Hunting the Abstract

Evaluating a Model for Analytics for OpenC2

Andreas Hverven

Thesis submitted for the degree of
Master in Information Security
30 credits

Department of Informatics
Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2023

Hunting the Abstract

Evaluating a Model for Analytics for OpenC2

Andreas Hverven

© 2023 Andreas Hverven

Hunting the Abstract

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Defensive security operations are tasked with protecting increasingly larger and more complex digital infrastructures and systems spanning across the whole fabric of modern societies including health, power, communication, and transportation. Higher complexity systems are associated with longer intrusion detection and response times, and higher costs.

Automation, advanced analytics, and artificial intelligence (AI) can contribute to faster mean time to detection (MTTD) and response (MTTR). To facilitate these functions in an increasingly complex ecosystem of tools and processes coupling communication interfaces through customized integrations is not sustainable and cannot scale. Consequently, defenders need to standardize command and control interfaces for the sake of interoperability.

In that regard, OpenC2 is being developed. A standardized language for the command and control of systems and components that perform or support cyber security. This thesis evaluates the ongoing work of developing an extension of the language to support threat hunting, against criteria relevant to a profile for analytical operations. We find that the ongoing work does not satisfy the requirements for an analytical extension of the language but instead aims at facilitating the automatic invocation of huntbooks, programmable intrusion detection. Further, we argue that previous extensions addressing technologies like packet filtering and endpoint response are more in line with important design principles.

Alternative approaches to a profile for analytics are presented and discussed, using the ongoing work as a starting point. We evaluate the alternatives under criteria such as agnosticism to existing implementations, the extent to which the models abstract the analytical function or specific operations, as well as the estimated amount of work to produce a comprehensive version of the proposed solution.

We were not able to produce complete, comprehensive versions of our suggestions in the form of an OpenC2 actuator profile under the time restrictions of a short master thesis. However, we do hope our discussions can be of some value to further work with developing a standardized language for analytics.

Contents

List of Figures

List of Tables

Preface

Thanks, Dr. Vasileios Mavroeidis, for the excellent supervision and feedback, for being invested, and for opening doors.

Thanks, OpenC2 technical committee, for letting me participate and observe their processes, especially Mr. Lemire for being helpful and friendly to new and inexperienced members such as myself.

Thanks, IBM senior researcher and Kestrel founder Dr. Xiaokui Shu, for being accommodating, and taking the time to meet and discuss the project.

Thanks to everyone contributing to god stemning on the 9th floor of IFI.

Glossary

Table 1: Acronyms and their full form

OpenC2	Open Command and Control
OASIS	Organization for the Advancement of Structured Information Standards
STIX	Standardized Threat Information eXchange
SCO	STIX Cyber Observable
SDO	STIX Domain Object
CACAO	Collaborative Automated Course of Action Operations
CTI	Cyber Threat Intelligence
OCA	Open Cybersecurity Alliance
EDR	Endpoint Detection and Response
AP	Actuator Profile
SLPF	Stateless Packet Filter
SIEM	Security Information and Event Management
SOAR	Security Orchestration Automation and Response
APT	Advanced Persistent Threat
IoC	Indicator of Compromise
IoB	Indicator of Behavior

Chapter 1

Introduction

1.1 Motivation

Modern society sees informational technology integrated into any and all of its digital services [1][2]. The benefits of digitization and automation of services do not come without costs. The increase in availability, interconnectivity, and complexity of our services, digital assets, and critical infrastructure, at the same time, expose potential vulnerabilities for exploitation. Cyber security has become integral to the security of basic needs, as power grids [3], water system facilities [4], healthcare organizations [5], and other critical infrastructure[6], are areas where informational and operational technology meet physical processes our safety depends on.

Adversaries can be resourceful, professional, and well-informed. They can choose when, what, and how to attack and utilize automation in their operations. However, defenders are challenged to monitor their assets, detect intrusions and respond in a timely manner (what experts call cyber-relevant-time [7]) to prevent actions on objectives. This produces an advantage to the adversaries, allowing the attackers to execute their offensive operations faster than defenders can detect and respond.

According to IBM security's Cost of a Data Breach 2022 report [8], it takes an average of 207 days to detect an incident and 70 days to contain it. There is a slight improvement from the previous year but no significant trend in the last five years. Longer data breach life cycles are associated with higher total costs in the same report.

One of the explanations for the slow detection and response time is that defenders still have to perform many of their operations manually, increasing detection time, response time, and, consequentially, the cost of attacks. This challenge is increasingly hard to overcome as the environments that defenders are tasked to protect become increasingly diversified regarding the composition of technologies and systems. As the set of distinct tasks defenders are required to solve grows, so does the size of their toolbox, referring to specialized software or tools that need to be integrated into already established procedures and infrastructures. This can be seen in IBM's analysis of critical factors associated with the costs of incidents. Out of the 28 factors analyzed, complex security systems are the factor associated with the highest average difference in the costs of incidents.

On the other end of the spectrum, and possibly the remedy to handle these complex systems, is the factor of artificial intelligence (AI) and automation platforms. This factor measures the average cost difference between incidents in systems with advanced security platforms using AI, machine learning, analytics, and automated security orchestration. Fully deployed AI and automation programs were associated with 28 days faster identification and containment of breaches on average, with partially deployed programs also performing significantly better than no automation[8]. See Appendix A for illustrations.

One way to approach the challenge of implementing AI, advanced analytics, and automation is to "buy your way out". Different devices and software produce non-standardized system event logs that need to be normalized and translated to support the needs of detection and analytical functions. Security vendors sell Security Information and Event Management (SIEM) solutions that handle the collection and presentation of data from relevant sources. The vendors create proprietary solutions that solve translation and support analytical functions. More advanced products like Security Orchestration Automation and Response (SOAR) products solve the problem of integrating tools and support higher maturity level functions like automated decision-making and response. However, the "buying your way out" solution does come with problems. Advanced system-wide security solutions in the SIEM/SOAR categories can lead to vendor-lock-in issues. The benefits of an out-of-the-box, fully integrated environment come with the condition that many or all devices, tools, or technologies are vendor-specific. They use proprietary and non-standardized solutions, and specific components can be hard to replace. When a customer wants to integrate a tool or device, either custom-made or from a competitor, they are not guaranteed that the vendor will support integrating solutions, especially if the vendor provides a tool or device that is close to equivalent.

To address the risk of system ossification and to facilitate the orchestration and automation of cyber security functions, it is a goal to standardize the command and control (C2) of devices integral to security operations. It is required that the different functional blocks communicate with each other using a common language. This standard language would need to capture an abstract representation of the function and describe its services in a way that makes them available to other parts of the system. If successful, C2 interfaces can be implemented in accordance with the standard, and plug-and-play interoperability can be achieved. One ongoing effort to create such a standard is OASIS OpenC2.

The OASIS OpenC2 project aims to develop a standardized technology- and vendor-agnostic language for the command and control of devices in security operations. Within the OpenC2 project, there is a sub-committee tasked with creating a threat-hunting actuator profile[9], an extension of the language for incorporating threat hunting. It is possible that the ongoing effort toward defining a threat-hunting actuator profile can address at least some of the challenges described above. The committee has involved threat-hunting experts associated with IBM security, including renowned security researcher and founder of Kestrel, Dr. Xiaokui Shu. It is possible that this cooperation between the OpenC2 technical committee and the Kestrel researchers can produce an information model that satisfies the requirements of a standard for analytics.

1.2 Research questions

The first goal of the thesis is to assess to what extent the ongoing work with developing a threat hunting actuator profile is applicable to the challenges described above, or:

RQ1: To what extent can the OpenC2 Threat Hunting Actuator Profile satisfy the requirements for an abstraction of analytical operations?

After evaluating the ongoing effort, the natural continuation of the work is to address the potential findings by investigating alternative approaches and evaluating them.

RQ2: How could alternative approaches address findings from RQ1 and be more in line with the OpenC2 philosophy and design principles?

1.3 Research methodology

Solheim and Stølen[10] differentiate between classical research method, the formulation of hypotheses, experimenting, observing, evaluating, to understand and explain worldly phenomena, and technology research. Where classical research is mainly motivated by the search for new knowledge, technology research tends to originate from a practical problem, seeking a solution. "Technology research is research for the purpose of producing new and better artifacts" [10].

The iterative process of technology research consists of problem analysis, identifying requirements for a solution, innovation, designing a potential solution, evaluation, testing, and proving whether the requirements are met.

Similarly, this research work evaluates a solution to a part of the problem of facilitating automation, advanced analytics, and AI in cyber defense operations. The part of the problem this thesis focuses on is creating an information model for analytics. We specifically evaluate the ongoing work of creating a threat hunting actuator profile but against the requirements of a profile for analytics. These definitions are discussed in detail in the background chapter.

To ensure that our evaluation is based on a sound understanding of the challenges and the ongoing work, that any potential critique has a basis and is justified, and that our alternative suggestions are credible, we will:

- Participate in OpenC2 technical committee weekly meetings
- Participate in OpenC2 Kestrel Threat hunting actuator profile bi-weekly sessions
- Survey relevant literature

Though it exists plenty of literature on standardization in cyber security, the specific issue of abstracting analytical operations is state of the art. As such, the most relevant information we have found has been in less formal publications, even documents from the working branch of GitHub repositories [9]. We have chosen to incorporate the literature review instead of a dedicated chapter. The main portion supports the background theory chapter for the readers understanding.

We did get a chance to discuss both the threat hunting profile and Kestrel with the founder of Kestrel himself, Dr. Xiaokui Shu. This could have been an opportunity to perform a semi-structured interview to include as part of the thesis. However, unfortunately, this was rather spontaneous and close to the deadline for the thesis, and we were not able to adhere to the academic standards for interview methodology. Instead, we had a casual thirty-minute conversation that, however valuable to the author's understanding of the project, does not meet the ethical requirements to be a part of the thesis.

Chapter 2

Background

2.1 OpenC2

2.1.1 OASIS

OpenC2 is developed and maintained under OASIS, an organization dedicated to open-source development and international standardization. OASIS has, amongst other fields, projects within the cyber security domain. They themselves describe their mission as "to advance the fair, transparent development of open source software and standards through the power of global collaboration and community" [11].

In addition to getting support from universities, like the University of Oslo [12], the OASIS foundation has been largely successful in engaging large tech companies in their work. IBM, Cisco, DELL, Google, and other influential stakeholders in information and communication technologies (ICT) give their support, participate in, and contribute to OASIS projects. One could argue that, in some cases, it is in large tech companies' self-interest to work against standardization. Vendor-lock-in is a situation that benefits the vendor, at least in the short term. Therefore, involving these types of stakeholders in cooperation, in efforts to increase interoperability, is a testament to the authority of a standard, and a key factor for producing standards that eventually get implemented. Amongst other important standards, OASIS is responsible for the development of STIX and TAXII, "the de facto standard for sharing threat intelligence" [13].

2.1.2 Overview

The OpenC2 project is a technology- and vendor-agnostic command and control language for machine-to-machine communication within the cyber defense domain. In OpenC2, producers generate and send commands to consumers, which in turn can execute operations and provide meaningful/informative responses. Messages are constructed as pairs of actions and targets, with the possibility to include arguments. The standard is comprised of several specifications. An introduction to general structures, design models, and guidelines for developing extensions, are documented in the architecture specification [14]. The core of the language is defined in the language specification [15]. Actuator profiles extend the core language to define actuator-specific actions and targets, but also to define the correct interpretation of action:target pairs in the context of a given actuator. Structuring OpenC2 in this manner is a design choice to achieve extensibility and to stay relevant when new functions are introduced. In addition to the approved and published specifications, there is ongoing or planned work on actuator profiles covering a plethora of central functions within the cyber defense domain.

2.1.3 OpenC2 Design Philosophy

OpenC2 is developed according to a set of design principles to guide contributors. Remembering the process of technology research, the principles can be thought of as identified high-level requirements for solutions, guiding the actual innovation, and natural criteria for evaluation.

Consise

Commands should be concise, meaning they should not contain redundant or irrelevant information.

Extensible

Extensibility is an important feature for models in a dynamic and young research field such as cyber security. The model needs to be designed in a way that allows additional currently unknown features to be added without compromising existing work, and without having to build new structures from scratch. This manifests in a modular design with common structures defined in the architecture specification, common features defined in the language specification, and actuator profiles defining function-specific features.

Vendor- and technology agnosticism

To facilitate interoperability it is important that the language is an abstraction of functionality. This means that it needs to be relevant and correct regardless of the underlying implementation.

Information model versus data model

OpenC2 is a collection of information models. An information model is a conceptual abstract representation of entities and their constraints and relationships within a domain. The model focuses on the meaning and context of data and how it is used. Information models guide the process of implementing the modeled function to what may, should, or must be implemented, but not how. An implementation of an information model is called a data model.

A description of a data model is low-level and rich in details, describing how a specific instance derived from an information model was implemented. Several data models can derive from a single information model, as the information model is a higher-level description of the environment, agnostic to the specific tools or software used or developed for the purpose.

2.1.4 Components

This subsection introduces a few terms that have a specific meaning in the context of OpenC2.

Architecture Specification

The architecture specification[14] is an overview and high-level description of OpenC2 components, their structures, and how they relate to each other, such as commands, responses, and types of devices. Further, it explains design models used to develop and extend the language, such as the action-target model and the introspection model.

Language Specification

The language specification[15] defines an abstract set of core functionality, actions, targets, and other type definitions, common to many devices and operations within the cyber defense.

An example is how the language specification defines types for representing points in time, intervals, or durations. Many actuators have this need, defining new types for each actuator would be redundant work, and could potentially lead to misunderstandings.

OpenC2 Actuator

An actuator is the implementation, device, or technology that performs a function in security operations. To put things in perspective, a security ecosystem could have a device functioning as a firewall. That device is an actuator.

OpenC2 Action

Action is a mandatory field in OpenC2 commands and a description of the operation to be performed by the actuator. Commonly a piece of abstracted functionality essential to the actuator's function. Present, in some form, in most implementations. For example, most firewalls have some implementation of rules that "block X". A different actuator might call it to deny. Different syntax, but semantically the same. In an OpenC2 information model, there will be one standardized Action.

OpenC2 Target

The second mandatory field, a target is the object on which the action is performed. The target can be further specified with Target-Specifiers. An example target for the endpoint response action "deny" is "file". The file target can be specified with a Target-Specifier such as a hash of the file.

Example from the endpoint response actuator profile [16]:

```
{
{
  "action": "deny",
  "target": {
    "file": {
      "hashes": {
        "sha256": "0a73291ab5607aef7db23863cf8e72f55
           bcb3c273bb47f00edf011515aeb5894"
      }
    }
  },
  "actuator": {
    "er": {}
  }
}
```

OpenC2 Action:Target pair

A command always consists of an action:target pair. An unambiguous interpretation of a given action:target pair in the context of a specific actuator should be described in the actuator profile.

OpenC2 Command Arguments

A command can be further specified using arguments when necessary. In an actuator profile, allowed arguments for a given pair should be defined in a Command Arguments matrix. Further, the interpretation of an argument being present for a given action:target pair should be defined. Some arguments require specific responses, such as the packet filter profiles "insert_rule" argument requiring the response_requested argument to be "complete".

OpenC2 Example Command

Example from the packet filtering actuator profile[17], serialized in JSON:

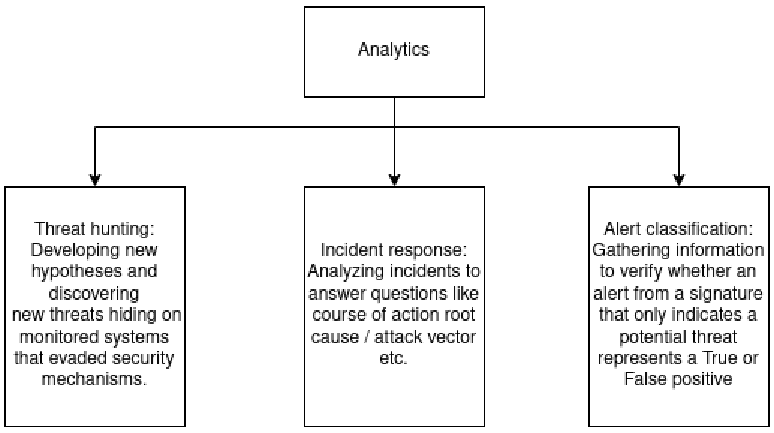
```json
{
  "action": "deny",
  "target": {
    "ipv4_connection": {
      "protocol": "tcp",
      "src_addr": "1.2.3.4",
      "src_port": 10996,
      "dst_addr": "198.2.3.4",
      "dst_port": 80
    }
  },
  "args": {
    "start_time": 1534775460000,
    "duration": 500,
    "response_requested": "ack",
    "pf": {
       "drop_process": "none"
       }
  },
  "actuator": {
        "pf": {
            "asset_id": "30"
        }
    }
}
```

OpenC2 Actuator Profile

An abstraction of the functionality that an actuator provides. Extends the core language specifying actuator-specific components as well as providing detailed and unambiguous interpretations of action:target pairs. This thesis evaluates the ongoing work on a specific actuator profile, the threat hunting profile.

2.1.5 A Profile for Threat hunting

Before we move on to background theory on analytics we explain the connection to the threat hunting actuator profile. Threat hunter is a term used to describe a role in security operations. Performing threat hunting is a process that revolves around ***proactively*** gaining an understanding of the target system, generating relevant threat hypotheses, and testing them against observations of system and network activities [18]. The process can require analytical operations such as data retrieval and data enrichment. Other more ***reactive*** analytical roles and processes, such as incident response and classification of alerts in a SOC environment, often rely on the same operations, only with different motivations.



Analytics
Threat hunting: Developing new hypotheses and discovering new threats hiding on monitored systems that evaded security mechanisms.
Incident response: Analyzing incidents to answer questions like course of action root cause / attack vector etc.
Alert classification: Gathering information to verify whether an alert from a signature that only indicates a potential threat represents a True or False positive

Figure 1: Cyber security roles, or functions, that rely on analytics

Our research question focuses on evaluating the profile's success in abstracting the functionality of analytics. If successful, this could be used to automate several procedures, roles, or responsibilities, within the domain. Not just threat hunting. In terms of naming the profile, the name implies to us that we will attempt to capture an abstraction of analytics because they rely on the same operations as other roles. However, to stay consistent with Kestrel and OpenC2 documentation, we also sometimes use "threat hunting" when we mean analytics.

2.2 Analytics

Analytics can be used to describe processes in security operations that involve retrieving data from a data source, filtering data based on some attribute, sorting or grouping data according to some order description, data enrichment, and visualization. This section briefly covers some important terms, as well as goes a little more in-depth on some fundamental intricacies of this function.

2.2.1 General

Data

Data is context-less low-level material used as basis for analysis. Typically answering 'what'.

Information

Information is organized and structured data. Typically answering 'what' and sometimes 'how'.

Intelligence

Contextualized information. Typically answering ’what’, ’how’, ’who’, and sometimes ’why’.

Security information and event management

SIEM solutions can span several functions within cyber defense. SIEMs can perform log collection. This involves data ingestion, normalization, and indexing. SIEMs can also perform both signature-based and anomaly-based detection, and handle the processing of alerts generated by intrusion detection systems. Another general function SIEM solution can perform is analytics through search and visualization. The indexed data can be queried, inspected, and visualized in dashboards.

Security orchestration automation and response

SOAR solutions are advanced systems for automating procedures in cyber defense.

Orchestration: SOAR solutions tie together several functions. One example is automating the consumption of threat intelligence and ingesting security information. This includes extracting relevant detection signatures and updating rule sets in the monitored environment. This requires common languages and standardized threat intelligence formats.

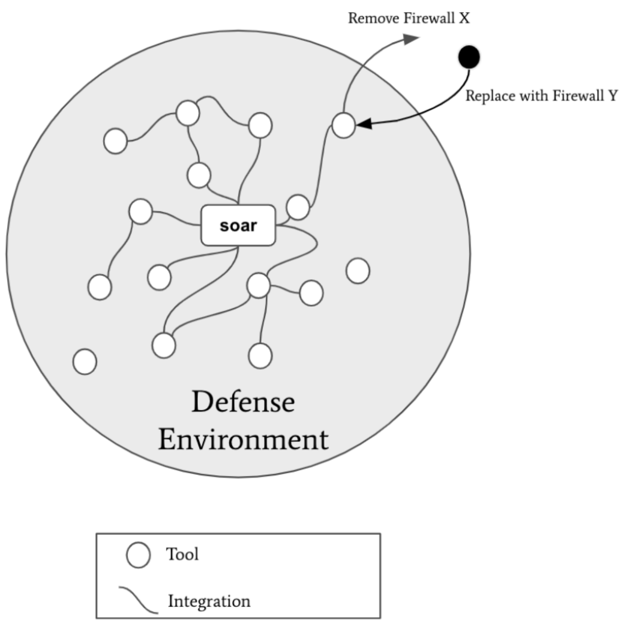


Figure 2: Replacing a firewall in an integrated environment [19]

Response: SOAR solutions can also perform automated decision-making and response to certain events. For example, if a critical vulnerability is announced for a service running on node(s) in a monitored environment, a SOAR solution could have logic to isolate and/or emergency patch without the need for human interaction. When a critical vulnerability for software or hardware is publicly announced a race begins between opportunistic adversaries and defenders to respectively exploit the vulnerability or evaluate exposure and take a course of action. SOAR solutions subscribing to critical vulnerability feeds can match announced vulnerabilities with inventory and perform automated predefined responses.

Automating procedures across functions in cyber defense systems demands standardized and machine-readable actions. The international community of cyber defense researchers is working on a standard model for action playbooks. Collaborative Automated Course of Action (CACAO) is an OASIS project working towards enabling organizations to create and share these playbooks. To put this in the perspective of OpenC2, an orchestrating device could execute a CACAO playbook that involves several actuators, communicating with them using OpenC2 commands. Figure 8 illustrates a use case.

2.2.2 Datatypes

Earlier we defined analytics as a function performing different operations involving data. In this subsection, we cover some different categories of datatypes that are necessary to discuss analytics.

Entity

Kestrel documentation defines entities as "a system, network, or cyber object that can be identified by a monitor" [20]. Entities are the vertices in a graph showing all monitored events within and between systems. Files, processes, hosts, network traffic, etc., are all examples of entities.

Attribute

In this context, attributes are additional information describing an entity. A relevant property. Entity attributes are fields that describe an entity, and therefore activity on a system or systems, and/or relations between systems.

Relation

A relation is a relevant connection between entities. Relations often contain information about why entities exist, and what entity is responsible for an event. This way, relations tie together observable artifacts into a graph that holds the information explaining relevant actions and causal effects both within and between systems. Typically, these relations are directed edges in a graph with natural counterparts such as created and created by. Examples: "process1 created process2", and "process2 created network-trafficA".

Event

In security operations, an event is an action performed by, and including at least one, entity. A record is machine-generated data describing the event, also referred to as an event log entry. Records can include information on more than one entity, such as when a process creates a child- process, in which case both processes are separate entities. An entity can also produce several events. A process can spawn more processes, create files, read files, and create network connections. These are all typical examples of system events that monitoring software will create records of.

In some cases, such as the case with network traffic, also called network flow data, the entity versus event differentiation is somewhat unclear. Typically flow data contains two entities, source and destination IP addresses are regarded as host entities. Flow data typically contain information such as the point in time the traffic started, network ports, the number of bytes sent, protocols, etc. As such, this datatype seems to be a perfect example of how we describe events. However, network traffic can also be considered an entity because it is something that can be observed by network monitoring devices, and that itself can relate to other entities.

An important insight is that records of events are a natural choice of the type of data we wish to create logs of in order to store the information of what happened in a monitored environment. The reason is that events are limited in extent by nature. The ability to investigate all information on a specific entity is valuable to analytics, but storing this information for all entities in one place is impractical in real life scenarios. For example, some processes persist for a long time, at what point should a log be made? What if the system crashes? There are many reasons why event logs are the obvious choice for storing historical information in monitored systems. Alternatives to event logging, such as graph databases of entities, could be a supplementing alternative but unlikely a replacement.

2.2.3 Data storage in cyber security

The following very brief introduction to database models explains a typical trade-off dilemma in the choice of data models and an important source of complexity in analytics in security operations.

Relational

Relational databases organize data in tables usually with columns describing variables and each row an instance of the data stored. The way the database is designed can follow norms called normal forms. The level of normalization of the data relates to the possibility of the tables containing redundant or wrong information. For example, 1NF, or first normal form, requires that a column can not contain more than one value. If there is a need to store more than one value for a type of relation to an entity then the data will need to be normalized. Meaning a separate table will have to be made with multiple rows indicating relations.

Simple example:

Table 2: Multiple values in a column (NOT 1NF)

uid	pid	binary_name	Network_Traffic_UID
123123	1234	cmd.exe	ABC,BCD,CDE

Table 3: Extract the relations

uid	network_traffic_uid
123123	ABC
123123	BCD
123123	CDE

Table 4: Example network traffic

network_traffic_uid	src_ip	src_port	dst_ip	dst_port
ABC	1.1.1.1	55550	2.2.2.1	80
BCD	1.1.1.1	55551	2.2.2.2	80
CDE	1.1.1.1	55552	2.2.2.3	80

Failure to normalize relational databases can result in insertion anomalies, deletion anomalies, or update anomalies. Consequentially the data stored in a relational database have to be structured according to a defined schema. An example where this can be impractical is if you want to collect event logs from a number of different devices that might support a varying range of debug and log features. Forcing the logs to fit a relational database might be disadvantageous because of the information that would need to be excluded in order to normalize.

Scaling

Relational databases scale vertically, typically by adding more resources such as memory and computational power to a single server. This can eventually become very cost-inefficient compared to a distributed design. Non-relational databases, like document storage, are designed to scale horizontally, meaning they can be distributed over several servers. The distributed design is preferable for use cases with high demands for performance and availability, which can be the case for security operations, and monitored environments. With a distributed design you sacrifice some of the strong guarantees the relational database gives when it comes to data consistency to be able to handle larger amounts of data.

JOINs

Relational databases are optimized for complex queries requiring operations such as SQL JOINs. The analytical function often requires such queries to be answered. Take the example from the tables above. Say we get a tip that ip-address 2.2.2.1 has been involved in malicious activities. We query for network traffic to this IP with "SELECT * WHERE dst_ip='2.2.2.1';". Now, we want to know more about the process that created this traffic to understand if our traffic is related to the malicious activities we received a tip about. To answer the question what is the process id of the process that created network traffic 'ABC', we need to join the tables. Luckily, relational databases are optimized for these types of operations, whilst non-relational databases are not.

This is a significant downside to non-relational databases. The distributed design makes join operations, nested complex queries with more than one hops in relations, "prohibitively expensive" [21]. For example "GET outgoing network traffic created by some processA created by some processB with (binary name not equal to 'WinMail.exe' AND created by a process with binary name equal to WinMail.exe)", see Figure 4 for illustration of pattern. Depending on the size of the queried storage inefficient beyond tolerance levels. A remedy can be fetching data from a distributed document storage (hunt)step by step, normalizing the data if needed, and building temporary relational databases for use in analytical processes.

Summary of data storage

For security operations, there is a trade-off between non-relational and relational databases. The trade-off is in terms of handling large amounts of schema-less and unstructured data versus handling complex pattern matching on the same data. Non-relational databases are not optimized for advanced analytics. On the other hand, it is not realistic to maintain a relational database for log collection, and monitoring for large environments in the long term. However, temporary relational databases can provide valuable functionality to analytical processes.

2.2.4 Patterns and other expressions for data description

When describing an entity we use patterns. Patterns can be comprised of combinations of small comparison expressions, combined by operators, and described by other qualities. We use patterns to describe how we want to filter data retrieved when we query data repositories. The pattern is matched against entries and positive matches are returned. To explain these terms in more detail we use the STIX framework definitions of patterns and their components.

STIX comparison expression

An expression stating a value for an attribute. Used for matching events or entities, they are the simplest form of filtering. Here presented as a STIX comparison expression, but in general composed of an attribute specified entity:attribute, a comparative operator such as equal to, not equal to, less than, etc., followed by a value valid for the attribute. Conventionally, indicators of compromise (IoC) is a term used for simple attribute values associated with some suspicious behavior, worthy of further investigation, but often lacking further information or intelligence.

Example: [ipv4-addr:value == '198.1.1.1/32']

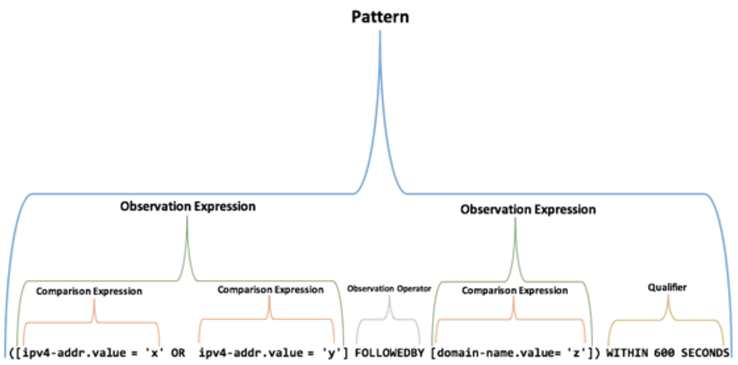
In the example, we have the format entity:attribute operator 'value'.

STIX observation expression

To express the concept of patterns in activity the STIX framework uses Observation expressions. These are composed of one or more of the simpler, more atomic comparison expressions. Several comparison expressions can be combined using operators such as OR/AND. Several observation expressions can also be combined by observation operators such as 'FOLLOWEDBY'. Finally, qualifiers can be included to express qualities such as the number of repetitions within a certain time frame. Combined following a set of rules these concepts form a pattern.

STIX pattern

A STIX pattern can include one or more STIX observation expressions combined by observation operators and can include a qualifier as the example shows. Conventionally, indicators of behavior (IoB) is a term used for larger patterns describing known or suspected malicious behavior.



```
([ipv4-addr.value = 'x' OR  ipv4-addr.value = 'y'] FOLLOWEDBY [domain-name.value= 'z']) WITHIN 600 SECONDS
```

Figure 3: STIX 2.0 Part 5: STIX Patterning [22]

2.2.5 Entity graphs

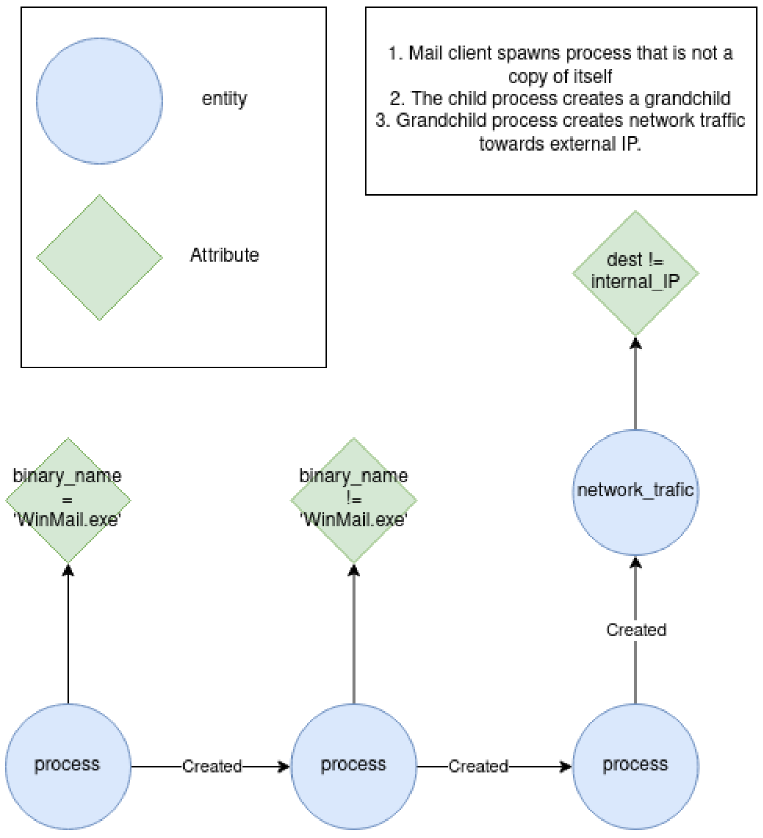
A common way to illustrate and reason around entities is using graphs. The following are some common types of graphs used in analytics.

Single node graph pattern

The single node is an entity. There are no edges or other vertices. The node can have labels describing attributes with values belonging to the entity.

Extended centered graph pattern

ECGP is a way to represent IoBs, a graph centered around one entity. Its described using its attributes, but also relations to other entities and possibly their attributes. A combination of comparison expressions describing specific events or entities.



entity
Attribute
1. Mail client spawns process that is not a copy of itself
2. The child process creates a grandchild
3. Grandchild process creates network traffic towards external IP.
dest != internal_IP
binary_name = 'WinMail.exe'
binary_name != 'WinMail.exe'
network_trafic
Created
process
Created
process
Created
process

Figure 4: Complex pattern illustrating EGCP / IoB

Property graph

A property graph can consist of vertices with labels describing that instance of an entity. Relations between entities are represented by the edges which are both directed and can have labels or properties. In other words, property graphs are graphs where both vertices and edges can have both labels and attributes.

2.2.6 Analytical operations

So far, we have covered some concepts and structures necessary to perform analytics in security operations, but we have not yet fully defined analytical processes. As a reminder, these processes or operations make up the functionality we expect a comprehensive version of an OpenC2 actuator profile for analytics to capture.

We have mentioned data retrieval. Any data repository needs to be able to output the stored data. We have discussed some categories of databases and how different models can have benefits and drawbacks when it comes to complexity in queries. Data transformation is the sorting or grouping of data. Data enrichment is processing data with external analytics, using for example threat intelligence or machine learning to add information. Data visualization and inspection functionality also fall under analytics. These categories are used by Kestrel to group some of their commands together.

2.3 Kestrel

Kestrel is an open-source project under the Open Cybersecurity Alliance (OCA) consisting of two parts. The Kestrel threat hunting language, and the Kestrel runtime software. It is maintained by IBM security researchers with two main contributors. One of which is the founder Dr. Xiaokui Shu, who has also demonstrated Kestrel at security gatherings such as Blackhat 2022[23]. In this section, we give a brief introduction to the most relevant features.

2.3.1 Overview

First, a short description of how Kestrel fits in security operations, see figure 5 for illustrations. As a high-level example: take A) monitored devices producing raw logs, B) a system for collecting and storing the logs in some data repository, and C) analysts crafting and sending queries to the data repository to perform different analytical tasks. Kestrel runtime will fit between the analyst and the data source. Instead of crafting queries in the query language native to the data source, the analysts will use queries written in the Kestrel language. The Kestrel runtime will parse the query and its internal logic decides how to hunt. The Kestrel runtime crafts queries for a STIX connector in order to receive back a STIX bundle of observed objects. This requires that there exists a STIX connector for the data source in question.

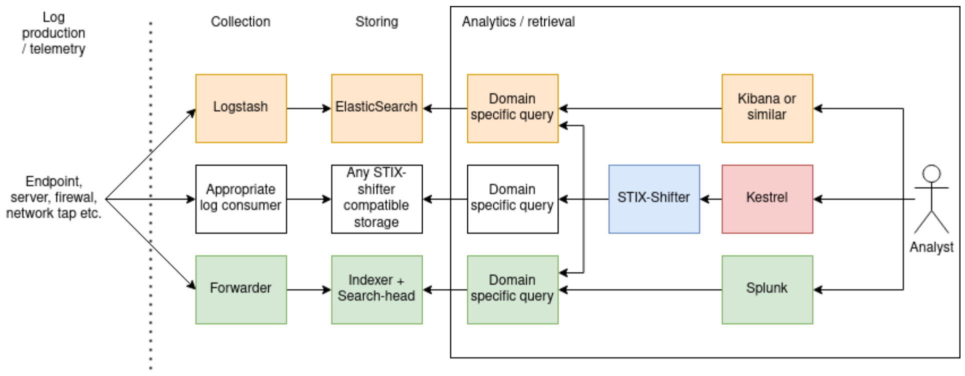


Figure 5: Kestrel in the analytics stack

2.3.2 Kestrel Threat Hunting Language

To our knowledge, Kestrel threat hunting language is the first language designed specifically to perform cyber threat hunting. In the following, a few terms defined in the Kestrel documentation will be explained.

Hunt

A threat discovery process, built up of one or more hunt steps. "A procedure to find a set of entities associated with a threat." [20]

Kestrel variables

Kestrel variables point to table-like data structures describing a homogeneous set of entities. Many Kestrel commands have Kestrel variables as output, some have Kestrel variables as input.

Hunt Step

A huntstep is a single Kestrel command. The syntax and vocabulary seem inspired by both SQL and Python. "variable = GET process FROM [datasource] WHERE [pattern]."

```
svchost = GET process
          FROM file:///tmp/lab101.json
          WHERE name = 'svchost.exe'
          START 2021-04-03T00:00:00Z STOP 2021-04-03T02:00:00Z
```

Figure 6: Kestrel huntstep [24]

Huntflow

Several hunt steps combined is also called a hunt flow. A huntflow can be expressed in huntsteps and stored in a Kestrel Huntbook. Flows can be split and merged. See figure 7 [20].

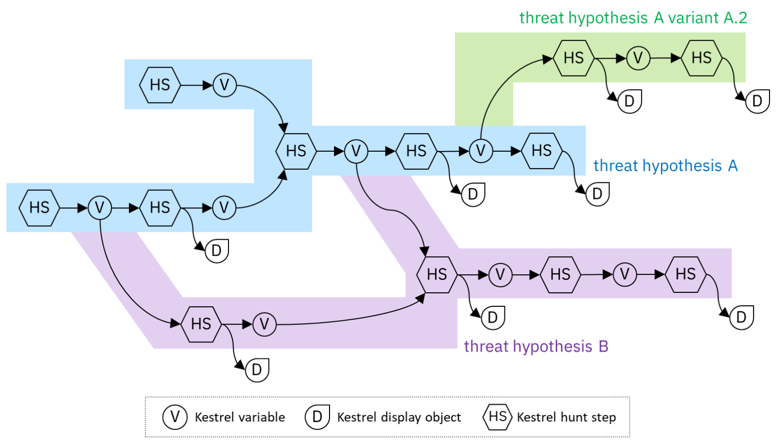


Figure 7: Kestrel huntflow [20]

2.3.3 Kestrel runtime

The Kestrel runtime is written in Python. It interprets commands written in Kestrel language and decides *how* to execute the hunt.

Frontend

Kestrel has several user interfaces such as a command line interface and python API. It can also be used interactively with Jupyter Notebook, and has the functionality to export Kestrel huntbook files under *File>Download>Kestrel (.hf)*. The huntbook files can be stored and executed as a high-level program language for intrusion detection.

Backend

The runtime interprets commands, executes queries, and caches results using Firepit, relational database temporary storage to allow faster processing of results, as well as smart pre-fetching of data to fill information gaps on entities already in storage.

At the time of writing, Kestrel has two ways of ingesting data; either through a STIX bundle, which can be stored locally or remotely, or through a STIX-shifter connector data interface.

2.3.4 STIX-Shifter

The STIX-Shifter[25] project is part of the Open Cybersecurity Alliance[26], which is an OASIS Open Project. STIX-Shifter is an open-source Python library defining field mappings from STIX patterns to a wide range of data repositories used in monitored environments. With many of the most popular data sources in cyber defense having their own STIX-shifter module, called STIX connectors, STIX-Shifter works as a federated search engine, allowing the same STIX pattern to be translated into many different query languages.

STIX connectors take STIX patterns as input, translate the pattern to the underlying technology's domain-specific query language, and output a STIX bundle object containing the data resulting from the query translated into STIX Observed Data objects.

It's important to note, that while it might exist STIX connectors for many of the most relevant data sources for cyber security operations today, the level of functionality you get access to for each of the data sources is restricted by the respective connector. Meaning, that you are likely to have less than full access to the normal functionality the repository provides. You might not have guarantees that the STIX connector is updated and maintained at the same pace as the underlying technology. This problem is passed along to the Kestrel runtime as the STIX-shifter data source interface is currently the only supported way to consume data, in addition to STIX bundle data source Interface.

STIX Cyber Observable object (SCO) [27]

An SCO holds a piece of structured data documenting something that existed at a point in time on a system or network. An example of captured data is the SCO File object, which must include the id and at least one of two fields filename or hash-value. In addition to the required fields, there are many optional fields that describe the object.

The SCO is a piece of *data*, which means it does not necessarily contain explicit information regarding who is responsible for something or when something happened, and "never the why" [27].

An SCO is often contained within a STIX observed data object when the data collector/producer has more information to communicate.

STIX Observed Data object [28]

The STIX observed data object is a STIX domain object (SDO) used to convey information about SCOs. The object can contain one or more SCOs as well as information such as timestamps for first seen, last seen, or the number of times observed. This way the object can be used to aggregate information. For instance, a firewall could output logs as a list of all SCOs seen at a time interval. However, its possible to aggregate the information by wrapping the SCOs in an Observed data object and including a "number of times observed" value. If a firewall-type device is experiencing a large number of observations related to a handful of IPv4 addresses, this information can be communicated efficiently, with fewer bytes, when using the STIX observed data object.

To be precise, a STIX observed data object must not be mistaken as intelligence. The object itself does not contain rich contextual information, only raw information about SCOs. It is not an attempt to communicate meaning, it does not describe why certain phenomena are being observed.

STIX bundle object

A STIX bundle object is a list of arbitrary STIX objects. It's important to note that no meaning can be inferred from the presence of an object or a combination of objects in a given bundle. The bundle conveys no semantic meaning, it is simply a container for objects. As such it is a flat structure. Objects within the same bundle could be related to each other, but information about any such relations must be conveyed by use of other types of objects such as STIX relationship objects.

2.4 Chapter Summary

This chapter has provided an overview of the OpenC2 project. As this thesis focuses on the challenge of extending OpenC2 to include an abstraction of analytics this chapter also hopefully covered how we define analytics, the basic building blocks, and why it might not be as straightforward to capture as other functions within the cyber security domain.

Further, as the committee has involved threat hunting experts and security researchers from Kestrel to develop the actuator profile this chapter has introduced some of the theory behind Kestrel and terms relevant for later discussion.

What follows is a chapter addressing our first research question, an evaluation of the ongoing work with the threat hunting actuator profile.

Chapter 3

The Threat Hunting Profile

RQ1: To what extent can the OpenC2 Threat Hunting Actuator Profile satisfy the requirements for an abstraction of analytical operations?

3.1 Overview

When it comes to extending the core language and defining actuators for specific functions it seems OpenC2 so far has found the most progress in packet filter-related functions. With a published stateless packet filtering actuator profile[17], and in progress with stateful packet actuator profiles, intended to be combined in the future[29]. Further, the technical committee has worked on an actuator profile for endpoint detection and response[16], efforts led by Martin Evandt, a master's student from UIO, and his supervisor Vasileios Mavroeidis. Their work, which was to be combined with writing a master's thesis, showed that it can be useful to define endpoint detection and endpoint response as separate functions. That endpoint detection's operations could in some cases have more in common with an analytical-type function.

To generalize it seems the OpenC2 project so far has seen mostly progress towards creating actuators that actuate simple atomic commands, changing the state of devices, and in turn responding with success or failure as results. Less efforts have so far been directed at analytical functions. This is a conscious choice by the committee, stating in the language specification that "aspects of coordinated cyber response such as sensing, analytics, and selecting appropriate courses of action are beyond the scope of OpenC2". However, now that the project has found success in abstracting some of the classical atomic operations it might be time to investigate more complex functions. There exist many types of data storage solutions and with them domain specific query languages. It is outside the OpenC2 scope to define how data should be stored, but the communication to and from a data source can be argued to be within the scope.

This chapter introduces the ongoing work with creating a OpenC2 threat hunting actuator profile. It investigates to what extent the profile addresses the need for a profile for analytics, and how well the profile adheres to OpenC2 philosophy. As a point of reference we compare the ongoing work with previous profiles.

3.2 Disclaimer

The work on the threat hunting actuator profile is an ongoing effort. We have to use the most recent data we have available, which might not be representative of the final product but a step in the direction of what the technical committee wants to achieve. OASIS enforces requirements for procedures regarding the standards developed under its name. This provides quality assurance for the product and credibility to the OASIS name brand. Therefore as the work is yet to be published, it has also yet to go through the entirety of due process including presentation for review and public comment. The product is however reviewed internally within the sub-committee. To get more information than can be read from the public GitHub we have participated in OpenC2 meetings as well as reviewed recordings of working sessions between the technical committee for the threat hunting actuator profile and the Kestrel team. Further, through a one-on-one interview with Dr. Xioakui Shu, we learned more about his perspectives and motivation for contributing to the efforts to develop OpenC2. This is how we have achieved our understanding of the direction the work is taking at the time of writing, or in other words, this is what we base the analysis and evaluation in this thesis upon.

3.3 How its made

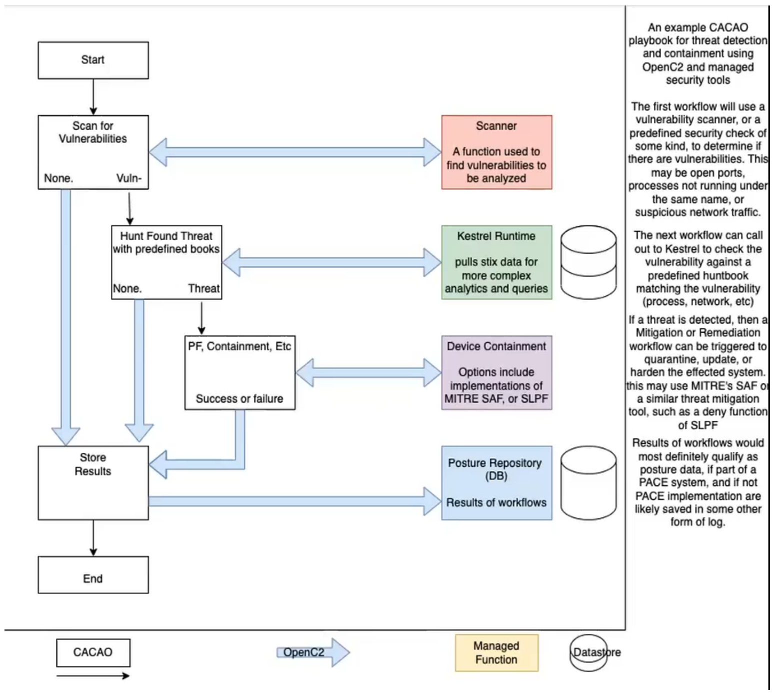
3.3.1 Participants, forum

OpenC2 is developed through OASIS technical committee (TC) program. In the TC program members develop specifications through a defined open process. Subcommittees are appointed if dividing the work is necessary, such as for the threat hunting AP. The TC meets every week to present progress on the different work items. Once a month, the weekly meeting is a voting meeting to vote on matters such as approving pull requests, voting to send matters up a level to OASIS, etc. All the different working items, tools, or documents under development have a person in charge, often called maintainers, as well as contributors. In terms of the threat hunting actuator profile, Mr. David Lemire is assigned the role of "contributor". He usually takes care of updating the rest of the OpenC2 TC on developments in the threat hunting subcommittee. Mr. Lemire's contributions to the OpenC2 initiative are substantial. Though he is the only publicly assigned contributor in the official GitHub repository there are multiple people involved with developing the schemas for the threat hunting actuator profile, and a larger international community participates in discussions.

In addition to the OpenC2 working meetings, the bulk of the actual work towards producing a document that meets OASIS's standards is done by the individual contributors, guided and consulted by a mix of experts who meet for bi-weekly two-hour discussions. The meeting is often called OpenC2 Hunt AP (Kestrel) meetup, as it is hosted by threat hunting experts associated with IBM, and developers and maintainers of the Kestrel software and threat hunting language. In addition to OpenC2 members and the Kestrel team, contributors to the Open Cybersecurity Alliance project are welcome to contribute to the discussion.

3.3.2 Design Goal

The ongoing work with the threat hunting actuator profile started in 2022. Their vision from the start has clearly been focused on facilitating the automation of defensive processes, as can be seen from this early-stage use case diagram. See Figure 8, borrowed with permission from the OCA. The use case involves a CACAO playbook using OpenC2 commands to execute a vulnerability scan, followed by targeted threat hunting, as well as other possible functions.



An example CACAO playbook for threat detection and containment using OpenC2 and managed security tools

The first workflow will use a vulnerability scanner, or a predefined security check of some kind, to determine if there are vulnerabilities. This may be open ports, processes not running under the same name, or suspicious network traffic.

The next workflow can call out to Kestrel to check the vulnerability against a predefined huntbook matching the vulnerability (process, network, etc)

If a threat is detected, then a Mitigation or Remediation workflow can be triggered to quarantine, update, or harden the effected system. this may use MITRE's SAF or a similar threat mitigation tool, such as a deny function of SLPF

Results of workflows would most definitely qualify as posture data, if part of a PACE system, and if not PACE implementation are likely saved in some other form of log.

Figure 8: Use case diagram, CACAO and OpenC2

This figure was presented and discussed during a bi-weekly meeting between OpenC2 and Kestrel. Enthusiastic comments mentioned how early visions are still relevant. This particular use case is a fully automated process using pre-defined pattern matching.

It is important to note that while the figure shows all machine-to-machine communication as OpenC2, the threat hunting actuator profile only covers the communication between the Kestrel Runtime and the CACAO security-action labeled "Hunt found Threat with predefined books". We can gather from this that there is information in the output from the previous CACAO security-action which influences which huntbooks are invoked.

3.4 Analysis

This section covers the current state of the threat hunting actuator profile. It includes an account of the tables in the working schemas and comparisons to other actuator profiles. First, we present the list of actions and their descriptions. Next, we look at action:target pair matrices.

3.4.1 Type: Actions enumerated

Table 5: Action Enumerated Threat Hunting AP

ID	Name	Description
3	query	Initiate a request for information.
30	investigate	Task the recipient to aggregate and report information as it pertains to a security event or incident.

Table 6: Action Enumerated Endpoint Response AP

ID	Item	Description
1	scan	Initiate a scan for binaries classified as malicious.
3	query	Query the ER actuator for a list of available features.
6	deny	Deny a process or service from being executed on the endpoint.
7	contain	Isolate a device from communicating with other devices on a network, quarantine a file.
8	allow	Un-isolate a previously isolated device.
9	start	Initiate a process, application, system, or activity.
10	stop	Halt a system or end an activity.
11	restart	Restart a device, system, or process.
15	set	Change a value, configuration, or state of a managed entity (e.g., registry value, account).
16	update	Instructs the Actuator to retrieve, install, process, and operate in accordance with a software update, reconfiguration, or other update.
19	create	Add a new entity of a known type (e.g., registry entry, file).
20	delete	Remove an entity (e.g., registry entry, file).

Comparing the valid actions for the threat hunting AP to previous actuator profiles it's clear that the threat hunting AP is more limited. Remembering that this work is ongoing, and especially matters such as providing precise descriptions might have been down-prioritized, it's still relevant to notice

Table 7: Action Enumerated stateless packet filter AP

ID	Name	Description
3	query	Initiate a request for information. Used to communicate the supported options and determine the state or settings
6	deny	Prevent traffic or access
8	allow	Permit traffic or access
16	update	Instructs the Actuator to update its configuration by retrieving and processing a configuration file and update
20	delete	Remove an access rule

that the action investigate is a rather general concept. One could argue an investigation is more a process than an action.

3.4.2 Command Matrix Threat Hunting AP

An OpenC2 command matrix shows valid action:target pairs for a given actuator profile. Further details and defined interpretations of the pairs are defined in dedicated sections of the schemas.

Table 8: Threat hunting AP Command Matrix

	Features	Datasources	Huntbooks	Hunt
Query	Valid	Valid	Valid	
Investigate				Valid

Table 9: Endpoint Response AP Command Matrix

	domain_name	device	features	file	ipv4_net	ipv6_net	process	registry_entry	account	service
query			x							
deny	x			x	x	x				
contain		x		x						
allow	x	x		x	x	x				
start				x						
stop		x					x			x
restart		x					x			
set					x	x		x	x	
update				x						
create								x		
delete				x				x		x

The point of showing these tables is to compare previous OpenC2 actuator profiles with the ongoing work. It's important here to note that the query action is an introspective query, simply allowing the consumer to inform about its capabilities and available resources. This leaves the threat hunting AP with a single valid command for performing an actual operation. For comparison, the ER AP has 22 valid action:target pairs in addition to the introspective query:features commands.

3.4.3 Pair descriptions

The introspection-model requires the actuator to have functionality to inform about its capabilities. This is accessed using the query action.

Query:features

This pair must be defined for all actuators. A device is required to be able to communicate its own features. While not defined for the threat hunting AP yet, we can see from other APs that this will return which actuator profiles are available on this device (as some devices perform more than one role) and information about what other commands/pairs are valid.

Result: Information about actuator features, valid action:target pairs

Query:Datasources

Building on the goal of introspection, a threat hunting actuator is required to be to be able to inform about the various data sources that can be queried from the consuming device. This command returns a list of data sources. The individual data sources can be specified as arguments in investigate:hunt commands.

Result: Type:Datasource-Array, data source names, and information

Query:Huntbooks

This command returns a list of huntbooks available to the consumer. The individual huntbooks could potentially be used as arguments in investigate:hunt commands. Based on our understanding of their intentions, the technical committee intends to add functionality for filtering huntbooks using tags. The tags could describe either a phenomena that triggered the investigation such as high-CPU-usage anomalies, or specific types of threats or vulnerabilities.

Result: Type:Huntbook-Info, Structured data about huntbooks

Investigate:Hunt

The target "Hunt" is described in the documentation as "a procedure to find a set of entities in the monitored environment that associates with a cyber threat" [9]. The purpose of this action:target pair is to invoke huntbooks. This seems to be the only action:target pair which in fact commands an actual operation under the function the AP represents.

This command will tell the Kestrel runtime software to run a specific huntbook, comprised of one or more huntsteps. The response will be a STIX bundle containing the STIX SCOs resulting from the huntflow.

3.4.4 Threat hunting AP arguments

An OpenC2 message can contain arguments. An actuator profile can define actuator specific arguments. The language specification defines OpenC2 arguments as properties that provide additional information on how to perform a command. Some standard arguments will likely be relevant and included such as the response-requested argument. It can be used to define what type of response is to be generated. Is the data that was retrieved from a query desired in the response? Or is a boolean type describing whether the hunt result was empty or non-empty enough? As such well-defined arguments can open for implementations to have higher efficiency, in this example by ending the operation early upon some condition.

The timestamp-type or time range-type arguments will be necessary to define in order to filter event log searches. Some use cases might encourage storing logs going years back, as certain APTs can remain hidden in systems for up to several years [6] stealing information, while they develop exploits to elevate privileges in or cause harm to, the systems they have infiltrated. This makes time range filtering a required option when performing data retrieval operations as searching the entire data repository will be prohibitively slow.

The threat hunting AP will define argument types specific to this actuator and the interpretation.

Table 10: Arguments in the threat hunting AP

ID	Name	Type	#	Description
1	string_arg	String	1	string arguments supplied as huntargs.
2	integer_arg	Integer{0..*}	1	integer arguments supplied as huntargs.
3	stix	STIX-Array	1	STIX arguments supplied as huntargs.
4	timeranges	Timeranges	1	Timeranges used in the execution of a hunt.
5	datasources	Datasource-Array	1	You must identify one or more available data sources for hunting. These may be a host monitor, an EDR, a SIEM, a firewall, etc.

The arguments will be necessary to provide huntbook input requirements. For example, when searching for a process with a given attribute, the value of the attribute to be used in a comparison expression on which to filter has to be provided to the consumer using arguments. In the following example, an argument is provided (uuid) and identified as "hunt_process". This has to be in response to a description of the huntbook stating the identifier and the expected type. An alternative that might provide simplification is describing a huntbook with a count for required arguments, and a sorted list with descriptions. This will have to be defined in the AP in the future.

Example command showcasing huntargs:

```
{
  "action": "investigate",
  "target": {
    "th": {
      "hunt": {
        "path_relative": "path/name/example"
      }
    }
  },
  "args": {
    "response_requested": "status",
    "th": {
      "huntargs": {
        "timerange": {
          "timerange_relative": {
            "number": "15",
            "time_unit": "Minutes"
          }
        },
        "datasource": "Datasource_Name",
        "hunt_process": {
          "uuid": "1234567890"
        }
      }
    }
  }
}
```

3.5 Discussion

3.5.1 Comparing the threat hunting AP to other APs

When we compare the current state of the ongoing work with the threat hunting actuator profile with other earlier profiles such as Endpoint Response AP, and stateless packet filter AP, it appears that the previous profiles are more successful in capturing the semantics/meaning of atomic cyber operation procedures. Earlier profiles include more actions, and the actions are conventionally related to the abstracted function it represents.

The tasks performed are not any less complex, rather on the contrary. The meaning of the commands is instead shifted from being contained in the OpenC2 message to being contained in the huntbooks. The abstraction is performed by the Kestrel threat hunting language. Further, an argument could be made that the complexity is shifted additionally one step by the Kestrel runtime, which is currently the only intended consumer for the threat hunting AP commands. The Kestrel runtime software parses the Kestrel commands. The Kestrel command is not necessarily one query, the runtime might perform many queries, and it might write queries to pre-fetch information about entities related to entities that are actual direct results from the query. In the case of other actuator profiles, the semantics is successfully captured.

3.5.2 Technology agnosticism

OpenC2 is supposed to capture the semantics to express what *should* be possible within a function, leaving the *how* to the actual implementation. The goal is to facilitate automation, interoperability, and to prevent vendor-lock-in issues. To succeed in these goals developers of the actuator profiles have to abstract the functionality commonly found in current implementations, while also attempting to stay extensible to accommodate innovation.

Instead, the threat hunting actuator profile seems to rather be tailored towards interacting with Kestrel runtime. It is a software that according to the GitHub repository contributions [30] is written and maintained to a large degree by only two persons. If Kestrel for whatever reason stops being maintained, the actuator profile has no consumers as far as we can tell. In fairness, the Kestrel team could be correct when stating that there exists no technology that does what Kestrel does. A counterpoint is again that no SIEM solution is exactly the same. The relevant question to ask is what functionality does Kestrel provide to make it deserving of its own actuator profile? Whether the AP has utility is currently 100% dependent on how useful Kestrel is. This should however imply that Kestrel's unique functionality could be interesting to abstract into OpenC2. Instead, the hunting profile is limiting itself to the simple function of invoking Kestrel.

Kestrel's utility is again dependent on STIX-shifter to access data repositories. If the data storage does not have a STIX-shifter module then Kestrel can not talk to it, and by the transitive property, neither can OpenC2. When STIX-shifter modules translate STIX patterns to other domain-specific languages and convert the results to STIX objects, we achieve something that is key to abstracting analytics. A common structure for the data, and a common language to write patterns. However, when the threat hunting actuator profile is implemented to have Kestrel as an intermediary technology, it could paradoxically make the abstraction that STIX provides less available. A complete version of the threat hunting profile, as its currently headed, will require support for STIX objects.

3.6 Summary

The ongoing work we have examined is presented as a threat hunting actuator profile. However, the actions included so far only cover the invocation of pre-defined sequences of Kestrel commands. The Kestrel language and runtime provide a plethora of analytical functionality that is not captured in the actuator profile. The fact that this functionality can be accessed, and automated, is of considerable pragmatical value. On the other hand, it is not meeting the expectations we have for a profile named threat hunting actuator profile.

Capturing analytical functionality has proven before to be a challenging task. It might even prove to be impractically hard to the extent that no "perfect" one-size-fits-all solution exists. However, the next chapter of this thesis will attempt to investigate if modifications to the current efforts can lead to steps in the direction of addressing the issues we have pointed out in this chapter.

Chapter 4

Investigating principled approaches

RQ2: How could alternative approaches address findings from RQ1 and be more in line with the OpenC2 philosophy and design principles?

4.1 Overview

In this chapter, we explore alternative approaches to a profile for analytics that is more in line with the OpenC2 philosophy and design principles. Our intentions are to expand on the ongoing work of creating a threat hunting actuator profile. More specifically we want to see how modifications can be made without disrupting the valuable work they are doing by facilitating automated invocation of huntbooks. We investigate different approaches to address the points presented in Chapter 3 and discuss their benefits and drawbacks.

4.1.1 Disclaimer

We do not have the resources to produce a comprehensive solution. Instead, we attempt to describe what a principled approach could look like. We do not claim our descriptions are easily developed or that the end result necessarily must be better than what is currently being produced. We simply explore alternatives that potentially make less compromise on OpenC2 principles and evaluate the consequences of these approaches.

4.1.2 Evaluation

We compare our alternatives to the ongoing work and evaluate their differences under the following criteria:

1. How well are analytical operations abstracted?
2. To what degree is the solution dependent on external technology?
3. How much work would it take to comprehensively develop the proposed changes?

4.2 "A better shortcut"

Our first option involves adding a second target "hunt step" with the required fields "language" and "query". As a follow-up, data sources could return the required language in query:data_source responses. If the actuator is Kestrel runtime, you can proceed to invoke huntbooks. Otherwise, the producer can use the information in the response to produce the next commands.

4.2.1 Define atomic huntstep target

The ongoing work facilitates the invocation of huntbooks, comprised of one or more hunt steps. This enables OpenC2-compliant orchestrators access to Kestrel's predefined hunt flows. It does not abstract analytical functionality. Since OpenC2 aspires to be technology agnostic, this begs the question of why it shouldn't facilitate predefined queries in other languages as well. We argue that simply invoking Kestrel huntbooks is a shortcut, and we ask if there exists a slightly better shortcut.

One approach to facilitate more languages would be to define a new target, the huntstep. An Investigate:huntstep command would send encoded predefined queries instead of invoking predefined huntbooks stored at the consumer. When invoking a huntbook, the target is defined as 'hunt'. The hunt target requires an ID as an argument to identify which huntbook to invoke, and the huntbook will contain hunt steps in the Kestrel language. When using the new target, the producer will explicitly send the hunt steps in the command. Critically, the hunt steps, or queries, do not necessarily have to be described in Kestrel. Instead, we open up to other languages as well in an attempt to be more technology-agnostic.

Example:

```
{
  "action": "investigate",
  "target": {
    "th": {
        "language":"SQL"
        "query":"BASE64-encoded SQL query=="
    }
  }
  "args": {
    "response_requested": "complete"
  }
}
```

4.2.2 Evaluation

This solution provides practically no abstraction of analytical operations. We are submitting to include the syntax defined for the underlying technologies we wish to perform the operation. On the other hand, the ongoing work, as we have argued, is tailored to a specific technology's syntax. The proposed change takes very little work to include and has no negative impact on the ongoing work.

4.3 Abstracting an analytical operation

In this section, we describe what a committed principled approach to an attempt to abstract an analytical operation could look like.

4.3.1 Actions

OpenC2 actuators should abstract meaningful functionality. The architecture specification says that an action should represent the task or activity [14]. The Kestrel threat hunting language has commands that provide different categories of operations, 4 of which could be considered analytical operations. In our opinion, this is an excellent place to start when thinking about what actions need to be defined for analytics.

1. Retrieve (Get, Find)
2. Transform (Sort, Group)
3. Enrich (Apply external analytics)
4. Inspect (Display, Metadata, Dashboards)
5. Flow-control (Save, Load, Session-handling)

4.3.2 Targets

The target should refer to the object the action is performed on. There are several candidates for targets. It's one of the many facets of analytics that is hard to abstract. To start with the most basic data retrieval operations. Following the definition from the architecture specification, the data retrieval is performed by a data repository. This is a more concrete mindset, connecting the data retrieval operations to what is performing the operation. However, in real-life scenarios, this makes little sense as the data sources available might be many, and of different types.

We can again look to Kestrel for inspiration, or even a more classic/well-adopted query language like SQL, which has influenced the Kestrel query language. In SQL you specify which columns to pull from rows in tables. In the "SELECT something FROM source WHERE pattern", the "something" refers to columns in a table. In our case, even though we don't care about the underlying data structure implementations, the rows will be records. Records include information about entities. The natural conclusion following this analogy would be to specify the targets as entities. This abstraction works well together with the Kestrel implementation, with its Kestrel variables containing data on homogeneous entities.

The OpenC2 language specification already has targets that correspond to entities, like process, file, ipv4-address, etc. All though, these have not been defined to facilitate analytical requirements, capturing essential data connected to cyber observables. However, the STIX standard has been created for this exact purpose. A good place to start could therefore be to look at the STIX-cyber observables as candidates for targets. This lines up well with how the current AP returns a STIX bundle with STIX Observed Data, a container for STIX Cyber Observables. A clean copy of the standard would not be likely to work. Whether or not this list is extensive enough or whether the requirements to the objects are strict enough would have to be evaluated. One example is how the SCO process only requires one of the fields (other than the two common properties type and the unique id) to be included. For analytical purposes, this requirement could likely benefit from being stricter to facilitate minimum functionality. Some examples of fields for the process type that are common in EDRs and useful in threat hunting and pattern matching: command-line arguments, reference to the parent process, current working directory, etc.

We have learned from following the Kestrel-OpenC2 meetings that the committee agrees that an OpenC2-schema-compliant version of STIX SCOs will have to be defined for the threat hunting actuator profile. A comprehensive solution would have to evaluate many things, for now, to have something to work with, we propose some of the basic entities as targets.

Examples: process, file, network-traffic, ipv4-address, ipv6-address, domain-name, URL.

4.3.3 Filtering in data retrieval operations

Since we define our targets as entities, one way to implement filtering in our data retrieval commands whilst staying true to design principles is to use OpenC2 target specifiers. From the OpenC2 target definition: "(...) properties of the Target, called Target Specifiers, further identify the Target to some level of precision, such as one specific Target, a list of Targets, or a class of Targets" [15]. Following this definition we could allow for target specifiers to define the pattern with which we are filtering.

Example:

```
{
  "action": "retreive",
  "target": {
    "process": {
      "binary_name": "cmd.exe"
    }
  },
  "args": {
      "response_requested": "complete"
  }
}
```

This alternative has the benefit that it's simple to write and translate meaning into machine-readable queries. It could however be challenging to define appropriate attributes. However, a lot of work has already been done toward this, and again the STIX standard is an excellent starting point.

Another alternative would be to take advantage of the work put into STIX patterns. This has the benefit of unlocking interoperability with STIX-shifter modules for data repositories, just like Kestrel does. To complete filtering we add arguments for specifying the relevant time window. This is an especially important feature for data repositories holding telemetry from large environments for longer time periods. Here the OpenC2 language specification has relevant timestamp data types defined that can be used to express start and stop for a time window. Alternatively, to only specify the start, implying that data older than the specified value should be ignored.

This approach could look something like this when serialized in JSON:

Example:

```
{
  "action": "retrieve",
  "target": "process",
  "args": {
      "response_requested": "complete",
      "th": {
          "pattern": "encoded STIX-Pattern",
          "start-time":"timestamp",
          "stop-time":"older-timestamp"
      }
  }
}
```

4.3.4 Command matrix

We propose expanding the command matrix with the action retrieve and the target observable, referring to the STIX observable returned objects.

Table 11: Analytics AP Command matrix

	Features	Datasources	Huntbooks	Hunt	STIX SCO
Query	Valid	Valid	Valid		
Investigate				Valid	
Retrieve					Valid

4.3.5 Evaluation

To create a satisfying abstract representation of analytical operations common actions performed in relevant implementations need to be defined. This proposed alternative does so to a higher degree than the current ongoing work. We evaluate this model to be more in line with what we expect from an actuator profile for threat hunting activities.

In terms of technology agnosticism, this solution still requires consumers to translate our choice of syntax for filtering. In this version, consumers will need to translate STIX Patterns. The benefit of STIX is that it can give access to a wide range of data sources, but not all. Further, we have not abstracted the idea of filtering, simply submitted to a standard for data representation. This is a constraint that is not introduced by us, it is also present in the ongoing work as Kestrel depends on STIX.

Developing a comprehensive solution following the description that we have started here would add considerable work compared to the simpler version which only invokes huntbooks.

There are still advanced analytical operations that software like the Kestrel runtime can perform that are hard to boil down to a single command. One example is how the FIND command takes a Kestrel variable as input, something that could facilitate considerable performance benefits over constructing longer and more complex nested/join type queries. Kestrel also has functionality for sending intermediary results to external analytics, sorting and/or grouping by different values, etc.

4.4 Reverse engineering Kestrel to create an information model for analytics

A third option is the idea of using sessions and session management for analytics. The reason for this is simply put to attempt to facilitate advanced analytics such as the functionality that Kestrel provides. Addressing the dilemma of both needing distributed data storage holding records and wanting complex queries against entities, still leaving the actuator the responsibility to decide *how*, but absorbing the *what* into OpenC2. Keeping in mind that OpenC2 is a standard language for command and control of technology, an information model to guide the design of data models, the process can be described either as absorbing the Kestrel threat hunting language or as reverse engineering an information model from the Kestrel runtime software.

Indicators of behavior, or extended centered graph patterns as Kestrel calls them, are potentially complex descriptions of patterns in event data. Patterns describing multiple jumps in relations are computationally heavy and not recommended in distributed systems. Even worse, in some cases, if the records the pattern is matched against only contain information on one relational jump and the pattern expresses two jumps, Kestrel will in some cases return no matches. Even with its internal relational model, Firepit, the Kestrel documentation recommends limiting the number of jumps per huntstep to increase performance [31], similar to ElasticSearch warnings against joins [21]. Hunters can increase performance by instead expressing their huntflow over several huntsteps and increase overall performance by letting Kestrel build temporary data structures optimized for entity reasoning.

A solution that expands the functionality that the analytics data model can provide beyond simple data retrieval might have to include some form of session management and state. We mention this as it could be a way to build a fully abstracted representation of analytical operations in the future, admitting that this possibly introduces new compromises to OpenC2 current guidelines. Although it facilitates advanced analytics and allows the consumer to implement features such as pre-fetching, consumer-side internal re-structuring of data, and building relational models to increase performance for complex queries, it strays away from other typical design choices of OpenC2, as other actuator profiles aim at defining more atomic commands.

Example:

```
#Create session
{
  "action": "create",
  "target": "session",
  "args": {
    "response_requested": "complete",
  }
}

#Session created successfully
{
"status": 200,
"results": {
    "th": {
        "session-id":"abc123"
    }
}

#First standard query
{
  "action": "find",
  "target": "process",
  "args": {
      "response_requested": "complete",
      "th": {
          "session-id":"abc123",
          "pattern": "encoded STIX-Pattern",
          "start-time":"timestamp",
          "stop-time":"older-timestamp"
      }
  }
}

#Response: query successful, return meta-data
{
"status": 200,
"results": {
    "th": {
        "session-id":"abc123"
        "meta-data":{
            "records":"18",
            "entities":"3"
        }
    }
}

#Perform external analytics
{
  "action": "analytics",
  "target": {
      "type":"process",
      "external-analytics":"127.1.1.100:80"
  }
  "args": {
      "response_requested": "complete",
      "th": {
          "session-id":"abc123"
      }
  }
}

#Response
{
```

```
"status": 200,
"results": {
    "th": {
        "session-id":"abc123"
        "meta-data":{
            "records":"6",
            "entities":"1"
        }
    }
}

#Retrieve results (STIX SCOs), end session
{
  "action": "retrieve",
  "target": {
      "type":"process"
  }
  "args": {
      "response_requested": "complete",
      "th": {
          "session-id":"abc123"
      }
  }
}

#Final response
{
"status": 200,
"results": {
    "session-id":"abc123"
    "th": {
        "meta-data": {
        "count":1
        }
        STIX-bundle: {
            "type": "bundle",
            "id": "bundle--NUMBERS",
            "objects": [
                {
                    "type": "process",
                    "spec_version": "2.1",
                    "id": "indicator--numbers",
                    "created_by_ref": "identity--
                       numbers",
                    "created": "timestamp",
                    "modified": "timestamp",
                    "object_marking_refs": ["marking-
                       definition--numbers"],
                        }]
                  }
            }
      }
}
```

4.4.1 Evaluation

This third option is the most ambitious in terms of attempting to abstract analytical operations. When it comes to dependency on existing technology, session management is to our knowledge not common functionality for data repositories. Meaning that to adhere to requirements to technology agnosticism this will likely be part of something like *extended* features for advanced analytics. We estimate the amount of work to produce a comprehensive version of an analytical session profile to be significantly larger compared to our other suggestions.

4.5 Summary

In this chapter, we have explored some options that address our findings in the previous chapter. Section 4.2, is an alternative that takes the approach of adding support for other languages than Kestrel threat hunting language, changing as little as possible. Section 4.3 attempts to describe how one could approach defining analytics in a way that involves less compromise in regard to OpenC2 philosophy and design principles. It is however a more comprehensive solution that requires substantial work to develop a complete version. It could take advantage of STIX-shifter, as Kestrel already does. Section 4.4 discusses the option of adding a layer on top of the option described in 4.3. A session layer to facilitate session management and advanced huntflows, merging and forking flows, sending temporary results to external analytics, etc.

Chapter 5

Conclusion

5.1 Answering our research questions

RQ1: To what extent can the OpenC2 Threat Hunting Actuator Profile satisfy the requirements for an abstraction of analytical operations? In Chapter 3, we found that the ongoing work with the threat hunting actuator profile does not satisfy the requirements for a profile for analytics but instead defines a way to invoke predefined custom intrusion detection in the form of Kestrel huntbooks. We have discussed how the ongoing work with the threat hunting actuator profile, when taken as a profile that should abstract the function of threat hunting, is not yet on par with previous actuator profiles. When considering the profile as a more narrow function than the name suggests, the role of invoking Kestrel huntbooks automatically, we want to acknowledge that the functionality it brings is valuable in security operations. Especially if successfully integrated with standards for orchestration such as CACAO [32].

However, we find it appropriate to highlight the limited scope and the lack of technology agnosticism. As far as we know, no other entity-reasoning multi-step threat hunting technologies exist, proprietary or open-source, and Kestrel is not a widely adopted technology yet. This makes the use case that this work addresses rather specific. Threat hunting is performed using techniques integral to other sub-categories of analytics. Researching methods to automate invoking huntbooks, gaining access to only a piece of the functionality that Kestrel itself provides, not gaining access to any other technologies, and still relying heavily on other standardization work through STIX, summarizes to conditions worthy of discussing and possibly re-evaluating. To some degree, the difficult task of abstracting analytics is left to the developers of the Kestrel threat hunting language.

RQ2: How could alternative approaches address findings from RQ1 and be more in line with the OpenC2 philosophy and design principles? We have proposed some alternative approaches that the technical committee could consider should they want to further abstract analytical functions into OpenC2, expanding on the language specifications defined scope in alignment with what ongoing work could imply.

We believe it could be valuable to continue exploring an actuator for analytics. A standardized way to smaller analytical operations, likely leaning on the STIX standard for data representation and pattern expressions, as well as more advanced huntflows such as what Kestrel implements. The Kestrel threat hunting language is a good candidate to be considered absorbed in order to be able to express technology-agnostic analytical commands in OpenC2.

5.2 Contributions

This thesis has, under the motivation that standardization is essential to automation and interoperability, and that automation seems to be a key factor in reducing the mean time to detect intrusions, found OpenC2s ambitions for the threat hunting actuator profile not sufficient in regards to abstracting analytical operations. As a follow-up, we have attempted to describe some promising alternative approaches we hope might be of value in future work with developing OpenC2.

5.3 Future work

As we merely sketch a way to approach the challenge of abstracting analytics, it is obvious that future work will include continuing to develop a comprehensive functioning actuator profile for analytics. Here are some other topics related to this thesis we think are interesting.

5.3.1 STIX-Shifter

It would be interesting to do a thorough analysis of the quality of STIX-shifter modules. Since the only work attempting to include analysis into OpenC2 depends on the STIX standard, it would be interesting to investigate how well STIX connectors are performing in real life on different data repositories.

5.3.2 Entity reasoning and future data repositories

Tau-calculus, and threat intelligence computing. In 2018 Dr. Xiaokui Shu published research on the theory behind Kestrel [33], a new methodology that models threat discovery as a graph computation problem. Our thesis has discussed some of the dilemmas with storing and querying data in security operations. Researching alternative data structures for storing historical system information as a foundation for analytics is a huge topic that could impact how we design our systems for security operations in the future.

5.3.3 Problems with standardizing too early

Dr. Xiaokui Shu mentions in our conversation with him that in the future, there should be technologies similar to Kestrel that would be accessible through the threat hunting AP as its currently being developed. If this is true, it raises a question about how early standardization of a new function within any domain currently only being implemented by one, or a few homogeneous instances, influences potential innovation within the development of the new function. Look to TCP as an example where new transport protocols could be wanted, but certain nodes in the internet have implementations of x/IP where if x does not equal UDP or TCP packets are dropped. How are new instances of emerging functions within a domain affected by having to comply with early standardization? Overfitting a standard to a training sample of the first implementations. It seems it could hinder innovation and paradoxically contribute to ossification, the evil it sought to defeat.

Appendix

Appendix A.1

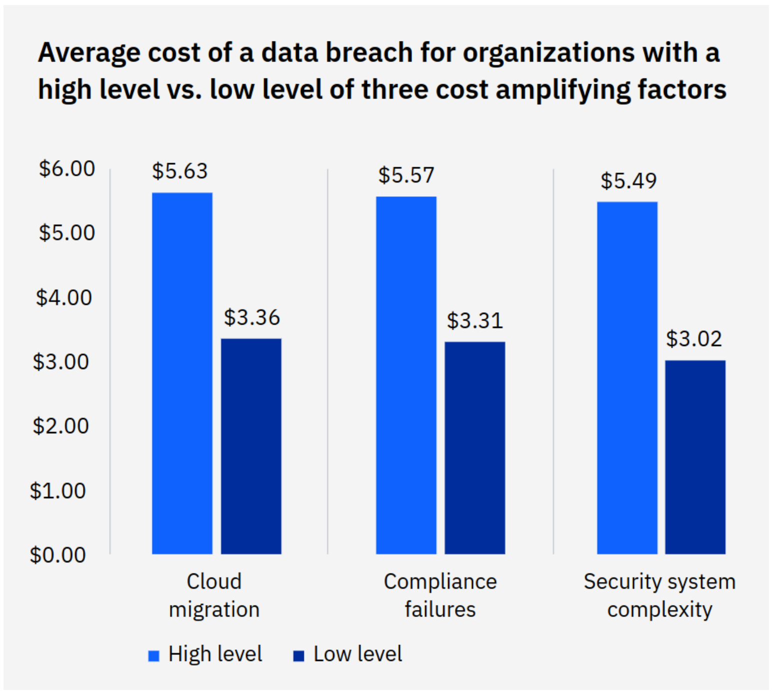


Figure 9: Factors associated with higher costs [8]

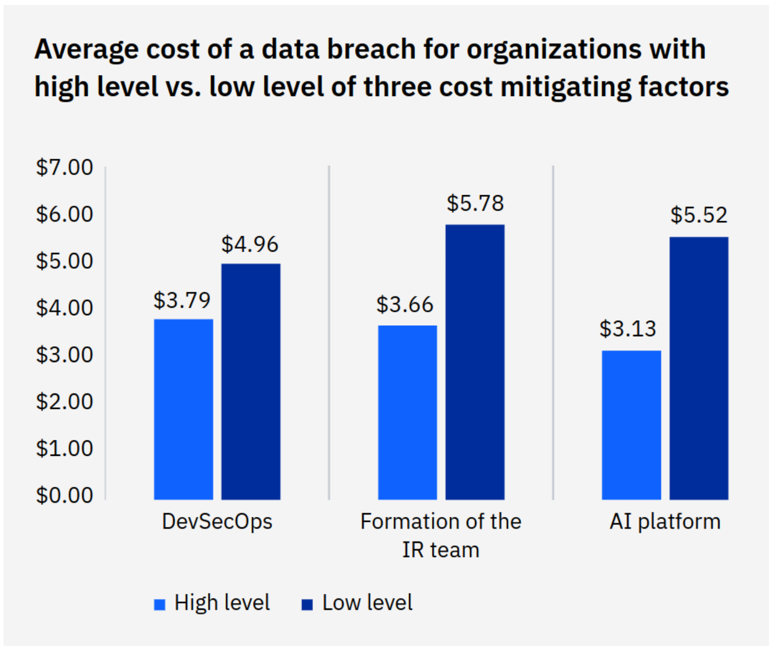


Figure 10: Factors associated with lower costs [8]

Appendix A.2

Bibliography

[1] Kommunal-og distriktsdepartementet. *Skal stake ut kursen med ny nasjonal digitaliseringsstrategi*. Regjeringa.no. Publisher: regjeringen.no. 21st Apr. 2023. URL: https://www.regjeringen.no/nn/aktuelt/skal-stake-ut-kursen-med-ny-nasjonal-digitaliseringsstrategi/id2972487/ (visited on 19/05/2023).

[2] Helse-og omsorgsdepartementet. *E-helse*. Regjeringen.no. Publisher: regjeringen.no. 8th June 2021. URL: https://www.regjeringen.no/no/tema/helse-og-omsorg/e-helse/id2479944/ (visited on 19/05/2023).

[3] 'U.S. firm blames Russian 'Sandworm' hackers for Ukraine outage'. In: *Reuters* (8th Jan. 2016). URL: https://www.reuters.com/article/us-ukraine-cybersecurity-sandworm-idUSKBN0UM00N20160108 (visited on 19/05/2023).

[4] Catherine Stupp. *Cyberattack Raises Pressure on European Water Providers During Drought*. WSJ. Section: WSJ Pro. URL: https://www.wsj.com/articles/cyberattack-raises-pressure-on-european-water-providers-during-drought-11661506201 (visited on 24/05/2023).

[5] Heather LandiMay 29 and 2019 09:00am. *Report: 40% of healthcare organizations hit by WannaCry in past 6 months*. Fierce Healthcare. Section: Fierce Healthcare Homepage, Hospitals, Practices. 29th May 2019. URL: https://www.fiercehealthcare.com/tech/lingering-impacts-from-wannacry-40-healthcare-organizations-suffered-from-attack-past-6-months (visited on 19/05/2023).

[6] *Attackers Deploy New ICS Attack Framework "TRITON" and Cause Operational Disruption to Critical Infrastructure*. Mandiant. URL: https://www.mandiant.com/resources/blog/attackers-deploy-new-ics-attack-framework-triton (visited on 19/05/2023).

[7] Vasileios Mavroeidis. 'Towards Automated Threat-Informed Cyberspace Defense'. Accepted: 2021-09-22T13:10:24Z ISSN: 1501-7710. Doctoral thesis. 2021. URL: https://www.duo.uio.no/handle/10852/88302 (visited on 10/04/2023).

[8] *Cost of a data breach 2022*. 25th Apr. 2023. URL: https://www.ibm.com/reports/data-breach (visited on 10/05/2023).

[9] *oasis-tcs/openc2-ap-hunt*. original-date: 2022-12-06T21:09:23Z. 13th Jan. 2023. URL: https://github.com/oasis-tcs/openc2-ap-hunt (visited on 14/05/2023).

[10] Ida Solheim and Ketil Stølen. *Technology Research Explained*. Accepted: 2016-04-28T11:47:35Z. 2007. ISBN: 978-82-14-04039-5. URL: https://sintef.brage.unit.no/sintef-xmlui/handle/11250/2387932 (visited on 14/05/2023).

[11] *About Us*. OASIS Open. URL: https://www.oasis-open.org/org/ (visited on 22/05/2023).

[12] *CyberHunt - Department of Informatics*. URL: https://www.mn.uio.no/ifi/english/research/projects/cyberhunt/index.html (visited on 10/05/2023).

[13] Clemens Sauerwein et al. 'Threat Intelligence Sharing Platforms: An Exploratory Study of Software Vendors and Research Perspectives'. In: ().

[14] 'Open Command and Control (OpenC2) Architecture Specification Version 1.0'. In: ().

[15] 'Open Command and Control (OpenC2) Language Specification Version 1.0'. In: ().

[16] *oasis-tcs/openc2-ap-er*. original-date: 2020-12-01T21:30:30Z. 29th June 2022. URL: https://github.com/oasis-tcs/openc2-ap-er (visited on 12/05/2023).

[17] *oasis-tcs/openc2-apsc-stateless-packet-filter*. original-date: 2018-03-06T23:17:47Z. 26th May 2021. URL: https://github.com/oasis-tcs/openc2-apsc-stateless-packet-filter (visited on 12/05/2023).

[18] *What is Kestrel? — Kestrel Threat Hunting Language*. URL: https://kestrel.readthedocs.io/en/stable/overview/index.html (visited on 24/05/2023).

[19] Vasileios Mavroeidis. *The Role of OASIS OpenC2 in Cybersecurity Automation and Orchestration*. 29th Dec. 2022. DOI: 10.13140/RG.2.2.27829.76005.

[20] *Terminology and Concepts — Kestrel Threat Hunting Language*. URL: https://kestrel.readthedocs.io/en/stable/language/tac.html (visited on 24/05/2023).

[21] *Joining queries | Elasticsearch Guide [8.7] | Elastic*. URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/joining-queries.html (visited on 24/05/2023).

[22] 'STIX™ Version 2.0. Part 5: STIX Patterning'. In: (2017).

[23] Open Cybersecurity Alliance. *Streamlining and Automating Threat Hunting With Kestrel - Black Hat 2022*. 12th Sept. 2022. URL: https://www.youtube.com/watch?v=tf1VLIpFefs (visited on 20/05/2023).

[24] *GitHub: opencybersecurityalliance/kestrel-huntbook/HEAD*. URL: https://mybinder.org/v2/gh/opencybersecurityalliance/kestrel-huntbook/HEAD?filepath=tutorial (visited on 24/05/2023).

[25] *Introduction to STIX-Shifter*. original-date: 2018-05-11T18:01:01Z. 10th May 2023. URL: https://github.com/opencybersecurityalliance/stix-shifter (visited on 12/05/2023).

[26] *Open Cybersecurity Alliance (OCA)*. original-date: 2019-09-09T17:12:59Z. 10th May 2023. URL: https://github.com/opencybersecurityalliance/oasis-open-project/blob/7171d36c35445aa70b9507e2bde1bd8c791160ba/CHARTER.md (visited on 12/05/2023).

[27] 'STIX™ Version 2.0. Part 4: Cyber Observable Objects'. In: (2017).

[28] 'STIX™ Version 2.0. Part 2: STIX Objects'. In: (2017).

[29] *oasis-tcs/openc2-ap-pf*. original-date: 2021-02-18T21:00:42Z. 3rd Feb. 2022. URL: https://github.com/oasis-tcs/openc2-ap-pf (visited on 12/05/2023).

[30] *Contributors to opencybersecurityalliance/kestrel-lang · GitHub*. URL: https://github.com/opencybersecurityalliance/kestrel-lang/graphs/contributors (visited on 21/05/2023).

[31] *Graph Pattern and Matching — Kestrel Threat Hunting Language*. URL: https://kestrel.readthedocs.io/en/stable/language/ecgp.html (visited on 24/05/2023).

[32] 'CACAO Security Playbooks Version 1.0'. In: (2021).

[33] Xiaokui Shu et al. 'Threat Intelligence Computing'. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18: 2018 ACM SIGSAC Conference on Computer and Communications Security. Toronto Canada: ACM, 15th Oct. 2018, pp. 1883–1898. ISBN: 978-1-4503-5693-0. DOI: 10.1145/3243734.3243829. URL: https://dl.acm.org/doi/10.1145/3243734.3243829 (visited on 01/03/2023).