POLITECNICO DI TORINO
Repository ISTITUZIONALE

A Formal Model of Security Controls' Capabilities and Its Applications to Policy Refinement and Incident Management

Original
A Formal Model of Security Controls' Capabilities and Its Applications to Policy Refinement and Incident Management / Basile, C., Gatti, G., Settanni, F.. - In: IEEE TRANSACTIONS ON NETWORKING. - ISSN 2998-4157. - 34:(2026), pp. 1659-1673. [10.1109/ton.2025.3629574]

Availability:
This version is available at: 11583/3005366 since: 2025-11-24T10:49:09Z

Publisher:
Institute of Electrical and Electronics Engineers

Published
DOI:10.1109/ton.2025.3629574

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

30 October 2026

A Formal Model of Security Controls' Capabilities and Its Applications to Policy Refinement and Incident Management

Cataldo Basile, *Member, IEEE*, Gabriele Gatti, and Francesco Settanni

***Abstract*—Enforcing security requirements in networked information systems relies on technical security controls to mitigate the risks posed by increasingly sophisticated threats. Configuring these controls is challenging; even nowadays, administrators must perform it without adequate tool support. Hence, this process is plagued by errors that result in insecure postures, security incidents, and a lack of promptness in addressing threats. This paper presents the Security Capability Model (SCM), a formal model that abstracts the features that security controls offer for enforcing security policies, which includes an Information Model that depicts the basic concepts related to rules (i.e., conditions, actions, events) and policies (i.e., conditions' evaluation, resolution strategies, default actions), and a Data Model that covers the capabilities needed to describe different types of filtering and channel protection controls. Following state-of-the-art design patterns, the model enables the generation of abstract versions of the security controls' languages and a model-driven approach for translating abstract policies into device-specific configuration settings. By validating its effectiveness in real-world scenarios, we demonstrate that SCM enables the automation of various and complex security tasks, including accurate and granular security control comparison, policy refinement, and incident response. Lastly, we present opportunities for extensions and integration with other frameworks and models.**

***Index Terms*—Formal model of security controls, model-driven security, security controls, incident response, policy-based security, intent-based security.**

I. Introduction

TECHNICAL Security Control (SC)s, such as firewalls and VPN gateways, are crucial to mitigate the cybersecurity risks that threaten the organizations' networks.

Recent networking technologies that deviate from on-premise infrastructure management, like cloud computing, have increasingly hidden these devices from users and administrators; nonetheless, they still need to be configured. Their configuration has been historically associated with human errors [1]. Indeed, SCs are complex to configure, and the complexity grows with the size and heterogeneity of the networks, the number of entities to consider (e.g., hosts, services, users), and the required rules in their configuration files [2].

Unfortunately, the tools to help users reduce human errors and support the management of SCs configurations never became mainstream due to their limitations and lack of generalization [3], [4]. Methods for automatic security configuration using refinement or anomaly analysis techniques have been investigated in recent years; they have had a minimal impact on the practice [5], [6].

Activities in complex distributed systems that rely on configuring SCs, such as incident management (e.g., reacting to attacks to limit the negative consequences) and risk mitigation (e.g., configuring SCs to prevent attacks), require promptness and human expertise. These activities can be delegated to computers but involve advanced modelling tasks of security experts' competencies and tasks. Automated solutions may be faster and, if adequately validated, less prone to errors, yet they still need to be achieved [7], [8], [9]. Hence, significant efforts are needed to investigate novel techniques to bridge the gaps in the current state of the art.

This research work is based on the hypothesis that the limitations in the current practices could also originate from the lack of formal models of SCs, which may capture the hidden semantics that allow human beings to operate. Still, it is not available to the machines. Hence, this paper aims to answer a crucial question: *Can a formal model of SCs enhance the network security landscape by mitigating some of the current issues plaguing it?*

To this purpose, this paper introduces a formal model of security control capabilities, the SCM, which represents the configurations these devices can accept based on their specific languages. In simpler words, the SCM captures what SCs can do to enforce security policies.

We have analyzed several open-source SCs to abstractly characterize their features and generalize individual controls' peculiarities. Then, we have modelled selected SCs and formally described their capabilities, by 1) characterizing their conditions by the data type they accept and the logical operators they support, and 2) enumerating all the actions, based on the effect they produce on the resources (e.g., datagrams, packets, network traffic) to which they are applied, and by 3) modelling how these can be used to form valid policies.

Received 4 April 2024; revised 4 December 2024, 21 May 2025, and 29 August 2025; accepted 27 October 2025; approved by IEEE TRANSACTIONS ON NETWORKING Editor J. S. Sun. Date of publication 14 November 2025; date of current version 8 January 2026. This work was supported in part by the Project Security and Rights in CyberSpace (SERICS) through National Recovery and Resilience Plan, Ministero dell'Università e della Ricerca Program, funded by the Next Generation EU and in part by the Smart Networks and Services Joint Undertaking (SNS JU) through European Union's Horizon Europe Research and Innovation Program, iTrust6G Project under Agreement 101139198. *(Corresponding author: Cataldo Basile.)*

The authors are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Torino, Italy (e-mail: Cataldo.Basile@polito.it; Gabriele.Gatti@polito.it; Francesco.Settanni@polito.it).

Digital Object Identifier 10.1109/TON.2025.3629574

© 2025 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

The SCM enables several management tasks, e.g., comparing different SCs to determine when they can enforce the same actions and when they have conditions in common. It supports informed decisions when selecting a SC that could implement a specific policy, comparing two products to determine the best one to use, and migrating from a control to a different one from another vendor.

The SCM also enables security-sensitive tasks. Via a set of association classes, the model determines how abstract capabilities can be converted into low-level configuration settings understandable by a target SC. This feature enables translations between different levels of abstraction (i.e., policy refinement), as well as among different SCs (e.g., two firewalls from distinct vendors).

The practical outcomes of this research include the SCM,[1] a UML model from which XML representations can be obtained, a Catalogue with different SCs from various categories, a set of web services for managing models and querying catalogues, and a policy translation engine.

The usefulness of the SCM lies in its ability to decompose security controls into their features, providing an interface for ensuring vendor-neutral security policy management. It supports daily tasks, it can be used to help administrators when (manually) translating high-level security requirements, e.g., stemming from regulations or internal security policies, into actual configurations, or to lock and isolate threats when reacting to security incidents. In this regard, the SCM has been validated with three relevant real-world scenarios:

- *Security management tasks*, the first scenario involved the evaluation of how the formal descriptions in the security capability model can reduce vendor lock-in;
- *Automatic configuration of software networks*, the second scenario dealt with developing a tool for refining security policies. The tool uses capabilities to identify suitable SCs for policy enforcement and employs a translator to generate the appropriate configurations.
- *Automated remediation*, lastly, the third scenario tested the ability to remediate security incidents with playbooks that use capabilities to determine the SCs to reconfigure or how to modify the network level when playbooks are not enforceable (e.g., adding more SCs to mitigate risks).

Our results prove that in an era where everyone seems to want to offload their tasks to AI-based chatbots, developing formal models is still a crucial and complementary activity (yet complex and time-consuming), especially in areas like cybersecurity, where even minor uncertainty or inaccuracies cannot be tolerated.

The rest of the paper is structured as follows. Section II presents the three relevant scenarios where the security capability model has been validated. Section III presents the SCM, the design decisions, and the features it exposes. Section IV presents the tools that rely on the SCM and that served to validate the importance of a formal model of security capabilities, as described in Section V. Section VI describes past works that presented or used formal models in policy-based security. Finally, Section VII concludes and provides hints for future work.

II. Motivating Examples and Requirements

Three main areas of application of the SCM highlight the progress over state of the art and allow answering our research questions about the usefulness of a formal model of SCs.

A. Cybersecurity Market and Vendor Lock-in

The first scenario originates from the enterprise market. Security administrators and network designers employ various types of SCs for enforcing security in their daily activities. They may want to switch products for better performance or features, reduced costs, or because they lost trust in the original vendor. But vendors hinder this process.

The SCs selection process has been widely studied in the literature [11], [12]. Still, these methods leave the final evaluation of the features offered by competing vendors to experts [13], who must guess what functionalities SCs provide and compare similar SCs. However, products are not easily comparable due to buzzword-driven data sheets, incremental updates, and rebranding of old ideas marketed as radical improvements, and limited testing options [14].

Although not well-documented in the literature, empirical observations of the enterprise security landscape have shown that easy migrations between different security products are, unfortunately, far from reality. This generally leads to the adoption of the same-vendor products, even if better alternatives are available on the market. For example, changing the firewall vendor requires time-consuming translation. This translation may be assisted by migration software[2] However, it still requires manual validation; hence, it remains mostly human-dependent and prone to misconfiguration.

Therefore, our research objectives include verifying that a formal representation of the SCs using predefined security capabilities allows for

- (RO_1) a simple evaluation and comparison of the functions covered by commercial products;
- (RO_2) mapping abstract policies into to ready-to-use low-level settings exported in the SC-specific language.

The scenario validation will be presented in Section V-A.

B. Policy Refinement

Enterprise security requirements are usually maintained as abstract statements. However, statements can only be enforced after a security policy refinement process [15]. First, one has to determine the resources, i.e., the SCs, that may be used to enforce the desired requirements. Then, requirements are translated into the low-level settings for the identified SCs.

Refinement is a complex problem that security architects face regularly, and its complexity increases with the size and heterogeneity of the target infrastructure. Currently, this process is mainly conducted by network and security administrators, a human-driven approach that has been linked to

[1]This model's preliminary version has already been presented in a previous publication [10].

[2]https://live.paloaltonetworks.com/t5/expedition/ct-p/migration_tool https://www.fortinet.com/products/next-generation-firewall/forticonverter

misconfigurations and conflicts [2]. Automatic policy refinement has been proposed in the literature; however, previous works focus on vertical areas (i.e., access control [16], channel protection [17]), rather than analyzing the problem from the perspective of what the generic SCs can do and the entities they interact with. Hence, automating the refinement in real-world contexts remains an open problem.

Refinement could benefit from a formal model of capabilities that can describe precisely how individual SCs work. Indeed, the generic concepts related to configurations, such as conditions and actions, are common to all, but it is not captured by the machines. Different controls share several configuration parameters, data types, and options (e.g., IP addresses are used to filter traffic by all firewalls). Still, they have their peculiarities (e.g., conditions on protocol states are typically SC-specific) and configuration languages. For this scenario, our research objectives include:

- (RO_3) accurately modelling the context, the general concepts and the individual elements that constitute the device configurations process, including allowing the selection with high precision the SCs that can be used to enforce security policies;
- (RO_4) permitting the refinement of high-level security policies into configuration settings for selected SCs.

The scenario validation will be presented in Section V-B.

C. Incident Response and Security Orchestration

Incident response teams and security personnel, in their day-to-day activities, must ensure that systems are secured and return to their operational state after incidents, but have to face limitations from the operational practice [18], [19], [20].

Many solutions have been proposed and designed to address this scenario over the years [21], [22], which promise optimal functionality and seamless operations and are all underscored by continuous enhancements over the others. Solutions include SOAR, "solutions that combine incident response, orchestration and automation, and threat intelligence platform management capabilities in a single solution" among many other hybrids that have emerged over the years, such as XDR [22], [23], [24].

These solutions may orchestrate security processes, provide remediation measures as security playbooks [25], and interact with the available SCs (virtual and physical). In most cases, they underwent real-world validation and on-the-ground experience from security operators. Unfortunately, as of now, these solutions lack effective integration with SCs to be considered capable of remediating or mitigating incidents [21], [26].

For this scenario, our research objectives include:

- (RO_5) defining a precise mapping between the SCs' features and the operations that are used to describe and enforce remediation procedures with the playbook abstraction, to mitigate risks and benefit incident management;
- (RO_6) augment cyber threat intelligence with additional context and artefacts that are useful to characterize incidents better, allowing sharing of supplementary data enabling fine-tuned reactions, and continuous monitoring of the appropriateness of the defensive apparatus.

The scenario validation will be presented in Section V-C.

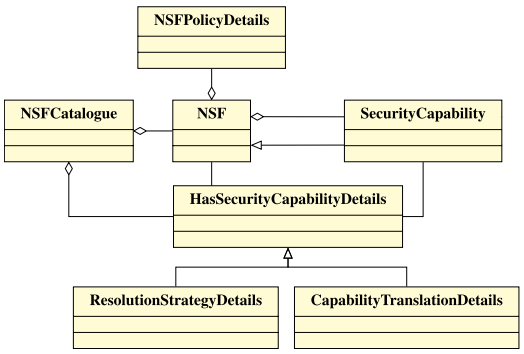


Fig. 1. The security capability information model.

III. The Capability Model

At a high level, SCs are well approximated with the IETF architecture [27], which consists of a Policy Enforcement Point that receives an object (e.g., a packet on a NIC) and asks a Policy Decision Point what to do with that object based on a security policy. In this context, the security policy is the SC's configuration specified with its specific language. Policies are (most often) formed by rules. Rules are the means to specify what to enforce (i.e., the actions), where (the objects to which these actions need to be enforced, i.e., the *conditions*), and when (the *events*). When a rule includes more than one, conditions are organised in logical formulas (*condition clause*) that allow determining when the rule matches the object. To ensure coherence, SCs support methods to determine what to enforce when no rules apply (*default actions*) or when multiple rules match the object (*resolution strategies*).

The SCM captures these relations through an Information Model (IM), presented in Fig. 1. The NSF and SecurityCapability classes abstract Network Security Function (NSF)s and generic Security Capabilities. Particularly, the IM implements the "decorator pattern" [28]. The HasSecurityCapabilityDetails class implements the association class used to "decorate" the security capabilities when they are associated with an NSF, and will be presented in more detail in Section III-C.

The aggregation of the SecurityCapability makes NSF instances containers of capabilities. Since SecurityCapability inherits from NSF, this pattern allows the inclusion of groups of specific capabilities into new NSF instances; hence, it supports creating and merging NSF description templates. To make a concrete example, the capabilities of filtering controls operating at the fourth OSI layer, like IpTables, can be obtained by extending the description of a generic packet filter with additional conditions on TCP states, MAC, bandwidth, etc.

The SecurityCapability class has been subclassed to describe the main concepts needed to model policies and emerged during the analysis of several SCs (see Fig. 2).

Four concepts are needed to model rules. As anticipated before, the *conditions* are the features available at the NSF

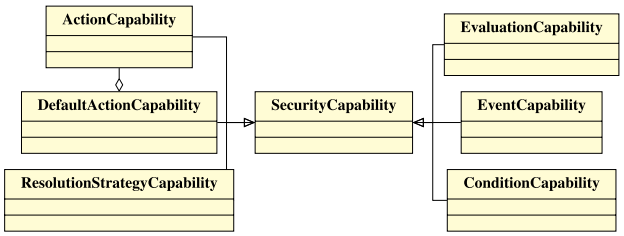


Fig. 2. Subclasses of SecurityCapability.

```
<nSF id="genericPacketFilter">
   <securityCapability ref="IpProtocolTypeConditionCapability"/>
   <securityCapability ref="IpSourceAddressConditionCapability"/>
   <securityCapability ref="IpDestinationAddressConditionCapability"/>
   <securityCapability ref="SourcePortConditionCapability"/>
   <securityCapability ref="DestinationPortConditionCapability"/>
   <securityCapability ref="AcceptActionCapability"/>
   <securityCapability ref="RejectActionCapability"/>
   <securityCapability ref="DefaultActionCapabilitySpec"/>
</nSF>
<securityCapability id="IpSourceAddressConditionCapability"
   xsi:type="IpSourceAddressType"/>
```

Fig. 3. Specification of the generic packet filter.

to trigger the rule enforcement. Conditions are expressed with specific constraints on different data types (e.g., matching an IP address or checking a regex on the HTTP MIME type allows selecting the traffic to allow). The *actions* are the operations an NSF can perform on individual objects (e.g., deny packets or encrypt flows). The *events* allow triggering the rules' evaluation for specific classes of controls (e.g., the reception of a network packet on an interface). The *condition clauses* indicates the formula for determining when a rule is activated depending on the evaluation of individual conditions, the most common ones being the Disjunctive Normal Form (DNF), i.e., the rule is activated when all the conditions are true, or the Conjunctive Normal Form (CNF), i.e., it is activated when just one condition is satisfied.

Two more concepts illustrate how to build security policies from individual rules. The *resolution strategies* describe how the SC behaves when multiple rules are fired. For instance, the First Matching Rule strategy applies the actions from the highest priority rule. The *default actions* determine how the SC behaves when no rule is fired (e.g., for implementing the "deny all" default behaviour). If some actions are not available at an NSF to specify default behaviour, the DefaultAction class allows aggregating the allowed actions.

Finally, the NSFCatalogue class is the central aggregator of NSF instances and decorations and the root element of the repositories. More details on SCM classes and attributes are available in a technical report.[3]

A. Modelling the Reference Security Controls

Individual capabilities are modelled with a Data Model. For this research, we have analysed controls that filter traffic up to OSI layer 4 and layer 7, channel protection, and VPN controls, namely:

- IpTables[4], a very widespread and effective filtering control available in the Linux distributions (only the filtering modules, no NAT/NAPT);
- XRFM[5], the native IPsec configuration for Linux;
- strongswan,[6] an advanced IPsec/IKE implementation;
- squid,[7] a web caching proxy with an application layer filtering module.

Moreover, the Data Model includes controls that abstract the features of theoretical SCs. Fig. 3 presents an XML description of a generic packet filter, which serves to specify the allowed or denied connections using five-tuples composed of source and destination IPs and ports, and protocol type [29]. Fig. 3 also presents the definition of one of the condition capabilities, the IpSourceAddressConditionCapability and its data type (i.e., IpSourceAddressType), defined within an XMLSchema.

The definition of the IpTables NSF can be obtained by extending the definition of the generic packet filter with its resolution strategy (i.e., the FMRResolutionStrategyCapabilitySpec), the additional conditions (187 ConditionCapability classes), and actions (140 ActionCapability classes) it supports.

The NSF Catalogue[8] describes seven SCs. The Data Model contains one single instance of capabilities for events, and Resolution Strategy (the FMR), two evaluations (CNF and DNF), one default action (the class that indicates that all the actions are suitable as default); considering all the SCs, it contains 270 conditions and 167 action capabilities.

B. Deriving the Abstract Language for an Nsf

The capabilities describe what SCs can do, which also correspond to the features that can be enabled with their configuration languages. Hence, capabilities can be associated with abstract language constructs. Starting from NSF instances, an automatic language generation process can derive their NSF Abstract Languages. The NSF Abstract Language can be used to specify the configuration for that specific NSF, named Middle-Level Policy (MLP). MLPs use a syntax that is independent of the SC for which they have been derived, but since they are validated against the NSF Abstract Language, it is ensured that they only contain instructions for capabilities owned by the SC.

Fig. 4 presents the abstract language of the generic packet filter in Fig. 3. The Policy class aggregates Rules. Rule instances only allow the use of conditions and actions specified in the NSF; they are associated with the proper data types. Moreover, rules include the necessary fields for supporting the resolution strategy and attributes helpful for specification purposes (e.g., descriptions and labels).

C. Model-Driven Approach for Translation

MLPs are not directly usable to configure SCs; hence, a further translation into the appropriate low-level settings is needed. An essential requirement in designing this translation process was to avoid constantly updating the Translator code.

[3]https://github.com/aldobas/SecurityCapabilityModelPub/tree/main/manual
[4]https://netfilter.org/news.html
[5]https://man7.org/linux/man-pages/man8/ip-xfrm.8.html
[6]https://www.strongswan.org/
[7]http://www.squid-cache.org/

[8]The Catalogue and the SCM Information and Data Models can be found at https://github.com/aldobas/SecurityCapabilityModelPub

```
<xs:complexType name="Policy">
<xs:sequence>
   <xs:attribute name="nsfName" type="xs:string"/>
   <xs:element maxOccurs="1" minOccurs="0"
        name="defaultActionCapabilitySpec"
        type="DefaultActionCapability"/>
   <xs:element maxOccurs="unbounded" minOccurs="0"
        name="rule" type="Rule"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="Rule">
    <xs:choice maxOccurs="unbounded">
        <xs:element name="ruleDescription" type="xs:string"/>
        <xs:element name="label" type="xs:string"/>
        <xs:element name="externalData" type="ExternalData"/>
        <xs:element name="defaultActionCapabilitySpec"
                    type="DefaultActionCapabilitySpec"/>
        <xs:element name="ipSourceAddressConditionCapability"
                    type="IpSourceAddressConditionCapability"/>
        <xs:element name="ipDestinationAddressConditionCapability"
                    type="IpDestinationAddressConditionCapability"/>
        <xs:element name="sourcePortConditionCapability"
                    type="SourcePortConditionCapability"/>
        <xs:element name="destinationPortConditionCapability"
                    type="DestinationPortConditionCapability"/>
        <xs:element name="ipProtocolTypeConditionCapability"
                    type="IpProtocolTypeConditionCapability"/>
        <xs:element name="acceptActionCapability"
                    type="AcceptActionCapability"/>
        <xs:element name="rejectActionCapability"
                    type="RejectActionCapability"/>
    </xs:choice>
    <xs:attribute name="id" type="xs:string"/>
    <xs:attribute name="ruleType" type="xs:string"/>
</xs:complexType>
<xs:complexType name="ExternalData">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="type" type="xs:string"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

Fig. 4. Abstract language of the generic packet filter.

The decorator pattern helped us reach our goal, as it allows providing additional information when capabilities are "attached" to the decorated objects. It includes the HasSecurityCapabilityDetails association class, which in the implementation is associated with both NSF and SecurityCapability. This class can be used to map abstract capabilities to device-specific settings at the model level, permitting the development of a model-driven translator that does not need NSF-specific code to perform the translations. Indeed, translation directives are available as association classes.

Different types of "details" (see Fig. 1) were needed in the SCM. NSFPolicyDetails aggregates instructions for generating valid policies for the target NSF and is described in Section III-D). CapabilityTranslationDetails specifies how to translate abstract capabilities into low-level settings and is discussed in Section III-E ResolutionStrategyDetails specifies the external data the Resolution Strategy uses, which are sketched in Section III-F.

D. NSFPolicyDetails

The NSFPolicyDetails class is an inner attribute of the NSF class and conveys information on how to translate policy-related information for a target NSF. It reports:

- strings and formats needed to generate valid rules and policies (with the attributes `ruleStart`, `ruleEnd`, `policyTrailer`, and `policyEncoding`);
- the capabilities that need to mandatorily be present for a valid policy, which may vary depending on the SC (`defaultSecurityCapability`);

```
<nSF id="IpTables">
  <nsfPolicyDetails>
    <ruleStart>iptables</ruleStart>
    <policyAttribute>
      <attributeName>targetRuleSet</attributeName>
    </policyAttribute>
    <defaultSecurityCapability>appendRuleActionCapability
    </defaultSecurityCapability>
  </nsfPolicyDetails>
  ...
</nSF>
```

Fig. 5. Policy details for IpTables.

```
<securityCapability ref="SourcePortConditionCapability" />
<commandName>
  <realCommandName>--sport</realCommandName>
</commandName>
<commandName>
  <realCommandName>! --sport</realCommandName>
  <commandNameCondition>
    <attributeName>operation</attributeName>
    <attributeValue>NOT_EQUAL_TO</attributeValue>
  </commandNameCondition>
</commandName>
```

Fig. 6. Source port connection.

- other parameters to be given to the Translator for a correct generation that cannot be obtained otherwise (`policyAttribute`).

Fig. 5 presents the NSF details for Iptables. Rules must start with the 'iptables' prefix; every IpTables rule must contain an option that indicates the policy chain where the rules need to be inserted (e.g., the `-A` option for appending them). The chain name where rules will be appended (e.g., INPUT, OUTPUT, FORWARD) is passed as an attribute.

E. CapabilityTranslationDetails

The CapabilityTranslationDetails class conveys data needed to translate individual capabilities to form correct NSF-specific rules. In other words, the attribute values reported for this class specify the low-level policy rule semantics and syntax, which are crucial for model-driven translation. It includes the keywords to invoke the capability (`realCommandName`) and additional attributes to adjust the command name when Boolean operators are used, e.g., to negate an individual condition (`commandNameCondition`), and how this keyword connects to the values (e.g., to put an equal after the condition name and a semicolon to end it with), as in the example below for source port conditions in IpTables:

Such a class also reports the way to parse the values from the MLP and merge them into a valid statement (`bodyConcatenator`), how to provide more individual values (e.g. port=80,81,82), build ranges (e.g., port=80–82 or port=[80–82]), and how the specification of conditions and actions is started and terminated (e.g., to put a comma before processing the following capability). The example in Fig. 7 presents how to treat multiple values for IpTables source ports and how this affects the use of specific IpTables extensions.

Finally, the CapabilityTranslationDetails class describes the dependencies among capabilities, e.g., if a capability can only be present if other ones are already present or absent in the same rule, also allowing the setting of their values (`dependency`). For instance, in IpTables, the source and

```
<bodyConcatenator>
  <operatorType>union</operatorType>
  <realConcatenator>,</realConcatenator>
  <concatenatorCondition>
    <preVariable>elementRange</preVariable>
    <postVariable>elementValue</postVariable>
  </concatenatorCondition>
  <newCommandName>
    <realCommandName>-m multiport --sports</realCommandName>
  </newCommandName>
  <newCommandName>
    <realCommandName>! -m multiport --sports</realCommandName>
    <commandNameCondition>
        <attributeName>operation</attributeName>
        <attributeValue>NOT_EQUAL_TO</attributeValue>
    </commandNameCondition>
  </newCommandName>
</bodyConcatenator>
```

Fig. 7. An example of concatenator.

```
<dependency>
    <presenceOfCapability>IpProtocolTypeConditionCapability
    </presenceOfCapability>
    <presenceOfValue>tcp</presenceOfValue>
</dependency>
```

Fig. 8. An example of dependency.

```
<policy nsfName="IpTables" targetRuleSet="INPUT"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="language_ipTables.xsd">
<rule id="0">
    <externalData type="priority">1</externalData>
    <appendRuleActionCapability>
        <chain>OUTPUT</chain>
    </appendRuleActionCapability>
    <inputInterfaceActionCapability>
        <inFa>eth0</inFa>
    </inputInterfaceActionCapability>
    <ipProtocolTypeConditionCapability operator="exactMatch">
        <capabilityValue>
                <exactMatch>TCP</exactMatch>
        </capabilityValue>
    </ipProtocolTypeConditionCapability>
    <ipSourceAddressConditionCapability operator="exactMatch">
    <capabilityIpValue>
        <exactMatch>192.168.1.1</exactMatch>
    </capabilityIpValue>
    </ipSourceAddressConditionCapability>
    <rejectActionCapability/>
</rule>
<defaultActionCapabilitySpec>
    <acceptActionCapability/>
</defaultActionCapabilitySpec>
</policy>
```

Fig. 9. An MLP for IpTables.

destination ports can only be used if a condition on the protocol type (requiring UDP or TCP) is included in the same rule, as described in Fig. 8.

An example of an abstract policy for IpTables can be seen in Fig. 9. The rule, added to the OUTPUT chain, drops TCP packets from a specific IP address if received on a given network interface, while all other packets are accepted. This policy has been validated against the language ipTables.xsd language, the automatically generated schema for IpTables, and translates to a file containing:

```
iptables -A OUTPUT -i eth0 -p TCP -s
192.168.1.1 -j DROP
iptables -P OUTPUT ACCEPT
```

F. ResolutionStrategyDetails Class

ResolutionStrategyDetails class serves to characterise the external data that a Resolution Strategy uses so that those values are properly generated during translation (e.g., `requiredExternalData`).

For example, the *First Matching Rule* resolution strategy, i.e., the default one for IpTables, associates rules to integers to express the priority. Based on priorities, the Translator will append rules to the selected chain in the proper order.

G. Addressing Other Translation Issues

In most cases, but more frequently for conditions, there is a need to express more values when specifying a single capability. The method for specifying multiple values varies depending on the type. The simplest types, like ports or IP addresses, are usually specified as individual values, ranges, or both (e.g., port=80,85-90,101,1024-). More complex types, like URLs or application-level payloads, usually rely on regular expressions[9]

The Abstract Languages generated from the NSF instances have no limitations in the specification formats, i.e., they support sets and ranges for integer-based types, sets and ranges (also in CIDR notation) for IP addresses, and Perl regex for string-based capabilities.

However, when generating the low-level configurations, some NSFs may not fully support all the expressiveness allowed at the abstract level. For example, before the development of the multiport extension, IpTables only accepted a single port number as a condition. Theoretically, multiple values within a single capability are associated using a set union. Thus, when a union at the capability level is unsupported, it needs to be expanded into a union of individual rules. The Cartesian Product needs to be used to expand the cases when multiple value capabilities are specified in a single rule. The Translator understands that an expansion is needed by interpreting data from the corresponding CapabilityTranslationDetails. The attributes of the CapabilityTranslationDetails serve to customise this process. The `PreferredExpansionType` attribute explicitly states how the expansion must be performed. By default, the Translator ensures a minimum number of rules are generated. In the worst case, e.g., NSFs that only accept a single value for each condition, the number of generated rules could be large. Nonetheless, we have not encountered these situations in all the NSFs we have analyzed.

IV. Implementation

We have implemented an entire ecosystem of components; some serve to manage the capabilities, and others use them to perform more sophisticated tasks, as depicted in Fig. 10.

The Security Capability Model is represented with the UML dialect of Modelio,[10] an open-source modelling environment for UML. However, we needed XML technologies for practical usability and to be applied to our use cases. The *XMIConverter* tool automatically generates the XML Schema (named capability data model.xsd) that describes the UML classes and

[9]Note that IP ranges are sometimes specified with a regex, but it is an overly sophisticated approach that easily maps to the set and range case.

[10]https://www.modelio.org

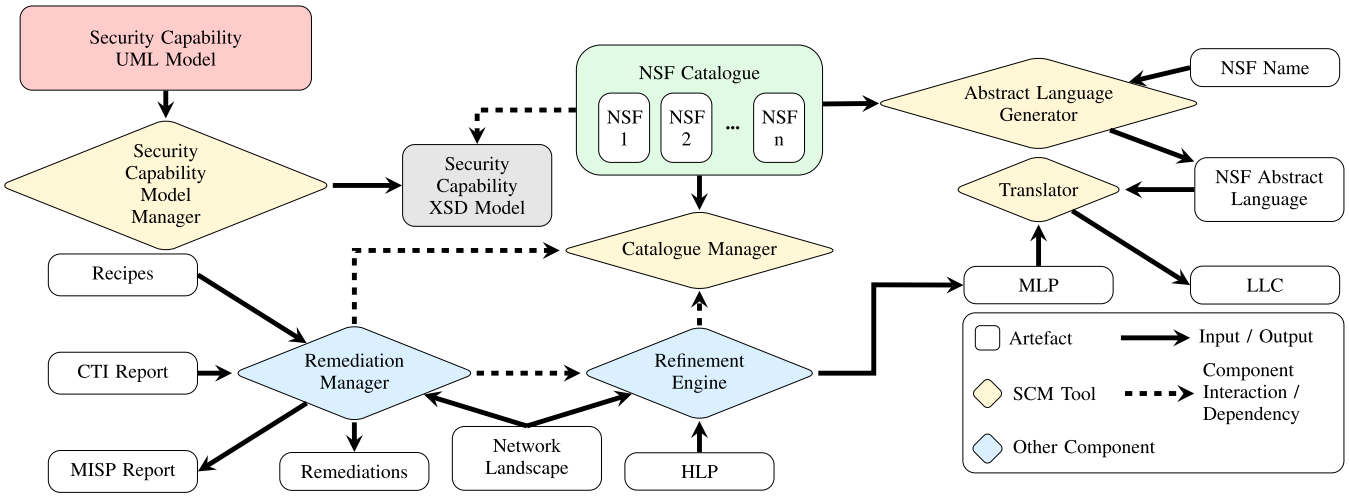


Fig. 10. Visual representation of the components built upon the SCM and their interactions.

properly maps the original UML features. This file contains all the data types to model the security controls.

Security Controls have been described in separate XML files and validated against the capability XML Schema. These XML files, rooted at the NSF class, list all the security capabilities, the associated policy details, and all the translation instructions as LanguageGenerationDetails instances. The Catalogue is an XML file that includes the SC files to use.

Given the name of an NSF and the XML file describing it, the *Abstract Language Generation* tool produces the XMLSchema file describing its abstract language that can be used to validate MLP policies for that NSF, written in XML. These two tools have been implemented in Java. They are expected to be executed occasionally, i.e., when changes are implemented in the model or new security controls are added to a catalogue.

The capability model can be managed and accessed through a web service named Catalogue Manager (CaM), which exposes a user-selected NSF Catalogue through an XML DB. We have used BASE X[11] which provides various visualizations to explore data interactively and implements a RESTXQ service, enabling the development of web applications in XQuery. A set of endpoints is exposed, as a set of XQueries, that implement essential functions/workflows:

- *Comparison*: it receives the names of two NSF and, if these are present in the catalogue, checks if the sets of capabilities they own are equivalent, their intersection is empty, or one is contained in the other. It returns the set of shared capabilities.
- *Substitute*: it receives the name of an NSF, then the service proposes other security controls in the catalogue that could be employed instead;
- *Policy enforcement*: it receives an MLP, and the service will identify other security controls that support the capabilities required to enforce the MLP. Knowing this information, users can select one NSF and use the Translator to obtain the policy in the specific language of that security control.
- *NSF search*: it receives a comma-separated list of capabilities and returns the list of NSFs supporting all of them.

The RefEng is a component that refines requirements for securing a computer network, expressed as High-Level Policy (HLP) statements, into MLP for the security controls in the target network. HLP statements are authorization policies expressed in the subject-action-verb-options paradigm [30]. RefEng analyses the subjects, objects, and optional fields present in the HLPs and associates them with the entities in the target network; the HLPs are then interpreted semantically to determine the capabilities needed for its enforcement. After identifying the required capabilities, the target network is explored. By querying the CaM, the RefEng identifies the NSFs that own them, also proposing to add new controls from the Catalogue to remediate non-enforceable policies.

Once the usable controls are known, the selection of the ones to use is done according to a refinement strategy (e.g., defence in-depth, minimization of the security control to configure, minimize the delay). Finally, RefEng generates the MLPs for all the selected NSFs.

The refinement has been implemented as a forward reasoning task that semantically analyzes and enriches via inference using CLIPS[12] and its Python bindings. Its features can be accessed natively, as a Python script, or offered via a web service that exposes the following actions:

- *Upload*: allows the upload of the HLP in XML format;
- *Upload Landscape File*: allows to upload a file containing the target network description;
- *Refinement*: perform an interactive refinement where the user is asked to provide missing information or make decisions when alternative refinement options are available (e.g., to select the devices to use);
- *Non-interactive Refinement*: the refinement process is performed using the user-selected refinement strategy;

[11]https://basex.org

[12]https://clipsrules.net/

- *MLP Generation*: generates the MLP policies from the facts contained in the reasoning engine's knowledge base and then outputs the list of involved NSFs;
- *Download MLP*: download the MLP policies for one or all of the involved NSF.

The *Translator* is the tool that, given an MLP and the Abstract Language of the target security control, generates the configuration settings that map an input MLP into the NSF-specific language. This tool is available both as a web service and natively as a JAR.

Finally, the Remediation Manager (ReM) is the component that orchestrates the remediations to security incidents/events notified by Cyber Threat Intelligence (CTI) reports. The ReM interprets the reports (e.g., infers the type of threat and the impacted entities), identifies the applicable remediations, and evaluates their potential effectiveness.

Remediations are enforced through the procedural execution of playbooks, derived from interpreting intents detailing defensive strategies, called Recipes, expressed in a custom DSL (Domain Specific Language). ReM recipes are interoperable with the CACAO standard and follow the best practices of playbook-based remediation description [31]. These recipes consist of high-level instructions that may propose to add new rules to an existing SC (e.g., new firewall filters) by proposing new HLPs, suggest network changes (e.g., isolating or shutting down machines), or propose adding new SCs to mitigate the threats. The CaM helps identify the controls to use from the ones available in the NSF Catalogue.

The ReM can automatically select the most effective recipe, as security experts rated every recipe's effectiveness on individual incidents and the related impacted environment. Otherwise, users can manually choose a suitable one from a GUI. Based on a DSL framework that employs a visitor pattern and a PEG grammar, the interpreter gathers all necessary data for recipe deployment and the execution of specific actions, and then runs the recipe step by step. The ReM interfaces with the RefEng to trigger the refinement process of HLP statements that may be mandated by the recipe or with the network infrastructure to initiate changes to the network or routing.

A. Security and Complexity Considerations

It is important to emphasize that the SCM, the artefacts it generates (e.g., the XML Schemas), and the associated catalogues are all sensitive assets. The effectiveness of security enforcement and remediation relies heavily on their content. Modifying these files can cause refinement systems or remediation tools to fail or produce mistaken configurations. Moreover, attackers accessing these artifacts may have an advantage in identifying vulnerabilities in the security posture. As a result, these assets must be protected against attacks aimed at compromising their confidentiality and integrity, whether those attacks come from external sources or insiders. Protecting these assets is not different from safeguarding other pieces of security-sensitive information by implementing risk mitigation measures. Nonetheless, we present a brief security analysis here.

Integrity is the most critical property of the SCM that must be safeguarded. Specifically, the integrity of the NSF Catalogue (i.e., the definitions of the modelled SCs, including the information required for policy translation) must be ensured. If an attacker is able to tamper with this data, security administrators may unknowingly generate and deploy incorrect security policies. For example, consider an attacker who modifies the *capabilityTranslationDetails* of the *SourcePortConditionCapability* in the IpTables NSF definition (shown in Listing 6) by inverting the two *realCommandName* tags. This seemingly minor alteration causes all IpTables policies that use this definition and rely on the *SourcePortConditionCapability* to behave contrary to their intended function (potentially allowing traffic from undesired TCP ports while blocking legitimate traffic when the policy is deployed).

Nevertheless, only vendors or the teams responsible for developing open-source security controls should be permitted to modify the model or the catalogue. Verifying digital signatures and leveraging trusted certification authorities can significantly mitigate the risks associated with most tampering attacks. Additionally, to further reduce the risks of insider tampering, strict non-repudiation mechanisms should be implemented alongside versioning capabilities, enabling prompt restoration of safe definitions in the event of a compromise.

Since the model and the NSF definitions are intended to be publicly available, confidentiality is not the primary focus of our security analysis. However, in certain scenarios (such as the modelling of non-public tools for internal use), the NSF definitions may contain information about the features and internal workings of these privately used tools. A data breach involving said data could lead to the unintended disclosure of intellectual property. Strong data protection mechanisms, both in transit and at rest, can help with this issue.

However, significant effort should also be dedicated to protecting the confidentiality of SCM artefacts related to policy refinement and translation (namely, the HLPs, MLPs, and Low-Level Configuration (LLC)s). A data leak could allow attackers to analyse the configuration of the organisation's deployed security controls, potentially leading to the discovery of vulnerabilities, misconfigurations, or other weak points [32]. To mitigate this risk, it is recommended to implement robust data protection mechanisms, both at rest and in transit.

Lastly, availability is another important security property for the SCM and NSF Catalogue information. The SCM tools should be designed and implemented to be resilient, also during critical operations (e.g., software updates). Moreover, the tools and databases they manage must also be able to cope with catastrophic events, such as failure or DoS attacks, against the online repositories.

The complexity related to the management of the SCM and its ecosystem would be largely compensated by the advantages given by the automated (or semi-automated) security policy enforcement and remediation that can be built on top of it.

B. Attack Model

The SCM is actually independent of any attack model. Nonetheless, it relates to the attacks that the SCs it allows to

model can prevent. The currently modelled SCs are capable of mitigating a broad range of cyber attacks performed by *active (remote) network adversaries*. These attacks include, but are not limited to, distributed denial-of-service (DDoS) attacks, port scanning, IP spoofing, brute-force attempts, man-in-the-middle (MITM) attacks, which can compromise the security of communication channels, traffic interception, replay attacks, session hijacking, application-layer exploits, bandwidth abuse, and information leakage. Therefore, the SCM is well-suited for typical network security scenarios in traditional network architectures as well as software-defined ones. The attackers could be either internal or external to the organization, depending on the specific application scenario where the SCM is used. We refer to the extensive corpus of existing literature for additional details on the attackers and scenario [33], [34].

V. Validation

Our validation focused on the three scenarios in Section II. The reference network[13] served as a representative example of a medium-sized company network, with ten subnets interconnected by eight security-enabled nodes, each implementing one or m ore security controls. All the performance tests were performed on a laptop equipped with an Intel Core i7-13700H processor averaging at 4 GHz and 32 Gbyte RAM.

A. Cybersecurity Market and Vendor Lock-in

The validation of the first proposed use-case aims to prove that the model is expressive enough to cover relevant SCs, in particular packet filters, and can be easily extended. Moreover, it aims to assess the feasibility of migrating configurations from similar yet not equivalent devices, mimicking what security operators would perform in a hypothetical company.

To this purpose, we simulated the migration of the company's internal firewall from IpTables to another product, in our case, the pfsense solution,[14] which builds on the BSD Packet Filter (PF) and is also a common basis for commercial products. Initially, PF was not supported in the SCM, hence we organized an empirical experiment, in which two members of our research group assumed the role of security operators and were tasked to describe PF according to the SCM, reusing, whenever possible, what was already available in the Data Model and Catalogue files for IpTables. The experiment was conducted asynchronously, not in a controlled environment.

Iptables and PF are not equivalent [35]. Still, even if significant challenges surfaced during the modelling of PF stateful capabilities (e.g., TCP three-way handshake and management of counters and other times) and traffic direction, the SCM was properly extended to support PF. Table I reports the capabilities added to model PF. Starting from a limited knowledge of the model but a good knowledge of packet filters, answering a questionnaire, the involved subjects reported they needed about two days to support PF, including a study of the SCM and a meticulous analysis of the PF features and syntax.

[13]Available in our repository: https://github.com/aldobas/SecurityCapabilityModelPub

[14]https://www.pfsense.org/

TABLE I
The PF Capabilities (PF-Specific Ones Are in Italic)

Capabilities of the PF-PacketFilter
AcceptActionCapability, RejectActionCapability, TcpFlagsConditionCapability, DestinationPortConditionCapability, SourcePortConditionCapability, IpDestinationAddressConditionCapability, IpSourceAddressConditionCapability, IpProtocolTypeConditionCapability, NumberConnectionsConditionCapability, LimitSAddrConditionCapability, OutputInterfaceConditionCapability, InputInterfaceConditionCapability, *InterfaceConditionCapability*, *StateInterfaceBoundConditionCapability*, *MaxRateConnectionsConditionCapability*

After PF was added to the Catalogue, all the queries in Section IV, completed in an average of 10 ms, were manually checked and confirmed that PF capabilities intersect with the capabilities offered by IpTables. Furthermore, we tested the performance of searching compatible NSF on synthetic policies containing three rules and achieved an average execution time of 10 ms. Increasing the number of rules to 100 resulted in an average query execution time of 30 ms, showing promising scalability.

This experiment demonstrated the model's flexibility in integrating new security controls. Supporting the general case of stateful packet filters may require additional refinements and adaptations to enhance stateful capabilities, which we will address as future work. Indeed, SCs within the same category, especially in the context of firewalls and packet filtering controls, exhibit evident similarities, even if some differences remain [35]. Similarities and differences have been leveraged in literature to assess the portability and perform translation of policies across different SCs [36], [37]. Our model allows the abstract description of SCs; it highlights in the model differences and similarities, and can serve to anticipate potential incompatibilities without the need for formal approaches. The model-driven approach used for translating policies also overcomes the limitations of transcompilation, as it models the differences upfront, allowing one to understand when they are reconciliable and including instructions in the model to properly manage the translation when compatible.

Then, we evaluated the model's ability to determine when a policy is enforceable on target devices. MLP policies for a generic 5-tuple packet filter, enforceable by IpTables and PF, were translated, leveraging the respective abstract language definitions, in an average of 7.7 s for PF and 9.3 s for IpTables. The higher translation time is due to the greater number of capabilities required to model IpTables, which determines more complex internal data structures during the translation process.

Moreover, we translated IpTables policies into PF policies using the Translator.[15] By checking the model, the Translator correctly noticed when the policies were not enforceable on

[15]Note that the conversion from IpTables configurations to MLP has been made with an *ad hoc* script. The Translator does not provide generic features for low- to medium-level transformation. Obtaining MLP from LLC is feasible and was considered initially, but has limited research applications and is more tedious to implement because languages allow different methods for expressing the same policies.

PF. In all cases, it translated all the PF rules that were compatible, generating a warning for the rules that were not.

The translation process for PF required an average of 361 ms and 3.4 s for the three-rule and 100-rule policies, respectively. Regarding iptables, the translation required 2.9 s and 6.6 s, a performance difference likely attributable to the greater complexity of the SC and the time needed to initialize the data structures for performing the translation.

These experiments proved that the SCM can be used to evaluate how different security controls can handle high-level requirements and select the most suitable ones, hence proving our research objective RO_1 presented in Section II-A. In concrete scenarios, our model can be used in workflows that enable policies to be translated between different formats, which renders vendor lock-in more complex to achieve, hence proving our research objective RO_2. It does it without necessitating costly assessments aided by the vendors of those security controls. Moreover, it showed enough flexibility to cope with adding more SCs. However, we acknowledge that incorporating all security controls can be time-consuming, to industrially exploit our approach.

B. Policy Refinement

The refinement has been initially validated against a synthetic network inspired by a real-world scenario. The network comprises ten different subnets, interconnected by six security controls, four filtering devices (IpTables), and two VPN gateways (XFRM and Strongswan). Additionally, three general-purpose nodes have been introduced as actors in the policy refinement process: two legitimate network users and an external network segment representing malicious users.

Authorization and channel protection policies were expressed using HLP statements. The RefEng identifies the capabilities needed to implement the policies, selects the SCs in the Catalogue,[16] generated the MLP, and, via the Translator, the LLC. We manually verified that the options and selections were correct, i.e., the devices provided the required capabilities and belonged to the correct network paths. An optimization model could help this selection process; it will be investigated in future work. A manual assessment also proved that the generated MLPs and LLC correctly implemented the input HLPs; the LLC were properly imported by the target SCs.

After that internal assessment, the RefEng and the Translator were used by the Consortium of the EC-funded FISHY project. It was integrated into the FISHY architecture and used in the project's Use Cases [38], [39]. This validation was conducted at the Consortium level. Through probe packets and tentative network misuses, the Consortium validated that the security policies were correctly enforced.[17]

The validations proved that the RefEng allows reaching the research objectives established in Section II-B; it identifies the security controls required for the specified policies (satisfying RO_3) and enables the refinement from HLPs to LLCs (satisfying RO_4). Being based on the features exposed by our SCM and the translation possible thanks to our Model-Driven approach, we validated the importance of a formal model of security capabilities to enable sophisticated security tasks in computer networks.

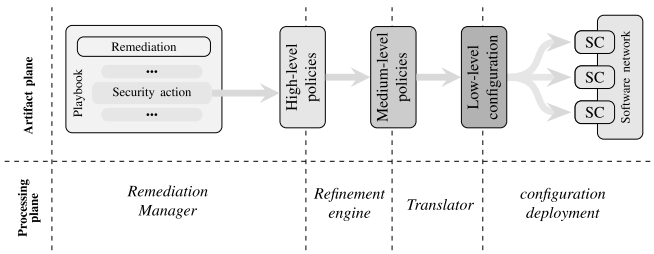


Fig. 11. Playbook-based security orchestration and enforcement.

C. Incident Response and Security Orchestration

The effectiveness of leveraging the SCM for automating tasks in the incident response pipelines and security orchestration was tested within the distributed systems of the use cases of the PALANTIR,[18] FISHY,[19] and Trust6G project[20] projects, on enterprise and IoT scenarios. The SCM was integrated in a playbook-based security orchestration platform and used to enact responses, see Fig. 11 for a high-level workflow [31].

We provide all artifacts related to the complete remediation pipeline in a public repository for ease of access. These include the original Cyber Threat Intelligence (CTI) alert and the final CTI report, which contains a response playbook and Indicators of Compromise (IoCs) for threat detection. The artifacts correspond to a remediation run targeting a botnet threat, conducted as part of the PALANTIR demonstration, as illustrated in Fig. 11.

Responses included two types of remediation. The first type was implementing changes to the network topology through a software network management layer, e.g., adding new security controls or redirecting traffic flows. The second involved the reconfiguration of security controls through the RefEng.

Orchestrating efficient incident responses requires automating the enforcement of the remediation playbook that prescribes high-level security actions, such as "block", "isolate", "quarantine", "audit", put together with workflow blocks enabled by conditions. In PALANTIR, the ReM responded to threats from crypto-mining and DDoS botnets. The remediation recipes for these scenarios queried the software-defined network to identify available security controls. These controls were then evaluated by the CaM against the required capabilities, e.g., filtering Layer 4 traffic and isolating or quarantining infected machines. In FISHY, the ReM reacted to increasing authentication failures to a distributed ledger based on Ethereum. The recipes involve filtering traffic at layer 7 (URL) and adding banning rules for specific Distributed IDs and Wallet IDs to the ledger authorization module. Moreover, the reactions also remediated DDoS and brute force attacks. In this case, the recipes proposed filtering traffic at layer 7 or

[16]When multiple controls were available to implement a policy, the user was asked to select the ones to use via an *ad hoc* web GUI (see Table II).

[17]https://www.youtube.com/channel/UCSDpfCPvFNjRS3RemG0iNQQ

[18]https://www.palantir-project.eu/

[19]https://fishy-project.eu/

[20]https://www.sns-itrust6g.com

4. In iTrust6G, the ReM dealt with different types of incident responses by implementing micro-segmentation, adding and configuring security controls. Moreover, it was integrated with a risk assessment module able to estimate the remediation impact before selection.[21]

In all cases, when a needed SC was unavailable in the software network, the recipes requested the deployment of a capable one in the correct portion of the network to protect target hosts if necessary. Finally, the security controls were properly reconfigured through the RefEng and the Translator that produced the LLC in the specific languages. The stakeholders in the consortia verified that recipes effectively mitigate the identified risks. They manually validated all the artifacts generated and checked using scripts that simulated attacks to ensure the threats were actually mitigated, verifying that the attacks were rendered ineffective [40].

With these experiments, the SCM concretely demonstrates its effectiveness firstly, by facilitating the selection of security mechanisms capable of enforcing the specified actions, and secondly, by enabling the generation of concrete policies and low-level configurations necessary for the playbook to take effect [41]. The SCM served in understanding which SCs could implement the high-level actions and then generate the LLC implementing the mitigation. It enabled the fine-grained selection of appropriate security controls and the effective interaction of the orchestration layer of the playbook-based platform with the security controls.

This practical validation in the three projects confirmed fulfilling the requirements in Section II-C. Specifically, it demonstrates the effectiveness of the SCM in informing the orchestration layer about the security controls' capabilities. It enables the interoperability of various security controls within automated, playbook-based incident handling. Moreover, it enables the interaction of the orchestration with the SCs via refinement and translation processes exposed by the accompanying tools (RO_5) [42].

Being driven by playbooks, the validation enhances cyber threat intelligence sharing, leading to improved actionability of such information and providing analysts with deeper insights into threats and their effective mitigation or remediation strategies (RO_6) [43]. The validation also demonstrated the model's and the Translator's ability to cope with security controls commonly used in the reference scenarios of the three projects.

VI. Related Works

We studied the state of the art on two levels. The first level focused on identifying standardization and modelling frameworks that aim to provide mappings, enabling the abstraction of configuration or, at a minimum, the representation of defensive strategies or security controls. The second level analyzed previous research on modelling approaches and formalization methods, particularly in the context of NSFs.

A. Abstract Models of Defensive Strategies

We investigated and compared several initiatives and projects that model and standardize defensive strategies and security controls. The objective of this comparison is not to provide a comprehensive survey of the security modeling landscape. In some cases, the identified initiatives are not overlapping but complementary to our work. They can be merged to contribute to achieving real-world interoperability in the field of security operations.

In earlier work, Basile et al. presented an ontology-based approach to abstractly translating policies into low-level configurations [44]. This approach exploited ontology as a tool for formal modelling knowledge about security controls. The representation of SC features was very coarse-grained due to the severe limitations posed by ontology-based reasoning and, hence, of limited usability. This work evolved into a more coherent framework for supporting refinement in software networks [5]. Still, instead of modelling individual capabilities, this work used SC categories to implicitly determine a standard set of features they exposed. Hence, it was not able to capture the differences between SCs in the same category.

When speaking of modelling individual capabilities offered by NSFs, the major efforts are presented by *CapIM* [45], the information model developed by the I2NSF WG to provide a standard interface for control and management of NSFs. The work described in this paper is a fork of *CapIM*, which inherits the basic design principles and most of the Information Model. However, the data models and the features for translation are a clear improvement over the original *CapIM*.

As shown by Geismann and Bodden [46], model-driven approaches towards security are currently a relevant research area, with various examples applied to cyber-physical systems. The reason is that having a formal model may eliminate threats and potential vulnerabilities in the early phases of design, while also providing flexibility during implementation. Similarly, adopting the same approach in the context of security functions results in a generalized representation of the entities that can be easily specialized to match the required security controls during implementation.

OpenC2 provides a generic, centrally maintained *abstract command language* for issuing cyber defense commands across different platforms and vendors. Its goal is to allow automated, machine-to-machine management of SCs [47]. OpenC2 does not specify how these commands are implemented; it only specifies their semantic meaning. The binding is performed for each category of security control, e.g., SLPF (StateLess Packet Filter), by means of *actuator profiles*. An actuator profile thus provides a standardized mapping schema from abstract OpenC2 language instructions to concrete device configurations.

It allows for extension to include vendor-specific features; however, these features are not semantically characterized. Hence, it is not possible to compare them. Hence, even if OpenC2 models security controls and provides translation features, it approaches the problem differently than we do, and, in some sense, it is a complementary approach.

OpenC2 categorizes the security controls very strictly and centrally determines the common features. We model SCs based on what they can provide, without an a priori categorization, and offer translation features by adhering to the Model-Driven paradigm. Therefore, concerning the modelling

[21]The details are omitted here due to confidentiality; they will be shared in upcoming public deliverables.

TABLE II
SAMPLE HLP STATEMENTS AND SC AVAILABLE TO ENFORCE THEM IN THE TARGET NETWORK

Subject	Action	Object	Available Security Control
Bob	is authorized to access	internet traffic	Path 1: firewall-1 (IpTables), Path 2: firewall-1 (IpTables), firewall-2 (IpTables)
Alice	protect integrity	Bob	Path 1: firewall-1 (XFRM), vpn-gateway-1 (StrongSwan)
Malicious_User	is not authorized to access	Alice	Path 1: firewall-1 (IpTables), firewall-2 (IpTables) Path 2: firewall-2 (IpTables)

of the low-level features of SCs, our approach can be considered more ambitious and flexible. It allows for a finer-grained specification of the capability, which permits checking if the SC's capabilities will be able to enforce security intents. However, we don't cover the same appliances they cover, and we lack a high-level imperative language that hides the complexity of low-level details. For a thorough analysis of OpenC2 we refer to [48].

We argue that the SCM can be leveraged with OpenC2 for bridging the gap between high-level actions and low-level configuration at runtime or by integrating the SCM in the production and consumption stages, instead of relying on the static approach based on compliance to actuator profiles. We aim to further investigate this possibility in future work.

The D3FEND framework is another example of how helpful formalization is in cybersecurity [49]. The D3FEND model focuses on a high-level semantic characterization of various countermeasures provided by cybersecurity products. It maps attacks, defence strategies or techniques, and digital artefacts, defined as generic computer systems or information used by them. Its knowledge graph enables practitioners to analyze and compare products based on their functionalities rather than relying on marketing materials. D3FEND can be used to assess the defence posture of a computer network infrastructure based on the countermeasures available. As it focuses on abstract countermeasures, D3FEND works at a higher level than our SCM. Hence, they can interoperate. The SCM can enrich the description of SCs, among the most used countermeasures, and can allow determining when they can be used to counter specific attacks.

On the other hand, we highlight that our modelling efforts have currently been scoped to a specific category of controls, namely NSFs, according to the NFV deployment models common in telco scenarios, in particular filtering appliances, and channel protection. While OpenC2 aims for a broader scope of SCs, its current public specifications actuator profiles also mainly target stateless and stateful packet filter profiles. Additionally, our modelling doesn't cover the management interface of the SCs, leaving the injection of configuration to the user of the SCM. For instance, in our validation, the ReM managed the deployment of the new configuration on the required SCs.

We summarize our findings in Table III. The features have been categorized in Focus, describing the frameworks' objectives, Layer, presenting their scope, Standardization, the modelling abstraction, Target, the objects covered, and, finally, Relation to SCs, which describes the aspects of the SCs that are covered. Based on the features they expose, it is possible to highlight the complementarity of these approaches.

TABLE III
COMPARISON OF SCM, MITRE D3FEND, AND OPENC2

Category	Feature	SCM	D3FEND	OpenC2
Focus	Capability description	✓	✗	✗
	Defensive technique taxonomy	✗	✓	✗
	Execution/Automation	✓	✗	✓
	Mgmt interface	✗	✗	✓
Layer	Conceptual model	✓	✓	✗
	Analytical/Strategic	✗	✓	✗
	Operational control	✓	✗	✓
Standardi-zation	Feature semantics	✓	✗	✗
	Knowledge graph	✗	✓	✗
	Command syntax/API	✓	✗	✓
Target	Vendor/Product developers	✓	✗	✓
	Cybersecurity analysts	✗	✓	✗
	Automation engineers	✓	✗	✓
Relation to SCs	Describes what they offer	✓	✗	✗
	Describes what they defend	✗	✓	✗
	Defines how to use them	✓	✗	✓

B. Applications

Analysing past research in automatic policy refinement, intents, and incident response gives insights into the abstract models in the literature and how they have been used. Despite being focused on firewalls, the work by Kovačević et al. offers the most recent systematic review of the literature related to automated policy refinement and low-level translation, highlighting a significant amount of work existing in the field [50]. In this context, Rivera et al. introduced a two-level mechanism for policy translation and automatic incident response [51]. However, the focus of this work is the development of a high-level model for a policy language and does not include a formal model that represents the security capabilities required to enforce such policies. Cheminod et al. developed an approach to refine and verify access control policies [16]; the set of actions modelled is limited and decoupled from the security controls that can enforce them, an issue that can be addressed with the introduction of the SCM.

Leivadeas and Falkner [52] explore various facets of the intent-based networking (IBN) paradigm in their survey. They introduce five distinct concepts that characterize IBN, namely intent profiling, translation, resolution, activation, and

assurance. In particular, translation enumerates the various approaches for converting intents into granular network configurations suitable for network devices, and activation addresses the mechanisms involved in deploying intents within the network system. Most of these approaches rely on general yet hardcoded refinement rules, built ad hoc for their specific use cases, instead of using a general and extensible model, as proposed in our research. Having a rigorous underlying model makes our approach more flexible and extensible, while also providing a ground truth that can be leveraged for building Agentic AI-based automatic risk management. Among the open challenges, they mention Intent fulfillment, i.e., translating intents into network policies and configurations. The authors emphasize the importance of vendor-agnostic fulfillment of intents, ensuring that infrastructure configuration mechanisms derived from intents are effective across multi-vendor environments. Moreover, they observe that translation mechanisms often involve multiple levels of refinement, each extracting different details of the intent and ensuring consistency with intent provisions, as in the framework we propose. Our SCM is scalable, flexible, and helps avoid the use of hard-coded decisions during the intents translation and fulfillment. The validation in the proposed scenarios proved that the SCM is an important step towards solving these issues.

VII. Conclusion

This paper presented the Security Capability Model, a precise and extensible formal model that describes several types of SCs. Its Information Model captures the general concepts, and its Data Model reports the capabilities needed to model packet filters and solutions for establishing secure channels. Thanks to a model-driven approach, it allows for implementing methods for translating abstract policies into SC-dependent configurations. It also supports the automatic creation of abstract languages to describe vendor-specific policies. The model proved effective in automating various crucial security tasks. Its basic features (e.g., comparing capabilities and identifying security controls based on the features required for enforcing policies) helped automate administrators' typical tasks, like migrating to new security controls. These basic features allowed for building more advanced services, i.e., security policy refinement and handling security incidents with procedural remediations. All the components developed upon the model have been validated on realistic scenarios and real-world applications in the context of EC-funded projects, leading to extremely promising results.

While writing this paper, we are already adding a new category of SC, namely ModSecurity, an open source, cross-platform web application firewall (WAF) engine that can be configured with an event-based programming language that is more sophisticated than the ones we currently support. Moreover, we plan to add several open-source and commercial packet filters to our catalogue to capture precisely all the facets of these stateful devices.

From the practical point of view, we are investigating how to integrate our model with the MITRE D3FEND and OpenC2 frameworks to propose contributions. Moreover, the SCM may also be used alongside different security standards and guidelines, e.g., the ISO 27000 family and NIST SP 800-53, to help link conceptual frameworks, attack models, and practical tools concerning security controls. Moreover, we will further consider how our model may impact OT security, as per the ISA/IEC 62443 standard. From the research point of view, we are investigating how to build an optimization model for determining the best resolution strategies in the RefEng and a risk-based evaluation of the mitigations proposed by the ReM. Furthermore, we are exploring the possibility of using the model for Retrieval-Augmented Generation to improve LLM performance in this policy-based management field. Moreover, as a long-term direction, we would like to use the model with AI techniques to develop methods that can autonomously build (the best) reactions to incidents.

Acknowledgment

The authors would like to thank Davide Colaiacomo for enhancing the existing SCM and tools to be used in the iTrust6G Project and for computing all the performance metrics reported in this article.

References

[1] A. Wool, "Trends in firewall configuration errors: Measuring the holes in Swiss cheese," *IEEE Internet Comput.*, vol. 14, no. 4, pp. 58–65, Jul. 2010, doi: 10.1109/MIC.2010.29.

[2] H. Hamed and E. Al-Shaer, "Taxonomy of conflicts in network security policies," *IEEE Commun. Mag.*, vol. 44, no. 3, pp. 134–141, Mar. 2006.

[3] D. Bringhenti, G. Marchetto, R. Sisto, and F. Valenza, "Automation for network security configuration: State of the art and research trends," *ACM Comput. Surv.*, vol. 56, no. 3, pp. 1–37, Oct. 2023, doi: 10.1145/3616401.

[4] A. Voronkov, L. H. Iwaya, L. A. Martucci, and S. Lindskog, "Systematic literature review on usability of firewall configuration," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 1–35, Dec. 2017, doi: 10.1145/3130876.

[5] C. Basile, F. Valenza, A. Lioy, D. R. Lopez, and A. P. Perales, "Adding support for automatic enforcement of security policies in NFV networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 707–720, Apr. 2019.

[6] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, and J. Yusupov, "Automated firewall configuration in virtual networks," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 1559–1576, Mar. 2023.

[7] P. K. Chouhan, A. Beard, and L. Chen, "Intrusion response systems: Past, present and future," 2023, *arXiv:2303.03070*.

[8] Z. Inayat, A. Gani, N. B. Anuar, M. K. Khan, and S. Anwar, "Intrusion response systems: Foundations, design, and challenges," *J. Netw. Comput. Appl.*, vol. 62, pp. 53–74, Feb. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804515002994

[9] C. Islam, M. A. Babar, and S. Nepal, "A multi-vocal review of security orchestration," *ACM Comput. Surv.*, vol. 52, no. 2, pp. 1–45, Apr. 2019, doi: 10.1145/3305268.

[10] C. Basile, D. Canavese, L. Regano, I. Pedone, and A. Lioy, "A model of capabilities of network security functions," in *Proc. IEEE 8th Int. Conf. Netw. Softwarization (NetSoft)*, Jun. 2022, pp. 474–479.

[11] L. Barnard and R. von Solms, "A formalized approach to the effective selection and evaluation of information security controls," *Comput. Secur.*, vol. 19, no. 2, pp. 185–194, Feb. 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404800878293

[12] I. Yevseyeva, V. Basto-Fernandes, M. Emmerich, and A. van Moorsel, "Selecting optimal subset of security controls," *Proc. Comput. Sci.*, vol. 64, pp. 1035–1042, Feb. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705091502760X

[13] J. Breier and L. Hudec, "On selecting critical security controls," in *Proc. Int. Conf. Availability, Rel. Secur.*, Sep. 2013, pp. 582–588.

[14] D. Dey, A. Lahiri, and G. Zhang, "Quality competition and market segmentation in the security software market1," *MIS Quart.*, vol. 38, no. 2, pp. 589–A7, Jun. 2014. [Online]. Available: https://www.jstor.org/stable/26634941

[15] A. K. Bandara, E. C. Lupu, J. Moffett, and A. Russo, "A goal-based approach to policy refinement," in *Proc. 5th IEEE Int. Workshop Policies Distrib. Syst. Netw. (POLICY)*, Jun. 2004, pp. 229–239.

[16] M. Cheminod, L. Durante, L. Seno, F. Valenza, and A. Valenzano, "A comprehensive approach to the automatic refinement and verification of access control policies," *Comput. Secur.*, vol. 80, pp. 186–199, Jan. 2019.

[17] D. Bringhenti, G. Marchetto, R. Sisto, and F. Valenza, "Short paper: Automatic configuration for an optimal channel protection in virtualized networks," in *Proc. 2nd Workshop Cyber-Security Arms Race*. New York, NY, USA: ACL, Nov. 2020, pp. 25–30, doi: 10.1145/3411505.3418439.

[18] M. Husák and M. Čermák, "SoK: Applications and challenges of using recommender systems in cybersecurity incident handling and response," in *Proc. 17th Int. Conf. Availability, Rel. Secur.* New York, NY, USA: ACM, Aug. 2022, pp. 1–10, doi: 10.1145/3538969.3538981.

[19] N. Gupta, I. Traore, and P. M. F. de Quinan, "Automated event prioritization for security operation center using deep learning," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 5864–5872.

[20] M. Nyre-Yu, "Identifying expertise gaps in cyber incident response: Cyber defender needs vs. technological development," in *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, 2021, pp. 1978–1987. [Online]. Available: https://api.semanticscholar.org/CorpusID:232414081

[21] C. Islam, M. A. Babar, and S. Nepal, "An ontology-driven approach to automating the process of integrating security software systems," in *Proc. IEEE/ACM Int. Conf. Softw. Syst. Processes (ICSSP)*, May 2019, pp. 54–63.

[22] J. Kinyua and L. Awuah, "AI/ML in security orchestration, automation and response: Future research directions," *Intell. Autom. Soft Comput.*, vol. 28, no. 2, pp. 527–545, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:234318365

[23] (2023). *Gartner Reviews: Security Orchestration, Automation, and Response Solutions*. Accessed: Nov. 9, 2023. [Online]. Available: https://www.gartner.com/reviews/market/security-orchestration-automation-and-response-solutions

[24] R. A. Bridges et al., "Testing SOAR tools in use," *Comput. Secur.*, vol. 129, Jun. 2023, Art. no. 103201. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404823001116

[25] A. Shaked, Y. Cherdantseva, P. Burnap, and P. Maynard, "Operations-informed incident response playbooks," *Comput. Secur.*, vol. 134, Nov. 2023, Art. no. 103454, doi: 10.1016/j.cose.2023.103454.

[26] Z. T. Sworna, C. Islam, and M. A. Babar, "APIRO: A framework for automated security tools API recommendation," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 1, pp. 1–42, Feb. 2023, doi: 10.1145/3512768.

[27] D. Pendarakis, D. R. Yavatkar, and D. R. Guerin, *A Framework for Policy-Based Admission Control*, document RFC 2753, Jan. 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc2753

[28] E. Gamma, *Design Patterns: Elementi Per Il Riuso Di Software a Oggetti*. Reading, MA, USA: Pearson Education, 2002. [Online]. Available: https://books.google.it/books?id=IkzmAAAACAAJ

[29] E. S. Al-Shaer and H. H. Hamed, "Modeling and management of firewall policies," *IEEE Trans. Netw. Service Manage.*, vol. 1, no. 1, pp. 2–10, Apr. 2004.

[30] V. Hu et al. (2014). *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*. [Online]. Available: https://csrc.nist.gov/files/pubs/sp/800/162/final/docs/nist.sp.800-162-201401.pdf

[31] F. Settanni, L. Regano, C. Basile, and A. Lioy, "A model for automated cybersecurity threat remediation and sharing," in *Proc. IEEE 9th Int. Conf. Netw. Softwarization (NetSoft)*, Jun. 2023, pp. 492–497.

[32] M. Conti, F. De Gaspari, and L. V. Mancini, "Know your enemy: Stealth configuration-information gathering in SDN," in *Green, Pervasive, and Cloud Computing*, M. H. A. Au, A. Castiglione, K.-K. R. Choo, F. Palmieri, and K.-C. Li, Eds., Cham, Switzerland: Springer, 2017, pp. 386–401.

[33] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. K. Bhattacharyya, and J. K. Kalita, "Network attacks: Taxonomy, tools and systems," *J. Netw. Comput. Appl.*, vol. 40, pp. 307–324, Apr. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804513001756

[34] O. Yurekten and M. Demirci, "SDN-based cyber defense: A survey," *Future Gener. Comput. Syst.*, vol. 115, pp. 126–149, Feb. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X20303277

[35] L. Ceragioli, P. Degano, and L. Galletta, "Are all firewall systems equally powerful?," in *Proc. 14th ACM SIGSAC Workshop Program. Lang. Anal. Secur.* New York, NY, USA: ACL, Nov. 2019, pp. 1–17, doi: 10.1145/3338504.3357340.

[36] C. Bodei, P. Degano, R. Focardi, L. Galletta, and M. Tempesta, "Transcompiling firewalls," in *Principles of Security and Trust*. Cham, Switzerland: Springer, Apr. 2018, pp. 303–324.

[37] L. Ceragioli, P. Degano, and L. Galletta, "Checking the expressivity of firewall languages," in *The Art of Modelling Computational Systems: A Journey From Logic and Concurrency to Security and Privacy*. Cham, Switzerland: Springer, Nov. 2019, pp. 86–100.

[38] J. Martinez, G. Kalogiannis, C. Basile, and J. M. M. Caliz. (2023). *D4.4 Security and Certification IT2 Integration*. [Online]. Available: https://fishy-project.eu/sites/fishy/files/public/content-files/deliverables/D4.4%20Security%20and%20Certification%20IT2%20integrationv1.0.pdf

[39] G. J. Prieto, A. R. Morgan, and A. Gonos. (2023). *D6.4 IT-2 Fishy Final Release*. [Online]. Available: https://fishy-project.eu/sites/fishy/files/public/content-files/deliverables/D4.4%20Security%20and%20Certification%20IT2%20integrationv1.0.pdf

[40] G. Xylouris, I. Miakar, and A. Lioy. (2023). *D6.2 Integration & Validation Report: Use Case Results and Playbook (Final Prototype)*. [Online]. Available: https://www.palantir-project.eu/documents/project-deliverables/

[41] H. Karlzen and T. Sommestad, "Automatic incident response solutions: A review of proposed solutions' input and output," in *Proc. 18th Int. Conf. Availability, Rel. Secur.* New York, NY, USA: ACM, Aug. 2023, pp. 1–9, doi: 10.1145/3600160.3605066.

[42] D. Sanvito, G. Athanasiou, and D. Santorinaios. (2023). *D5.2 Hybrid Threat Intelligence Framework—Second Release*. [Online]. Available: https://www.palantir-project.eu/documents/project-deliverables/

[43] I. Mlakar, L. Regano, M. Compastié, and N. Gkioules. (2023). *D4.3 Palantir Dashboard, Reporting and Threat Sharing Platform—Second Release*.

[44] C. Basile, A. Lioy, S. Scozzi, and M. Vallini, "Ontology-based security policy translation," *J. Inf. Assurance Secur.*, vol. 5, no. 1, pp. 437–445, 2010.

[45] L. Xia, J. Strassner, C. Basile, and D. Lopez, "Information model of NSFs capabilities," draftietf-i2nsf-capability, Internet Eng. Task Force, Fremont, CA, USA, Apr. 2019. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-i2nsf-capability/05/

[46] J. Geismann and E. Bodden, "A systematic literature review of model-driven security engineering for cyber-physical systems," *J. Syst. Softw.*, vol. 169, Nov. 2020, Art. no. 110697. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121220301461

[47] V. Mavroeidis and J. Brule, "A nonproprietary language for the command and control of cyber defenses—OpenC2," *Comput. Secur.*, vol. 97, Oct. 2020, Art. no. 101999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820302728

[48] M. Repetto, "Interface to security functions: An overview and comparison of I2NSF and OpenC2," *IEEE Commun. Standards Mag.*, vol. 7, no. 4, pp. 60–67, Dec. 2023.

[49] P. E. Kaloroumakis and M. J. Smith, "Toward a knowledge graph of cybersecurity countermeasures," MITRE Corp., Bedford, MA, USA, Tech. Rep. MITRE-TR-2021-02, 2021.

[50] I. Kovacevic, B. Štengl, and S. Groš, "Systematic review of automatic translation of high-level security policy into firewall rules," in *Proc. 45th Jubilee Int. Conv. Inf., Commun. Electron. Technol. (MIPRO)*, May 2022, pp. 1063–1068.

[51] D. Rivera, F. Monje, V. A. Villagrá, M. Vega-Barbas, X. Larriva-Novo, and J. Berrocal, "Automatic translation and enforcement of cybersecurity policies using a high-level definition language," *Entropy*, vol. 21, no. 12, p. 1180, Nov. 2019.

[52] A. Leivadeas and M. Falkner, "A survey on intent-based networking," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 625–655, 1st Quart., 2023.



Cataldo Basile (Member, IEEE) received the M.Sc. and Ph.D. degrees in computer engineering from the Politecnico di Torino, in 2001 and 2005, respectively. He is an Associate Professor with the Politecnico di Torino. His research concerns policy-based security management, general models for detecting, resolving, and reconciling security policy conflicts, software protection, and software attestation.



Gabriele Gatti received the M.Sc. degree in cybersecurity from the Department of Control and Computer Engineering, Politecnico di Torino, in 2022, where he is currently pursuing the Ph.D. degree. He is with the TORSEC-Research Group. His research focuses on the automation of cyber risk assessment and management, as well as the integration of cyber risk models into modern cloud infrastructures.



Francesco Settanni received the M.S. degree in cybersecurity from the Politecnico di Torino, Italy, in 2022, where he is currently pursuing the Ph.D. degree with the Control and Computer Engineering Department. He is a Research Fellow with the TORSEC-Research Group. His research focuses on cybersecurity threat management, cloud-native networks, kubernetes security, and policy-based security management.