ICSC: Intent-Based Closed-Loop Security Control System for Cloud-Based Security Services

Patrick Lingga, Jaehoon (Paul) Jeong, and Linda Dunbar

The authors present novel contributions to intent-based networking, emphasizing intent fulfillment and intent assurance within network security.

Abstract

This article proposes an intent-based closed-loop security control (ICSC) system for intelligent and effective security service management. Recent advancements in computer network technologies have led to the emergence of intent-based networking (IBN), significantly improving network security management. This article presents novel contributions to IBN, emphasizing intent fulfillment and intent assurance within network security. The proposed approach in this article utilizes a standardized framework called interface to network security functions (I2NSF) with standardized communication protocols and data models, allowing the deployment of security policies across multi-vendor environments. Furthermore, the existing security policy translator for an intent is extended to support dynamic translation, enabling the immediate integration of new security solutions into the network. An analytics component with machine learning is also introduced for continuous network monitoring, proactively identifying anomalies, and triggering automated threat mitigation. Additionally, the ICSC system's performance is assessed in various scenarios and configurations, providing a thorough understanding of its strengths and limitations. Thus, it is shown that the ICSC system can establish robust and adaptive network security management.

Introduction

The evolution of technology has revolutionized computer network design and management. Traditional networks relied on specialized, proprietary hardware devices like routers, switches, and firewalls, leading to inflexible and expensive infrastructures. To address these limitations, new networking paradigms called Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) have emerged in attempts to provide a more agile, efficient, and cost-effective networking infrastructure. However, the complexity and dynamism of these networks pose new challenges for network administrators in terms of the effective management and orchestration of networks. To address these challenges, a new paradigm called Intent-Based Networking (IBN) has emerged.

IBN takes the concept of SDN and NFV a step further by adding a layer of abstraction that allows network administrators to define the intent or objectives of the network, rather than specifying the specific configuration of network devices. This approach simplifies network management by automating the translation of a high-level intent into low-level network configurations. Thus, allowing network administrators to focus on the desired outcome of the network rather than the configuration. As defined in RFC 9315 in [1], the concept of IBN encompasses two crucial functionalities:

- *Intent Fulfillment:* This functionality enables users to communicate their network intent and perform the necessary actions to achieve it. It includes optimizing outcomes, orchestrating configuration operations, and translating a high-level network intent into lower-level network configurations.
- *Intent Assurance:* This functionality allows users to validate and monitor network alignment with the desired intent. It provides feedback to optimize fulfillment functions, assess effectiveness, and address network behavior that gradually deviates from the desired intent over time.

The concept of IBN has been developed by several works discussed in [2], each with different strategies to achieve the solution. These works have contributed significantly to the advancement of IBN, demonstrating various strategies and implementations. However, most of the works are implemented in a very specific way for their respective system. This poses a limitation when attempting to scale up across different environments created by various vendors.

To overcome this issue, the Interface to Network Security Function (I2NSF) Framework [3] provides standardized software interfaces and data models designed specifically to support different types of Network Security Functions (NSFs). To apply IBN to network security, J. Kim *et al*. [4] proposed Intent-Based Cloud Services (IBCS) that enable an automated and virtualized security system with easy and flexible security services by utilizing the I2NSF Framework. Furthermore, C. Basile *et al*. [5] provide automated enforcement of security policies with optimal selection of NSFs.

While works in [4] and [5] represent a notable attempt to automate and virtualize security systems, some inherent challenges and shortcomings need to be addressed in the overall current state of the art.

Digital Object Identifier: 10.1109/MCOM.001.2400022

Patrick Lingga and Jaehoon Jeong (corresponding author) are with Sungkyunkwan University, Republic of Korea; Linda Dunbar is with Futurewei Technologies, USA.

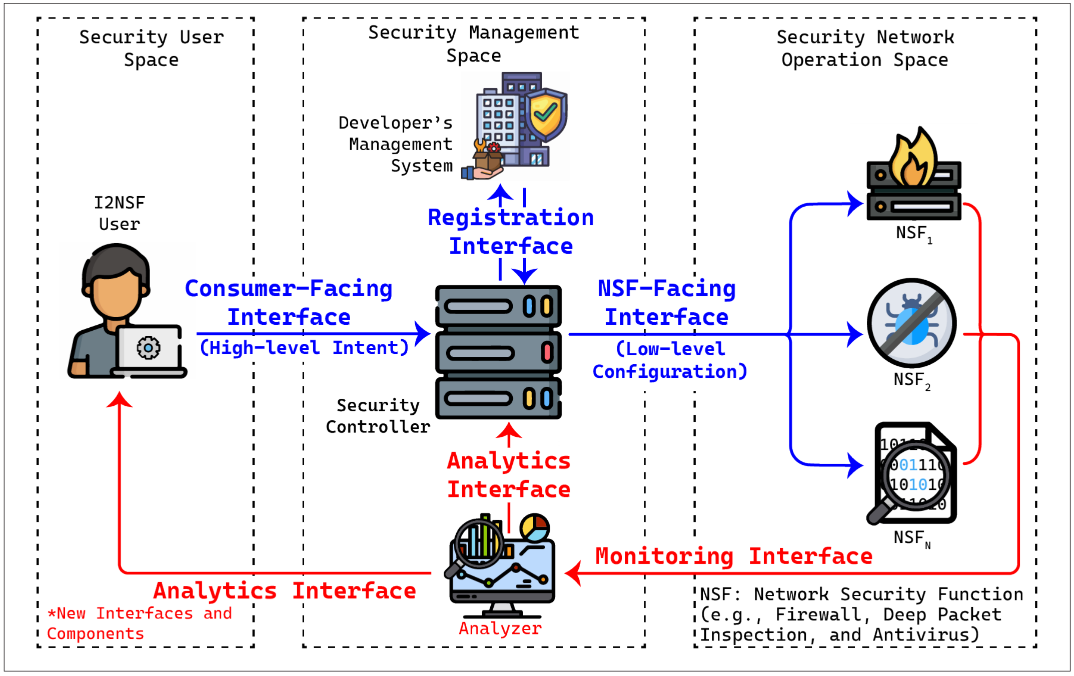
Security User Space
Security Management Space
Security Network Operation Space
Developer's Management System
I2NSF User
Registration Interface
Consumer-Facing Interface
(High-level Intent)
NSF-Facing Interface
(Low-level Configuration)
Security Controller
NSF$_1$
NSF$_2$
NSF$_N$
Analytics Interface
Monitoring Interface
Analyzer
Analytics Interface
*New Interfaces and Components
NSF: Network Security Function (e.g., Firewall, Deep Packet Inspection, and Antivirus)

FIGURE 1. A new logical architecture view of I2NSF framework [4].

Vendor-Specific Implementations: Existing IBN solutions are often tailored to specific systems, limiting their interoperability and scalability across different vendor environments. For instance, a solution optimized for one vendor's hardware may not function seamlessly with another's.

Static Security Policies: These approaches rely heavily on predefined security policies, which require continuous manual updates to adapt to emerging threats. This reliance makes it difficult to swiftly respond to the rapidly evolving cyber landscape.

Limited Real-Time Adaptation: Current solutions lack robust mechanisms for real-time network behavior analysis and adaptive threat response. The absence of these capabilities to analyze network behavior in real time means missed opportunities for identifying and mitigating emerging threats before they can exploit vulnerabilities.

This article aims to advance the state of the art by addressing the critical limitations of current IBN solutions. Hence, this article makes the following contributions to solve the limitations and provide both Intent Fulfillment and Intent Assurance.

An Open Standard Security Framework: The proposed approach adopts a standard I2NSF framework with standardized communication protocols and data models, which enables the deployment of unified security policies effectively across multi-vendor environments, reducing complexity and improving overall security posture. The standard framework enables interoperability and scalability to deploy security strategies effectively across different vendor environments.

Adaptive Security Policies: The proposed approach extends the existing security policy translator to enable dynamic translation with the ever-evolving network security. It allows new security solutions to be involved immediately within the network by allowing autonomous updates within the security policies and data models.

Real-Time Adaptive Response: The proposed approach integrates a new analytics component that utilizes machine learning algorithms to enable real-time network monitoring and adaptive threat response. It continuously collects and analyzes network traffic patterns and behaviors to detect anomalies and potential threats proactively, triggering automated responses to mitigate risks swiftly and effectively.

Evaluation of the Proposed System for Viability: The proposed approach is verified by evaluating the overall performance. A better understanding of the proposed system's strengths and limitations can be obtained by testing the system in various scenarios and configurations.

To make such contributions, we extended the architecture described by J. Kim *et al*. [4] by evolving the framework from a passive observer to an active guardian with the addition of a standardized Monitoring Interface [6] and a proposed Analytics Interface [7]. The new standardized interfaces allow the collection and analysis of network information to detect anomalies or deviations from the intended security policies. Additionally, the existing security policy translator is extended to allow dynamic YANG data models. The subsequent sections will explain the details of the architecture, design, implementation, and evaluation of the proposed Intent-Based Closed-Loop Security Control (ICSC) system.

Architecture of Intent-Driven System

This section provides an overview of the system that describes how it is organized to help the reader reach a common understanding of the structure and functionality of the proposed system design. In this article, the previous system described in Intent-Based Cloud Services (IBCS) [4] is extended with a new component and new interfaces, as shown in the logical view in Fig. 1. The main components are as follows:

- *I2NSF User:* A user interacts with the I2NSF framework to manage network security by expressing their security requirements in terms of a high-level policy (i.e., intent).

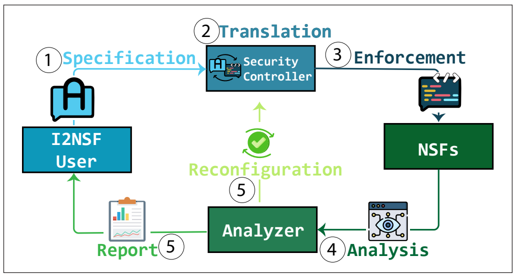

FIGURE 2. Managing an intent with a closed-loop security control in the I2NSF framework.

The user can be a system administrator or authorized personnel responsible for network security management.

- *Security Controller:* A key component of the I2NSF framework that is responsible for orchestrating the NSFs. It acts as a central entity that receives a high-level policy and translates it into low-level configurations that can be understood and implemented by the NSFs.
- *Developer's Management System (DMS):* A component of the I2NSF framework that caters to the needs of the developers or vendors of the NSFs. This system provides a platform for developers to register and advertise their NSFs' capabilities.
- *Network Security Function (NSF):* A virtualized instance that provides network protection and enforces the defined security policies. NSFs can include various security functions such as firewalls, intrusion prevention systems (IPS), and antivirus solutions.
- *Analyzer:* A component dedicated to collecting and analyzing security-related data and events. It performs traffic analysis and the generation of reports or alerts. It helps to assess the effectiveness of security policies, detect potential threats, and make informed decisions about network security management.

Communication between these components is built with the standardized interfaces by the I2NSF Working Group in the Internet Engineering Task Force (IETF). The following interfaces facilitate the I2NSF Framework.

Consumer-Facing Interface (CFI): The interface through which users interact with the Security Controller of the I2NSF Framework to express their security requirements and policies. The CFI abstracts the underlying complexity of the NSFs and provides a simplified view that users can use to interact with the I2NSF framework [8].

Registration Interface (RI): The interface that allows the DMS to register NSFs' capabilities [9] with the Security Controller. The Registration Interface can provide information such as their capabilities, configuration requirements, NSF access information, and other relevant metadata [10].

NSF-Facing Interface (NFI): The interface that allows the Security Controller to interact with individual NSFs. This interface provides the necessary mechanisms for the configuration and control of the NSFs, thus allowing the I2NSF framework to enforce the desired security policies to the appropriate NSFs [11].

Monitoring Interface (MI): The interface that enables the monitoring of status, performance, and security events related to the NSFs. It enables the implementation of an analyzer with real-time visibility in the operation of the NSFs and the overall security posture of the network [6].

Analytics Interface (AI): The interface that facilitates the analysis of security-related data for advanced security enhancement purposes. This Interface lets the security controller receive policy reconfigurations for the enhanced security services and allows users to gain feedback information for understanding and handling NSFs (e.g., overload and malfunction) [7].

Enabling Intent-Driven Network Management with Intent Assurance

In an Intent-Driven approach, administrators or users express their desired intent or objectives in a higher-level language that is more easily readable by humans. The specified intent is automatically translated into understandable configurations, abstracting complexity and technical details from the user. Intent-Based Cloud Services (IBCS) were proposed by J. Kim *et al.* [4], who provided a detailed explanation of the translation approach.

The extended I2NSF Framework enables the procedure of intent-driven network management by creating a closed-loop management system, as shown in Fig. 2. The whole procedure of intent-driven network management can be divided into several stages:

- Intent specification
- Intent translation
- Intent enforcement
- Intent analysis (i.e., monitoring and observation)
- Intent adaptation (i.e., reconfiguration and report).

Intent Specification

In this stage, the I2NSF User expresses their intent or desired outcome in terms of high-level security requirements. The user specifies their security requirements by using a user-friendly interface, which simplifies the process of entering inputs related to network security, making it accessible to users who may not have expertise in network technologies or security protocols. Then, it will construct a high-level security policy in an XML form. An example of a high-level security policy can be seen in the "high-level-policy.xml" available at https://codeocean.com/capsule/0816676/tree/v2. In this example, the I2NSF User wants to let their clients access the web servers.

With this method, the underlying complexities of the network (i.e., types, resources, and topology) are hidden from the users. This method shields users from the intricacies of network protocols, device configurations, capabilities, and security mechanisms. Hence, rather than having to deal with technical details and complex configurations, the users can focus on articulating their specific security needs and objectives using familiar terminology and high-level concepts.

In the I2NSF Framework, intent specification is provided by the CFI YANG data model [8] that specifies the data types and encoding schemes so that high-level security requirements can be delivered to the Security Controller in the form of security policies. The YANG data model provides a complete method of security management with high-level security policies.

However, this concept heavily relies on the next stage (i.e., intent translation) to properly fulfill the necessary requests of the user. Without intent translation, the expressed high-level security policies would remain as abstract concepts that cannot be implemented in the security infrastructure.

Intent Translation

Intent translation is a crucial intermediary stage that transforms the user's security requirements into the actual configuration for the NSFs. This process ensures that the security requirements are accurately captured and enforced within the network. The example high-level security policy will be translated into the corresponding low-level configuration shown in the "low-level-policy.xml" file available at https://codeocean.com/capsule/0816676/tree/v2. With the I2NSF framework, the translator can generate the necessary low-level configuration based on the NSFs' capabilities [9]. The I2NSF framework allows the NSFs of multiple vendors to be used by the syntax for this low-level configuration, which is standardized with the NFI YANG data model [11].

This translation process has been addressed in a previous work by J. Kim, *et al*. [4]. The proposed security policy translator performs intent translation using three important components: data extractor, data converter, and policy generator. The data extractor extracts and verifies a high-level security policy using the deterministic finite automaton (DFA) concept. The data converter transforms the extracted information into specific low-level configuration data for the appropriate NSFs. The policy generator produces the actual low-level configurations in XML form according to the NFI YANG data model [11].

This previous work did not consider an approach to perform the mapping between the high-level and low-level YANG data models and how the translation service works over time when the data models are updated. The previous approach lacks a method to provide a fully automated translation. Hence, as shown in Fig. 3, this article improves the translator by adding a component that handles the mapping between high-level and low-level data models automatically, that is, a data model mapper [12].

The data model mapper is designed specifically for the I2NSF Framework. It leverages the design similarity between the high-level and low-level YANG data models. As the data structure of a YANG data model is a tree data structure, the two YANG data models can be compared on each linear node (i.e., an attribute entity from the root to the leaf in a YANG tree) to find the closest similarity. The similarity is calculated using a tree edit distance algorithm called Zhang-Shasha (ZSS) algorithm. In this algorithm, linear nodes from the high-level YANG data model are systematically compared to every linear node in the low-level YANG data model.

Through this comparison, the mapper identifies the most similar linear nodes (i.e., the nodes requiring the minimum number of operations) which are projected as the mapping nodes. For example, the high-level nodes of "i2nsf-cfi-policy/condition/firewall/destination" will have high similarity with the low-level nodes of "i2nsf-security-policy/condition/ipv4/destination-ipv4-network" and "i2nsf-security-policy/condition/tcp/destination-port-numbers." Hence, the two linear nodes are projected as the mapping nodes. The result is saved to the NSF database so that it can be used by the data converter to correctly transform security requirements into security policies.

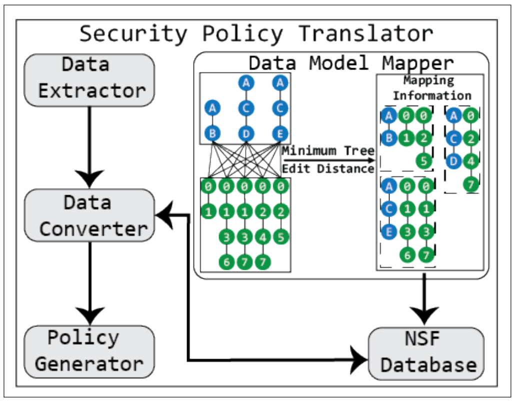

FIGURE 3. Extended architecture of security policy translator for I2NSF framework.

Furthermore, the Policy Generator is updated by utilizing a PyangBind-based policy generation [12] to automatically generate low-level security policies and ensure that they comply with the NFI YANG data model [11]. Note that PyangBind is open source, available at https://github.com/robshakir/pyangbind. The integration of PyangBind allows data models to be dynamic and ensures responsiveness to evolving network threats.

The addition of the data model mapper and policy generator allows the translation process to become fully automated and effective whenever the data models experience updates or extensions. By dynamically adapting to evolving data models, the improved security policy translator can help maintain a robust and up-to-date network security infrastructure. Note that the detailed explanation of the latest security policy translator can be seen in our work in [12].

Intent Enforcement

After the translation, the security policies are enforced on the NSFs accordingly. The Security Controller communicates with the NSFs through the standardized NFI [11], allowing for the configuration of NSFs from multiple different vendors. The NSFs start enforcing the security policies defined in the translated intent. This includes firewalls, URL filtering, intrusion detection and prevention systems (IDS/IPS), traffic monitoring, and antivirus solutions. The NSFs apply these policies to inspect and control network traffic, thus ensuring compliance with the user's intent and providing the desired security services.

During intent enforcement, the Security Controller ensures synchronization and coordination among the NSFs with Service Function Chaining (SFC). As shown in Fig. 4, the framework ensures that the NSFs operate collectively and consistently to enforce the intended security policies across the network. For example, network flows with HTTP and HTTPS protocols must be directed through the Firewall first and then through the Web Filter for further inspection, while other protocols will only be directed through the Firewall. This is done with a flow classifier to classify the type of network flows that enter or exit the net-

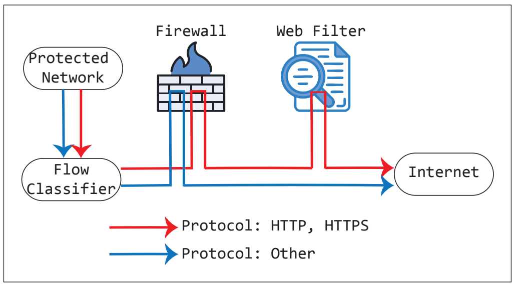

FIGURE 4. Enforcement of service function chaining.

work. It then activates SFC by pairing the ports according to the translated intent.

The Security Controller configures SFC automatically through the deployment of the SFC configurations to the NSFs involved in the service chain and the routing configuration with the flow classifier by directing packets to the appropriate NSFs in the previously defined order. This configuration deployment ensures that the NSFs are aware of both their placement within the SFC and the sequence in which they need to be traversed.

Intent Analysis

Once the intent is enforced, the framework performs an intent analysis (i.e., monitoring and observation) to ensure that the implemented security services align with the user's high-level security requirements. This stage involves monitoring data from the NSFs in real time using the Monitoring Interface YANG data model [6]. This data includes network traffic, system logs, and security events to provide visibility into the potential vulnerabilities, attack patterns, and the impact of security measures on the overall network.

To prevent data overload, the Analyzer employs a subscription-based model. This model ensures that only specific types of data relevant to the intent analysis are collected and analyzed, thus reducing the overall data size and optimizing resource utilization. In this subscription-based model, the Analyzer initiates an agreement with the NSFs to define the types and categories of data that need to be collected. Moreover, the subscription-based model allows for flexibility and scalability in data collection. The agreements between the Analyzer and NSFs can be dynamically adjusted based on evolving security needs.

The intent analysis also enables the detection of security events and anomalies within the network to identify potential threats. The analysis also helps to evaluate the inefficiencies in the implemented security measures based on the performance metrics. To enhance the analysis results, the Analyzer utilizes machine learning techniques to improve the accuracy and efficiency of an intent. These results are expected to help provide appropriate response measures to mitigate the identified risks and prevent further security breaches.

As an example, in the Intent Analysis stage, the Analyzer monitors incoming network traffic in real time. It collects monitoring data from various network devices to analyze traffic patterns and identify potential Distributed Denial of Service (DDoS) attacks. At one point, the Analyzer detects a sudden surge in incoming traffic to a specific web server's IP address. It analyzes the traffic patterns, such as the source IP addresses, packet sizes, and request rates, to determine if the surge is consistent with a DDoS attack.

Using machine learning algorithms, for example, decision tree algorithm, the Analyzer correlates these traffic patterns with known DDoS attack signatures to assess the likelihood of an ongoing attack. It may also analyze network flow data to identify anomalous behaviors, such as a large number of connections from a single IP address or unusual packet fragmentation patterns.

Intent Adaptation

Based on the results obtained in the previous stage, the I2NSF framework may initiate adaptation actions if any issues are identified. Another important role of the Analyzer is to provide feedback about necessary adjustments that must be made. It is important to ensure that the network security measures remain aligned with the desired intent and address the evolving security challenges.

Two models are providing intent adaptation in the I2NSF Framework: reconfiguration and report [7]. Reconfiguration involves rearranging security policies in different forms to mitigate threats and improve the quality of network security. Reconfiguration aims to adapt the network security measures to align with the desired intent and address any identified issues. By reconfiguring the security policies based on the analysis results, the I2NSF framework can effectively respond to evolving security threats and changing network conditions, thus ensuring the continuous alignment of security measures with the desired intent.

A report is the feedback information given to the I2NSF User to handle risks that cannot be handled by reconfiguration, for example, system resource malfunction or threats exceeding the capabilities of existing services. It highlights specific areas that require further action from the user to handle the identified risks, thus enabling the user to make informed decisions [7].

In the previous example of a DDoS threat detected in the Intent Analysis, the Analyzer must create a reconfiguration to try and stop the threat. A decision-making machine learning algorithm (e.g., decision tree algorithm) is implemented for the reconfiguration. An action space such as adjusting firewall rules, implementing rate limiting, or rerouting traffic through a DDoS protection service. The system observes itself and verifies whether the system successfully reduces the impact of the attack or whether the attack persists. An example of a reconfiguration security policy can be seen in the "reconfiguration.xml" available at https://codeocean.com/capsule/0816676/tree/v2.

If the impact of the attack persists, the Analyzer will try different actions in the action space by removing the previous reconfiguration action. Changing the policies every time may cause instability in the system. Hence, the reconfiguration can be paired with the report to make sure that a human administrator can monitor the changes, and if any reconfiguration action fails, the administrator can take different actions to secure the network.

Implementation and Performance Evaluation

This section discusses the implementation and evaluation of ICSC to assess its effectiveness and efficiency with practical aspects of deploying ICSC in a specific target scenario.

Proof-of-Concept Implementation

The ICSC is implemented over the OpenStack cloud computing service. The I2NSF Framework components are built in a virtual cloud environment using an Ubuntu 20.04 cloud image. The intent specification is built with the Consumer-Facing Interface YANG data model [8] on a Node.js server environment with the React library and delivered over RESTCONF [13] to the Security Controller. The intent translation in the Security Controller is performed using Python and MongoDB, while enforcement is implemented with ConfD, running the NETCONF server [14] accepting an XML configuration based on the NSF-Facing Interface YANG data model [11]. NSFs leverage an open source IDS/IPS software called Suricata for network analysis and threat detection. For intent analysis, NSFs employ a subscription-based notification system for the Analyzer through NETCONF event notifications [15] with ConfD. The Analyzer subscribes to monitoring data via the Monitoring Interface [6] with multiple NSFs, utilizing a simple machine learning technique (e.g., decision tree algorithm) for Intent Assurance in the I2NSF Framework.

In this Proof of Concept (PoC), all components of the I2NSF Framework play a role in defending the protected network through continuous intent analysis and intent adaptation. In this scenario, normal clients are accessing a protected network that can cope with a DDoS attack. Each component has distinct responsibilities for mitigating the attack as follows. The I2NSF User monitors the system and assists it if it is necessary to handle attacks when alerts are received. The Security Controller receives reconfigurations and dispatches them to the NSFs, sometimes requesting additional NSFs from the DMS to aid in attack mitigation. The DMS may receive requests to deploy new NSFs. NSFs continue working to mitigate attacks based on reconfigurations and to collect network information. Finally, the Analyzer remains vigilant, observing the network to assess changes in the threat.

Note that the implementation of the I2NSF system has been proved through several IETF hackathon projects. This implementation involves the final I2NSF YANG data models that are approved as RFCs by the IETF. The source code and demonstration video clip of the implementation are available at https://github.com/jaehoonpauljeong/I2NSF-Closed-Loop-Security-Control and https://youtu.be/OWNJbF7wGgs, respectively.

Performance Evaluation

To evaluate our proof-of-concept implementation, we performed a simple simulation with various numbers of clients to acquire its impact on the Mean Time to Detect (MTTD) and the Mean Time to Respond (MTTR) as performance metrics. MTTD is the average time it takes to discover a potential security issue, while MTTR is the average time it takes to completely eliminate the problem after it has been discovered.

Figure 5 shows the MTTD and MTTR of both ICSC and a manual operation. The MTTD and MTTR of the manual operation increase exponentially with the involvement of more clients. A constant number as a common duration is used to represent a baseline time that is required for a human operator's operations, independent of the number of clients. This number includes factors such as the time it takes for a human operator to recognize a security threat, access relevant information, and take appropriate actions. The exponential increase is due to the higher workload and complexity of manually detecting and obtaining the necessary information when more clients are involved.

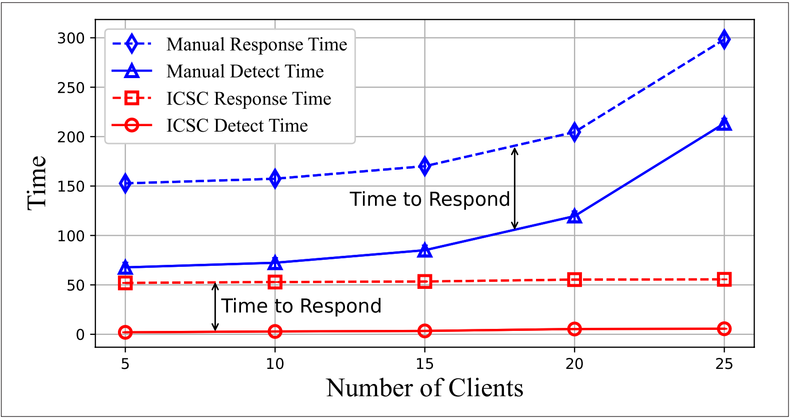

FIGURE 5. The performance of ICSC vs. manual operation.

It is important to note that the MTTD and MTTR values for the manual operation are derived from a controlled Proof-of-Concept environment. By conducting experiments in the controlled environment, specific variables can be isolated and systematically measured for their impact on MTTD and MTTR. This controlled setting eliminates external variables that could otherwise affect those MTTD and MTTR values, providing a clear benchmark for evaluating the effectiveness of the ICSC system.

On the other hand, as shown in Fig. 5, the MTTD and MTTR of ICSC are not significantly impacted by the number of clients. Furthermore, ICSC enables faster detection and reaction time compared to manual operation. Moreover, the automatic adaptation of ICSC allows for continuous surveillance of the network, regardless of the availability of an operator.

Research Challenges

The idea of ICSC has been proposed herein with a proof-of-concept implementation. ICSC needs to be polished before it can be deployed in the industry. The following research challenges must be addressed to enhance its effectiveness:

Adaptive Policy Enforcement: It is important to ensure the effectiveness of security policy enforcement in a dynamic and evolving network environment. This includes addressing challenges that are related to policy conflicts, policy updates, and consistent enforcement across different NSFs. The framework must ensure that the policies are well-adapted with no conflict between any of them.

Real-time Threat Detection: Detecting threats in a real-world network is a large-scale research task that requires advanced techniques for real-time threat detection and analysis to identify and respond to emerging security threats. This involves leveraging machine learning, anomaly detection, and behavioral analysis to detect and mitigate both known and unknown threats in real time.

ICSC provides two IBN functionalities: Intent Fulfilment and Intent Assurance. ICSC expands upon IBCS by adding a data model mapper into the security policy translator to enable fully automated and effective translation to achieve Intent Fulfilment. ICSC also provides Intent Assurance by utilizing a closed-loop security control system to align the security policies that have been deployed with the desired intent based on an analysis of numerous metrics obtained from the monitoring process.

Proactive Defense: The feedback delivered to the Security Controller to prevent concurring attacks must be developed with adaptability and responsiveness in mind. This involves designing feedback mechanisms that not only provide timely and accurate information about ongoing attacks but also dynamically adapt to the changing threat landscape. This requires continuous refinement of the feedback mechanisms to proactively stay ahead of emerging threats.

Conclusion

ICSC provides two IBN functionalities: Intent Fulfilment and Intent Assurance. ICSC expands upon IBCS by adding a data model mapper into the security policy translator to enable fully automated and effective translation to achieve Intent Fulfilment. ICSC also provides Intent Assurance by utilizing a closed-loop security control system to align the security policies that have been deployed with the desired intent based on an analysis of numerous metrics obtained from the monitoring process. As future work, ICSC will be enhanced for real-world network environments, including 5G core networks and Open Radio Access Network (O-RAN).

Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Ministry of Science and ICT (MSIT), South Korea (No. RS-2022-II221199 and No. RS-2022-II221015). The figures were designed using images obtained from Flaticon.com.

References

[1] A. Clemm *et al.*, "Intent-Based Networking — Concepts and Definitions," RFC 9315, Oct. 2022; available: https://www.rfc-editor.org/info/rfc9315

[2] A. Leivadeas and M. Falkner, "A Survey on Intent-Based Networking," *IEEE Commun. Surveys Tutorials*, vol. 25, no. 1, 2023, pp. 625–55.

[3] D. Lopez *et al.*, "Framework for Interface to Network Security Functions," RFC 8329, Feb. 2018; available: https://rfc-editor.org/rfc/rfc8329.txt

[4] J. T. Kim *et al.*, "IBCS: Intent-Based Cloud Services for Security Applications," *IEEE Commun. Mag.*, vol. 58, no. 4, 2020, pp. 45–51.

[5] C. Basile *et al.*, "Adding Support for Automatic Enforcement of Security Policies in NFV Networks," *IEEE/ACM Trans. Networking*, vol. 27, no. 2, 2019, pp. 707–20.

[6] J. P. Jeong *et al.*, "I2NSF NSF Monitoring Interface YANG Data Model," Internet Engineering Task Force, Internet-Draft draft-ietf-i2nsf-nsf-monitoringdata-model-20, June 2022, Work in Progress; available: https: //datatracker.ietf.org/doc/draft-ietf-i2nsf-nsf-monitoring-data-model/20/

[7] P. Lingga, J. P. Jeong, and Y. Choi, "I2NSF Analytics Interface YANG Data Model for Closed-Loop Security Control in the I2NSF Framework," Internet Engineering Task Force, Internet-Draft draft-lingga-opsawg-analytics-interface-dm-00, Aug. 2024, Work in Progress; available: https://datatracker.ietf.org/doc/ draft-lingga-opsawg-analytics-interface-dm/00/

[8] J. P. Jeong *et al.*, "I2NSF Consumer-Facing Interface YANG Data Model," Internet Engineering Task Force, Internet-Draft draft-ietf-i2nsf-consumer-facing-interface-dm-31, May 2023, Work in Progress; available: https://datatracker.ietf.org/doc/ draft-ietf-i2nsf-consumer-facing-interface-dm/31/

[9] S. Hares *et al.*, "I2NSF Capability YANG Data Model," Internet Engineering Task Force, Internet-Draft draft-ietf-i2nsf-capability-data-model-32, May 2022, Work in Progress; available: https://datatracker.ietf.org/ doc/draft-ietf-i2nsf-capability-data-model/32/

[10] S. Hyun *et al.*, "I2NSF Registration Interface YANG Data Model for NSF Capability Registration," Internet Engineering Task Force, Internet-Draft draft-ietf-i2nsf-registration-interface-dm-26, May 2023, Work in Progress; available: https://datatracker.ietf.org/doc/draft-ietf-i2nsf-registration-interface-dm/26/

[11] J. T. Kim *et al.*, "I2NSF Network Security Function-Facing Interface YANG Data Model," Internet Engineering Task Force, Internet-Draft draft-ietf-i2nsf-nsf-facinginterface-dm-29, June 2022, Work in Progress; available: https: //datatracker.ietf.org/doc/draft-ietf-i2nsf-nsf-facing-interface-dm/29/

[12] P. Lingga *et al.*, "SPT: Security Policy Translator for Network Security Functions in Cloud-Based Security Services," *IEEE Trans. Dependable and Secure Computing*, 2024, pp. 1–14.

[13] A. Bierman, M. Björklund, and K. Watsen, "RESTCONF Protocol," RFC 8040, Jan. 2017; available: https://www.rfc-editor.org/ info/rfc8040

[14] R. Enns *et al.*, "Network Configuration Protocol (NETCONF)," RFC 6241, June 2011; available: https://www.rfc-editor.org/info/rfc6241

[15] H. Trevino and S. Chisholm, "NETCONF Event Notifications," RFC 5277, July 2008; available: https://www.rfc-editor.org/info/ rfc5277

Biographies

Patrick Lingga is a Ph.D. candidate in the Department of Electrical and Computer Engineering at Sungkyunkwan University, Republic of Korea.

Jaehoon (Paul) Jeong is a professor in the Department of Computer Science and Engineering at Sungkyunkwan University, Republic of Korea.

Linda Dunbar is a distinguished engineer at Futurewei Technologies in the United States and the IETF I2NSF Working Group Chair.