOO, illustrating the complete workflow of the framework. To utilize VEREFOO, users need to provide two inputs:

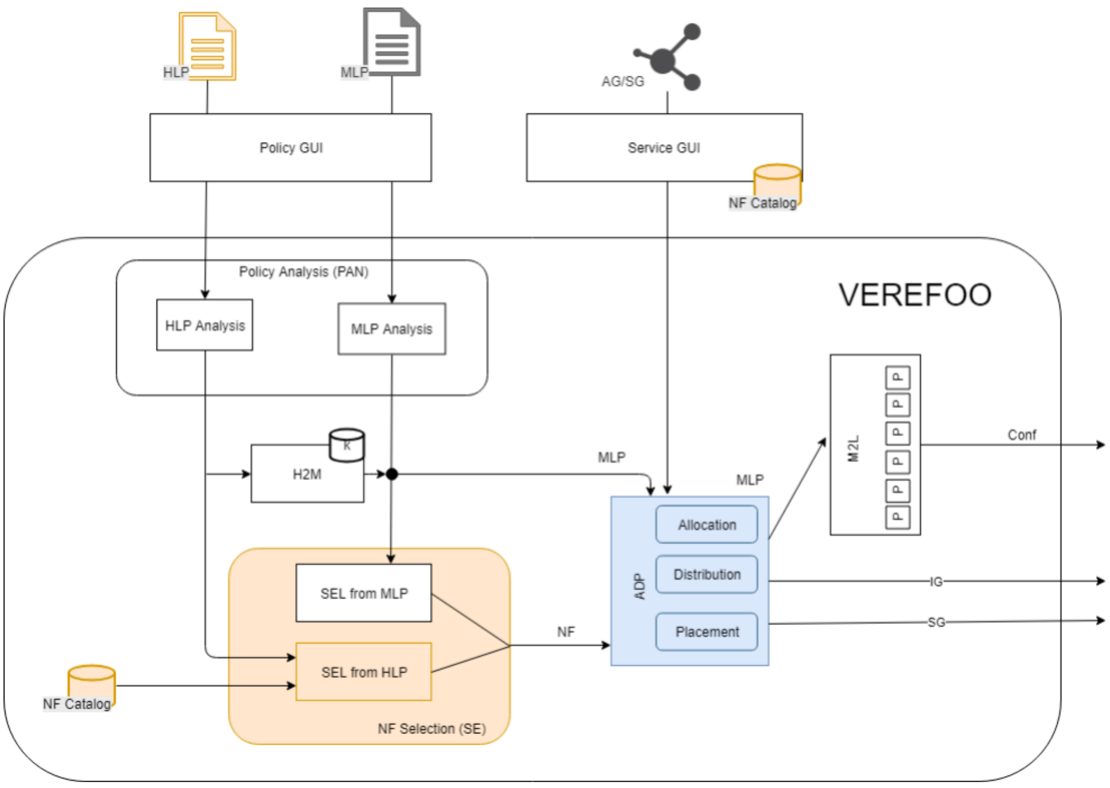

Figure 3.8: Verefoo Architecture

- Network Security Requirements (NSR): These represent the security constraints that must be met. Depending on the security administrator's expertise, they can be specified either using a high-level or medium-level language. A Policy Graphic User Interface simplifies the creation of the NSRs.

- Service Graph: This logical topology consists of network functions such as NAT, forwarder, and web cache, which form an end-to-end service. Multiple paths or loops can exist between endpoints. In VEREFOO, additional network security functions like Packet Filters or VPN Gateway can be incorporated into the allocation graph later to meet security requirements. Therefore, they are not initially present in the Service Graph.

  Instead of inserting as input a Service Graph, users can directly provide an Allocation Graph, which present the same network functions that can be found in the Service Graph, with the addition of some different nodes known as Allocation Places, that constitute the potential locations for allocating Network Security Functions. The choice if choosing a Service Graph or an Allocation Graph as input depends on if the security administrator already know these locations. Additionally, VEREFOO's graphic user interface provides access to a catalog where specific network security functions can be selected.

The framework's initial step involves conflict analysis of the input Network Security Requirements to detect conflicts and establish the minimal set of constraints that must be adhered to in the network. The Policy Analysis (PAN) module performs this analysis and generates a non-enforceability report if conflicts that cannot be resolved automatically are detected.

The High-to-Medium (H2M) module do the refinement in the case high-level language is used to formulate the security requirements in a way to derive a set of Network Security Requirements expressed using a medium level language. These medium-level requirements include all the information needed to create policies for the automatically allocated Network Security Functions on the graph and the low-level configuration of the Virtual Network Functions (VNFs) placed on the physical network.

The NF Selection is a Verefoo module that has the objective to fulfill the specified requirements by choosing the most suitable Network Security Functions from a predefined catalog. This selection process may incorporate optimization techniques to identify an optimized set of Network Security Functions. However, one drawback of this module is its limited awareness of the Allocation Graph's topology

The central component of the Verefoo framework is the Allocation, Distribution, and Placement (ADP) module. It takes as input the list of selected Network Security Functions, a Service Graph or an Allocation Graph, depending on the expertise of the experience of the security administrator, and a set of Network Security Requirements, expressed using a medium-level language and generates a Service Graph that contains the functions already existing from the input graph plus the newly allocated network security functions. This module also provides a vendor-independent configuration, utilizing a medium-level policy language, for each allocated network security function.

Finally, a vendor-dependent low-level configuration is generated through a translation process, starting from the medium-level configuration.

3.4.2 ADP Module

Up to this point, we have discussed the overall model and workflow of VEREFOO, which is capable of accommodating various types of network security requirements based on the desired security objectives. However, due to the specific focus of this thesis on automating the allocation and configuration of Parental Control systems to enforce parental control requirements, certain limitations have been imposed on the model:

1. The security administrator is limited to specifying parental control requirements only.

2. The only type of security function that can be automatically allocated and configured is the Parental Control System.

The Allocation Deployment Placement (ADP) module of VEREFOO, to which this thesis has contributed in terms of implementation, performs two crucial tasks for the proper functioning of the system:

- Automatic Orchestration and Configuration (AOC): This task involves allocating the appropriate Network Security Functions (NSFs) on the allocation graph provided as input or generated from the Service Graph. Additionally, policy rules are computed for each allocated network security function to fulfill the given security requirements.
- Automatic VNFs Placement (AVP): This task, which occurs after the Automatic Orchestration and Configuration task, is responsible for placing the Virtual Network Functions (VNFs) from the output Service Graph (including the original Service Graph and the added NSFs) onto the Physical Graph, representing the physical infrastructure.

During the execution of the Automatic Orchestration and Configuration task, the provided inputs can lead to four different scenarios. These scenarios arise due to the fact that Verefoo can receive different inputs, resulting in varying outputs from the ADP module.

Although each scenario is unique, they all share a common input: the list of medium-level Parental Control Requirements that need to be fulfilled.

The first three scenarios also involve selecting specific Network Security Functions to meet the corresponding Network Security Requirements, that in this thesis work is the Parental Control System. In contrast, the fourth scenario focuses solely on verifying an existing Service Graph against a set of Network Security Requirements and does not require any specific Network Security Function. This represents a special case.

The only input that varies across each scenario is the logical network topology. This can be in the form of a Service Graph or an Allocation Graph, with or without certain Security Functions already allocated.

In the initial scenario of the AOC task, as depicted in Figure 3.9, the ADP module can receive the following inputs:

1. The Service Graph, representing the logical topology comprising network functions necessary for providing a complete end-to-end service. It may also include indications regarding the preferred or prohibited positions for placing Network Security Functions, if the security administrator has enough security knowledge to make such decisions.

2. A list of medium-level Parental Control Requirements that must be fulfilled.
3. The list of selected Network Security Functions intended to meet the corresponding Network Security Requirements, i.e. the Parental Control Systems in this case.

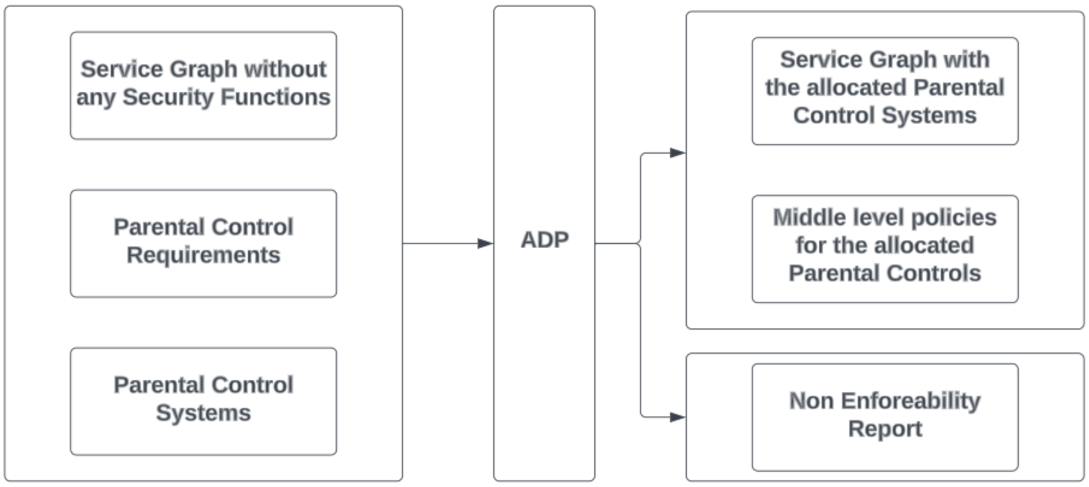
Service Graph without any Security Functions
Parental Control Requirements
Parental Control Systems
ADP
Service Graph with the allocated Parental Control Systems
Middle level policies for the allocated Parental Controls
Non Enforeability Report

Figure 3.9: Scenario 1

The outputs can be as follows:

- In the case of successful execution, an enriched Service Graph is generated. It contains the Parental Controls that have been optimally allocated within the Allocation Graph, which is automatically generated based on the provided logical topology. Additionally, the ADP module automatically creates a configuration for each allocated Parental Control. The medium-level policy rules are presented to the service designer to satisfy the specified Parental Control Requirements.
- In the case of failure, a non enforceability report that notify the inability to compute the Service Graph along with the description of the problem. The main reasons may include insufficient Network Security Functions to fulfill all the Network Security Requirements, insufficient placeholders in the Allocation Graph for allocating all the required Network Security Functions, or the inability to compute a suitable configuration due to the topology of the input Service Graph.

In the second scenario, depicted in Figure 3.10, the ADP module is provided with the following inputs:

1. A Service Graph, which already includes certain Network Security Functions with pre-computed or future configurations managed by the framework. This Service Graph may result from a manual operation or a previous execution of the ADP element on an original Service Graph that lacked security properties. The framework cannot modify the presence or configuration of these existing Network Security Functions unless dictated by user preferences. Furthermore, the service designer can provide guidance on potential placement or exclusion of new Network Security Functions, based on their expertise in security considerations.

2. A list of medium-level Parental Control Requirements that need to be fulfilled.

3. A list of selected Network Security Functions that are intended to satisfy the corresponding Network Security Requirements, i.e. the Parental Control Systems in this case.

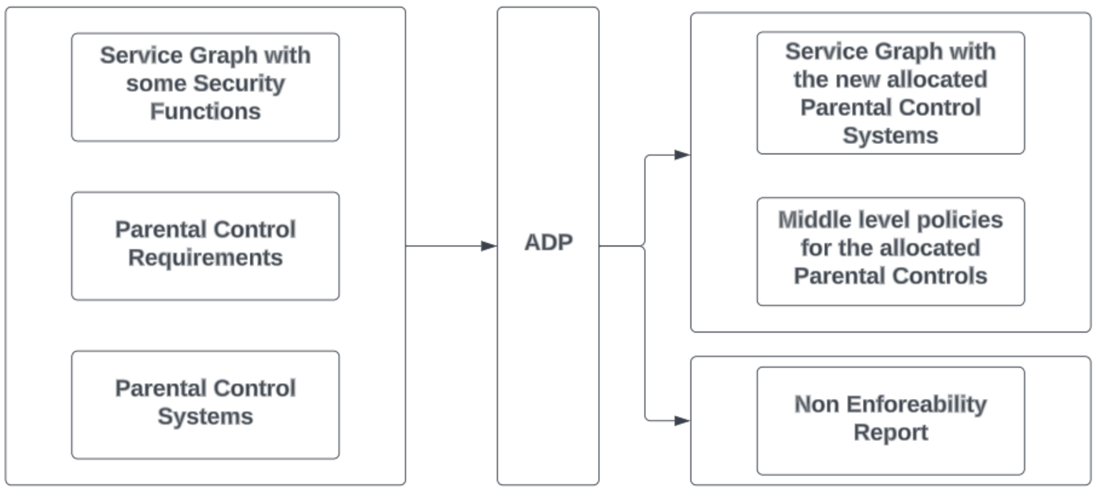
Service Graph with some Security Functions
Parental Control Requirements
Parental Control Systems
ADP
Service Graph with the new allocated Parental Control Systems
Middle level policies for the allocated Parental Controls
Non Enforeability Report

Figure 3.10: Scenario 2

In this situation, the ADP module can produce the following outputs:

- In the case of successful execution, a new Service Graph is generated where additional Parental Control Systems are optimally allocated in the Allocation Graph, while preserving the existing NSFs. The module automatically configures these newly allocated Parental Controls to meet the input medium-level Parental Control Requirements. It is important to note that In cases where the Network Security Functions (NSFs) already included in the input Service Graph lack configuration, the ADP module can compute their configuration without modifying their position within the service.

- In the case of failure, a non-enforceability report is generated to notify about the inability to compute the Service Graph and highlight the encountered issues. The main reasons for the failure of the ADP element in this scenario may include the existing Network Security Functions or their previous configurations being insufficient to satisfy all the input security requirements, particularly if the user does not wish to modify them. Another possible reason is the inadequacy of the available Allocation Places to allocate the required new NSFs for fulfilling the requirements.

In the third scenario, shown in Figure 3.11, the ADP module receives the following inputs:

1. An Allocation Graph where the security administrator specifies also all the feasible Allocation Places where the Parental Control Systems can be automatically allocated. Additionally, the security administrator can include pre-configured Parental Controls or Parental Controls to be configured later by the framework. The framework cannot modify the position and the configuration of these Parental Controls, except for potential modifications based on user preferences.
2. The list of medium-level Parental Control Requirements that need to be satisfied
3. The list of selected Network Security Functions chosen to fulfill the corresponding Network Security Requirements, i.e. the Parental Control Systems in this case

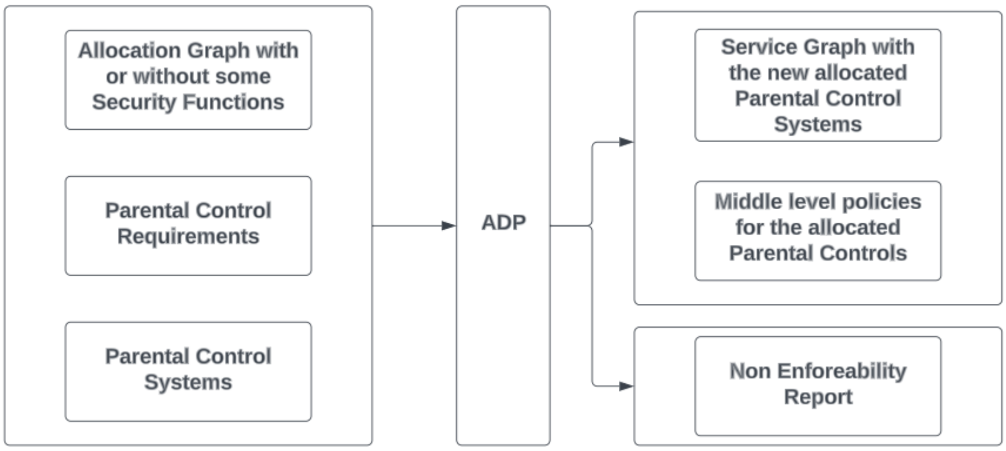
Allocation Graph with or without some Security Functions
Parental Control Requirements
Parental Control Systems
ADP
Service Graph with the new allocated Parental Control Systems
Middle level policies for the allocated Parental Controls
Non Enforeability Report

Figure 3.11: Scenario 3

In this scenario, the ADP module produces the following outcomes:

- In the case of successful execution, a Service Graph is generated where additional Parental Control Systems are optimally allocated within the Allocation Graph. The existing ones, if any, are retained, and their configuration is automatically generated to fulfill the medium-level Parental Control Requirements. It should be noted that if the existing Parental Controls in the input Allocation Graph are not configured, the ADP module can compute their configuration without modifying their position in the graph.

- In case of failure, a non-enforceability report is generated to notify about the inability to compute the Service Graph and to highlight the encountered problems. The main reasons for the failure could be that the existing Network Security Functions or their previous configuration cannot satisfy the input security requirements, particularly if the user does not wish to modify them. Another possibility is that the Allocation Places provided by the designer are insufficient to allocate the required new Parental Control Systems and meet all the requirements.

In the last scenario, depicted in Figure 3.12, a special case arises where the objective is to verify an existing Service Graph against a set of Network Security Requirements without allocating new Network Security Functions. This scenario is commonly employed when the security administrator aims to determine if a prior computation of the ADP module is adequate for meeting new or modified Parental Control Requirements, without the need to allocate and configure new Parental Control Systems. It is also applicable when the security administrator utilizes the framework to automatically compute the configuration of the Parental Control Systems. This approach saves time and resources.

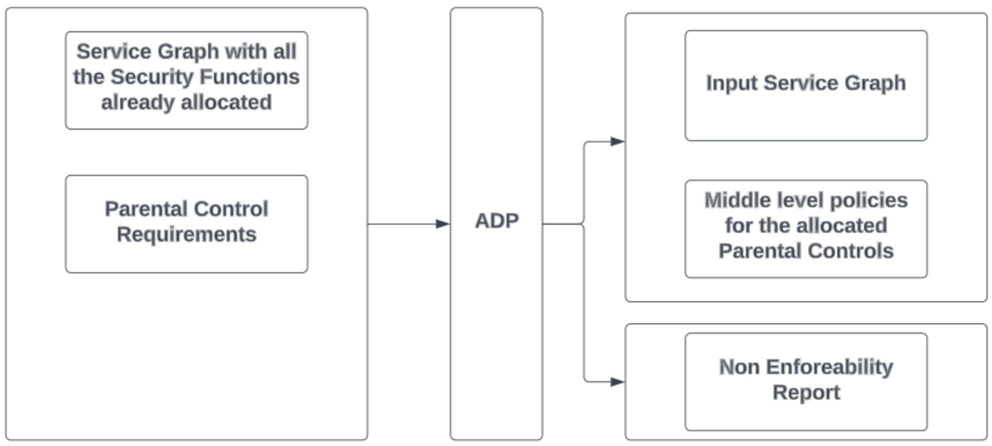
Service Graph with all the Security Functions already allocated
Parental Control Requirements
ADP
Input Service Graph
Middle level policies for the allocated Parental Controls
Non Enforeability Report

Figure 3.12: Scenario 4

To fulfill this purpose, the ADP module accepts the following inputs:

1. A Service Graph where the Parental Control Systems are already allocated on the graph. These functions' configurations can either be empty, awaiting VEREFOO to populate them, or pre-defined from a previous execution of the framework or manual configuration.
2. The list of medium-level Parental Control Requirements that need to be fulfilled.

In this scenario, the output can take the following forms:

- In the case of successful execution, the input Service Graph is given in output, and if requested, the configuration of the already allocated Parental Control Systems is computed to meet the specified medium-level Parental Control Requirements and provided to the security administrator.
- In the case of failure, a non-enforceability report is generated to notify about the inability to satisfy the Parental Control Requirements with the given Service Graph or to compute a suitable configuration for the provided Parental Control Systems. There are several factors that could contribute to the failure, including the possibility that the newly defined set of Network Security Requirements is more stringent compared to the requirements used in a previous run of the framework. Another potential reason could be that the provided Network Security Functions are inadequate to meet the specified requirements.

3.4.3 Z3

Z3, developed by Microsoft Research, is an advanced theorem prover widely used for software analysis and verification tasks. It excels in solving Satisfiability Modulo Theories (SMT) problems, which encompass a broader range of reasoning beyond the traditional Boolean Satisfiability (SAT) problem.

The Maximum Satisfiability Modulo Theories (MaxSMT) problem extends the scope of the SMT problem to include optimization. In MaxSMT, given a set of predicate clauses containing predicate variables, the objective is to find the optimal values for these variables that maximize the satisfiability of the clauses.

Similar to the SMT problem, MaxSMT is NP-complete in terms of worst-case computational complexity. However, the key distinction lies in the introduction of weights for each clause, typically set to 1 in the standard version. This means that finding a solution that satisfies the predicate clauses is not sufficient. Instead, among all possible solutions, the chosen solution must maximize the number of satisfied clauses.

There are several common variants of MaxSMT:

- Weighted MaxSMT: Each clause can have a different weight assigned to it. The satisfiability of higher-weighted clauses os prioritized in the research.
- Partial MaxSMT: Some constraints must be satisfied since are non-relaxable, other clauses, instead, do not necessarily need to be satisfied for a valid solution to be achieved.
- Weighted Partial MaxSMT: This variant combines the aspects of both weighted and partial MaxSMT problems.

In the context of this thesis, the approach adopted is weighted partial MaxSMT, which involves two distinct categories of constraints:

- Hard Constraints: These constraints are non-relaxable and must be fulfilled for a solution to be considered satisfiable. They are essential for ensuring the proper functioning of the ADP module.
- Soft Constraints: These constraints are relaxable, meaning their fulfillment is not mandatory for a satisfiable solution. In the weighted MaxSMT problem, each soft constraint is assigned a specific weight that represents its priority. The solver selects the solution that maximizes the sum of satisfied soft constraints, taking into account their respective weights. Soft constraints are useful for representing preferences and optimizing the solution.

A formal definition of the weighted partial MaxSMT problem involves considering a set of hard constraints (H) and a set of soft constraints (S):

$$max \sum_{i=1}^{s} \omega_i * s_i$$
$$\text{subject to } h_j, \forall j \in [1, H]$$

Z3 provides convenient APIs available in various programming languages such as C, Java, Python, and C++. For this thesis work, the Java APIs version 4.8.15 designed for 64-bit machines were utilized.

In Figure 3.13 the workflow of the z3 framework is presented. Upon receiving a set of formulas through a programming interface, z3 translates them into a SMTLIB2 file format. Subsequently, z3 employs tactics like pre-processing or heuristics to optimize computation time or obtain suboptimal solutions. Finally, it employs specific solvers such as SAT to obtain a solution. If an optimization phase is required, z3 employs an optimizer engine named z3Opt to obtain the optimal solution.

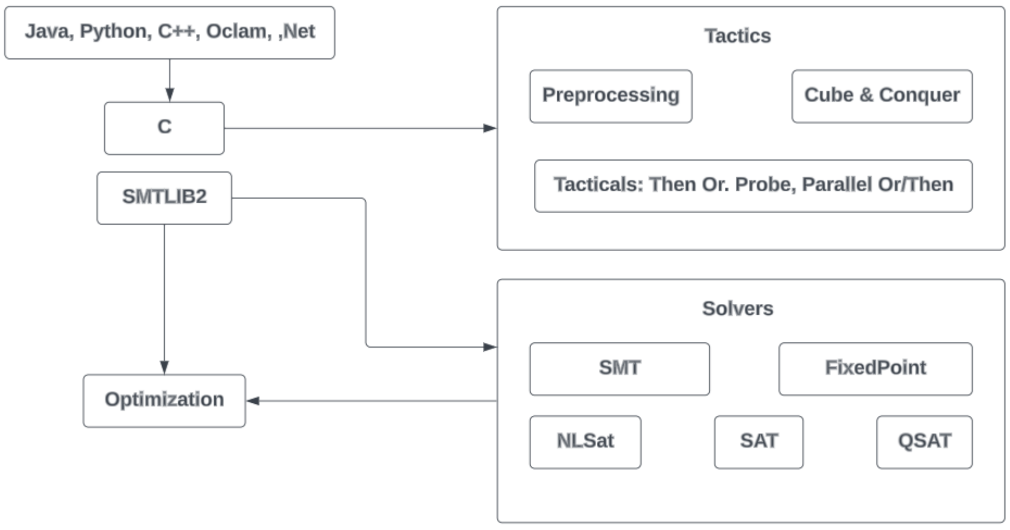

Figure 3.13: Z3 Workflow

The upcoming chapter will delve into a detailed discussion of the modifications made to the ADP module in this thesis. Specifically, it will focus on the development of distinct formal models for various components involved in executing the ADP Module for implementing a Parental Control System. These formal models were crucial in ensuring the effective functioning of the system.

Chapter 4

Parental Control Model: Implementation and Validation

4.1 Network Security Requirements

The Network Security Requirements serve as one of the inputs for VEREFOO. Users of the framework can formulate these requirements using either a high-level representation or a medium-level security representation, depending on their security expertise. Within VEREFOO, the H2M module (described in Section 5.2) translates high-level requirements into medium-level representations. As a result, the ADP module exclusively receives constraints expressed in the medium-level language without requiring further modifications. In this thesis work, this aspect was considered as an assumption when defining the model for representing security requirements. Another assumption made is that the set of input Parental Control Requirements is expressed in the medium-level language and does not contain any conflicts. The thesis analyzed the Parental Control Requirements, which specify the need to allow only the traffic suitable for the user utilizing a particular endpoint.

4.1.1 Parental Control Requirements Model

Consider the set R representing the Parental Control Requirements that a security administrator can specify for an input Service Graph. Each security requirement in this set is formulated using the following components of a medium-level representation:

[ruleType, user, contentCategory, filterLevel, deviceIP, dailyTimeLimit]

- *ruleType*: Indicates the type of security requirement to be satisfied. In the scope of this thesis, it assumes the value *PC_Property*, representing a Parental Control Property. However, it can be extended in future work to integrate other types of requirements.
- *user*: Specifies the user to whom the requirement applies.
- *contentCategory*: Identifies the category of the traffic to which the requirement applies.
- *filterLevel*: Represents the level of traffic filtering specified by the requirement.
- *deviceIP*: Refers to the IP address of the device specified by the requirement. If this field is left unspecified, it indicates that the requirement applies to all devices used by a specific user.
- *dailyTimeLimit*: is the maximum amount of time that a particular device can be used within a day, as specified by the requirement.

In the original version of the framework, the *user* and *contentCategory* components were restricted to accepting only IP addresses. This constraint was in place to signify their roles as the source and destination components of the Network Security Requirements, specifically pertaining to endhosts. However, in the context of this thesis, modifications have been made to enable non-IP inputs for these components.

Regarding the *user*, it now expects usernames of users present in the input Graph as input. It is assumed that these usernames are unique and correspond to the ones used in a typical Parental Control application.

As for the *contentCategory*, it currently assumes the value *INTERNET_ENDHOST* in this thesis. This introduces a level of abstraction compared to most commercial parental controls available in the market. For instance, the current implementation does not allow for specifying a specific URL to be blocked for a particular user or selecting a category from a predefined list. Furthermore, it does not provide the ability to block a specific app or game. However, there is potential for expansion by defining additional functionality, which would enable a more detailed and nuanced specification.

The *filterLevel* component accepts only the following input values: *NONE*, *KID*, *TEEN*, and *ADULT*. By specifying the most appropriate value, users can ensure that the traffic content aligns with their age or personal preferences.

As for the *deviceIP*, it expects a specific IP address in the traditional dot-decimal notation:

$$ip_1.ip_2.ip_3.ip_4$$

where ip_i, $\forall i \in \{1, 2, 3, 4\}$, can be an integer ranging from 0 to 255, inclusive.

The *dailyTimeLimit* component only accepts strings formatted as *HH:MM*. The hours (*HH*) can have values from 00 to 23, while the minutes (*MM*) can have values from 00 to 59. Alternatively, the value 24:00 can be used to indicate no restrictions, allowing device usage for 24 hours per day.

In the scope of this thesis, two types of Parental Control implementations have been developed: *simpleParentalControl* and *advancedParentalControl*. The first type, *simpleParentalControl*, represents a simple implementation of Parental Control. When allocating a Parental Control Requirement for this type, it is only necessary to specify the *src*, *dst*, and *filterLevel* attributes. This type of Network Security Requirement applies to all devices used by a specific user.

On the other hand, the second type, *advancedParentalControl*, is more complex. Allocating a Parental Control Requirement for this type requires the specification of *src*, *dst*, *filterLevel*, *device*, and *dailyTimeLimit* attributes. If the *device* and *dailyTimeLimit* are not specified, the behavior of Parental Control will be the same as the first type.

By specifying the *device* attribute, it is possible to create a Network Security Requirement that applies specifically to the device used by the source User, with reference to the provided IP address value. Additionally, the second type of Parental Control allows the inclusion of the *dailyTimeLimit* attribute in the Parental Control Requirement, which adds a constraint regarding the duration of time the device can be used.

4.1.2 Implementation in the XML Schema of the Parental Control Requirements

The XML schema implementation of the Parental Control Requirements, inherited from the previous version of the framework, utilizes *Property* elements. Each *Property* element is characterized by *name*, *src*, *dst*, *filterLevel*, *device* and *dailyTimeLimit*. In the implementation phase, the terms *src* and *source* were used to refer to the user to whom the Requirement or Policy applies. Conversely, the terms *dst* and *destination* were used to refer to the contentCategory to which the Requirement or Policy applies. Additionally, the term *parental_control_t1* has been used to represent the *simpleParentalControl* implementation, while the term *parental_control_t2* has been used to refer to the *advancedParentalControl* implementation.

Two examples extracted from an XML input file of the framework are provided below.

In Listing 4.1, a Parental Control Requirement aiming to allocate a *simpleParentalControl* as final goal is depicted. According to this requirement, only the traffic that corresponds to the *filterLevel* "KID" is allowed for the *User* "James" and the *Internet_EndHost*. The *Internet_EndHost* is a fictional representation of all traffic directed to the Internet. Regardless of the device from which the *User* is connecting, only a specific type of traffic is permitted, while all other traffic is denied.

```xml
    <Property graph="0" name="PC_Property" src="James" dst="INTERNET_ENDHOST"
        filterLevel="KID"/>
```

Listing 4.1: XML example of a medium-level Parental Control Requirement for the allocation of simpleParentalControl

In Listing 4.2, a Parental Control Requirement is shown, aiming to allocate a advancedParentalControl as the final goal. This requirement specifies that only traffic corresponding to the "KID" *filterLevel* is allowed for the *User* "James" and the *Internet_EndHost*. Additionally, the Parental Control Requirement is specified for a specific device with the IP address 10.0.0.1. It also includes a *dailyTimeLimit* of 07:30, indicating a restriction on the device usage time of 7 hours and 30 minutes per day.

```xml
    <Property graph="0" name="PC_Property" src="James" dst="INTERNET_ENDHOST"
        filterLevel="KID" device="10.0.0.1" dailyTimeLimit="07:30"/>
```

Listing 4.2: XML example of a medium-level Parental Control Requirement for the allocation of advancedParentalControl

4.2 EndPoints

As discussed in Section 3.4.1, Verefoo can accept two types of input graphs: the Service Graph and the Allocation Graph. The Service Graph is an extension of the Service Function Chain concept, representing the logical topology of an end-to-end service composed of multiple network functions. On the other hand, the Allocation Graph includes the same network functions as the Service Graph, but additionally incorporates Allocation Places. In the scope of this thesis work, the types of endpoints that can be defined within both the Service Graph and the Allocation Graph has been expanded. Originally, the Verefoo framework was developed to automate the generation of various configuration aspects in an enterprise network. However, since this thesis focuses on Parental Control Systems in the smart home environment, the need arose to model new types of endpoints to accommodate this specific domain.

The first type of endpoint introduced is the *Internet_EndHost*. It serves as a fictional representation of the destination for all traffic directed towards the Internet. To simplify the definition of Network Security Requirements related to Parental Control Systems, IP addresses are no longer used. Instead, two key actors define a Parental Control Requirement: the *User* to whom the requirement applies (a new concept introduced in this thesis) and the *Internet_EndHost* as the designated destination.

Additionally, the following endpoints have been introduced: *FIC_Device*, *BIC_Device*, and *LIC_Device*. A *FIC_Device* (Full Internet Capabilities Device) represents devices that provide users with complete access to the Internet, such as smartphones, tablets, laptops, e-readers, and smart TVs. These devices are capable of installing a Parental Control application and configuring User Profiles.

LIC_Device (Limited Internet Capabilities Device) refers to devices that have restricted Internet functionality and do not support the installation of Parental Control applications despite having internet access. This category includes smart appliances such as Alexa or smart refrigerators.

Lastly, *BIC_Device* (Basic Internet Capabilities Device) includes devices with basic Internet functionality necessary for their intended usage or to provide users with essential functions. This kind of devices comprises intelligent appliances like Alexa or smart refrigerators. Similar to *LIC_Device*, *BIC_Device* devices do not support the installation of a Parental Control application.

4.2.1 Implementation in the XML Schema of Endpoints

An *Internet_Endhost* in the XML schema is depicted as a node element with the functional type attribute set as *INTERNET_ENDHOST*. Apart from the neighbor elements that represent standard internal elements of a node, an *Internet_Endhost* possesses a distinct configuration that incorporates an an *internet_endhost* element. Currently, this element solely encompasses the *name* field, which is not utilized within the scope of this thesis. Implementation of the *Internet_Endhost* in the XML Schema is demonstrated in Listing 4.3.

```
<xsd:element name="internet_endhost">
   <xsd:complexType>
         <xsd:attribute name="name" type="xsd:string" default="
            internet_endhost"/>
   </xsd:complexType>
</xsd:element>
```

Listing 4.3: Internet_EndHost XML element

The *User*, instead, is represented as a node element with a functional type attribute labeled as *USER*. Like other nodes, the *User* node also possesses a neighbors attribute. Additionally, a *User* node has a specific configuration that includes a *user* element. The *user* element currently includes the *currentDevice* field, which is not utilized in this thesis. However, it could potentially be used in future extensions of the framework instead of the neighbor attribute. The implementation of a *User* in the XML Schema is shown in Listing 4.4.

```
<xsd:element name="user">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="currentDevice" maxOccurs="unbounded"
                         minOccurs="0" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Listing 4.4: User XML element

In addition, the XML schema presented in Listing 4.5 also includes the implementation of the *currentDevice* field:

```
<xsd:element name="currentDevice">
    <xsd:complexType>
        <xsd:attribute name="name" type="xsd:string"
                       use="required" />
        <xsd:attribute name="type" type="deviceType"
                       use="required" />
    </xsd:complexType>
</xsd:element>

<xsd:simpleType name="deviceType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="FICDevice" />
        <xsd:enumeration value="BICDevice" />
        <xsd:enumeration value="LICDevice" />
    </xsd:restriction>
</xsd:simpleType>
```

Listing 4.5: currentDevice XML element

The *currentDevice* field is defined as an element within the *User* element. It consists of two attributes: "name", which is a required string attribute defining the name of the device, and "type", which is a required attribute of type "deviceType". The "type" attribute allows specifying whether the device is a *Fic_Device*, *Lic_Device*, or *Bic_Device*.

Fic_Device, *Lic_Device*, and *Bic_Device* are represented in the XML schema by node elements with functional type attributes set as *FIC_DEVICE*, *LIC_DEVICE*, and *BIC_DEVICE*, respectively. These nodes, like others, have internal elements such as *neighbors*. They are also characterized by a specific configuration that includes *fic_device*, *lic_device*, and *bic_device* elements, respectively. These elements currently consist of the *specificDevice* field, which is not utilized in the context of this thesis. The XML Schema implementation of these elements is demonstrated in Listing 4.6.

```
<xsd:element name="fic_device">
    <xsd:complexType>
        <xsd:attribute name="specificDevice" type="xsd:string" />
    </xsd:complexType>
</xsd:element>

<xsd:element name="lic_device">
    <xsd:complexType>
        <xsd:attribute name="specificDevice" type="xsd:string" />
    </xsd:complexType>
</xsd:element>

<xsd:element name="bic_device">
    <xsd:complexType>
        <xsd:attribute name="specificDevice" type="xsd:string" />
    </xsd:complexType>
</xsd:element>
```

Listing 4.6: Fic_Device, Lic_Device and Bic_Device XML elements

An example of an input Allocation Graph containing these new implemented Endpoints is presented in Listing 4.7.

```
<graph id="0">

    <node functional_type="USER" name="James">
        <neighbour name="10.0.0.1"/>
        <configuration description="A simple description" name="confA">
            <user>
                <currentDevice name="10.0.0.1" type="FICDevice"/>
            </user>
        </configuration>
    </node>


    <node functional_type="FIC_DEVICE" name="10.0.0.1">
        <neighbour name="James"/>
        <neighbour name="30.0.0.1"/>
        <configuration description="A simple description" name="confB">
            <fic_device specificDevice="tablet"/>
        </configuration>
    </node>


    <node functional_type="FIC_DEVICE" name="10.0.0.2">
        <neighbour name="30.0.0.1"/>
```

```
            <configuration description="A simple description" name="confB">
                <fic_device specificDevice="computer"/>
            </configuration>
        </node>


        <node name="30.0.0.1">
            <neighbour name="10.0.0.1"/>
            <neighbour name="10.0.0.2"/>
            <neighbour name="INTERNET_ENDHOST"/>
        </node>


        <node functional_type="INTERNET_ENDHOST" name="INTERNET_ENDHOST">
            <neighbour name="30.0.0.1"/>
            <configuration description="A simple description" name="confC">
                <internet_endhost/>
            </configuration>
        </node>
    </graph>
```

Listing 4.7: XML example of an Allocation Graph containing the new implemented Endpoints

In this example, the Network topology includes a single *User* named "James" connected to a *Fic_Device* with the IP address "10.0.0.1." In this specific case, the *Fic_Device* represents a Tablet. Additionally, there is another *Fic_Device* present in the Allocation Graph, representing a computer with the IP address "10.0.0.2." Currently, this computer is not being used by any user. Both these *Fic_Device* are directly connected to the *Internet_Endhost* through a link, in which has been placed an Allocation Point.

4.3 Parental Control Systems

In the context of a smart home network, the implementation of a parental control system becomes crucial to ensure the safety of children and adolescents while using the Internet. Consequently, it is necessary to extend the capabilities of Verefoo to address the new challenges that arise in this distinct scenario.

Parental control encompasses two main aspects: the built-in parental control systems found in smart TVs and the network-level parental control system that encompasses all devices capable of connecting to the Internet, such as smartphones, laptops, and tablets. Smart TVs typically have parental control features as a device function, often lacking comprehensive documentation on their functionality. On the other hand, network-level parental control can be implemented at various levels, including the edge router within the network, the ISP level, or as device-installed software, offering a wide range of solutions.

Given the abundance of commercial solutions available, it is beneficial to identify the most common functionality to create a model that aligns with the majority of these solutions.

Regarding smart TV parental control systems, the following functionality are commonly implemented:

- Blocking channels: When the parental control feature is active, specific channels are blocked for all users, requiring the entry of a 4-digit PIN code to unlock them temporarily. The lock channel feature can be deactivated temporarily by entering the correct PIN code or by power-cycling the TV.
- Blocking apps: Similar to channel blocking, certain apps can be blocked for all users. The lock app feature requires the entry of a 4-digit PIN code to access the app temporarily. Like the lock channel feature, it can be deactivated temporarily by entering the correct PIN code or power-cycling the TV.
- Blocking programs based on TV ratings: Programs rated for ages above the specified age limit can be blocked for all users. The lock program feature requires the entry of a 4-digit PIN code to view such programs temporarily. The lock can be deactivated temporarily by entering the correct PIN code or power-cycling the TV. The rating system relies on information from the broadcasting station.
- Blocking TV input sources: When the parental control feature is active, access to certain TV input sources can be blocked for all users. The lock TV input source feature requires the entry of a 4-digit PIN code to watch the content from the input source. The lock can be deactivated temporarily by entering the correct PIN code or power-cycling the TV.
- Sleep timer: Users can set a sleep timer, causing the smart TV to automatically turn off when the timer duration elapses.
- Daily screen time limit: Parents can create user profiles and set a maximum time period (e.g., 30 minutes, 1 hour) during which a specific child user can use the smart TV.

These functionality were identified based on the smart TV parental control solutions provided by devices such as Google TVs, Apple TVs, Samsung TVs, LG TVs, Hisense TV H5507 Model, Hisense TV H65B7500 Model, and Sony TVs.

Regarding general parental control systems operating at the network level, the following functionality are commonly implemented:

- Child profiles and restrictions: Parents can create child profiles and define specific restrictions for each child.

- Web filtering: Specific URLs can be blocked, websites can be filtered based on predefined categories, websites can be filtered based on age-related categories, and adult content can be filtered.
- Blocking inappropriate apps and games.
- Screen time limitations: Parents can schedule specific times during the day or week when a child can access the Internet and set a limit on the number of hours per day the child can use the Internet. After reaching the time limit, either all websites become unavailable or only predefined categories remain accessible.
- Real-time Internet cutoff: Parents can disable Internet access for a child instantly.
- Bedtime and Awake time: Scheduling allows parents to define times when the Internet should be turned off during a child's bedtime and reactivated afterward.
- Distraction-free focus time: Parents can create a list of permitted educational websites that a child can access during specific days and times. Additionally, they can customize a block list of distracting sites.

These functionality were considered based on the parental control solutions offered by Vodafone Rete Sicura Family, TP-Link HomeCare, Net Nanny, Circle, Bitdefender, Norton Family, and Qustodio.

In this thesis, the focus primarily revolves around analyzing and modeling the general parental control system, as it covers the widest range of device types.

Specifically, two types of parental control systems have been modeled. The first type, *simpleParentalControl*, is a basic system that permits personalized traffic control based on pre-defined filter levels for each user. The second type, *advancedParentalControl*, is a more sophisticated system that introduces device-level restrictions, such as establishing a maximum daily usage time for a device.

Both Parental Control Systems are regulated by a Parental Control Policy, which serves as the configuration utilized by the Network Security Function to govern the permissible traffic for particular users. The Parental Control Policy for the simpleParentalControl system is referred to as *parental_control_t1*, while the policy for the advancedParentalControl system is named *parental_control_t2*. The Parental Control Policy can be set up either by a knowledgeable security administrator or automatically configured by the ADP module in a precedent run of the framework in a way to enforce parental controls at designated locations within the input Service Graph.

In the Allocation Graph each Allocation Place p_k where a Parental Control can be allocated is associated with a Parental Control Policy. This Policy can include one or more policy rules Ψ_k, referred to as *PC_t1_elements* for the simpleParentalControl and *PC_t2_elements* for the advancedParentalControl. These rules specify that all traffic generated by a specific user and directed towards an *Internet_EndHost* (i.e., all Internet-related traffic) is allowed only if it satisfies the designated filter level. It is assumed that the policy rules are non-redundant.

Each *PC_t1_elements* element is characterized by the following attributes:

- *user*: Represents the user to which the rule applies, identified by a unique username.
- *contentCategory*: Specifies the category of the traffic, which is always *INTERNET_ENDHOST* in the context of this thesis.
- *filterLevel*: Determines the level at which the traffic is filtered to make it more suitable for the specific user, ensuring that inappropriate content is not accessible.

In addition to the attributes of the *PC_t1_elements* element, each *PC_t2_elements* element is characterized by the following attributes:

- *device*: Indicates the particular device used by the user to which the policy applies. It is uniquely identified by its IP address.
- *dailyTimeLimit*: Specifies the maximum amount of time a particular device can be used within a day.

Drawing inspiration from existing commercial solutions in the market, a new feature has been developed in this thesis. In real-world scenarios where certain devices, such as *LIC_Devices* and *BIC_Devices*, do not support direct installation of a Parental Control, a common approach is to implement the Parental Control feature in a network router. This is achieved by identifying each device using its unique IP address.

In addition to the option of allocating the Parental Control in an Allocation Place p_k along the path between the source and destination of the requirement, this thesis introduces the capability to allocate the Parental Control directly within the device used by the specific User. However, this is only feasible if the device has full internet capabilities, specifically if it is classified as a *Fic_Device.* In this particular scenario, when examining the generated Allocation Graph as the output, the *Fic_Device* element will include a properly configured Parental Control Policy, rather than incorporating the *specificDevice* element as a configuration element.

It is important to note that the assumption made in this thesis is that each device is used by a single user. The challenge arises when a single device is shared among multiple users, making it difficult to identify which user generated the traffic.

Since the traffic is associated with the device's IP address, which is shared among all users of the device, it becomes problematic to assign the traffic to a specific user.

One potential solution to this problem could involve identifying the user through login credentials used to access the Internet or other applications. However, it should be acknowledged that not all routers support this user identification feature, making it impractical to use this solution universally. As a general approach, it can be assumed that each device is assigned to only one user, as implemented in Circle Parental Control.

4.3.1 Implementation in the XML Schema of Parental Control Systems

The XML schema presents the model of a single Parental Control Policy rule for a simpleParentalControl, illustrated in Listing 4.8.

```
<xsd:element name="PC_t1_elements">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="source" type="xsd:string"/>
            <xsd:element name="destination" type="xsd:string" />
            <xsd:element name="filterLevel" type="PC_filterLevel" minOccurs="
                0" default="NONE"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Listing 4.8: Parental Control Policy XML element for a simpleParentalControl

An example of an automatically generated Parental Control Policy for a simpleParentalControl can be found in Listing 4.9. This policy is generated by the framework when allocating the Parental Control System in an Allocation Place between the source and destination of the requirement.

```
<node name="30.0.0.1" functional_type="PARENTAL_CONTROL_T1">
            <neighbour name="10.0.0.1"/>
            <neighbour name="10.0.0.2"/>
            <neighbour name="INTERNET_ENDHOST"/>
            <configuration name="ParentalControlAutoConf">
                <parental_control_t1>

                    <PC_t1_elements>
                        <source>James</source>
                        <destination>INTERNET_ENDHOST</destination>
```

```
                            <filterLevel>KID</filterLevel>
                        </PC_t1_elements>

                    </parental_control_t1>
                </configuration>
            </node>
```

Listing 4.9: XML Schema for a Parental Control Policy of a simpleParentalControl allocated in an Allocation Place between the source and destination of the requirement.

In contrast, Listing 4.10 displays an example of a Parental Control Policy that can be automatically generated by the framework for a simpleParentalControl when the intention is to allocate the Parental Control System directly inside a *Fic_Device*.

```
    <node name="10.0.0.1" functional_type="FIC_DEVICE">
                <neighbour name="James"/>
                <neighbour name="30.0.0.1"/>
                <configuration name="ParentalControlAutoConf">
                    <parental_control_t1>

                        <PC_t1_elements>
                            <source>James</source>
                            <destination>INTERNET_ENDHOST</destination>
                            <filterLevel>KID</filterLevel>
                        </PC_t1_elements>

                    </parental_control_t1>
                </configuration>
            </node>
```

Listing 4.10: XML Schema for a Parental Control Policy of a simpleParentalControl allocated in a Fic_Device.

In this scenario, when dealing with a simpleParentalControl, it is only possible to define requirements that apply to a specific User without specifying the device to which the requirement pertains. Therefore, if the intention is to allocate the Parental Control Policy directly inside the device to satisfy the requirement, a separate Parental Control System must be allocated in each device used by the User. However, it should be noted that if any of the devices used by the user is a *Bic_Device* or a *Lic_Device*, the framework's result will result in an *UNSAT* outcome.

Similarly, the XML schema depicted in Listing 4.11 showcases the structure of an individual Parental Control Policy rule for a advancedParentalControl.

```
<xsd:element name="PC_t2_elements">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="source" type="xsd:string"/>
            <xsd:element name="destination" type="xsd:string" />
            <xsd:element name="filterLevel" type="PC_filterLevel" minOccurs="
                0" default="NONE"/>
            <xsd:element name="device" type="xsd:string" default="*"/>
            <xsd:element name="dailyTimeLimit" type="PC_dailyTimeLimit"
                default="24:00"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Listing 4.11: Parencal Control Policy XML element for a advancedParentalControl

An illustration of an automatically generated Parental Control Policy for a advancedParentalControl can be observed in Listing 4.12. This policy is generated by the framework when allocating the Parental Control System in an Allocation Place between the source and destination of the requirement.

```
<node name="30.0.0.1" functional_type="PARENTAL_CONTROL_T2">
            <neighbour name="10.0.0.1"/>
            <neighbour name="10.0.0.2"/>
            <neighbour name="INTERNET_ENDHOST"/>
            <configuration name="ParentalControlAutoConf">
                <parental_control_t2>

                    <PC_t2_elements>
                        <source>James</source>
                        <destination>INTERNET_ENDHOST</destination>
                        <filterLevel>KID</filterLevel>
                        <device>10.0.0.1</device>
                        <dailyTimeLimit>07:30</dailyTimeLimit>
                    </PC_t2_elements>

                </parental_control_t2>
            </configuration>
        </node>
```

Listing 4.12: XML Schema for a Parental Control Policy of a advancedParentalControl allocated in an Allocation Place between the source and destination of the requirement.

On the other hand, Listing 4.13 presents an instance of a Parental Control Policy that is automatically generated by the framework for a advancedParentalControl when the objective is to allocate the Parental Control System directly within a *Fic_Device*.

```
   <node name="10.0.0.1" functional_type="FIC_DEVICE">
              <neighbour name="James"/>
              <neighbour name="30.0.0.1"/>
              <configuration name="ParentalControlAutoConf">
                  <parental_control_t2>

                      <PC_t2_elements>
                          <source>James</source>
                          <destination>INTERNET_ENDHOST</destination>
                          <filterLevel>KID</filterLevel>
                          <device>10.0.0.1</device>
                          <dailyTimeLimit>07:30</dailyTimeLimit>
                      </PC_t2_elements>

                  </parental_control_t2>
              </configuration>
          </node>
```

Listing 4.13: XML Schema for a Parental Control Policy of a advancedParentalControl allocated in a Fic_Device.

In this situation, considering the advancedParentalControl, it is possible to set requirements that are specific to a particular user and device. If the objective is to allocate the Parental Control Policy directly within the device to meet the requirement, it is necessary to allocate the Parental Control System solely within the devices that are relevant to the requirement.

4.4 Validation

4.4.1 Use Case 1

Consider the given input Allocation Graph in Verefoo depicted in Figure 4.1.

In this network topology, there is a *User* connected to a *Fic_Device*. The service designer has created three Allocation Places as placeholder positions. These positions allow the MaxSMT solver to determine if allocating a Parental Control is necessary to achieve the optimal allocation schema after solving the optimization problem.

Now, let's suppose that a simpleParentalControl needs to be allocated along the path connecting source and destination of the requirement. The specific Parental Control Requirement is the following:

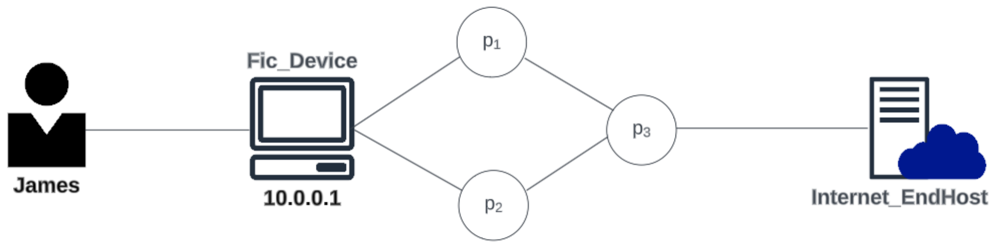

Figure 4.1: Input Allocation Graph

1. All the traffic from James to Internet_EndHost must satisfy the filterLevel "KID"

Considering that the Allocation Places are strategically positioned along the path between the specified sources and destinations of the Parental Control Requirement, the framework must consider the option of allocating a Parental Control in these positions. To incorporate this consideration, a set of hard and soft constraints is introduced in the MaxSMT problem formulation.

After the MaxSMT problem is solved, there are two possible outcomes for each Allocation Place:

- If a Parental Control is allocated, the Allocation Place is retained in the framework's output. It is then configured with a Parental Control Policy, which specifies the type of traffic that is deemed appropriate for a particular User.

- If no Parental Control is allocated, the Allocation Place functions with a simple forwarding behavior. It forwards each received packet to all possible out-ports. In a post-processing task, this Allocation Place can be removed from the system.

In this particular scenario, if the ADP module successfully solves the MaxSMT problem and finds a satisfiable solution, a new Service Graph is generated. This graph incorporates the allocated and/or configured Parental Control functions, as illustrated in Figure 4.2.

The solution illustrates the allocation and configuration of one Parental Control on the Graph. The Parental Control includes a defined Parental Control Policy with a single rule. According to this rule, all traffic generated by the *User* named "James" and directed towards the Internet must adhere to the *filterLevel* "KID".

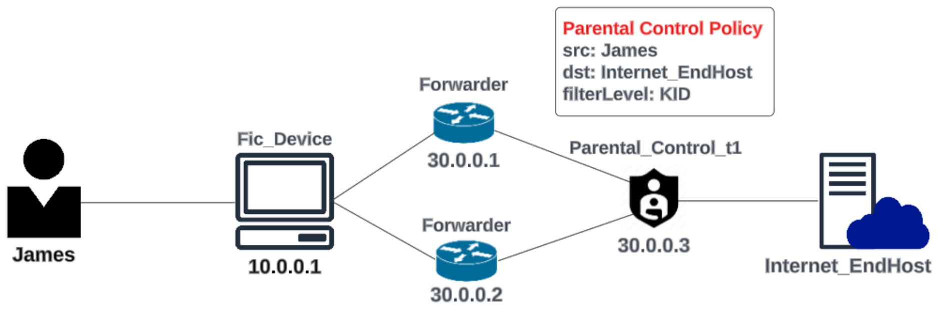

Figure 4.2: Output Service Graph

4.4.2 Use Case 2

Let's consider the given input Allocation Graph in Verefoo, as shown in Figure 4.3.

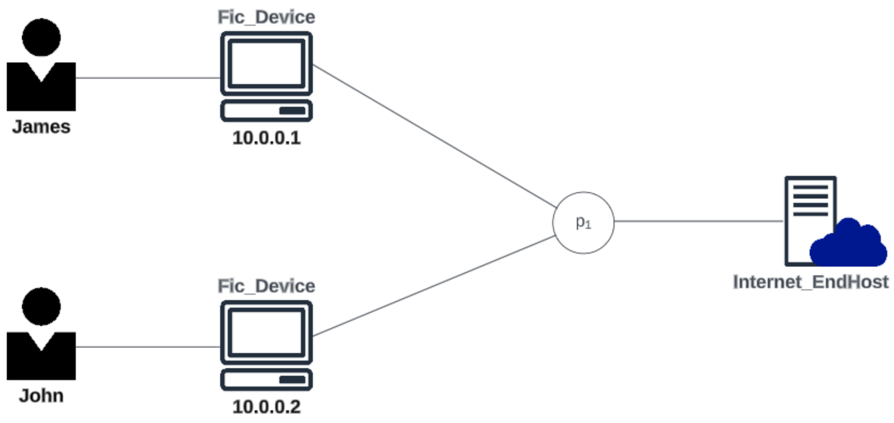

Figure 4.3: Input Allocation Graph

In this network topology, there are two Users, each connected to their respective *Fic_Device*. The service designer has created an Allocation Place as a placeholder. This position can be considered by the MaxSMT solver to determine whether allocating a Parental Control is necessary to achieve the optimal allocation schema after solving the optimization problem.

Now, suppose that a advancedParentalControl needs to be allocated along the path connecting source and destination of the requirement. The Parental Control Requirements are the following:

1. All the traffic from James to Internet_EndHost, while using the device with

IP address 10.0.0.1, must satisfy the filterLevel "KID". Set a dailyTimeLimit of 7 hours and 30 minutes for that device

2. All the traffic from John to Internet_EndHost must satisfy the filterLevel "ADULT"

In this scenario, the requirements include the device to which the Policy should be applied. Additionally, the first Property specifies a *dailyTimeLimit* for James's device.

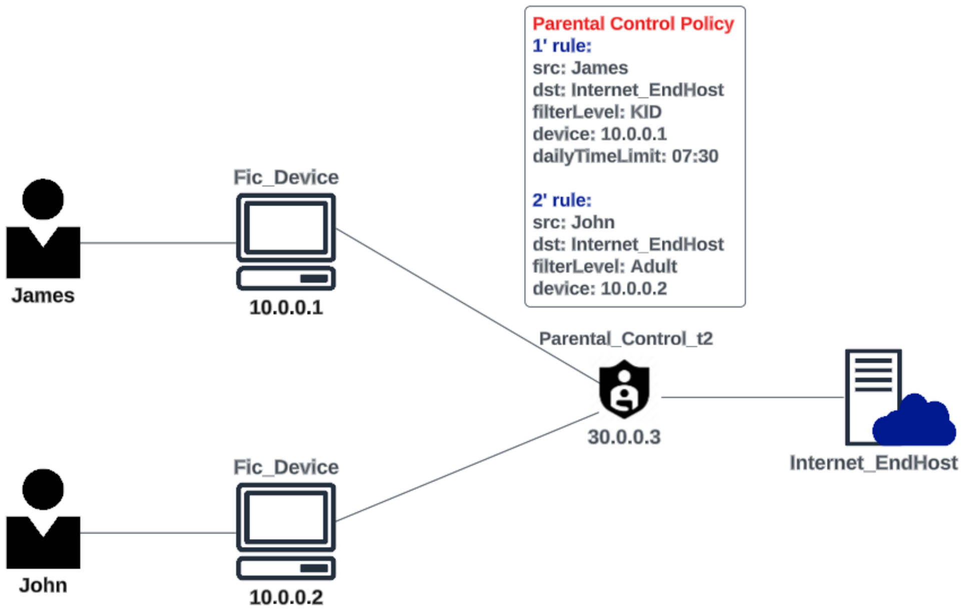

Figure 4.4: Output Service Graph

If the ADP module finds a satisfactory solution, a new Service Graph is generated, which incorporates the allocated and/or configured Parental Control functions, as depicted in Figure 4.4.

The solution demonstrates the allocation and configuration of one Parental Control on the Graph. The Parental Control includes a defined Parental Control Policy with two rules. The first rule states that all traffic generated by the *User* "James", using the device with IP address 10.0.0.1, and directed towards the Internet, must comply with the *filterLevel* "KID". Additionally, this rule specifies the *dailyTimeLimit* of "07:30", as required in the original requirement. The second rule states that all traffic generated by the *User* "John", using the device with IP address 10.0.0.2, and directed towards the Internet, must adhere to the *filterLevel* "ADULT".

4.5 Discussion

A series of tests were conducted to evaluate the effectiveness of the developed solution. The tests were performed on a laptop equipped with an i5-8250U CPU and 8 GB of RAM. The z3 version used for verification was 4.8.15, specifically designed for 64-bit machines.

The tests encompassed different network topologies and Parental Control Requirements as inputs, covering a range of configuration scenarios. All the tests were successful, and the time required to solve the MaxSMT problem and allocate and configure the Parental Control Systems varied depending on the inputs. However, the time ranged from approximately 200ms to 2000ms, which is considered acceptable.

It is worth noting that when using z3 version 4.8.8, some tests did not finish within the expected timeframe. Specifically, the "testPC_t1Correctness05_FicDevice" test in the TestParentalControl_t1.java class took around 5 minutes to complete successfully, while the "testPC_t2Correctness05_FicDevice" test in the TestParentalControl_t2.java class did not finish execution even after 10 minutes. Both tests had the same network topology as input, with the only difference being the type of allocated Network Security Function (simpleParentalControl in one case and advancedParentalControl in the other). This discrepancy is believed to be solely attributable to the version of z3 being used.

Another important limitation of the thesis work is that optimization aspects were not taken into account during the development of this solution, with a focus on effectiveness rather than efficiency. Consequently, scalability tests have not been conducted, leaving room for future work to expand the current solution.

Future research can prioritize the expansion of the current parental control implementation to incorporate a wider range of capabilities commonly found in commercial parental control systems. This entails integrating additional features that are currently not included in the framework. The developed solution encompasses two distinct implementations, offering both a simpler and a more advanced approach. These implementations provide the flexibility to specify constraints based on devices and enable the setting of limitations on daily device usage time.

However, there are additional functionality offered by commercial systems that can be incorporated. For example, the ability to block specific URLs, select specific content categories from a predefined list, block specific apps or games, and schedule internet access during specific times or days. Other potential enhancements include the addition of features such as bedtime and awake time, where internet access is disabled during a child's designated bedtime and reactivate it afterward. Real-time internet cutoff, allowing parents to instantly disable internet access as needed, and distraction-free focus time, during which only educational websites are permitted, would further enhance the parental control capabilities of the system.

However, the existence of self-configuration tools for home networks alone is not sufficient to address network security concerns, as these tools are typically accessible only to experts. to make these tools accessible to non-expert users, it is crucial to design user-friendly interfaces that allow an easy and a real-time interaction with these tools. These interfaces can leverage natural language rules, descriptions of desirable and undesirable situations, and other intuitive methods to abstract specific rules. This approach would enhances the system's usability, making it more user-friendly and intuitive.

Furthermore, addressing the interaction between commercial parental control solutions and Verefoo poses technical challenges. Each system may have unique functionality that differ from one another. One potential strategy is to incorporate all the diverse capabilities provided by commercial parental control systems and dynamically present the user with a subset of compatible features based on their chosen Parental Control System instance. By tailoring the available functionality to align with the selected solution, users can effectively utilize the features that are compatible with their specific requirements.

Although the solution developed in this thesis has limitations and room for improvement, it represents a significant extension to the functionality available in the Verefoo framework, enabling the automatic allocation and configuration of a broader range of Network Security Functions. Additionally, it tackles the specific context of home networks, which distinguishes it from the traditional contexts typically covered by Verefoo, marking a significant milestone for the project.

Chapter 5

Conclusions

The primary objective of this thesis was modifying Verefoo to support the automatic allocation and configuration of Parental Control Systems, which is a critical problem in the context of home networks. To successfully accomplish this objective, the ADP module of Verefoo has been extended. This expansion empowers Verefoo to allocate and configure Parental Control Systems as Network Security Functions (NSFs) in an automatic and optimal manner, ensuring the fulfillment of the associated Parental Control Requirements.

One of the key contributions of this thesis is the introduction of a new type of Network Security Requirements, namely the Parental Control Requirements. These requirements enable the specification of constraints regarding the suitable traffic for a particular user or device, as well as constraints related to usage time.

The incorporation of novel EndPoints into the Verefoo framework, such as Internet_Endhost, Users, and diverse devices with varying Internet capabilities, has been a noteworthy advancement. The modeling of these new elements has played a vital role, enabling Verefoo to effectively address scenarios that deviate from traditional ones and extend its applicability to the domain of home networks. This expansion of the framework's capabilities has facilitated the handling of diverse scenarios and broadened its scope beyond conventional contexts.

Furthermore, the ADP module has been enhanced to automatically allocate and configure new Network Security Functions that fulfill the newly introduced Parental Control Requirements. Two types of parental control systems have been modeled: a basic system allowing personalized traffic control based on predefined filter levels for each user, and a more advanced system introducing device-level restrictions, such as setting a maximum daily usage time. Notably, the framework also allows for the direct allocation of these functions within devices with full Internet capabilities.

A significant portion of this work focuses on formulating the MaxSMT problem, which takes as input the newly developed Network Security Requirements, the

Service Graph or Allocation Graph containing the newly introduced elements, and the Parental Control Systems. The output of this formulation is a new Service Graph that incorporates the allocated and configured Parental Control Systems in an automated and optimized manner.

Expanding the applicability domain of the Verefoo framework in this thesis opens up possibilities for future research in the field of home networks. Further extensions of the ADP module could introduce additional functionality. For instance, enhancing the capabilities of the developed Parental Control Systems to align with those offered by commercial systems in the market could be explored. This could include the ability to refer to specific Internet categories from a predefined set, rather than just an *Internet_Endhost*, in a Parental Control Requirement. Another avenue could involve addressing a specific type of parental control, such as Smart TVs, which would require modeling new elements. Additionally, implementing new NSFs, such as an Access Control System capable of managing multi-user and multi-device environments, could be considered.

Overall, this thesis not only enhances the Verefoo framework by enabling the automatic allocation and configuration of Parental Control Systems in home networks, but it also presents opportunities for further expansion of the current solution. This allows for the exploration of additional functionality and the development of a more comprehensive solution for home network security.

Bibliography

[1] Vishwajeet Bhosale, Lorenzo De Carli, and Indrakshi Ray. «Detection of Anomalous User Activity for Home IoT Devices.» In: *IoTBDS*. 2021, pp. 309–314 (cit. on p. 7).

[2] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. «IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT». In: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. 2017, pp. 2177–2184. DOI: `10.1109/ICDCS.2017.283` (cit. on p. 7).

[3] Curtis R Taylor, Tian Guo, Craig A Shue, and Mohamed E Najd. «On the feasibility of cloud-based SDN controllers for residential networks». In: *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE. 2017, pp. 1–6 (cit. on p. 8).

[4] Andy Dolan, Indrakshi Ray, and Suryadipta Majumdar. «Proactively extracting IoT device capabilities: An application to smart homes». In: *Data and Applications Security and Privacy XXXIV: 34th Annual IFIP WG 11.3 Conference, DBSec 2020, Regensburg, Germany, June 25–26, 2020, Proceedings 34*. Springer. 2020, pp. 42–63 (cit. on p. 9).

[5] Mahdi Nobakht, Craig Russell, Wen Hu, and Aruna Seneviratne. «IoT-NetSec: Policy-Based IoT Network Security Using OpenFlow». In: *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 2019, pp. 955–960. DOI: `10.1109/PERCOMW.2019.8730724` (cit. on p. 10).

[6] Nirnay Ghosh, Saket Chandra, Vinay Sachidananda, and Yuval Elovici. «SoftAuthZ: A Context-Aware, Behavior-Based Authorization Framework for Home IoT». In: *IEEE Internet of Things Journal* 6.6 (2019), pp. 10773–10785. DOI: `10.1109/JIOT.2019.2941767` (cit. on p. 11).

[7] William Jang, Adil Chhabra, and Aarathi Prasad. «Enabling multi-user controls in smart home devices». In: *Proceedings of the 2017 workshop on internet of things security and privacy*. 2017, pp. 49–54 (cit. on p. 11).

[8] Amit Kumar Sikder, Leonardo Babun, Z Berkay Celik, Hidayet Aksu, Patrick McDaniel, Engin Kirda, and A Selcuk Uluagac. «Who's controlling my device? Multi-user multi-device-aware access control system for shared smart home environment». In: *ACM Transactions on Internet of Things* 3.4 (2022), pp. 1–39 (cit. on pp. 12, 26).

[9] Pardis Emami-Naeini, Henry Dixon, Yuvraj Agarwal, and Lorrie Faith Cranor. «Exploring how privacy and security factor into IoT device purchase behavior». In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–12 (cit. on p. 13).

[10] Serena Zheng, Noah Apthorpe, Marshini Chetty, and Nick Feamster. «User perceptions of smart home IoT privacy». In: *Proceedings of the ACM on human-computer interaction* 2.CSCW (2018), pp. 1–20 (cit. on p. 13).

[11] Julie Haney, Susanne M Furman, Mary Theofanos, and Yasemin Acar Fahl. «Perceptions of smart home privacy and security responsibility, concerns, and mitigations». In: (2019) (cit. on p. 14).

[12] Oksana Kulyk, Kristina Milanovic, and Jeremy Pitt. «Does my smart device provider care about my privacy? Investigating trust factors and user attitudes in IoT systems». In: *Proceedings of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society*. 2020, pp. 1–12 (cit. on p. 14).

[13] Hosub Lee and Alfred Kobsa. «Understanding user privacy in Internet of Things environments». In: *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE. 2016, pp. 407–412 (cit. on p. 14).

[14] Michael Fagan, Mary Yang, Allen Tan, Lora Randolph, and Karen Scarfone. *Security review of consumer home Internet of Things (IoT) products*. Tech. rep. National Institute of Standards and Technology, 2019 (cit. on p. 14).

[15] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, Fulvio Valenza, and Jalolliddin Yusupov. «Towards a fully automated and optimized network security functions orchestration». In: *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. IEEE. 2019, pp. 1–7 (cit. on pp. 15, 40).

[16] Daniele Bringhenti, Riccardo Sisto, and Fulvio Valenza. «A novel abstraction for security configuration in virtual networks». In: *Computer Networks* 228 (2023), p. 109745 (cit. on p. 15).

[17] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, and Fulvio Valenza. «A novel approach for security function graph configuration and deployment». In: *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE. 2021, pp. 457–463 (cit. on p. 16).

[18] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, and Fulvio Valenza. «Short paper: Automatic configuration for an optimal channel protection in virtualized networks». In: *Proceedings of the 2nd Workshop on Cyber-Security Arms Race*. 2020, pp. 25–30 (cit. on p. 16).

[19] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, Fulvio Valenza, and Jalolliddin Yusupov. «Automated firewall configuration in virtual networks». In: *IEEE Transactions on Dependable and Secure Computing* 20.2 (2022), pp. 1559–1576 (cit. on p. 17).

[20] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, Fulvio Valenza, and Jalolliddin Yusupov. «Introducing programmability and automation in the synthesis of virtual firewall rules». In: *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE. 2020, pp. 473–478 (cit. on p. 18).

[21] Daniele Bringhenti, Jalolliddin Yusupov, Alejandro Molina Zarca, Fulvio Valenza, Riccardo Sisto, Jorge Bernal Bernabe, and Antonio Skarmeta. «Automatic, verifiable and optimized policy-based security enforcement for SDN-aware IoT networks». In: *Computer Networks* 213 (2022), p. 109123 (cit. on p. 18).

[22] Fulvio Valenza, Erisa Karafili, Rodrigo Vieira Steiner, and Emil C Lupu. «A hybrid threat model for smart systems». In: *IEEE Transactions on Dependable and Secure Computing* (2022) (cit. on p. 19).

[23] Daniele Bringhenti, Fulvio Valenza, and Cataldo Basile. «Toward cybersecurity personalization in smart homes». In: *IEEE Security & Privacy* 20.1 (2021), pp. 45–53 (cit. on p. 20).

[24] Gudrun Jonsdottir, Daniel Wood, and Rohan Doshi. «IoT network monitor». In: *2017 IEEE MIT Undergraduate Research Technology Conference (URTC)*. 2017, pp. 1–5. DOI: `10.1109/URTC.2017.8284179` (cit. on p. 27).

[25] Anna Maria Mandalari, Daniel J Dubois, Roman Kolcun, Muhammad Talha Paracha, Hamed Haddadi, and David Choffnes. «Blocking without breaking: Identification and mitigation of non-essential iot traffic». In: *arXiv preprint arXiv:2105.05162* (2021) (cit. on p. 28).

[26] Fulvio Valenza and Antonio Lioy. «User-oriented Network Security Policy Specification.» In: *J. Internet Serv. Inf. Secur.* 8.2 (2018), pp. 33–47 (cit. on p. 30).

[27] Joel M. Halpern and Carlos Pignataro. *Service Function Chaining (SFC) Architecture*. RFC 7665. Oct. 2015. DOI: `10.17487/RFC7665`. URL: `https://www.rfc-editor.org/info/rfc7665` (cit. on p. 33).