MONITORING AND IMPROVING MANAGED SECURITY SERVICES INSIDE A SECURITY OPERATION CENTER

Mina Khalili

A thesis
in
The Department
of
Concordia Institute For Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science in Information Systems Security
Concordia University
Montréal, Québec, Canada

December 2015

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Mina Khalili**

Entitled: **Monitoring and Improving Managed Security Services inside a Security Operation Center**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science in Information Systems Security

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. Chadi Assi _____ Chair

Dr. Benjamin Fung _____ CIISE Examiner

Dr. Todd Eavis _____ External Examiner (CSSE)

Dr. Lingyu Wang _____ Supervisor

Approved _____
Chair of Department or Graduate Program Director

_____ 20 _____ _____

Amir Asif, Ph.D.,Dean
Faculty of Engineering and Computer Science

Abstract

Monitoring and Improving Managed Security Services inside a Security Operation Center

Mina Khalili

Nowadays, small to medium sized companies, which usually cannot afford hiring dedicated security experts, are interested in benefiting from Managed Security Services (MSS) provided by third party Security Operation Centers (SOC) to tackle network-wide threats. Accordingly, the performance of the SOC is becoming more and more important to the service providers in order to optimize their resources and compete in the global market. Security specialists in a SOC, called analysts, have an important role to analyze suspicious machine-generated alerts to see whether they are real attacks. How to monitor and improve the performance of analysts inside a SOC is a critical issue that most service providers need to address. In this paper, by observing workflows of a real-world SOC, a tool consisting of three different modules is designed for monitoring analysts' activities, analysis performance measurement, and performing simulation scenarios. The tool empowers managers to evaluate the SOC's performance which helps them to conform to Service-Level Agreement (SLA) regarding required response time to security incidents, and see the need for improvement. Moreover, the designed tool is strengthened by a background service module to provide feedback about anomalies or informative issues for security analysts in the SOC. Three case studies have been conducted based on real data collected from the operational SOC, and simulation results have demonstrated the effectiveness of the different modules of the designed tool in improving the SOC performance.

Acknowledgments

Firstly, I would like to express my heartfelt gratitude to my supervisor Prof. Lingyu Wang for his endless support, motivation, patience, and immense knowledge. His guidance lightened my research way all the time. Dr. Wang has shown me how to improve myself, a lesson for the rest of my life.

I would like to thank Mr. Nicandro Scarabeo and Mr. Michel-Ange Zamor from the operational SOC, Above Security Inc., helping me to proceed in my study by their technical knowledge.

I would like to thank all CIISE faculty members, employees, and students to provide such a great educational environment. I thank my labmates for their limitless support, specially Mengyuan Zhang.

Last but not the least, I would like to thank my family: my grandmother, parents, aunt, uncles, and siblings for supporting me spiritually throughout my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Advantages of employing Managed Security Services (MSS), such as cost effectiveness, skilled security experts, appropriate facilities, up to date security awareness, and 24 hours continuous service encourage different companies to outsource their security services rather than having in-house security employees [1]. Network security monitoring (NSM) was born as a different term to specify the new feature of MSS for continuous monitoring of networks by human experts rather than just installing security appliances. NSM is defined by Bejtlich [2] as "the collection, analysis, and escalation of indications and warnings to detect and respond to intrusions." Describing the Bejtlich's definition, in order to provide NSM service, Managed Security Service Providers (MSSP) deploy various sensors in the client site, such as Intrusion Detection Systems (IDS) to gather various suspicious alerts from each client's computer network, and send them to the Security Operation Center (SOC). Then, SOC as a heart of NSM correlates and analyzes the alerts by its human security analysts to confirm whether they are successful exploits. A security incident is detected and confirmed by True Positive (TP) alerts as indications. In case of an incident, results of analysis need to be exposed to decision makers, in a process called escalation, to react in an appropriate way.

Emerging demand of outsourcing security services from different companies makes the business world increasingly competitive for MSS providers. Monitoring and improving performance of the SOC becomes more crucial to the managers to optimize their resources and improve the quality of service. Challenges faced by the operational SOC include: 1. To the best of our knowledge, there does not exist any model

for the SOC analysis workflow to elaborate analysts' detailed tasks. By modeling the analysis workflow, we can obtain a clear picture about analysis steps allowing the system to track analysts' activities. 2. There do not exist automated tools for monitoring and evaluating SOC performance. Consequently, there is no clear understanding of SOC capability and different analysts' performance. 3. There does not exist any tool to simulate potential improvement options of the SOC in order to assess their effectiveness. 4. There does not exist any tool to support convenient knowledge transfer among analysts. Analysts usually possess different knowledge, since they gain different knowledge during each investigation related to different clients. The above mentioned challenges prevent managers to prioritize the effort on improving SOC performance.

There are three main categories of related work (a more detailed review is given in Chapter 6). The first domain discusses different aspects of MSS. There are some works ([3] [4] [5] [6] [7] [8] [9]) proposing different designs for a SOC architecture, such as employing recognition mechanism of immune system, cloud based NSM, hierarchical mobile-agent-based approaches, etc. Some papers ([10] [11] [12]) study various aspects of different operational SOCs to compare their functionality. These works are different from our work, since we neither modify the SOC architecture, nor provide a classification framework for SOCs. The second domain is alert correlation techniques helping to provide more accurate alerts, and reducing the rate of False Positive (FP) ([13][14][15][16]). The third domain reviews studies about Call Centers (CC), since SOC and CC are similar regarding their performance evaluation. In a CC, operators answer to different calls in a queue where in a SOC, analysts analyze incoming logs in a queue. Different queueing models are employed in this area to solve the problem of staffing, scheduling, and routing jobs policy ([17][18][19][20]). Overview of the literature shows there is little work related to performance measurement and improvement of a SOC.

The designed system consists of a Graphical User Interface (GUI) for managers with three modules; monitoring, measuring, and simulation, respectively. Moreover, it consists of a background service to generate feedbacks to SOC's analysts about anomalies and informative issues. The tool helps managers to be updated about current analysts' activities of the SOC with the monitoring module. This module illustrates details of recent investigations which are in progress or recently completed

by the analysts. Managers can check overall and detailed SOC performance with the measuring module. Consequently, they may see the need for adding new analysts, recognize demanding clients, or optimize certain analysis steps average duration. With the simulation module, they can assess different performance improvement options to see the potential effect of each possible improvement on the performance result of real production data, without affecting the normal operation of the SOC. Two improvement options can be automating certain steps to increase the efficiency of human analysts' tasks or the way of dispatching incoming alerts among analysts. Analysts benefit from the feedback module where they get hints about next probable steps, and feedback whether they have performed their tasks in normal range of time or they have missed a step. Moreover, they can learn from the results of previous analyses performed by other analysts. The development team of the operational SOC can improve the SOC Console application based on provided evidence by the simulation module. Furthermore, some measuring results can be used as evidence to demonstrate quality of services to the clients.

Contributions of our work are; 1. modeling the alert analysis workflow based on a real operational SOC, 2. developing a practical tool capable of; a) monitoring analysts' analysis workflow, b) measuring analysts' performance by defining different performance metrics, c) providing simulation opportunities to assess possible improvement options. d) enabling knowledge transfer among analysts by feedback module, 3. conducting three case studies based on real life activity logs of analysts over a period of 57 days to show how the performance would be affected by different simulation scenarios.

The rest of the thesis is organized as follows. Chapter 2 describes the required background knowledge to understand this work. Chapter 3 illustrates our modeling methodology of the analysis workflow, and the logging phase of analysts' activities. Chapter 4 provides an overview of the system's functionality, and the modules employed methodology and implementation. Chapter 5 evaluates three case studies to assess different improvement options of the SOC performance. Chapter 6 reviews related work. Chapter 7 concludes our work and addresses the future work.

Chapter 2

Background

In this chapter, the operational SOC workflow and its related characteristics and notations are explained to provide background knowledge. Besides illustrating SOC characteristics, a brief description of the designed system, and related definitions are given to show how the proposed system functions alongside the SOC main workflow.

A conceptual example of a typical service deployment model is drawn in Figure 1 from an infrastructure perspective (in reality, a client deployment architecture can be more complex). The operational SOC uses Virtual Private Network (VPN) to connect the client site to the SOC. Depending on the number of sensors, sensors can be placed at different locations in the client network, such as outside the firewall facing internet, behind the firewall inside the demilitarized zone (DMZ), which are accessible internal points of client's network from outside, or in the local network behind the firewall.

Different devices and sensors, such as Intrusion Detection Systems (IDS), asset detection tools, and flow analysis tools can be deployed to gather various suspicious events, called *alerts*[1]. Once sensors are deployed on the client site, they start generating alerts, based on their configurations, and sending them to the SOC via VPN. In the SOC, analysts receive the alerts of each client in real time. Alerts (suspicious events) should be analyzed within a specific time period which is defined in *Service Level Agreement (SLA)* signed between MSS provider and the client. For instance, response time of the operational SOC studied in this work is two hours defined in

[1]To clarify notations, alerts are specifically IDS-generated. However the event notation includes all kinds of machine generated alerts and logs. Logs could be referred to normal operating system events (e.g., unsuccessful attempt to login a system) that can be useful for a security investigation.

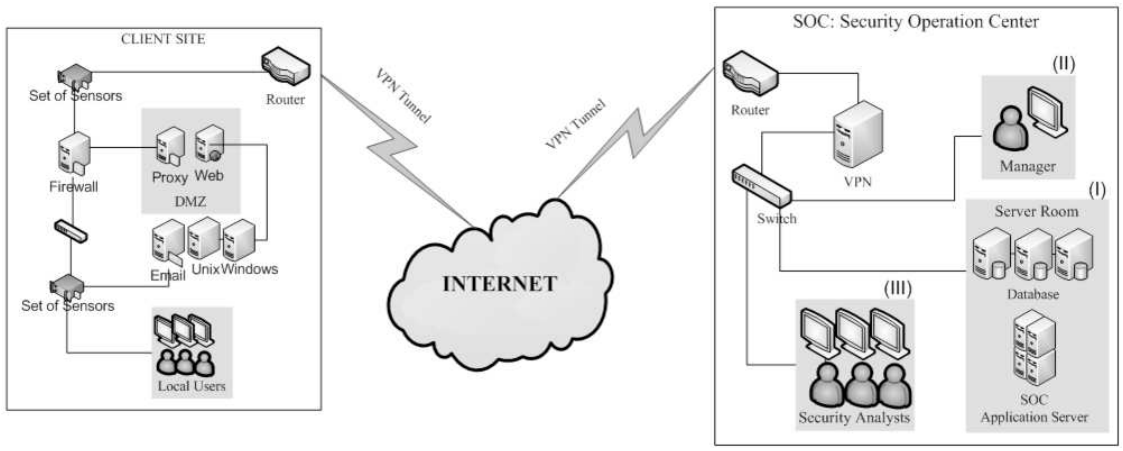


Figure 1: An example of a typical deployment model for Network Security Monitoring service representing an operational SOC and a client infrastructure

their SLA.

Analysts rely on the main tool of the SOC, named *SOC Console*, to review related alerts of each client and conduct analysis. The workflow of alert analysis starts from the SOC Console consisting tasks, such as receiving alerts, cross referencing network map of a client, examining a client's assets and vulnerabilities, opening content of the alert packet, and contacting the client in case of a true positive alert.

The analysis task of sensor-generated alerts by an analyst is called *investigation* (investigation and analysis notations can be used interchangeably). If the result of an alert analysis is true positive indicating a real threat, it is called an *incident*. The analyst needs to inform the related client about the incident in order to perform a proper action to tackle it. Such a process of informing client is called *escalation*. Based on severity of a threat and client's preference, different approaches could be defined for informing the client about different types of incident in the SLA, documented in *an escalation grid file*. Escalation approaches include sending an email, calling the administrator, or informing the client in a monthly report.

Investigations conclude in one of the following three different results, a) creating a new incident when a new threat is detected, b) updating a current incident when either more indications (TP alerts) related to one recorded incident are on hands and need to be added to the same incident, or the incident needs to be updated after contacting the client, or c) closing and archiving the alert, since there is no enough indication to declare the alert as a threat. From now on, *investigation class* points

at one of the mentioned investigation result categories.

Moreover, investigations can be categorized based on their alert types. An *investigation type* implies an analysis related to a specific attack type, e.g., malware infection. Different alerts related to a same attack type are grouped together to represent an attack type e.g., different snort rule IDs indicating the malware infection.

There are eight *investigation types* in two main categories from the SOC perspective, security-related incidents and policy violation incidents. *Investigation types* inherit their names from *attack types*, and can be used interchangeably, as they are implying the same concept. Table 1 represents different investigation types. Table 2 describes attack definitions related to investigation types. And, Table 3 provides one EventID example for each investigation type.

Incident category	**Investigation type**	**Abbreviation**
Security related	BruteForce Attack	BFA
	Denial Of Service	DOS
	External Vulnerability Scan	EVS
	Malware Infection	MI
	Abnormal Activities	AA
	Others	OTHERS
Policy violation	Policy Violation	PV
	Peer to Peer Bittorrent	P2P

Table 1: List of investigation types related to the incident categories

Inside an investigation workflow, each investigation has some major *steps*. As is discussed, an investigation process is conducted through the SOC Console. Generally, for analyzing each alert, 1. analysts expand details of the alert (step A). 2. Then, they may check network assets to see whether the reported alert is related to discovered vulnerabilities (step C). 3. They also check whether a similar incident with the same alert type is recently recorded (step C). 4. Some investigations might need more in-depth analysis, such as exporting alert packet content and using specific tools (e.g., Wireshark) to analyze them (step O). 5. In the end, if they infer the alert is not a threat based on the current situation, they will close and archive the alert (step B). If they deduce the alert is true positive, then they need to record and escalate the incident to the related client. 6. If the similar incident is still in

Investigation type	Description
Brute Force Attack	It is a probabilistic approach to obtain username and password credentials by trying all possible keys [21].
Denial of Service	It implies flooding resources by many requests making them out of service [22].
External Vulnerability Scan	It is a legitimate service for evaluating vulnerabilities of the network which can be threatful while it is launching from unauthorized sources [23].
Malware Infection	indicates all kinds of malicious activities resulting in exploiting network assets [24].
Abnormal Activities	Four categories of Snort alerts [25]; miscellaneous activity, miscellaneous attack, access to vulnerable web applications, and detection of TCP connection in the client's netwrok are grouped together forming this category.
OTHERS	Alerts that do not fit in any mentioned attack types are placed under this category.
Policy Violation	This category is considered for both adult content access and chatting applications which can be illegal based on the client's policy.
Peer to Peer Bittorrent	It is a legitimate file sharing protocol which can be defined as a policy violation activity inside a client network.

Table 2: Investigation types and related descriptions

the process, they update the current incident with more indications (step D). 7. If there is no recent similar incident, they record it as a new incident (step E). 8. Since recording an incident in the system is accompanied by escalating the incident, analysts may check the required escalation method from escalation grid document (step F). Each mentioned task is considered as one step of an investigation which the analyst performs to accomplish the alert analysis workflow. Table 4 describes different steps.

For each investigation type, there are usually several possible investigation approaches. We collect all possible analysis approaches in one integrated *model* consisting of different investigation paths of all investigation types. The complete investigation workflow modeling phase will be described in details in Section 3.1.

A simple flowchart is drawn as an "model example" in Figure 2. Three end nodes

Investigation type	EventID example	EventID description
BFA	31304	"SERVER-WEBAPP PocketPAD brute-force login attempt" [26]
DOS	2522	"This event is generated when an attempt is made to exploit a known vulnerability in the Microsoft implementation of SSL Version 3." [27]
EVS	632	"This event is generated when an external user scans an internal SMTP server using Network Associates' Cybercop vulnerability scanner." [28]
MI	35462	"MALWARE-CNC Win.Trojan.Kazy outbound connection" [29]
AA	18180	"This event is generated when an attempt is made to exploit a known vulnerability in flash player." [30]
OTHERS	101010	"A customized rule for the sensors to verify DNS queries by comparing to a blacklist file."
PV	34463	"APP-DETECT TeamViewer remote administration tool outbound connection attempt" [31]
P2P	12426	"This event is generated when network traffic that indicates Ruckus P2P broadcast domain probe is being used." [32]

Table 3: An EventID example for each investigation type category

show the three possible investigation results (classes). Each directed path from the start node to the end node is one possible investigation path. As is shown, two investigation paths result in scenario#1 which is closing and archiving the alert. The shorter path indicates that the analyst only by expanding the alert detail in the SOC Console determines that the alert is false positive. The other path shows the decision is made after conducting more analysis steps, such as step O which is about deep alert packet content analysis. The other scenarios#2 and 3 represent two possible investigation paths resulting in updating an incident and creating a new incident respectively.

Steps consist of sub-steps called *actions*. Actions are minor tasks grouped into one step. Actions are track-able points for each step performed by analysts. They correspond to single mouse click on the SOC console listed in Table 5, such as clicking on a specific button, or opening a window. Series of mouse clicks (actions) represent

StepIDs	Description
A	Checking global view of the SOC Console which shows all alerts related to each client, expanding a specific alert to check the details and related destination and source IP addresses
B	Classifying an event as not-suspicious and labeling it as FP
C	Checking the network assets to see whether the reported alert is related to discovered vulnerabilities. Moreover, checking whether the same attack is reported recently
D	Updating an existing incident for the client, since a similar incident is reported recently. Or updating the incident description, since the client responded to the incident (clarifying the incident)
E	Creating a new incident for the client, since indications confirm the alert as suspicious (true positive alert)
F	Checking the escalation grid for the related client. In this grid, analysts are provided information about how the client prefers to be informed about different kinds of attacks depends on criticality (e.g. call, email, report)
O	Opening the alert packet content by a specific tool to analyze deeply, such as Wireshark.

Table 4: Step IDs and their description

a task (step) which is performed. To avoid discussing unnecessary details, we focus on steps rather than actions mostly throughout the paper.

The pre-condition of the proposed system to monitor, measure, simulate, and feedback the investigation workflow is to find a way to track and log analysts' activities on the SOC Console. Logging activities becomes possible by defining steps and actions. As is discussed, actions are series of mouse clicks on the SOC Console. A script is written to log SOC Console activities with different required attributes which will be discussed in details in Section 3.2.

The designed system runs at different locations of the infrastructure, as it assists both managers and analysts. Referencing back to Figure 1, three locations of the SOC are labeled in that figure which are involved by the proposed system. Label I indicates the location at the server room where the tool for auditing and logging analysts' activities is running continuously keeping the logs up to date. Label II is the manager's desktop where the user interface (UI) with three modules (monitoring, measuring, and simulation) is installed. Label III is analysts' desktop where they can

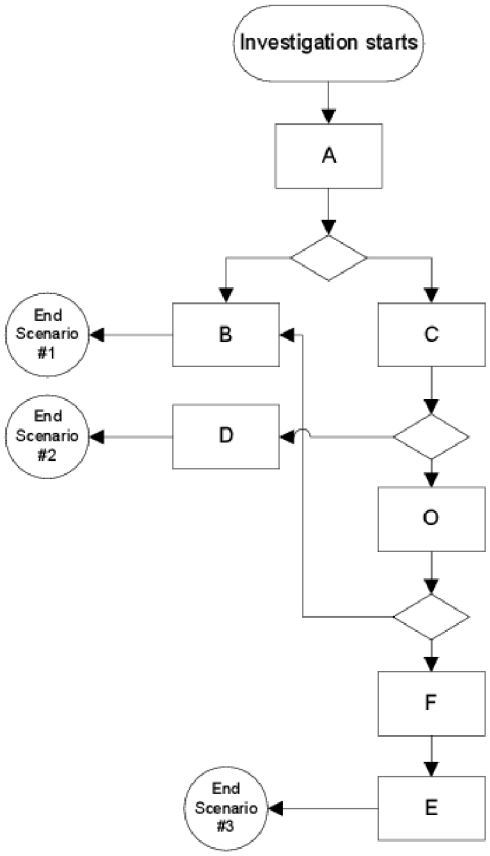


Figure 2: An example of a model with four possible investigation paths in flowchart format

receive notifications from the feedback module UI. Both tools, managers' GUI and the feedback module read the logs placed in the server room.

Step	ActionID	Description
A	130	Opening a specific client alert view
	105	Expanding the aggregated alert detail
	107	Clicking on the alerts to view alert details by “Editor”
B	109	Clicking on the button “Acknowledge” from alert detail
	125	Clicking on the button “Acknowledge” from alert list
C	104	Clicking on “View Asset” that will open a small window from source IP
	126	Clicking on “View Asset” that will open a small window from destination IP
	110	Clicking on the tab “Incidents”
	112	Clicking on “Edit incident” to open and view the incident
D	114	Clicking on the button “Add to incident”
	116	Clicking on the button “Add alerts to incident”
	117	Under comment box, clicking on the button “Apply” or “OK”
E	119	Clicking on the button ”Create incident”
	121	Clicking the button ”Finish”
F	122	Opening the documents dashboard
	123	Selecting the client
	124	click on the document ”Escalation Grid”
O	103	clicking on the button ”Export to pcap”
	127	Opening the packet using Wireshark

Table 5: Steps with sub-steps as Action IDs and related description

Chapter 3

Modeling and Logging the Investigation Workflow

In this chapter, the modeling phase of the investigation workflow is discussed. Then, the logging phase of analysts' activities will be detailed.

3.1 Modeling the Investigation Workflow

Before going through the modeling details, Section 3.1.1 justifies the employed method, UML activity diagram, than another popular modeling method, petri net, to model the investigation workflow.

3.1.1 UML Activity Diagram vs. Petri Net

There are two popular workflow modeling methods, petri net [33] and Unified Modeling Language (UML) activity diagram [34], in the literature. In this section, they are reviewed for modeling SOC's workflows. Both petri net and activity diagram methods intend to model workflow systems (WFS) with ordered and parallel activities considering decisions and iterations in a graphical representation.

Petri net is a bipirate graph for modeling systems' workflows graphically. It is enriched with precise mathematical definitions. It consists of places, transitions, and arcs which are circles, vertical boxes, and directed arrows respectively. A simple schema is provided in figure 3. Places represent conditions; transitions imply events which may occur; and arcs are connectors of places and transitions. A transition could

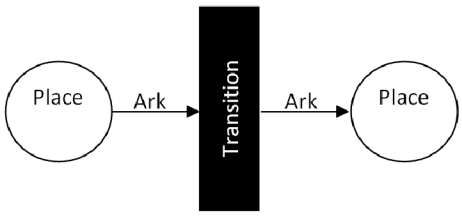


Figure 3: a simple representation of Petri net

be fired either immediately or later, when input places (preconditions) are satisfied to transmit the system to a new place (post condition).

Activity diagram could be considered as the extension of flowcharts [35] in software engineering with more standardized notations to model workflows. Activity diagram has a black circle as start point, rounded rectangles as actions, diamonds as decisions, narrow boxes as fork and join points, and a partial black circle as an end point.

Workflow systems are considered as open reactive systems, since they interact with their environments to perform business processes [36] [37]. A study [38] comparing petri net and UML activity diagram methods shows standard petri net is not suitable for modeling open reactive systems, where activity diagram does properly. Petri net models closed systems, since transitions imply activities from the system itself, not from the system's environment. Also standard petri net models active systems, once preconditions for a transition are satisfied the transition could fire immediately or later; in contrast, an activity diagram's action can happen where preconditions are done and environment trigger appears also, then the action will be done immediately which is the characteristic of reactive systems. Another comparison from Eshuis [39] between an action and a transition indicates a transition does not take time to be done, whereas an activity does. One given solution to this problem is timed or stochastic Petri nets method [40] [41].

On the other hand, lack of precise semantics in UML activity diagrams till version 1.x was one of this method's shortcomings [42]. A study [43] on defining semantics of UML 2.0 activity diagrams shows this method has been enriched by employing petri net semantics. Denotational semantics of activity diagrams are defined based on procedure call variant of Petri nets covering control flow, concurrency and procedure call.

As is discussed, petri net does not consider environment triggers, which can be important at some point of time for SOC investigation workflows. Since analysts is

in contact with the related client (environment) to clarify certain situations. Actions take time to be completed in the SOC, which basic petri net does not support. In our work, we need to model sequence of possible actions without detailing pre or post conditions.

The discussed issues and comparisons between petri net and activity diagram lead us to choose UML activity diagrams to model SOC's workflows. Activity diagram notations are concise, enough, and easy to understand for our need, where each node of a diagram represents an action either from the system or environment beside considering probable duration to perform the action.

3.1.2 Modeling Methodology

In this section, the modeling methodology of the investigation workflow is detailed. Hofstede [44] defines a workflow as an executable process, a detailed description of tasks with their chronological dependencies.

Modeling the investigation workflow for the designed system is a precondition to develop different functionalities of the system. Through the modeling phase, a good understanding of the investigation workflow is achieved. Consequently, measurable details which can be monitored and logged are identified as steps and actions discussed in Section 2.

The modeling of the investigation workflow is performed in two phases. The first phase is to gather the expert knowledge from analysts to identify different investigation types and relevant approaches. By completing this phase, Tables 1, 4, and 5 are provided in Section 2 to describe investigation types, and relevant steps and actions. To model different tasks (steps) and their relationship, UML activity diagram has been employed to visualize the model. The second phase is to visualize the model for the designed tool. Graphviz V.2.38 [45] is employed to layout the activity diagram to represent the investigation workflow model in the system.

Different analysis approaches regarding all investigation types are modeled in one integrated model by an activity diagram. Initially, it was assumed different investigation types have different analysis approaches. We had eight distinct investigation models. Later by comparing them, it became clear, however related approaches of different investigation types vary slightly, they can be integrated into one general model regardless of the investigation types. Slight difference, for instance having one

step more or less, can be managed within the same model. We aggregate all possible paths into one integrated model in which they do not contradict each other but just make the model bigger covering different investigation types paths.

The integrated model is shown in Figure 4. A labeling system is used to address each node of the activity diagram, than the node description. Each node of activity diagram is the representation of one single action belonging to a step. In other words, the label of each node indicates the node belongs to which action and step.

As is explained in Section 2, steps are tasks labeled by alphabet characters (step ID), and actions as single mouse clicks are labeled by numbers (action ID). The combination of step ID, and action ID is the identifier of each node of the activity diagram in the format of 'step ID : action ID'. For instance, we label the node of activity diagram as A:107 implying the related action 107 belonging to step A. Related description of different steps and actions are provided in Tables 4, and 5 in Section 2.

A sequence of nodes following from first node to the last node of the model forms an investigation path. One example of the investigation path is Specified by double boxes in Figure 4.

The logical relationship among different nodes of the model could be AND or OR. By traversing a single investigation path in the model, all existing nodes in the path have AND logical relationship from the start to the end node. When it is OR, it implies possibility of different investigation paths. By facing diamond shapes, which are decision nodes of the model, we can see OR relation among different nodes representing different investigation paths. The different investigation paths can be followed based on different conditions, such as different investigation types, or previous node's results. Following a single path of the investigation model, there is an AND logical relationship among the all related nodes. AND logical relation implies mandatory nodes. If an analyst misses one mandatory node between previous and next node of the path, the feedback module will warn the analyst about missing step. It will be discussed in details in Sections 4.1.4, and 4.2.4.

It is important to have a unique actionable label (stepID:actionID) for each node of the model since incoming analysts' activity logs will be mapped to each related node of the model.

The tool determines an investigation is completed if the next node of the current node is End node. Since we have several nodes in the model which could be considered

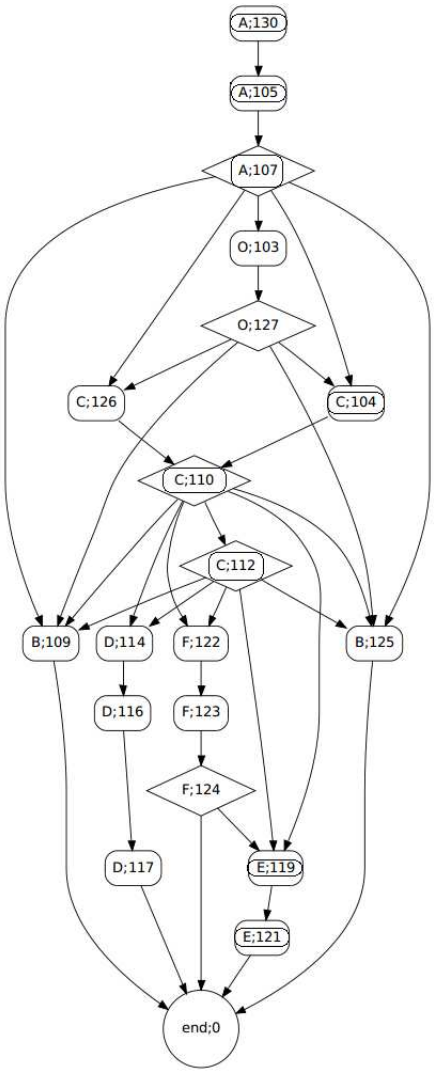


Figure 4: The investigation model

as the last node, we connect all of them to one dummy node called End. The tool by traversing next node of current node, can find out whether the next node is the End node. If so, then the current node is the last log of the investigation.

Graphviz [45] is employed to generate the investigation model as the graph output in a DOT file format. The DOT file can be fed into the system and traversed in order to track analysts' activities by comparing incoming logs with the related model. NetworkX package [46] in python is used as the data structure to read the DOT file.

Since preparing activity diagrams by the Graphviz tool requires writing codes in a special format, which is not user-friendly for the analysts, we introduce an open

source extension for Microsoft Visio called GraphVisio [47]. This extension eases the phase of generating a DOT file where analysts draw activity diagrams with Visio and then export their model directly to a DOT file.

3.2 Logging Analysts' Activities

Based on the extracted model from Section 3.1.2, the investigation workflow is defined by different steps (tasks) and their dependencies. Each task is defined by different actions as single mouse clicks on the SOC Console. These mouse clicks are logged by a python code which is part of the system. Each log represents one single action performed by an analyst through the SOC Console.

Each row of logs consists of eleven attributes. Attributes are listed and described in Table 6. Each log contains information about start and end time of the action as two attributes: TimeStart, and TimeEnd. We have three other attributes as InvestigationTypeID, StepID, and ActionID. The SourceID represents the alert is from which source (e.g., IDS), and the EventID represents Snort signatures or customized sensor rules. The AnalystID shows which analyst is performing the investigation, where the ClientID shows the related client. The IncidentID represents whether previously related client had the same type of incident. The InvID is the identifier of the ongoing investigation, a unique ID for all related investigation logs in the database.

Database Attributes	**Description**
ID	Auto increment primary key of the table
TimeStart	Starting time of the action
TimeEnd	Ending time of the action
InvestigationTypeID	Related investigation type
StepID	Related step
ActionID	Related action
SourceID	Event (alert) comes from which source e.g., IDS
EventID	The rule identifier of sensors to raise the alert
AnalystID	Which analyst performs the investigation
ClientID	Investigation is related to which client
IncidentID	If currently the incident is recorded in the SOC Console
InvID	One unique ID assigned to all logs of the investigation

Table 6: Related attributes and description

Based on the identified actions through the SOC Console, the logging script logs the actions started and ended via the SOC Console. Start time of actions are bound to the specific buttons, and end time of actions are bound to the start time of the next recognizable action by the tool. In this way, no time in the middle of an investigation is skipped in case of using other tools not being monitored by our logging system. End time of the last action is a specific button to close an investigation.

The output of the logging tool is a text file which needs to be adjusted by preprocessing phase (will be detailed in Section 5.1) to fit into the table of database.

As is shown in Table 6, each row of activity logs has related StepID and ActionID which the row will be matched to the investigation model. As is discussed in Section 3.1.2, we have the same identifier for each node of the model (StepID and ActionID). Therefore, by having StepID:ActionID from the log, we can examine the model to see which node can be mapped to the log. In this way, by tracking analysts activities the system is able to understand what is going on in each investigation, such as analyst is working on which step currently, or what is the next action. By mapping all logs of one investigation to the investigation model, the tool can also identify which actions are missed from the followed investigation path.

During the first phase of our implementation, due to lack of real-life activity logs, a program is developed to generate random log entries simulating different investigations by multiple analysts, for different investigation types, decisions (paths), and clients. By traversing the investigation model, different random paths are generated with random arrival times. The program could simultaneously generate events for multiple analysts performing investigations in parallel for multiple clients, and store generated logs in the database as well. On the other hand, the tool queries the database incrementally after configurable time intervals and updates results in measuring and monitoring modules.

In this phase, we made an assumption about identifying each single investigation log, that the combination of following six attributes forms a key : InvestigationTypeID, SourceID, AnalystID, ClientID, IncidentID, EventID, since we assumed such a combination would always be unique for the all investigation logs (e.g., MI;01;01;25;50001;36505). However, in real logs we found it is not possible to have all values of these six values present in all logs. In other words, in each row related to one investigation, some attributes may not have their related value. For instance, we

ID	TimeStart	TimeEnd	InvestigationTypeID	StepID	ActionID	SourceID	EventID	AnalystID	ClientID	IncidentID	InvID
203547	1432954820387	1432954834974	MI	A	130	0	33906	4	15	0	1210176
203548	1432954834974	1432954851571	MI	A	105	0	33906	4	15	0	1210176
203549	1432954851571	1432954857971	MI	C	104	0	33906	4	15	0	1210176
203550	1432954857971	1432954868877	MI	C	110	0	33906	4	15	0	1210176
203551	1432954868877	1432954934995	MI	A	107	1	33906	4	15	0	1210176
203552	1432954934995	1432955078419	MI	E	119	0	33906	4	15	0	1210176
203553	1432955078419	1432955083059	MI	C	112	0	33906	4	15	500204	1210176
203554	1432955083059	1432955083059	MI	E	121	0	33906	4	15	500204	1210176

Table 7: Log entries of one investigation related to MI investigation type

may have SourceID in just second or third row of logs. By this observation, we added a new attribute, InvID, to the log format as a primary key of each single investigation record. Dataset pre-processing is detailed in Section 5.1.

Table 7 shows a sequence of logs belonging to one investigation, containing three steps and eight actions(A:130 221A:105 221C:104 221C:110 221A:107221E:119 221C:112 221E:121). This investigation path is illustrated in the investigation model shown in Figure 4 as specified by the double boxes. By comparing the combined key of each log, which is StepID:ActionID, the tool can map the log to a node of the model (the analyst is performing which step and action). By looking at all logs and mapping them to related nodes in the investigation model, we can see which path is followed by the analyst. Furthermore, By identifying the path followed by an analyst in the investigation model, the feedback module can notify analysts about probable next steps or missing steps (will be discussed in Section 4.2.4).

Chapter 4

Methodology and Implementation

In this chapter, we elaborate the system modules and their functionalities. Then, we describe the employed methodology and detailed implementation of each module.

4.1 Overview of The System

The designed system assists both managers and analysts as a value-added component of the SOC Console. Once investigation modeling and logging phases are completed, it is possible to implement four modules with different functionalities. The independent graphical user interface with the three modules; monitoring, measuring, and simulation helps managers to evaluate the operational SOC performance. In addition, the feedback module assists analysts of the SOC by notifying them with different information through the Microsoft Windows operating system tray, while they are performing investigations through the SOC Console.

The implemented system by four components satisfies different objectives. *The monitoring module* makes tracking and monitoring analysts' activities possible. *The measuring module* empowers managers to check SOC's performance with different metrics provided by the tool. Furthermore, customizable OLAP analysis is integrated into the tool, providing more detailed analysis performance results. *The simulation module* facilitates evaluating improvement options by two different approaches: 1. Modifying different analysis steps' duration to see how the total analysis duration would be affected by an improved step. 2. Applying a different queueing model and alert dispatching approach to see how the overall performance would be affected.

The feedback module, by employing the investigation model and comparing it with ongoing investigations, can reveal anomalies in terms of analysis approach or time. It can also remind analysts about next possible steps. Furthermore, this module considers completed investigations as valuable historical results to be retrieved and shared with other analysts.

A general workflow of the designed system is shown in Figure 5, where activity logs are being gathered from analysts' machines and stored in the database. The main GUI including the monitoring, the measuring, and the simulation modules keeps reading the database and provides different capabilities to the managers. The feedback module works as a background service to give feedback to analysts in real time.

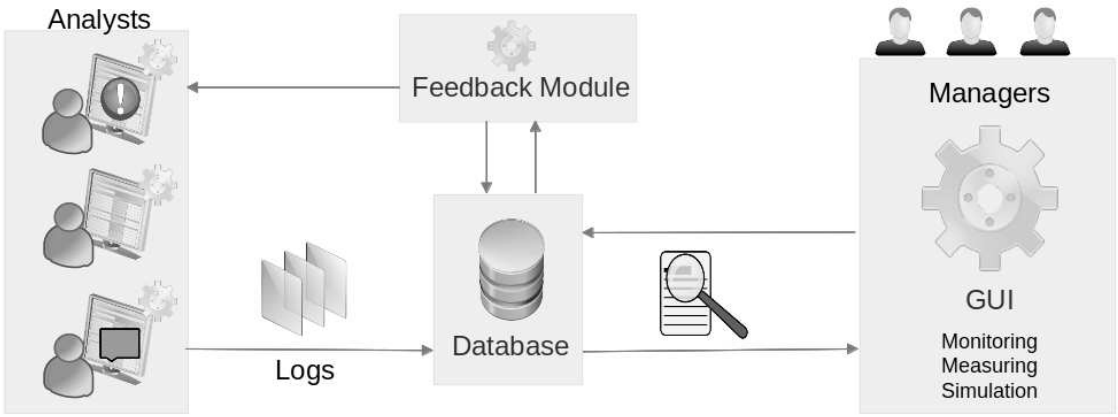


Figure 5: The proposed system architecture

4.1.1 Monitoring Module

Figure 6 shows the monitoring module of the managers' GUI. This module is provided to illustrate the detailed information of the SOC ongoing investigations in an easy-to-understand way. The main purpose is showing which analyst is performing which type of investigation for which client. Investigations started within a configurable period (e.g., 30 minutes) stay in the graphical plot.

By looking at the plot, the y-axis shows different investigations, the x-axis shows the investigation duration. Each stacked bar is an ongoing investigation in the SOC. A vertical line, drawn on the plot dividing a stacked bar to two, represents the current time. The stacked bar on the left side of the vertical line shows completed steps of the investigation, and a bar on the right side is the representation of the next predicted

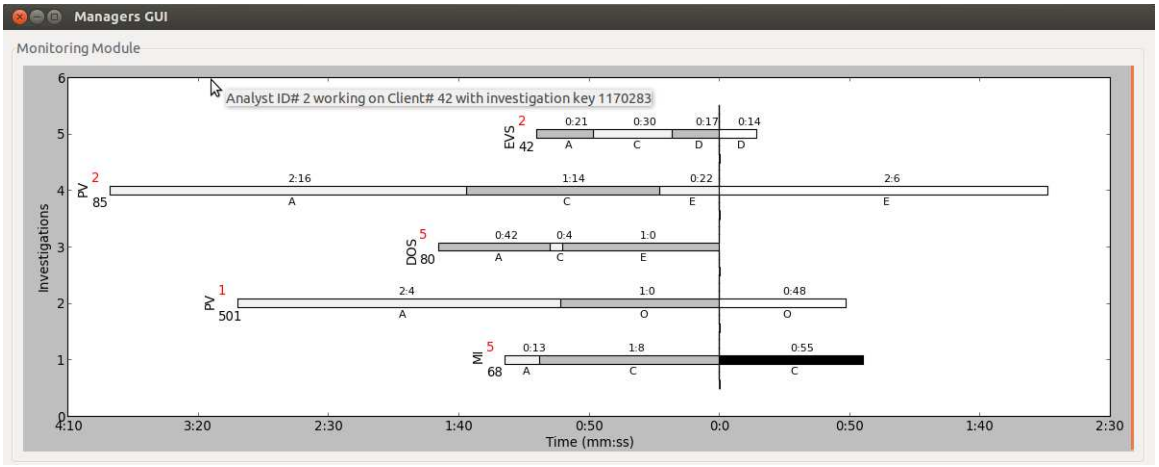


Figure 6: The monitoring module dashboard

step. Next step prediction is done by mapping the ongoing investigation to the relevant model, which will be detailed in section 4.2.1. The plot is continuously updated after configurable time intervals by reading new data from database incrementally.

By considering investigation number 2 in the monitoring module in Figure 6, we describe an example to elaborate the plot. As is shown, steps A, and O are performed for the investigation number 2 in the figure, since they are on the left side of the vertical line. Step IDs are displayed at the bottom side of the bar, where related duration are shown on the top. For instance, for step A, duration is 2:04. The time format is minutes and seconds (mm:ss). The next step predicted by the tool is shown next to the completed steps after the vertical line. The predicted step is O in this case, as the tool expects step O is still going on. The duration of the predicted step is based on the historical average of a step calculated from previous investigations in the database.

On the left side of a stacked bar, some information such as investigation type, analyst ID and client ID related to the investigation are shown. As we can see, for the investigation number 2, Investigation type is PV. Analyst ID and Client ID are 1 and 501 respectively. By clicking on the plot more detailed information will be provided. For instance, by clicking on the left side of the plot, as we can see in Figure 6, more information related to analyst ID, client ID and investigation ID is appeared.

4.1.2 Measuring module

To understand the SOC performance better, measuring becomes necessary. A recent book on IT security metrics [48] states that every metric which can reduce the uncertainty is a good metric. Qualitative measurement usually is ignored as it is difficult or expensive to be scaled. Therefore, the measuring module provides managers with quantitative measurements of the SOC performance.

To the best of our knowledge, there do not exist SOC analysis performance studies in the literature. In a recent paper [11], Jacobs et al. examined different aspects of three real-world operational SOCs. They expressed one performance metric considered in one operational SOC as the number of analyzed incidents per analyst daily. It is described that counting the number of incidents solely is not a good metric, if the time duration of analysis is not considered. Efforts on difficult investigations which take more time to analyze reduce the total number of processed incidents. As a result, analysts are not motivated to analyze carefully, since it results in showing lower productivity for them in managers' point of view. In a university SOC assessed in the same work [11], a ticketing system dispatching alert tickets to analysts, provides a performance metric which is time spent on each ticket; however more detailed information is not provided in [11].

The measuring module is designed to provide various SOC's performance metrics to evaluate SOC behavior. By having analysts' activity logs from the SOC Console, providing different performance metrics from different perspectives is possible.

We enumerate different metrics provided by the measuring module. Important results, such as the maximum values, are shown in the main GUI where detailed analysis is reachable through different buttons. Figure 7 shows the measuring module dashboard, and the two detailed analysis reports are open in the two windows.

The first group of metrics provides different average analysis time (AAT) from different perspectives. Total AAT which is the average duration of investigations regardless of any condition gives a general inference about overall performance of the SOC. AAT of each investigation type, each analyst, and each client are metrics focusing on those parameters. By these performance metrics, it is possible to identify time consuming investigation types, demanding clients, and fast analysts. With the last 4 metrics, more detailed information is provided for managers to see which analyst is faster for which investigation type, which client is demanding for which investigation

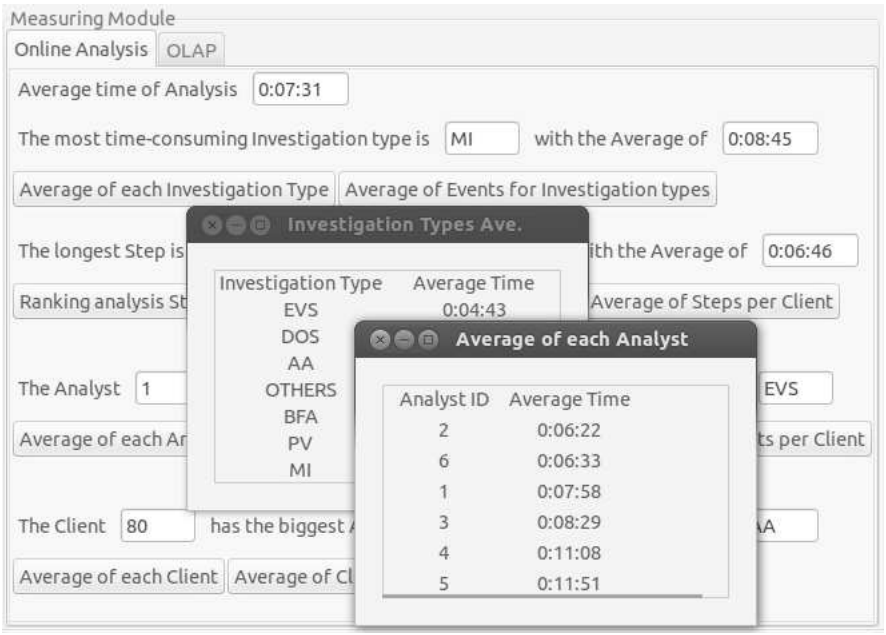


Figure 7: The measuring module dashboard

type, and which analyst is taking more time for which client, which can imply not knowing the client well.

- Total average analysis time.
- Average analysis time of each investigation type
- Average analysis time of each analyst
- Average analysis time of each client
- Average analysis time of different analysts for each investigation type.
- Average analysis time of different investigation types for each client.
- Average analysis time of different analysts for each client.

The second group of performance metrics concentrates on average duration of different investigation steps. This group attempts to show steps' average duration in general, for different analysts, and different clients.

- Ranking analysis steps according to the time spent in executing them.
- Average analysis time of different analysts for each Step.

- Average analysis time of different clients for each Step.

The third group is about showing maximum values for some performance metrics in the managers' GUI. This group helps managers to determine if maximum values are significantly different from average durations.

- Identifying which investigation type takes more time to be analyzed.
- Which step of which investigation type takes more time to be analyzed.
- Which analyst has the highest average time for which investigation type.
- Which client has the highest average time for which investigation type.
- Which analyst has the highest average time for which client.

The fourth group is the number of created and updated incidents for different parameters.

- The number of created and updated incidents for each investigation type.
- The number of created and updated incidents for each analyst.
- The number of created and updated incidents for each client.

OLAP component

An Online Analytical Processing (OLAP) tool is employed beside the designed measuring module inside the managers' GUI. OLAP empowers managers to create new analysis queries with mouse dragging and clicking instead of modifying code or writing complicated SQL queries. An open-source web-based OLAP engine, Community Edition Saiku (CE Saiku) [49], is integrated into the GUI providing customized data analysis opportunities. CE Saiku is implemented by Pivot4J Java API using Mondrian OLAP server. We integrate web-based Saiku to the managers' desktop GUI by embedding a browser.

Saiku configuration for SOC performance metrics is discussed beside an example in section 4.2.2.

4.1.3 Simulation Module

The simulation module is designed to show effects of potential changes on the investigation workflow. It simulates the effect of a change on real activity logs, and recalculates performance metrics. It helps the managers to prioritize efforts on potential improvements of the SOC. Two potential changes as simulation options are provided. One is modifying current steps' duration, and the other is changing the dispatching method of alerts.

The first simulation capability is modifying specific steps' duration by a specified percentage to see how different performance metrics would be affected. This simulation helps managers to find out whether it is the best option to optimize one specific step to reduce the time taken by that task.

By assessing optimization options for each step, the manager decides one or more steps duration to be modified and by a specific percentage. An optimization option can be a possible automation for a specific step to reduce analysts' tasks. The reduction percentage is the manager's prediction as a result of that potential change in the investigation workflow. For example, if one step is going to be automated completely, the related duration of the step should be removed completely (reduction percentage is 100%). The simulation is designed in a way that the simulator considers all historical database investigations containing that specific step, and modifies all current sub-steps' (actions) durations by the mentioned value, and recalculate all performance metrics based on the managers assumption.

The second simulation capability is simulating the dispatching phase of incoming alerts among analysts with a different queueing model. Different ways of dispatching services (alerts) among servers (analysts) is usually studied as queuing models [50]. In our work, a different alert dispatching method is simulated to assess the employed approach effect on the SOC performance metrics.

The result of both simulation scenarios is shown side by side with the real one (measuring module) in the GUI to ease the comparison process for managers. Figure 8 shows an example of the provided simulation results beside the measuring module. In this way, comparing the effect of the simulation scenario on actual SOC performance metrics is easy for the managers. For instance, as we see in Figure 8, the simulation reduces the average time of analysis from 6:41 to 5:53. By the measuring module results, we see the most time consuming investigation type is PV with the average

of 10:17, the conducted simulation reduces it to 6:03 however the investigation type is not changed. Moreover, we can compare easily that the biggest investigation type average duration belongs to AnalystID 2 for PV with the average of 45:05, and this simulation changes it to AnalystID 4 for MI with the average of 11:42.

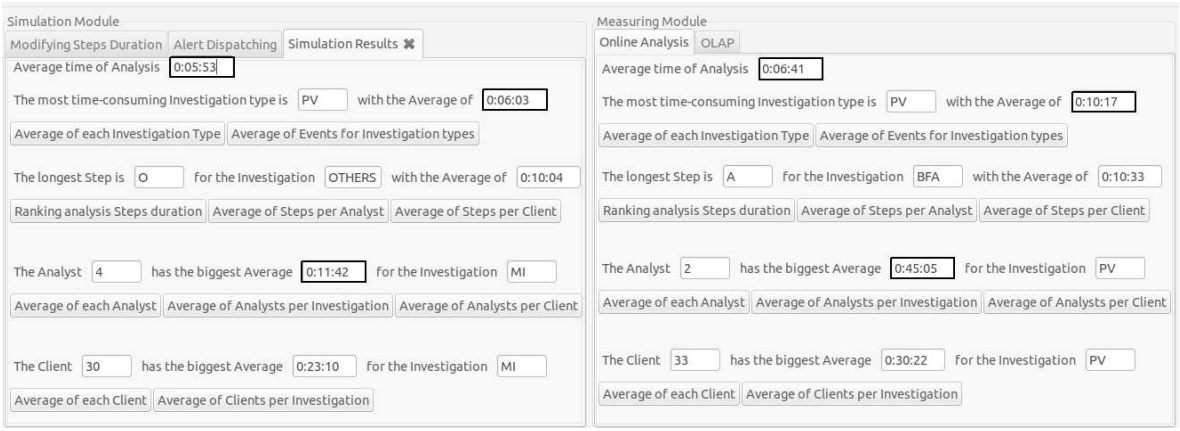


Figure 8: The results of the simulation module alongside the measuring module

4.1.4 Feedback Module

The feedback module is designed for knowledge transfer among SOC analysts. Knowledge can be a hint about next required action to proceed in the investigation, notifying the analyst about anomalous durations and missing steps, or showing the result of previous similar investigation types performed by other analysts. The analysts are firstly notified about mentioned information in the Windows operating system tray, and the summary of all notifications are accessible in a desktop GUI.

The feedback Module works as a background service uninterruptedly as analysts activity logs are being stored in the database. It keeps reading the database, and mapping them to the investigation model. As a result, the system can notify analysts about next probable steps, find anomalies, and warn analysts about them. The module reminds the analyst what is the probable next step based on the step he is currently performing. Anomaly notifications are about spending normal duration on the steps, or not missing an action. For instance, If the analyst takes 50% (which is configurable) less or more time than the historical average duration of the step, he will be warned. If the analyst's activities does not match one of the investigation model's paths, he will receive a warning about the missing steps.

Moreover, once an analyst starts to perform an investigation, the feedback module shows previous investigations activity logs of the same type by different analysts. This feature provides background knowledge from other analysts' approaches with different clients, or result of similar investigations for the same client. The results can be filtered by a specific client to provide knowledge whether the client recently had the same event type, what was the result, which source or destination IP were involved, etc. The main investigation approaches adopted by analysts are the same. However, analysts' knowledge can be improved over time with experiences and knowing clients' environment better. Moreover, the feedback module shows results of investigations which can help the next analyst to have an inference about similar situations, for instance, the same event is generated for the same source and destination IP which was false positive before. Then, the next analyst by knowing the history of the client about this specific event, and result of previous analyst's investigation, can take a faster action with the knowledge of previous analyst.

The feedback module is shown in Figure 9. As is shown, different hints are provided to the analyst. The analysts receive notifications in real time about their ongoing investigations in the Windows operating system tray, where all notification reports are accessible by the GUI.

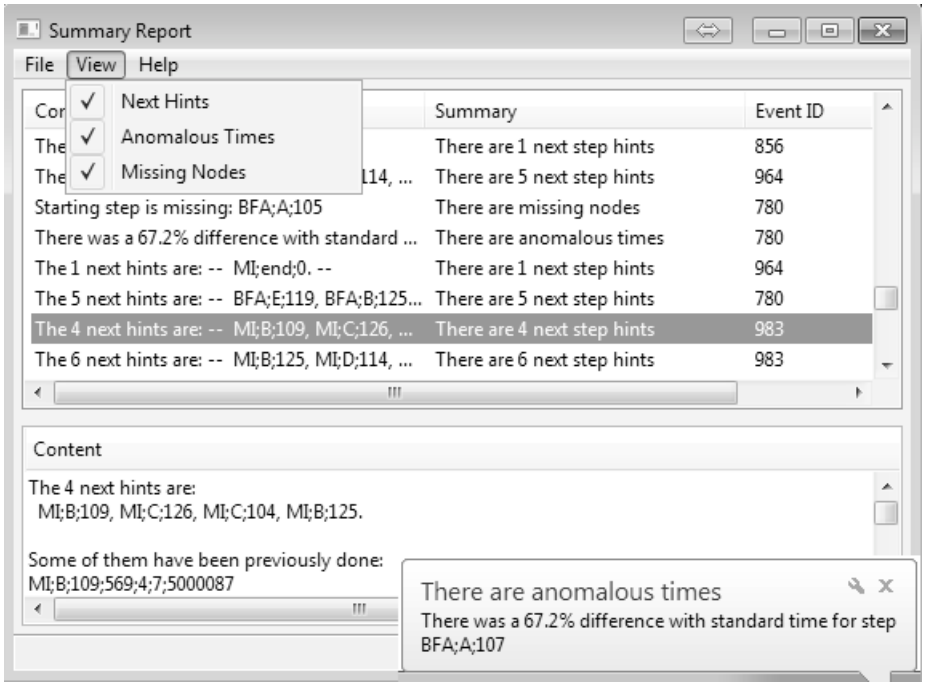


Figure 9: The feedback module notifications and reports

4.2 Implementation

In this section, the methodology and implementation details of each module are discussed. All modules are implemented in Python 2.7 programming language and existing libraries.

For the managers' GUI, since three modules (monitoring, measuring, and simulation) are integrated in the same window, threading objects [51] are used to serve each module's refreshing time. In this way, different updating times for each module (specifically, monitoring and measuring) does not block the main program. Both monitoring and measuring modules are updated periodically after configurable time intervals in the managers' GUI.

The smallest scale of time duration shown in different modules of the tool is step's duration. Series of actions are grouped to a step representing a task.

Moreover, All configurable parameters mentioned in the work are reachable in a single table in the database.

4.2.1 Monitoring Module

The characteristics of the visualized plot are discussed in Section 4.1.1. Visualization of the monitoring module is implemented with Matplotlib package [52] in Python. As is discussed, the plot is being continuously updated after configurable time intervals. With each update, the plot is refreshed, and recent logs, which are performed steps from the last update, are added to the plot. Here, the employed methodology to implement the module is discussed.

In each update, the module fetches StepID and ActionID from the last log of the investigation, and composes the mapping key. Then, it maps the key value to the relevant node of the investigation model. By traversing the investigation model, it can recognize what is the next probable action. If the next node is End, it shows the investigation is completed, otherwise it shows the next probable action and step. Moreover, it is possible to have more than one action as the next probable actions, since there are several paths due to decision nodes and integration of all possible paths. In this case, the predicted bar after the vertical line (shown in the plot) will be in black-color meaning there are some probable actions based on the current action. A white-color bar means there is only one possibility as a next action.

The estimated duration of a predicted step in the plot is calculated based on the average duration of a step from historical data. It is also possible that the analyst is in the middle of one step, in this way the next predicted StepID is the same as the performed step. For plotting it, the StepID will be the same, where the related estimated duration will be the subtraction of performed duration of the step from the total average of the step.

Differentiating between performed steps (past) and predicted steps (future) in the plot is achieved by the vertical line. To plot the vertical line representing current time, the tool fetches all performed steps' durations from the database into the memory as negative values, and historical average durations for the predicted steps as positive values.

Two dictionaries are defined to read data into the memory to plot investigation details as stacked bars. Dictionaries are a data structure in Python as a pair of key-value. Considering a single investigation, there are two dictionaries containing steps' durations of the investigation. The key of both dictionaries are InvID, where the values of keys are different. One dictionary values are series of performed StepIDs duration, and the other is average duration of predicted step (which could be more than one). InvID is the dimension of y-axis which is the same for both dictionaries, and the duration of different StepIDs, which are the length of different steps, are illustrated as x-axis. As is described, performed steps' durations are stored as negative values in one dictionary, and predicted step's duration is stored as a positive value in another. In this way, negative and positive values related to the same InvID are plotted on the same y axis, where performed steps and the predicted step can be differentiated by the vertical line.

4.2.2 Measuring Module

The measuring module is designed to provide SOC performance statistics with different metrics. The SOC performance metrics are described in Section 4.1.2. By having the measurable investigation workflow, performance metrics can be calculated by running SQL queries on the database. The duration of an investigation is measurable from provided analysts' activity logs with different attributes.

PostgreSQL [53] as a free and open source object-relational database system is employed in the implementation supporting SQL standard queries.

The first and most important basic SQL query is the one to calculate a single investigation duration. For calculating the related duration of one investigation, by considering the InvID, all actions' durations belonging to the same step are added together. Then, all steps of the InvID are summed to represent the investigation duration.

Since in reality there is a high chance of either skipping or redoing actions by analysts, we do not put any limitation regarding number of actions that could be added together for one step. As long as they are related to the same step of same investigation with the unique InvID, duration times are added together.

By the employed methodology, we have different durations related to each action, each step, and the entire investigation. Having duration of each step of the investigation model provides the opportunity to perform average queries from specific perspectives, such as average of each step by each analyst. For example, it is possible to look at performance of each analyst for each step to see who can manage which step in a better way.

Having the duration of each single investigation in the database provides the opportunity to perform average queries to calculate the investigation average duration from different point of views, such as investigation average duration of each analyst, investigation average duration of each client, etc.

Saiku OLAP Configuration

The functionality of web-based OLAP application, Saiku, is discussed in the subsection of Section 4.1.2. Saiku is integrated into the managers' GUI next to measuring module to provide more statistics on investigation logs. Besides OLAP configuration for the SOC database, it is described how to integrate web-based OLAP in the desktop GUI.

The employed package for integrating browser into the desktop GUI is CEF python [54]. By this package, A web browser is embedded in one tab of the GUI. By setting the web address of configured Saiku for the browser, managers can work with Saiku application through the GUI.

Configuring the OLAP tool is accomplished by designing a schema. *Schema* as a logical model defines the data format for calculations. *Dimensions* are the attributes that we want to measure (e.g., InvestigationTypeID attribute). *Measures*

are those quantitative attributes (e.g., TimeStart attribute) related to dimensions. Calculations on measures are defined by different *aggregators*, such as sum, count, avg, etc. One cube can contain several dimensions and measures to represent data in a multidimensional form. A schema (logical model) can lead to many cubes containing different dimensions and measures. In the following, the implemented cube to measure investigation durations is described.

The schema containing one cube is designed by considering different attributes of analysts' activity logs to provide detailed investigation measurement to managers. Dimensions considered for the cube are: investigation types, analysts, and clients. Related measures are: TimeStart, TimeEnd, and InvID. For each measure, we should set a proper aggregator. For TimeStart and TimeEnd attributes, the aggregator is sum. For InvID, the aggregator is distinct-count counting the number of distinct values (the measure is called "Number of Investigations").

By simply using aggregators on measures of dimensions, SOC performance metrics cannot be directly achieved. We employ the method, *calculated member*, to facilitate writing formulas to combine different measures together. For instance, as a basic calculation, different investigations' durations are calculated. Then by using the calculated member, we calculate the average of investigation durations. The calculated member is "Average per Investigation".

Once the cube is configured in the XML format, managers can make different queries by dragging and dropping listed dimensions and measures. For example, by selecting the dimension "Investigation Type" and the measure "Number of Investigations", we can see how many investigations are performed for each investigation type in the dataset. If we add other dimensions, such as analysts and clients, as well as more measures, such as "Average per Investigation", detailed information will be shown. Figure 10 is part of the result of this example, and the time is in minutes. It shows which analysts performed which investigation types (here analysts #3 and #4 performed the investigation type AA), which analysts performed the investigation type for which client (here analyst#3 performed the investigation for client#67), and the investigation average duration for the client analyzed by the analyst.

Investigation Types	Analysts	Clients	Number of investigations	Average per Investigation
AA	3	67	1	0.525
	4	68	1	1.809
BFA	1	19	1	22.415
		30	2	4.6
		67	1	49.942
		500	1	6.536
	2	30	1	4.674
		67	2	12.037
	4	67	1	0.593
		80	1	2.404
		504	1	7.721
	5	67	1	2.125
		504	1	7.911
	6	45	2	1.945
		48	1	0.507
		67	1	4.673
DOS	2	80	1	7.784
	4	45	1	1.396
	5	19	1	15.6

Figure 10: Partial OLAP analysis results

4.2.3 Simulation Module

None of the two simulation scenarios affect the production database. The alert dispatching simulation is provided in a replicated database by storing the simulated logs, and the modifying steps' duration simulation is performed by SQL queries in real time.

Modifying Steps' Duration

For modifying steps duration, different SQL queries are provided to simulate the same performance metrics as measuring module metrics. There are different SQL queries than measuring module queries, since simulation queries firstly apply the reduction percentage for the target steps in every investigation, then calculate different performance metrics and show the results with the same format of the measuring module.

Alert Dispatching Method

The default alert dispatching method is assigning the same number of clients to the available analysts of a work shift. Analysts are responsible for their assigned clients and work independently. For instance, we have five available analysts and 15 clients, each three clients are assigned to one analyst regardless of any consideration. This approach has some pitfalls, such as the possibility of assigning clients with high load of alerts to the same analyst, where the other analyst is assigned clients with low load of alerts. It is also possible that two clients assigned to the same analyst are under attack at the same time, and the analyst cannot handle both of them at the same time, where none of clients of the other analyst are in such a critical situation.

The employed simulation method follows single-queue, multiple-servers methodology. Considering several servers serving clients from a single queue is known as the $M/M/c$ model where c is the number of servers [50]. The discipline in these systems is first-come, first-served and the arrival rate of jobs is based on Poisson process.

We employ a job routing idea proposed by Armony [55], named Fastest Servers First (FSF). The idea is routing the incoming jobs to the fastest servers first resulting in the overall performance improvement for the system. In our work, a fastest server is the analyst with the lower average analysis duration for an assigned investigation type.

In order to dispatch each alert from the queue, average analysis durations of different analysts for the related investigation type are calculated, and the alert is assigned to the most efficient analyst for that investigation type. For example, if the incoming investigation type is MI; the simulator knows the analyst#6 is the most efficient available analyst for this type, and it assigns the alert to the analyst#6. In case the most efficient analyst is busy, it assigns it to the next best analyst.

In this simulation, two metrics will be provided to assess the effect of the simulated method. One is the average analysis duration, and the other is the alert waiting time. The waiting time is the time the alert stays in the SOC Console (queue) to be analyzed. It is the subtraction of the start time of an investigation from the arrival time of an alert in the queue. In our study, waiting time implies the response time of the SOC to incidents which is an important factor for the managers to respect SLA for different clients. By considering waiting time, we can also assess how the new approach would affect the waiting time of the alerts. The simulation will be detailed

in a case study in Section 5.3.

4.2.4 Feedback Module

WxPython user interface library [56] is used to implement the feedback module to notify the SOC analysts of various information in the MS Window operating system tray and the desktop GUI. Different methodologies, such as finding missing steps, predicting the next step and its estimated duration, revealing duration anomalies, and displaying previous similar investigations details to analysts are employed by this module.

Mapping analysts' activity logs to the investigation model to find the path followed by analysts is discussed in section 3.2. Predicting the next step and its estimated duration is also discussed in details in section 4.2.1, since the monitoring module has the same feature to show next probable steps to the managers.

The algorithm to find missing steps maps analysts' incoming activity logs to the investigation model continuously to see whether a step is missed. It starts checking logs of one investigation from the first log. It checks every two successive logs (adjacent) in the database to see whether they are also successive in the model. If they are not adjacent in the model, there is one or more missing nodes between them. Another algorithm is employed to find a shortest path between two nodes. Since it is possible to have several paths between two nodes, the shortest path, including less nodes, is selected to report missing actions or steps.

In order to report duration anomalies, as some analysts may not spend enough time on some steps, a dynamic duration standard is provided base on historical data for each specific step. The standard duration is considered as real-time average of historical data in a range, as follows.

$$(1-n) * AverageDuration < StandardDuration < (1+n) * AverageDuration \quad (1)$$

For instance, by applying the above formula, if n is 20% and Average is 60 sec, normal duration range is between 48 and 72. The alternative range percentage is configurable.

Once an analyst starts to perform an investigation, previous investigations logs of the same EventID will be shown to the analyst. As is discussed in Section 2, several alert types (EventID) are categorized to investigation types. Since investigation types

are general categories, investigations of the same EventID are shown to the analysts than the investigations related to a same investigation type.

Chapter 5

Case Studies

Three case studies are elaborated in this section to show the effect of the designed system on improvement of the SOC performance. Firstly, we go through the dataset pre-processing and provide different statistics of the dataset.

5.1 Dataset Pre-processing

A pre-processing step is implemented to remove data-gathering mistakes resulting in out-of-range values, impossible data combinations, and missing values. As is discussed in Section 3.2, each row of analysts' activity logs represents one single action of one investigation performed by an analyst with different attributes (TimeStart, TimeEnd, InvestigationTypeID, StepID, ActionID, SourceID, EventID, AnalystID, ClientID, IncidentID, InvID). The duration of each action is calculated by the subtraction of TimeStart from TimeEnd. The duration of relevant actions of one specific step will be added together to represent the step's duration. Then, accumulated steps' durations form the investigation duration.

Out of range values implies those steps' durations which are too long compared to real durations and need to be normalized. For instance, when normal step's duration is in a range of 10 minutes and information from the gathered log shows the duration is one hour, which is abnormal. One possible reason for these abnormal cases is that the TimeEnd of each action is considered as TimeStart of the next action. Therefore, if there is a gap (e.g., analysts take a leave) between sequential actions, the large duration is possible. Such large duration for steps is considered as a noise in our

dataset and is normalized. More specifically, steps duration more than 30 minutes is considered as a noise and normalized in the preprocessing phase.

Impossible data combination emerges after analyzing each single investigation logs. There can be different ClientIDs in one investigation related to a specific client, since logging the analysts' activities by our system is tracking their mouse clicks and it is possible that analysts mistakenly click on the other parts of the SOC Console relevant to other clients than the one they are performing an investigation for. This kind of noise also needs to be removed from the dataset to fit data into the tool.

Missing values are about non-existent values for different attributes of a log entry. During the logging phase, the recording log system does not guarantee to provide values for all attributes (InvestigationTypeID, SourceID, EventID, IncidentID) for each single action of investigation which is one row of log entries. It is possible that some values are not retrievable, and they will be filled by zero automatically; however, those values can be retrieved by looking at the entire investigation logs. Zero values could appear in two situations; either when all values of one attribute are zero in the investigation log entries, or some of them are zero. If all values of one attribute related to an investigation are zero, it means the value could not be fetched during logging phase which is considered as unknown. If even one of the values is filled by a value rather than zero, other zero values related to the same attribute should be replaced by that value in the pre-processing phase.

It is observed there are some investigations with zero or milliseconds duration, which are more related to false positive or contextual events. Analysts close these events immediately resulting in very fast investigations. Contextual events can be different assets syslogs such as failure login attempts that could be suspicious in specific situations. Once analysts infer nothing goes wrong, they clean them from the alert queue without much investigation. Since these short investigations affect different performance metrics enormously, investigations without expanding alert content are eliminated from the dataset.

Different activity logs from different analysts' machines are being gathered and stored at the same time by the logging script. Firstly, different analysts' activity logs are separated from each other ordered by time. Then, different investigations of the same analyst are distinguishable from each other by a specific ActionID. A new investigation starts with ActionID# 130. By separating each investigation from

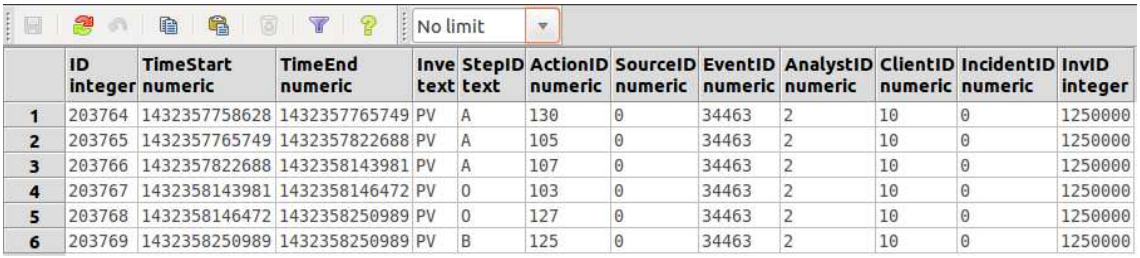
others, we are able to assign each investigation a unique identifier as InvID.

Here, we provide an example of one investigation activity logs to show the pre-processing phase which is performed to prepare raw logs to fit into the table of database. Table 8 represents raw data in a text format file and Figure 11 shows processed data placed in the database. At first glance, Figure 11 has two more attributes than Table 8, the first attribute is ID, an auto increment primary key of the table, and the last attribute is InvID. All related log entries of a single investigation receive the same unique identifier which is 1250000 here.

TimeStart	**TimeEnd**	**InvestigationTypeID**	**StepID**	**ActionID**	**SourceID**	**EventID**	**AnalystID**	**ClientID**	**IncidentID**
1432357758628	1432357765749	0	1	130	0	0	2	10	0
1432357765749	1432357822688	0	1	105	0	0	2	10	0
1432357822688	1432358143981	1	1	107	0	34463	2	10	0
1432358143981	1432358146472	0	16	103	0	0	2	10	0
1432358146472	1432358250989	0	16	127	0	0	2	10	0
1432358250989	1432358250989	1	2	125	0	34463	2	10	0

Table 8: Raw activity logs in the format of text file

InvestigationTypeID and StepID values are mapped from numbers to predefined codes of the system in the pre-processing phase. For instance, in the text file, value 1 for the InvestigationTypeID attribute is an identifier of Policy Violation (PV) investigation type. Consequently, those numbers are mapped to the abbreviated forms of their investigation type names. Similarly, it is done for StepID attribute, codes 1, 2, 3, ... are mapped to A, B, C, ... respectively. In Table 8, we can see values of InvestigationTypeID attribute are zero and one, firstly all values of this attribute are changed to one. Then all values are mapped to PV as we see in Figure 11. If all those values related to the InvestigationTypeID were zero, it would mean the InvestigationTypeID was not retrievable. And, the algorithm would assign OTHERS type



No limit

	ID integer	TimeStart numeric	TimeEnd numeric	Inve text	StepID text	ActionID numeric	SourceID numeric	EventID numeric	AnalystID numeric	ClientID numeric	IncidentID numeric	InvID integer
1	203764	1432357758628	1432357765749	PV	A	130	0	34463	2	10	0	1250000
2	203765	1432357765749	1432357822688	PV	A	105	0	34463	2	10	0	1250000
3	203766	1432357822688	1432358143981	PV	A	107	0	34463	2	10	0	1250000
4	203767	1432358143981	1432358146472	PV	O	103	0	34463	2	10	0	1250000
5	203768	1432358146472	1432358250989	PV	O	127	0	34463	2	10	0	1250000
6	203769	1432358250989	1432358250989	PV	B	125	0	34463	2	10	0	1250000

Figure 11: Rows of logs related to one investigation placed in the database

for the InvestigationTypeID, since it was not related to a known investigation type.

There is a possibility when an analyst is not convinced about an action and he will repeat it. Consequently, it is possible to see two or more log entries with a same ActionID happening one after another with different TimeStart and TimeEnd values. As the algorithm adds actions' durations related to the same step together, only spent durations on one step will be considered.

5.1.1 Statistics of Dataset

Our dataset is categorized into three different classes based on investigations results. As is discussed, an investigation can result in creating a new incident, updating one of current incidents, or closing the alert. When the related investigation of an alert results in creating an incident, it indicates a possible threat confirmed by an analyst's indications (proofs). Then the analyst contacts the related client for the incident escalation based on the escalation grid. The incident is open till the related client either confirms or clarifies the situation. In this gap (client's response), which could be from one day to one week depends on incident criticality, incoming alerts will be added to the current incident. Updating the recorded incident with more alerts or client's response forms the category of updating incidents. False positive alerts are categorized into the closed alerts class.

The time period of the dataset is from June 2015 to August 2015 for 57 days. 6 analysts perform alert analysis for 40 clients. The time duration format in Table 9, and all following tables is mm:ss. As is shown, 40.7% of investigations result in creating or updating incidents, where 59.3% of total investigations are about closing alerts after an investigation. Average analysis duration of investigations ending in creating new incidents is 17:52, while average duration of updating an incident and closing

Parameters	New Incidents	Updated Incidents	Closed Alerts	Total Dataset
Number of log entries	3301	1558	9380	14239
Number of investigations	194	331	765	1290
Proportion	15.04%	25.66%	59.3%	100%
Total average analysis duration	17:52	06:41	05:16	07:32
Average duration of fastest analyst	12:10	03:59	04:44	06:22
Average duration of slowest analyst	22:38	09:46	11:27	11:52

Table 9: The dataset statistics

an alert are 06:41, 05:16, respectively. Moreover, different analysts have different average analysis durations. The slowest analyst's average duration is 11:52 whereas the fastest one is 06:22.

The different average investigation durations of different investigation types are shown in Table 10 based on three investigation results. For each column, the first element is the number of related investigations, and the second is the average investigation duration. For all investigation types, we can see the average duration of creating a new incident category is longer compared to updating the incident and closing the alert categories. EVS, MI, BFA and PV investigation types take more time to gather indications to create an incident than AA, DOS, and OTHERS. Most popular attack type is MI with 47 distinct incidents in the dataset, whereas DOS has the least number of created incidents, 4. After MI, other two popular attack types are BFA and PV. Since the number of cases for OTHERS type is much more than other known categories, it is not mentioned in our comparisons for known investigation types.

For almost all investigation types, average time of updating an incident is more than closing the alert of the same type. Exception is the AA type whose number of related cases (#2) for updated ones is not enough to be considered as a counterexample. However EVS attack type takes more time to be confirmed as an incident, average duration of getting closed is the least (03:27) among other types. The most time consuming investigation type for updating an incident is PV type, which is reasonable as it mostly needs communicating with the client to clarify the situation.

Looking at the total dataset statistics, MI, BFA, and PV are most time consuming alert types in the SOC regardless of the investigation result. The number of received

Investigation Type	New Incidents		Updated Incidents		Closed Alerts		Total Dataset	
-	#	Avg	#	Avg	#	Avg	#	Avg
EVS	6	20:50	11	06:28	91	03:27	108	04:43
MI	47	20:31	68	07:04	183	06:22	298	08:45
BFA	24	20:18	17	08:38	102	05:16	143	08:11
PV	20	19:46	10	10:17	80	05:41	110	08:40
AA	6	19:27	2	01:10	67	05:25	75	06:25
DOS	4	17:35	5	08:13	77	04:55	86	05:41
OTHERS	87	15:02	218	06:17	165	04:58	470	07:26

Table 10: The dataset statistics regarding different investigation types. Here, average values are time duration in the format of mm:ss, and the other column (#) is the number of instances.

alerts from these attack types is highest beside EVS, although EVS alerts are analyzed quickly. Trend shows MI type has the highest number of investigations (298), and the highest average analysis time (08:45).

5.2 Case Study I, Modifying the Duration of Steps

This case study is about assessing potential steps which could be automated to improve overall performance. Every change in a system needs to be assessed before going into production. After going through all steps that analysts perform for an alert investigation, two possible automation options are recognized. One is about checking clients' assets and the other is about checking escalation grid. These two are considered as potential improvements to the current investigation workflow.

5.2.1 Checking clients' assets (part of step C)

As described before, there is one step about checking a client's assets. The reason behind checking the client's assets is to assure the raised alert matches to the assets' vulnerabilities. For instance, if the raised alert is about vulnerabilities of a specific version of software and the client does not have that specific version, then raised alert is apparently false positive. This example could be extended to all kind of assets,

resources, and different related versions and patches.

A possible solution to automate this step is correlating alerts with the events triggered by vulnerability scan of a client's assets. In other words, alerts can be correlated based on the client's assets vulnerabilities for not raising FP alerts. In this way, this task could be handled by the SOC automation solution than the human analysts.

In this simulation scenario, we assess how the possible solution would affect the SOC performance. As we know, investigations can result in three different categories, creating a new incident, updating the current incident, or closing the alert as FP. For the first two categories (creating or updating incidents), the simulation is done in a way that the related actions' duration to asset checking are eliminated completely, then average analysis durations are recalculated to show the improvement. For the investigations of the closing alerts category containing this step, a careful observation is made whether the reason for closing the alert was about mismatching asset's vulnerabilities and the raised alert. If the condition is true, we claim the entire investigation was useless, since by implementing the provided solution, there would be no more FP alert in this regard. As a result, FP alert reduction aside, analysts' time would be saved. If the alert is not closed immediately after checking this step, we just deduct this step duration from the investigation duration.

For the first two categories of investigations (creating and updating an incident), we remove related actions for checking assets, then following simulated results are calculated. Table 11 shows the results for creating a new incident class, where the dataset average, the simulated average and the reduction ratio are represented in this table. Generally, 48.45% of investigations in this class contains an asset checking step which is involved in this simulation. The total average of this investigation class is decreased by 7.55%. The most affected investigation types are AA, EVS, and MI. Table 12 depicts the simulation result of the updating incidents class, where 12.69% of investigations has the asset checking step. The total average of this class is decreased by 6.98%.

For the closing alerts category, the step for checking a client's asset is performed in 30.2% of investigations (231 out of 765 investigations) which can be eliminated from the analysis process similar to other two investigation categories scenarios. Specifically, 41 out of 231 (17.75%) alerts of these investigations are closed immediately

Investigation Type	Dataset Average	Simulation Average	Reduction Ratio
EVS	20:50	18:17	12.24%
MI	20:31	18:15	11.05%
BFA	20:18	19:22	4.6%
PV	19:46	19:25	1.77%
AA	19:27	16:13	16.62%
DOS	17:35	16:57	3.6%
OTHERS	15:02	14:01	7.62%

Table 11: Class: creating new incidents; 94 out of 194 investigations (48.45%) contain step C. By the simulation, the total average analysis duration decreases from 17:52 to 16:31, or 7.55%.

Investigation Type	Dataset Average	Simulation Average	Reduction Ratio
EVS	06:28	05:10	20.10%
MI	07:04	06:24	9.43%
BFA	08:38	06:47	21.43%
PV	10:17	09:08	11.18%
AA	01:10	00:30	57.14%
DOS	08:13	08:13	0.0%
OTHERS	06:17	06:04	3.45%

Table 12: Class: updating incidents; 42 out of 331 investigations (12.69%) contain step C. By the simulation, the total average analysis duration decreases from 06:41 to 06:13, or 6.98%.

Investigation Type	Dataset Average	Average after eliminating investigations	Final Simulation	Reduction Ratio
EVS	03:27	03:28	3:11	8.17%
MI	06:22	06:14	5:44	8.02%
BFA	05:16	05:19	5:01	5.64%
PV	05:41	05:55	5:36	5.35%
AA	05:25	05:08	4:59	2.92%
DOS	04:55	04:55	4:46	3.05%
OTHERS	04:58	04:50	4:35	5.17%

Table 13: Class: closing alerts; 5.36% of investigations are removed, 231 out of 724 investigations (31.9%) are involved in this simulation. By the simulation, the total average analysis duration decreases from 05:12 to 04:54, or 5.77%.

after checking the step indicating useless investigations. Since they are closed immediately after this step, it is assumed that the raised alerts did not match the assets vulnerabilities. As a result, they could be cleaned from the alerts repository before reaching analysts' SOC Console. By implementing the solution, the rate of false positive alerts would be decreased by 5.36% (41 out of 765) in this class, where 266 minutes of analysts time (around four and a half hours) would be saved. For the remaining 190 investigations containing this step, we eliminate the asset checking step and recalculate the results for the closing alerts category. Results are shown in Table 13 illustrating 7.28% decrease in the total average analysis duration.

Considering all investigation categories together, Table 14 illustrates different simulated average analysis durations for different investigation types. After removing 41 investigations from the closed alerts category, 29.38% of investigations are affected by this scenario, and the total average analysis duration decreases by 6.83%. The most affected attack types are EVS, MI, and BFA, where PV and DOS attacks are improved to a lower degree.

Considering investigation classes, the summary of simulation results is shown in Table 15.

5.2.2 Checking escalation grid (step F)

One analysis step is checking escalation grid to find out how the related client should be informed in case of a new or updated incident. Each client's escalation grid as an informative document is accessible through some mouse clicks in the SOC Console.

Investigation Type Type	Dataset Average	Average after eliminating investigations	Final Simulation	Reduction Ratio
EVS	04:43	04:46	4:15	10.84%
MI	08:45	08:49	7:58	9.64%
BFA	08:11	08:19	7:42	7.41%
PV	08:40	08:59	8:36	4.27%
AA	06:25	06:11	5:47	6.47%
DOS	05:41	05:44	5:33	3.2%
OTHERS	07:26	07:26	7:03	5.16%

Table 14: Total dataset; 29.38%, 367 out of 1249 investigations contain step C (41 investigations are removed from 1290 total investigations, as is discussed for the closed alerts category). By the simulation, the total average analysis duration decreases from 07:34 to 07:03, or 6.83%.

Investigation Class	Dataset Average	Simulation Average	Involved Proportion	Reduction Percentage
Creating new incidents	17:52	16:31	48.45%	7.55%
Updating incidents	06:41	06:13	12.69%	6.98%
Closing alerts	05:12	04:54	31.9%	5.77%
Total dataset	07:34	07:03	29.38%	6.83%

Table 15: The summary of simulation results for different investigation classes

Investigation Type	Dataset Average	Simulation Average	Reduction Ratio
EVS	20:50	20:36	1.12%
MI	20:31	20:06	2.03%
BFA	20:18	20:06	0.98%
PV	19:46	19:43	0.25%
AA	19:27	19:01	2.23%
DOS	17:35	17:28	0.66%
OTHERS	15:02	14:52	1.11%

Table 16: Class: creating new incidents; 58 out of 194 investigations (29.9%) contain step F. By the simulation, the total average analysis duration decreases from 17:52 to 17:38, or 1.3%

The contacting approaches can be different for each client based on severity of the incident and the client's preference.

A possible improvement for this step is providing information about the required escalation method for an incident in the window of creating and updating incidents in the SOC Console. By correlating related client escalation grid and related incident type, the proper contact approach can be fetched and shown to the analyst as a hint which saves him time to go through different buttons to find out the required information. In this scenario, we observe how frequent this step is beside the average duration analysts spending on it.

We found that 58 out of 194 (29.9%) investigations contain checking escalation grid for the category of creating incidents. Besides, 18 out of 331 (5.44%) and 23 out of 765 (3.0%) investigations include checking escalation grid for the categories of updating and closing incidents respectively. Since the number of involved investigations for the last two investigation classes are not much (as expected), the simulation is only done for the creating new incidents category. Table 16 shows the detailed simulation results for the class of creating new incidents, where the related step is eliminated completely. Our simulation illustrates the average investigation duration is decreased by just 1.3%. MI and AA attack types are mostly checked for the proper escalation method.

Investigation Type	Dataset Average	Average after eliminating investigations	Simulation Average	Reduction Ratio
EVS	04:43	05:18	04:46	10.06%
MI	08:45	09:46	08:48	9.9%
BFA	08:11	08:59	08:18	7.61%
PV	08:40	09:32	08:59	5.77%
AA	06:25	06:40	06:11	7.25%
DOS	05:41	05:59	05:43	4.46%
OTHERS	07:26	07:52	07:25	5.72%

Table 17: Combined effect: 41 investigations are removed resulting in an increase in the average analysis durations shown in the third column, 377 out of 1249 investigations (30.18%) are affected by the combination of two simulation scenarios, and decreases the total average analysis duration from 8:09 to 7:33, or 7.36% beside saving four and a half hours man-hour.

5.2.3 Combination of two possible improvements

By combining the above two improvement options, we assess how averages would be affected. Correlated alerts with assets' vulnerabilities and automated shown escalation grid together are simulated to show the effect on the averages. Table 17 shows the simulation results. Firstly, by removing 41 investigations which were closed immediately after checking assets' vulnerabilities, new analysis averages are shown in the second column indicating on an increase which means removed investigations were part of short investigations. Then by eliminating the steps related to checking assets and escalation grid, simulated average analysis durations are calculated and shown in the fourth column. By employing both automation solutions, the total average analysis duration would be decreased by 7.36%. The most effected attack types are EVS, and MI.

5.3 Case Study II, A Different Alert Dispatching Method

In this case study, the simulation of a different dispatching method of incoming alerts among analysts is assessed.

The number of analysts working in the SOC is different from one work-shift to another work-shift. For week days (Monday to Friday), there are three work-shifts

per day; day-shift from 8:00 to 16:00, evening-shift from 16:00 to 00:00, and night-shift from 00:00 to 8:00. For weekends, there are two 12 hours shift per day. Since during weekend and weekdays evening and night shifts only one analyst is working, these shifts are not considered in the dataset of this simulation. Only investigations of the day-shift for week days with more than two analysts are considered for this simulation.

In Section 5.1, it is mentioned that short investigations (with milliseconds durations) are removed from the dataset. Since the number of short investigations is significant, they affect the average time of analysis enormously. By removing short investigations, we are able to look at the dataset and its statistics in a more accurate way. However, in this case study since we want to simulate dispatching phase of alerts, we need all alerts. By those removed investigations, we would have time gaps between investigations showing that the analysts were free, which was a wrong assumption for this simulation. Consequently, we consider a dataset containing all investigations no matter how long they are. We consider all situations in which analysts are busy.

As is discussed, the alert waiting time is also considered as a metric to show the effect of the simulation. In this regard, having the arrival time of alerts is necessary, however we do not have arrival time in the provided dataset. Poisson process [57] is employed to simulate the arrival time of alerts. Using the Poisson distribution, we generate the desired number of random durations with a specific average in total. Summation of the waiting time with the investigation start time (timestamp) simulates the arrival time of the alert in the queue.

Two parameters are needed to be fed the Poisson process to generate random durations as the waiting time. One parameter is average duration time of the waiting time which alerts stay in the queue to be analyzed. We have the average duration as two hours asked from the SOC experts. Second parameter is the number of alerts which we need to generate time for them. Number of alerts is counted based on the number of investigations which we have for each work shift. For each work-shift, the simulator counts the number of investigations, and considers 120 minutes as the average waiting time. Then by Poisson process, it generates the same number of durations by different values with the average of 120 minutes in total.

The arrival time stamp of an alert is the constant value which is generated by the Poisson process. The start time of investigations changes from the original dataset

to the simulated data. Since investigations are performed by different analysts with different analysis durations, they affect the start time of the next investigation in the work-shift. Consequently, we have different waiting times for the simulated investigations.

In order to dispatch each alert of the queue, the simulator checks the average analysis duration of the available analysts of the work-shift for the related investigation type. Then, the alert is assigned to the most efficient and available analyst of the shift. The available analyst means analysts working in the work-shift, and not being occupied by other investigations. When all analysts are occupied, next investigation would be postponed till one analyst becomes free.

Once an investigation is assigned to an analyst by the simulator, activity logs of the simulated investigation are simulated from the real investigation logs. The simulator considers the same investigation path with the same number of activity logs for the simulated investigation, where duration of the different actions (activity logs) in that investigation are calculated based on the assigned analyst's average analysis duration. In other words, the duration of the simulated investigation is based on the average analysis duration of the selected analyst for the simulation.

We know based on the different results of an investigation (e.g., closing the alert or creating new incident), the average of the same actions are different. For simulating each action duration of one investigation, the simulator calculates the average duration of the action based on the investigation result. For example, the average duration of expanding alert content for investigations resulting in creating new incidents is 4 minutes where it is 30 seconds for investigations resulting in closing alerts. Since the simulator knows the result of each alert investigation from the beginning, it simulates the duration of each action based on the average duration of the related alert investigation class.

Totally, 33 day work-shifts are considered for the simulation with the minimally two analysts working per work-shift. For 17 work-shifts, two analysts are working per work-shift. For 15 work-shifts, three analysts and for one work-shift four analysts are working in the SOC in parallel. These 33 day work-shifts are selected with the entire dataset. Table 18 shows statistics on the selected dataset for 33 day work-shifts and simulation results. Each column represents one investigation class, such as new incidents which first sub-column is dataset average analysis time, second one is

simulated average analysis time, and third sub-column is the reduction ratio.

By comparing Table 9 with Table 18, we can see the proportion of the closing alerts category increases from 59.3% to 94.12%, since all investigations of the work-shift are considered regardless of their duration.

The simulation result in Table 18 represents the effect of the simulated dispatching approach on the total average analysis time and waiting time of different investigation classes and the total dataset. Total average analysis duration and waiting time of the total dataset are improved by 4.42% , and 2.18% respectively.

Investigation classes	**New Incidents**			**Updated Incidents**			**Closed Alerts**			**Total Dataset**		
# of log entries	1059			731			11181			12971		
# of investigations	75			163			3807			4045		
Proportion	1.85%			4.03%			94.12%			100%		
Parameters	**Real**	**Simu**	**%**	**Real**	**Simu**	**%**	**Real**	**Simu**	**%**	**Real**	**Simu**	**%**
Total average analysis duration	18:13	16:54	7.23%	05:19	04:16	19.75%	01:25	01:25	0%	01:53	01:48	4.42%
Waiting time	120:16	116:02	3.52%	120:00	116:02	3.3%	120:06	117:34	2.11%	120:06	117:29	2.18%

Table 18: The alert dispatching method dataset statistics for 33-day work-shifts, the simulation results and the reduction ratios.

Table 19 shows the simulation results for the different investigation types. For each column, first sub-column is the number of investigations related to the investigation type, second sub-column is the real dataset average analysis duration, third is the simulated average analysis duration, and the fourth is the reduction ratio. In some columns related to the reduction ratio, "+" is shown (e.g., +2.77%) implying the increase percentage rather the decrease.

By looking at the results, we can say when the reduction ratio is high, it is infer-able that either the dataset durations for those specific investigations are so different from the average, or analysts efficiency are so different from each other for that investigation type. For instance, we can see for the updating incidents class related to the PV investigation type, simulated average time is decreased by 98.94%; first point is that the number of investigations are few (#5), and the second point is that by the simulation we interchange the real investigation duration (which can be so far from the average) with the average analysis duration of the best analyst. Moreover, for investigations belonging to the updating incidents class, if they are about contacting the client, they can just contain few actions with long durations,

and we simulate them with the average duration of that action which is shorter.

Looking at the reduction ratios of the total dataset, the most affected investigation types are EVS, and PV showing different efficiency of analysts. Since by dispatching the alerts to different analysts, the average analysis durations are decreased significantly. However DOS and OTHERS investigations are increased in their average analysis duration by +5.71% and +1.98% respectively.

Inv. Types	New Incidents				Updated Incidents				Closed Alerts				Total Dataset			
-	#	Real	Simu	%	#	Real	Simu	%	#	Real	Simu	%	#	Real	Simu	%
EVS	3	15:56	13:05	17.89	7	03:56	00:25	89.41	92	01:04	00:59	7.81	102	01:42	01:18	23.53
MI	21	21:50	22:54	+4.88	38	06:47	05:42	15.97	300	02:21	01:58	16.31	359	03:58	03:35	9.66
BFA	12	22:16	22:52	+2.77	8	02:59	01:59	33.52	493	01:21	01:05	19.75	513	01:52	01:37	13.39
PV	7	18:34	11:28	38.24	5	06:18	00:04	98.94	222	01:05	01:04	1.54	234	01:43	01:21	21.36
AA	2	16:23	13:44	16.17	2	01:10	01:08	2.86	58	01:37	01:27	10.31	62	02:04	01:50	11.29
DOS	-	-	-	-	3	05:55	01:16	78.59	256	01:07	01:14	+10.45	259	01:10	01:14	+5.71
OTHERS	30	14:40	12:13	16.7	100	05:04	04:33	10.2	2386	01:23	01:28	+6.02	2516	01:41	01:43	+1.98

Table 19: The alert dispatching method dataset statistics for different investigation types, the simulation results and the reduction ratios (the sign "+" implies an increase than the reduction)

The proposed approach is supposed to have a better performance result when the number of working analysts in the work-shift is more. Since we have more analysts working per work-shift, we have more diversity in analysts' skills. Diversity implies the analysts with different expertise, as each analyst could be more skilled on specific alert type and analyze that type faster. In the dataset of this simulation, we can see that we have choices of two, three and four analysts working per work-shift.

We divide the dataset to three distinct datasets based on the number of analysts working in work-shifts (2, 3, and 4 analysts working per workshoft). Separate three alert dispatching simulations are done for the three datasets to show the effect of the employed simulation approach on work-shifts with different number of analysts.

Tables 20 and 21 represent the simulation results for the work-shifts with two and three analysts respectively. The total average analysis duration is decreased 3.88% and 0.85% for the work-shifts with two and three analysts respectively. Moreover, the waiting time is decreased 2.23% for the work-shifts with two analysts, and 2.15% for the work-shifts with three analysts.

An observation from the simulation results of the work-shifts with two and three analysts shows one important factor to improve the efficiency by the employed approach is about combination of the selected analysts' expertise for one work-shift. If

analysts with different expertise are chosen to work in the same work-shift, it would increase the efficiency of the SOC by the proposed dispatching model more. For instance, if we have two analysts that each one has a good performance result for a different range of investigation types, the performance improvement is more than the situation that we have three analysts with the same expertise level.

Parameters	**New Incidents**			**Updated Incidents**			**Closed Alerts**			**Total Dataset**		
Number of investigations	24			57			1535			1616		
Parameters	**Real**	**Simu**	**%**	**Real**	**Simu**	**%**	**Real**	**Simu**	**%**	**Real**	**Simu**	**%**
Total average analysis duration	15:17	15:31	+1.53%	05:11	03:37	30.22%	01:23	1:20	3.61%	01:43	01:39	3.88%
Waiting time	120:01	114:59	4.19%	120:27	115:40	3.97 %	120:01	117:54	1.76%	120:28	117:47	2.23%

Table 20: The alert dispatching method dataset statistics for 17-day work-shifts with 2 analysts, the simulation results and the reduction ratios (the sign "+" implies an increase than the reduction)

Parameters	**New Incidents**			**Updated Incidents**			**Closed Alerts**			**Total Dataset**		
Number of investigations	44			100			2125			2269		
Parameters	**Real**	**Simu**	**%**	**Real**	**Simu**	**%**	**Real**	**Simu**	**%**	**Real**	**Simu**	**%**
Total average analysis duration	19:31	17:49	8.71%	05:22	04:49	10.25%	01:26	1:28	+2.32%	01:57	01:56	0.85%
Waiting time	120:25	116:02	3.64%	119:48	116:05	3.1%	119:58	117:29	2.07%	119:58	117:23	2.15%

Table 21: The alert dispatching method dataset statistics for 15-day work-shifts with 3 analysts, the simulation results and the reduction ratios (the sign "+" implies an increase than the reduction)

Table 22 is the simulation results for one work-shift with 4 analysts. As is shown, the total average analysis duration and the waiting time are decreased by 32.21% and 6.58% respectively. This experiment shows also the employed approach improves the SOC performance where the number of analysts is increased and the analysts working in the same work-shift have different expertise regarding different investigation types.

Parameters	**New Incidents**			**Updated Incidents**			**Closed Alerts**			**Total Dataset**		
Number of investigations	7			6			147			160		
Parameters	**Real**	**Simu**	**%**	**Real**	**Simu**	**%**	**Real**	**Simu**	**%**	**Real**	**Simu**	**%**
Total average analysis duration	19:55	11:34	41.92%	05:51	01:37	72.36%	01:31	1:13	19.78%	02:29	01:41	32.21%
Waiting time	120:06	119:33	0.45%	118:58	118:37	0.29%	119:54	115:29	3.68%	119:52	115:47	3.41%

Table 22: The alert dispatching method dataset statistics for 1-day work-shift with 4 analysts, the simulation results and the reduction ratios

5.4 Case Study III, The Feedback Module

Our study on the analysts' activity logs shows investigations' average durations for different attack types are usually different from each other. Different analysts' average durations are usually also different even for the same attack type implying the different efficiency. The different efficiency can be due to the analysts' different levels of knowledge regarding analysis approaches or familarity about clients' environments.

As is discussed in Sections 4.1.4 and 4.2.4, one feature of the feedback module is showing previous investigation logs whose alert type is similar with which the analyst is currently analyzing. In this case study, we evaluate how such a feedback module would affect the analysts' performance.

We consider one analyst as the senior analyst who has better efficiency than others. The other analysts who may benefit from the senior analyst's knowledge and experience are called junior analysts. We model the trend of investigation durations of the senior analyst, and partially apply the model to future investigation durations of the junior analysts, assuming that, since the junior analysts can see what the senior analyst performed through the feedback module, they can potentially improve their efficiency.

Linear regression analysis [58] is employed to model the senior analyst's investigation durations. By the regression analysis, the mathematical function representing investigation durations is extracted from the dataset. The duration of each investigation completed by the senior analyst is considered as a data point for that analyst. Different data points of the analyst is ordered by time chronologically as they are performed in different days. By extrapolating the established model, we can predict future investigation durations of the senior analyst based on his historical data.

It is discussed in Section 2, several alert types (EventID) indicating the same attack type are grouped together to form an investigation type. EventID can be the signature of an exploited vulnerability of an application belonging to an attack type. Since investigation types are general categories containing different kinds of EventIDs, the feedback module will show the investigations of the same EventID to the analysts by the feedback module.

In order to simulate a scenario, we choose an EventID with the highest number of investigations in the dataset, since we would like to have sufficient investigation samples (data points) to more accurately model the analyst's investigation durations

AnalystID	Average	Variance	The Number of Investigations
1	16:05	-	1
2	01:17	-	1
3	21:34	717.72	2
4	05:46	100.57	5
5	07:29	35.45	6
6	03:16	7.85	45

Table 23: Different analysts' statistics; the average investigation duration, the variance of investigations durations, and the number of investigations related to EventID 101010 which is a custom EventID for verifying DNS queries.

and assess the effect of our feedback module. The most common EventID is 101010 which has the largest number of investigations compared to other EventIDs. This EventID is a custom EventID corresponding to sensors in verifying DNS queries. Each DNS request is verified by comparing it to a blacklist in the SOC automatically. If the requested domain is in the blacklist, the sensor raises an alert.

After determining the EventID, we must also choose a senior analyst to model his investigation durations. Some parameters are considered in choosing the senior analyst. The senior analyst is the one with the lower average investigation duration, the lower data points variance, and the highest number of investigations related to the selected EventID. In Table 23, we can see different analysts' statistics regarding the mentioned parameters. From the table, we can see the most suitable senior analyst is the AnalystID 6 who has the lower average investigation duration, the lower variance, and the higher number of conducted investigations compared to the other analysts.

We note that having more investigations by the AnalystID 6 in the dataset does not mean other analysts pay less attention to EventID 101010. Since we have some gap (missing days) in the dataset, it is likely that we may have missed some analysts' work-shifts, or AnalystID 6 simply has more work-shifts during the dataset's particular period.

A realistic assumption here is that the junior analyst will partially benefit from the knowledge of the senior analyst by observing the latter's investigation details but such benefit is not likely sufficient to enable the former to perform those analyses with exactly the same efficiency as the latter. In other words, the knowledge transfer is partial instead of complete. Accordingly, we assign a percentage range by which a junior analyst can improve the efficiency of his/her investigations of the same type

after observing the senior analyst's approach and results. It is assumed in this case study that a junior analyst can gain 10% to 60% of the senior analyst's knowledge to improve his investigations. To obtain a more accurate estimation of such a range from real data is a future work.

The average investigation duration of a junior analyst for EventID 101010 is considered as the default value for his future investigation durations in our simulation. This default value can be improved by learning from the senior analyst. For example, when we assume the junior analyst gains knowledge by 10%, his average investigation duration is calculated as the summation of 90% of the estimation proportion ($90\% * Junior\_Investigation\_Average$) and 10% of the senior analyst's investigation duration (as predicted by the model) ($10\% * Model\_Investigation\_Duration$). As another example, when we assume the junior analyst gains knowledge by 60%, his average investigation duration is equal to 40% of his own duration plus 60% of the senior analyst's duration.

The reason behind considering the average investigation duration of the junior analyst instead of modeling his investigations durations (as well as the senior analyst) is that we do not have sufficient data points to establish the model for the junior analysts. Since the dataset does not provide enough investigations for any single EventID, the average investigation duration is considered for the junior analysts.

For the regression analysis, the X axis represents time series ordering investigations chronologically and the Y axis is the investigation duration for the data points. In practice investigations might be performed on the same day or across different days in the period of the dataset, but the time distance between data points considered in this case study is limited to one day in this simulation. We aim to obtain the main trend of the investigation durations in chronological order as either an increase or decrease in the average investigation durations of the senior analyst during the dataset period.

There turn out to be a lot of fluctuations for the 45 data points representing investigation durations for the AnalystID 6. To smooth the curve, every five adjacent investigation durations are averaged and represents one data point. In the end, we obtain a model of the senior analyst's investigation durations as the exponential equation shown below.

$$y = 2.8352e^{-0.005x} \tag{2}$$

In Table 24, some of the data points are shown. The first 10 data points are averaged investigation durations from 45 investigations of the dataset for the senior analyst, and the next 10 data points are the extrapolation of the model. The average duration of the dataset data points is 3:04, where the average duration of the extrapolated data points is 2:37 showing the decreasing trend of the senior analyst's model, which indicates that the analyst's efficiency for this type of investigations slowly improves over time.

X; Time Series	**Y; Investigation Duration**	**X; Future Time Series**	**Y; Extrapolated Investigation Duration**
1	2:11	11	2:41
2	2:20	12	2:40
3	2:11	13	2:40
4	5:57	14	2:38
5	3:31	15	2:38
6	2:32	16	2:37
7	1:59	17	2:36
8	2:54	18	2:35
9	5:46	19	2:35
10	1:16	20	2:34

Table 24: First 10 investigation durations represent data points from the dataset for the senior AnalystID 6 and EventID 101010, and the next 10 investigation durations are extrapolated under the model.

As is discussed, in order to estimate the junior analyst's efficiency, a percentage range of gaining knowledge is considered from 10% to 60%. We simulate 10 future investigation durations for the junior analysts by combining their own investigation average duration and the effect of the senior analysts knowledge using the percentage.

Table 25 shows the simulation results, where the junior analyst is AnalystID 5 with the default investigation average duration of 7:29. Estimation results show that, if the junior AnalystID 5 gains 10% of the senior analyst's knowledge through the feedback module, the average investigation duration will change from 7:29 to 6:59, decreased by 6.68%. If he/she gains 60% of the senior analyst's knowledge, his/her average investigation duration will change from 7:29 to 4:33, decreased by 39.2%.

Table 26 shows the simulation results where the junior analyst is AnalystID 4 with the default average of 5:46. Simulation results show that, if the junior AnalystID 4

Simulated Time Series	Junior's average investigation duration*90%	Model's investigation duration*10%	Simulated investigation duration
1	6:44	0:16	7:00
2	6:44	0:16	7:00
3	6:44	0:16	7:00
4	6:44	0:15	6:59
5	6:44	0:15	6:59
6	6:44	0:15	6:59
7	6:44	0:15	6:59
8	6:44	0:15	6:59
9	6:44	0:15	6:59
10	6:44	0:15	6:59
Simulated Time Series	**Junior's average investigation duration*40%**	**Model's investigation duration*60%**	**Simulated investigation duration**
1	2:59	1:37	4:36
2	2:59	1:36	4:35
3	2:59	1:36	4:35
4	2:59	1:35	4:34
5	2:59	1:35	4:34
6	2:59	1:34	4:33
7	2:59	1:34	4:33
8	2:59	1:33	4:32
9	2:59	1:33	4:32
10	2:59	1:33	4:32

Table 25: The simulation results of the feedback module's impact for the junior AnalystID 5 with the default average investigation duration of 7:29 for the EventID 101010. The first part simulates the junior's efficiency for 10 future investigations by considering the knowledge transfer percentage as 10%, and the second part simulates the junior's efficiency for 10 future investigations by considering the knowledge transfer percentage as 60%.

gains 10% of the senior analyst's knowledge, the average investigation duration will change from 5:46 to 5:26, decreased by 5.78%. If he/she gains 60% of the senior analyst's knowledge, his/her average investigation duration will change from 5:46 to 3:53, decreased by 32.66%.

In summary, in this case study, the impact of showing previous investigations to the analysts, which is one of the important features of the feedback module, is assessed based on some assumptions. The AnalystID 6 is considered as a senior analyst, and AnalystsIDs 5 and 4 are juniors. Based on the simulation results, if the junior analysts gain 10% to 60% of the professional analyst's knowledge; the efficiency of AnalystID 5 can be improved by 6.68% to 39.2%, and the efficiency of AnalystID 4 can be improved by 5.78% to 32.66%. Those results clearly demonstrate

Simulated Time Series	Junior's average investigation duration*90%	Model's investigation duration*10%	Simulated investigation duration
1	5:11	0:16	5:27
2	5:11	0:16	5:27
3	5:11	0:16	5:27
4	5:11	0:16	5:27
5	5:11	0:16	5:27
6	5:11	0:16	5:27
7	5:11	0:16	5:27
8	5:11	0:15	5:26
9	5:11	0:15	5:26
10	5:11	0:15	5:26

Simulated Time Series	Junior's average investigation duration*40%	Model's investigation duration*60%	Simulated investigation duration
1	2:18	1:37	3:55
2	2:18	1:36	3:54
3	2:18	1:36	3:54
4	2:18	1:35	3:53
5	2:18	1:35	3:53
6	2:18	1:34	3:52
7	2:18	1:34	3:52
8	2:18	1:33	3:51
9	2:18	1:33	3:51
10	2:18	1:32	3:50

Table 26: The simulation results of the feedback module's impact for the junior AnalystID 4 with the default average investigation duration of 5:46 for the EventID 101010. The first part simulates the junior's efficiency for future 10 investigations by considering the knowledge transfer percentage as 10%, and the second part simulates the junior's efficiency for future 10 investigations by considering the knowledge transfer percentage as 60%.

the potential benefit of the feedback module of our tool. The summary of results is shown in Table 27.

Junior AnalystID	**Default Investigation Average Duration**	**Estimation 10% - 60%**		**Reduction 10% - 60%**	
5	7:29	6:59	4:33	6.68%	39.2%
4	5:46	5:26	3:53	5.78%	32.66%

Table 27: The summary of results

Chapter 6

Related Work

Related work is categorized into three different groups. Section 6.1 reviews existing works regarding Managed Security Services (MSS), Managed Security Monitoring (MSM), Network Security Monitoring (NSM), and Security Operation Center (SOC). Those range from preliminary works on this emerging service to its different aspects, such as design, classification, information sharing, etc.

In section 6.2, alert correlation techniques are reviewed, which draws a lot of attention in academia and industry. It plays an important role to generate accurate suspicious events for the security services. Moreover, alert correlation is a tangible aspect of the SOC that can be improved. It has also led to other research areas either military or non-military intelligent systems.

Section 6.3 reviews studies about Call Centers (CC), since SOC and CC are similar regarding their performance evaluation. Existing works in this domain focus on different queueing models and methods to solve the problem of staffing and scheduling. It also provides background knowledge for the alert dispatching simulation scenario.

6.1 MSS, MSM, NSM, SOC

Managed Security Monitoring (MSM) and *Network Security Monitoring* (NSM) were the terms for a new generation of *Managed Security Services* (MSS). At the present time, those three terms point to the same concept.

In the work of Allen et al. [1], different security services are reviewed, from

network boundary protection services, vulnerability assessment and penetration testing, anti-virus and content filtering services, information security risk assessments, data archiving and restoration, on-site consulting to security monitoring and incident management. Different guidelines about MSS request proposal, evaluating an MSS proposal, MSS service level agreement, and transitioning to MSS are discussed in details.

Managed Security Monitoring (MSM) is introduced as a network security solution of this century by Schneier in the year 2000 [59]. He claims simply installing firewalls and other security-related tools cannot tackle intrusions. MSM is compared with Managed Security Services (MSS) in 2001 [60]. Where MSS is more about providing updated firewalls, IDSs, and other security products for companies, MSM is monitoring a client's network to recognize and respond to threats simultaneously in real time.

NSM is defined for the first time by Bejtlich in [2] as "the collection, analysis, and escalation of indications and warnings to detect and respond to intrusions." The book describes different terms and security processes beside deployment considerations. It also explains a case study as an "intrusion reference model". Moreover, the different types of data needed to be collected from network are discussed beside useful open source tools for NSM, such as traffic modifying tools. The book provides technical best practices regarding event handling and incident response processes. It also allows analysts and supervisors to learn weapons and tactics, telecommunication, system administration, scripting and programming, and management and policy. 16 different case studies are provided to show analysts how they can employ principles to intrusion scenarios. And it discusses attacker's perspective by describing attacking approaches to products, processes, and people.

Implementing Network Security Monitoring (NSM) for cloud services is discussed by Shin and Gu [5]. The CloudWatcher framework is proposed to direct network packets of the cloud passing through defined network security monitoring points. Another work in NSM area focuses on routing network traffic to monitoring devices by introducing a system called OpenSAFE [8]. OpenSAFE provides configurable network traffic routing by employing OpenFlow-supported devices [61] to preserve high line rates performance, and introduces ALARMS as a flow specification language to ease the management of network monitoring devices.

Main concepts and components of a SOC as a heart of NSM are discussed in [3] by Renaud Bidou for the first time in 2005. Different modules' functionalities, from event generating, collecting, and storing to analysis approaches and related response methods, are covered to explain the process of building a SOC, where integrating all modules is considered as a challenge. Other work [4] on designing a SOC proposes a solution utilizing recognition mechanism of immune system to detect intrusions. The SOC is designed in a way that it detects the self from non-self. The protected network is preserved among immune cells.

Hu and Xie [7] present a novel design for a SOC by applying Dempster-Shafer Theory (DST) on the basic SOC model proposed by Renaud Bidou [3]. By using multi sensor data fusion techniques, they claim the presented approach reduces the rate of both false positive and false negative alerts. The authors believe integrating security services from a SOC is more beneficial since SOCs correlate different sources' events to detect threats with higher accuracy compared to single sensor events.

A recent work [6] on designing SOCs by Li et al. proposes a hierarchical mobile-agent-based SOC to avoid one fixed location for alert correlating and improve computational efficiency. Event collectors and correlators are distributed in the monitored network to prevent single point of failure attacks resulting in downing of the service.

Since independent SOCs usually are reluctant to share security information about new incidents, a study [62] introduces a mechanism for trusted sharing of security incidents information among SOCs by minimum information sharing. Information, such as time of occurrence, origin of attack, consequence, severity, and path of attack can be shared in Security Incident Data Exchang (SIDEx) format by not revealing sensitive information.

Ganame et al. [9] develops a SOC providing a global view of the monitored network in a graphical way to help analysts detecting attacks. Different reports are made from the aggregated alerts from different sources. Performance evaluation of the proposed system is evaluated and compared with simple IDS. This is different from our work since we evaluate analysis process performance performed by human security analysts.

Another work [10] proposes a classification model to assess SOC services. By the proposed framework, either SOC clients or owners can measure the maturity of SOC processes regarding certain aspects, such as log collection, log retention and archival,

log analysis, monitoring of security environments for security events, diversity of devices integrated, event correlation and workflow, incident management, reaction to threats, threat identification, and reporting.

Sundaramurthy et al. [11] study three different operational SOCs to understand their functionalities in details by the anthropological approach. Three different students with computer science background are assigned as analysts in each SOC, and trained to observe SOC analysts' tasks. List of different aspects of each SOC is detailed in this work, such as team structure, training methodology for new analysts, operational workflows, employed tools and software, work-shifts scheduling, SOC efficiency metrics. They mention one performance metric considered in one of the operational SOCs is the number of analyzed incidents per analyst daily. It is also described that counting the number of incidents solely is not a good metric, if the time duration of analysis is not considered. Consequently, efforts on difficult investigations taking more time to analyze reduce the total number of analyzed incidents. As a result, analysts are not motivated to analyze deeply, since it results in showing lower productivity for them in a managers' point of view. In a university SOC assessed by the same work [11], a ticketing system dispatching alert tickets to analysts, provides a performance metric as time spent on each ticket. To the best of our knowledge, there is no work presenting a clear approach to model SOC analysis process by human analysts, and evaluating SOC performance by different metrics in the literature.

Michail studies SOC from business perspective in [12], and answers different questions about SOC, which vary from how SOCs can be different from each other with the same goals to how being well-established helps a SOC to improve its performance. The author claims comparing SOCs is not a valid question since they employ different security solutions, organizational structures, and services. However the same high level goal is shared among them to protect against any cyber attack. The author also rejects the existence of any scientific literature supporting the idea that a well-established SOC results in a better performance.

6.2 Alert Correlation Techniques

Every raw alert of an IDS can show a single attack step of an intrusion, however usually it is a part of failed attack attempt or normal traffic. Consequently, SOCs

mostly receive a huge amount of raw alerts from deployed sensors. Considering such this massive load of raw alerts, two SOC-related studies [3], and [7] claim correlating gathered events from different sensors plays an important role to generate accurate suspicious events, and reduce the rate of false positive. There are plenty of works for alert correlation, since it is considered as the most improvable and effective part of a SOC regarding performance improvement. Data fusion techniques have been employed earlier in other fields either military or non-military intelligent systems, such as [63],[64].

In a recent survey on alert correlation by Leau et al. [65], correlation approaches are classified to four categories, correlation methods based on similar semantics in alerts description, predefined attack scenarios, preconditions and post conditions of attacks, and data mining. First method targets different alerts' attributes, such as Source IP address, destination port number, etc. and correlates alerts by their similarity score. Second method correlates alerts based on predefined attack scenarios. Learning phase of this method can be user defined signatures from attack scenarios or automatically extracted from training dataset. Third method correlates alerts based on each attack scenario prerequisites and consequences. For the last two methods, complexity of the design, and identifying new attacks are part of mentioned issues. Fourth category is about methods employing data mining techniques to identify attack patterns and correlations.

Elshoush and Osman [14] devise a correlation framework combining 10 components, called normalization, pre-processing, prioritization, alert verification, alert fusion, focus recognition, uncorrelated removal, multi-step correlation, intention recognition, and impact analysis. This approach aims to reduce FP alerts in initial phases by removing unrelated alerts from fused alerts. Then, it attempts to employ correlation approaches, such as attack scenarios, to have correlated alerts.

Wang et al. [13] propose a memory efficient correlation approach by employing attack graphs which are predefined attack scenarios. By introducing a novel approach *Queue Graph* (QG), nested loop based correlation is solved, and it is possible to match alerts to related nodes of the attack graph. Zali et al. [15] presents a correlation approach by pre-defining simple relations among minor attacks to identify attack scenarios in real time. By benefiting from TVA model[13], they model relations among minor attacks as pre and post conditions of each attack pattern in a new

way. Then, the algorithm extracts attack scenarios in real time with polynomial time complexity.

A correlation method is introduced by Zhu and Ghorbani [66] to recognize different attack scenarios without experts' knowledge background. Multilayer perceptron (MLP) and support vector machine (SVM) are employed as neutral network approaches to evaluate correlation probability of each pair of alerts. Correlation probability estimation results are stored in Alert Correlation Matrix (ACM), and ACM will be used to extract high level attack scenarios.

Ramaki et al. [16] presents a correlation framework to detect multi-step attack scenarios in real time as an Early Warning System (EWS). An EWS aims to identify hidden risky behaviour of a system which might expose the system to threats[67]. Statistical and stream mining (sequence analysis) techniques are employed to design the correlation scheme. The correlation framework works in two modes, offline and online. In offline mode, aggregated alert types, called hyper alerts, are checked by an episode mining algorithm to find aggregation options for hyper alerts. Then, in learning phase, it learns multi-step attack scenarios while it is constructing an off-line attack tree from the processed dataset. In online mode, it constructs an attack tree in real time with the learned knowledge.

6.3 Call Centers And Queuing Models

Performance evaluation of a SOC is similar to Call centers (CC) modeling. In both SOC and CC, humans serve different clients with different service requests in a queue. In the SOC, security analysts are the servers and incoming alerts are considered as different service requests, whereas in the CC, operators respond to different calls. In both cases, incoming service requests are coming in a queue and the service needs to meet certain service level agreement (SLA) specified between clients and service company.

Brown et al. [68] describe queueing-theoretic models for service systems where the number of human servers, arrival rate of requests, serving time, and authorized waiting time of requests in a queue are inputs of the queueing model, and outputs of the model could be for example distribution of waiting time for service requests and the fraction of requests not being handled within the authorized time. They present

a basic common queueing model M/M/N system or Erlang-C [69] which considers arrival rate based on Poisson process, exponentially distributed service time, and servers and clients act independently. Limitations of the basic model include not considering time-dependant parameters, clients' or their requests heterogeneity, and servers' skill levels.

Green et al. [17] discuss different methods of queuing-theory for setting a service system to serve clients whose request-pattern is predictable during a day (how much demanding in which periods). Different aspects of service systems are discussed, such as setting the system capacity (overall size of the workforce), single-skilled vs. multi-skilled human servers, and queueing models which consider demanding periods beside service time.

Excoffier et al. [18] try to solve staffing problem with a solution that determines the minimum number of servers that could conform to SLA. The proposed solution is based on linear approximation, and considered arrival rate as random. the other similar work [19] presents a robust solution guaranteeing that the proposed shift schedule with the minimum number of servers can conform to SLA. It computes the solution by considering the probability distributions of uncertain parameters.

Other recent works on queueing models of call centers [20][70] mostly focus on considering a new input parameter for queueing models called impatient customers. This parameter indicates those customers abandoning the queue before being served.

Chapter 7

Conclusion and Future Work

In this chapter, we conclude our work by summarizing the contributions and discussing the directions of future work.

7.1 Conclusion

In this thesis, by modeling the main workflow of an operational Security Operation Center (SOC), a system for improving the SOC's performance has been designed consisting of four modules; monitoring, measuring, simulation, and feedback. The first three modules empower the SOC managers to evaluate the current SOC performance, and assess potential improvement options through simulations. The feedback module enables knowledge transfer among SOC analysts in their ongoing workflows to improve their performance.

By deploying a logging component inside the main SOC console, analysts' activity logs from a real production SOC are collected from June to August 2015 for 57 days in a dataset for evaluating our system. Three case studies have been conducted based on the dataset to study the designed system's effectiveness, namely, modifying the duration of steps, a different alert dispatching method, and the feedback module's impact. In the case study of modifying the duration of steps, we provide two improvement scenarios for the SOC workflow. The simulation result of the combined scenarios demonstrates a performance improvement of 7.36%. In the case study of a different alert dispatching method, the results indicate that one important factor to improve the efficiency by the employed approach is about combination of the selected

analysts' expertise for one work-shift. If analysts with different expertise are chosen to work in the same work-shift, it would increase the efficiency of the SOC by the proposed dispatching model more. The simulation results for the 33-day work-shifts state a 4.42% improvement in the average investigation duration and 2.18% in the alerts waiting time. In the case study of the feedback module's impact, knowledge transfer rates (10% to 60%) are considered for two junior analysts gaining knowledge from a senior analyst regarding a specific EventID. The average performance improvement for the two junior analysts ranges from 6.23% to 35.93% depending on their knowledge transfer rate. In order to asses the improvement results, it should be considered that the all improvement percentages point at the time duration reduction in one single investigation.

7.2 Future Work

We address our future work by two directions. We will employ data mining for automated analysis of investigations logs. The examples are employing classification and association techniques. By classification, we can label analysts performance regarding their performance evaluation. Association rules can be employed to find frequent patterns, such as if one analyst has a certain habit in investigation, or if one EventID is closed all the time for a client, it can be suggested to the SOC to filter it.

We will also apply and simulate some queuing theories on the dataset to show how different models affect the overall performance of the SOC differently to find the optimum approach.

Moreover, we would like to extend case studies on a larger scale dataset.

Bibliography

[1] J. Allen, D. Gabbard, C. May, E. Hayes, and C. Sledge, "Outsourcing managed security services," tech. rep., DTIC Document, 2003.

[2] R. Bejtlich, *The Tao of network security monitoring: beyond intrusion detection*. Pearson Education, 2004.

[3] R. Bidou, "Security operation center concepts & implementation," *available at http://www. iv2-technologies. com*, 2005.

[4] Y. Niu, Q. Zhang, Q. Zheng, and H. Peng, "Security operation center based on immune system," in *International Conference on Computational Intelligence and Security Workshops, CISW'007.*, 2007.

[5] S. Shin and G. Gu, "Cloudwatcher: Network security monitoring using openflow in dynamic cloud networks," in *20th IEEE International Conference on Network Protocols (ICNP)*, 2012.

[6] J. S. Li, C. J. Hsieh, and H. Y. Lin, "A hierarchical mobile-agent-based security operation center," *International Journal of Communication Systems*, vol. 26, no. 12, pp. 1503–1519, 2013.

[7] X. Hu and C. Xie, "Security operation center design based on ds evidence theory," in *Proceedings of the IEEE International Conference on Mechatronics and Automation*, 2006.

[8] J. R. Ballard, I. Rae, and A. Akella, "Extensible and scalable network monitoring using opensafe," *Proceeding of INM/WREN*, 2010.

[9] A. K. Ganame, J. Bourgeois, R. Bidou, and F. Spies, "Evaluation of the intrusion detection capabilities and performance of a security operation center.," in *SECRYPT*, pp. 48–55, 2006.

[10] P. Jacobs, A. Arnab, and B. Irwin, "Classification of security operation centers," in *Information Security for South Africa, 2013*, pp. 1–7, IEEE, 2013.

[11] S. C. Sundaramurthy, J. Case, T. Truong, L. Zomlot, and M. Hoffmann, "A tale of three security operation centers," in *Proceedings of the ACM Workshop on Security Information Workers*, 2014.

[12] A. Michail, "Security operations centers: A business perspective," Master's thesis, Utrecht University, 2015.

[13] L. Wang, A. Liu, and S. Jajodia, "Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts," *Computer communications*, vol. 29, no. 15, pp. 2917–2933, 2006.

[14] H. T. Elshoush and I. M. Osman, "An improved framework for intrusion alert correlation," in *Proceedings of the World Congress on Engineering*, vol. 1, pp. 1–6, 2012.

[15] Z. Zali, M. R. Hashemi, and H. Saidi, "Real-time intrusion detection alert correlation and attack scenario extraction based on the prerequisite-consequence approach," *The ISC International Journal of Information Security*, vol. 4, no. 2, 2013.

[16] A. A. Ramaki, M. Amini, and R. E. Atani, "Rteca: Real time episode correlation algorithm for multi-step attack scenarios detection," *Computers & Security*, vol. 49, pp. 206–219, 2015.

[17] L. V. Green, P. J. Kolesar, and W. Whitt, "Coping with time-varying demand when setting staffing requirements for a service system," *Production and Operations Management*, vol. 16, no. 1, pp. 13–39, 2007.

[18] M. Excoffier, C. Gicquel, O. Jouini, and A. Lisser, "Comparison of stochastic programming approaches for staffing and scheduling call centers with uncertain

demand forecasts," in *Operations Research and Enterprise Systems*, pp. 140–156, 2014.

[19] S. Mattia, F. Rossi, M. Servilio, and S. Smriglio, "Robust shift scheduling in call centers," in *Combinatorial Optimization*, pp. 336–346, 2014.

[20] H. Takagi and Y. Taguchi, "Analysis of a queueing model for a call center with impatient customers and after-call work," *International Journal of Pure and Applied Mathematics*, vol. 90, no. 2, pp. 205–237, 2014.

[21] K. Apostol, *Brute-force Attack*. SaluPress, 2012.

[22] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, *Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security)*. Prentice Hall PTR, 2004.

[23] M. Vieira, N. Antunes, and H. Madeira, "Using web security scanners to detect vulnerabilities in web services," in *IEEE/IFIP International Conference on Dependable Systems & Networks, DSN'09*, 2009.

[24] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation.," in *USENIX Security*, 2007.

[25] "Snort default alert classifications." http://manual.snort.org/node318.html.

[26] "Snort sid 31304 description." https://www.snort.org/rule_docs/1-31304.

[27] "Snort sid 2522 description." https://www.snort.org/rule_docs/1-2522.

[28] "Snort sid 632 description." https://www.snort.org/rule_docs/1-632.

[29] "Snort sid 35462 description." https://www.snort.org/rule_docs/1-35462.

[30] "Snort sid 18180 description." https://www.snort.org/rule_docs/3-18180.

[31] "Snort sid 34463 description." https://www.snort.org/rule_docs/1-34463.

[32] "Snort sid 12426 description." https://www.snort.org/rule_docs/1-12426.

[33] J. L. Peterson, *Petri net theory and the modeling of systems*, vol. 132. Prentice-hall Englewood Cliffs (NJ), 1981.

[34] M. Fowler, *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004.

[35] L. A. Schultheiss and E. M. Heiliger, "Techniques of flow-charting," *Clinic on Library Applications of Data Processing*, 1963.

[36] D. Harel and A. Pnueli, "Logics and models of concurrent systems," ch. On the Development of Reactive Systems, pp. 477–498, 1985.

[37] R. J. Wieringa, *Design methods for reactive systems: Yourdon, statemate, and the UML*. Elsevier, 2003.

[38] R. Eshuis and R. Wieringa, "Comparing petri net and activity diagram variants for workflow modelling–a quest for reactive petri nets," in *Petri Net Technology for Communication-Based Systems*, pp. 321–351, 2003.

[39] H. Eshuis, "Semantics and verification of UML activity diagrams for workflow modelling," 2002.

[40] M. A. Marsan, "Stochastic petri nets: an elementary introduction," in *Advances in Petri Nets 1989*, pp. 1–29, 1990.

[41] M. Ajmone Marsan, G. Conte, and G. Balbo, "A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems," *ACM Transactions on Computer Systems (TOCS)*, vol. 2, no. 2, pp. 93–122, 1984.

[42] M. Dumas and A. H. Ter Hofstede, "Uml activity diagrams as a workflow specification language," in *UML 2001The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, pp. 76–90, 2001.

[43] H. Störrle and J. Hausmann, "semantics of uml 2.0 activities," in *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, 2004.

[44] A. H. Ter Hofstede, W. van der Aalst, M. Adams, and N. Russell, *Modern Business Process Automation: YAWL and its support environment*. Springer Science & Business Media, 2009.

[45] "Graphviz, open source graph visualization project." http://www.graphviz.org/.

[46] "NetworkX , python package providing data structures for graphs, digraphs, and multigraphs." https://networkx.github.io/.

[47] "Extension for MS-Visio, GraphVisio." http://www.calvert.ch/graphvizio/thumb.

[48] L. Hayden, *IT Security Metrics: A Practical Framework for Measuring Security Protecting Data*. McGraw-Hill Education, 2010.

[49] "Community edition Saiku." http://community.meteorite.bi/.

[50] N. Gautam, *Analysis of Queues: Methods and Applications*. CRC Press, 2012.

[51] "Python threading package." https://docs.python.org/2/library/threading.html.

[52] "Matplotlib , Python 2D plotting library." http://matplotlib.org/.

[53] "PostgreSQL, open source object-relational database system." http://www.postgresql.org/about/.

[54] "CEF Python, browser embedding package for popular Python GUI toolkits." https://code.google.com/p/cefpython/.

[55] M. Armony, "Dynamic routing in large-scale service systems with heterogeneous servers," *Queueing Systems*, vol. 51, no. 3-4, pp. 287–329, 2005.

[56] "WxPython, Python ui programming library." http://www.wxpython.org/.

[57] D. L. Snyder and M. I. Miller, *Random point processes in time and space*. Springer Science & Business Media, 2012.

[58] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*, vol. 821. John Wiley & Sons, 2012.

[59] B. Schneier, "Managed security monitoring: Closing the window of exposure," *Counterpane Internet Security*, 2000.

[60] B. Schneier, "Managed security monitoring: Network security for the 21st century," *Computers Security*, vol. 20, no. 6, pp. 491–503, 2001.

[61] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[62] D. Tsai, W. Chen, Y. Lu, and C. Wu, "A trusted security information sharing mechanism," in *43rd Annual International Carnahan Conference on Security Technology*, 2009.

[63] R. C. Luo and M. G. Kay, "Multisensor integration and fusion in intelligent systems," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 5, pp. 901–931, 1989.

[64] D. L. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, ISCAS'98.*, 1998.

[65] L. Yu Beng, S. Ramadass, S. Manickam, and T. Soo Fun, "A survey of intrusion alert correlation and its design considerations," *IETE Technical Review*, vol. 31, no. 3, pp. 233–240, 2014.

[66] B. Zhu and A. A. Ghorbani, *Alert correlation for extracting attack strategies*. PhD thesis, University of New Brunswick, Faculty of Computer Science, 2005.

[67] S. Chen and S. Ranka, "An internet-worm early warning system," in *IEEE Global Telecommunications Conference, GLOBECOM'04.*, 2004.

[68] L. Brown, N. Gans, A. Mandelbaum, A. Sakov, H. Shen, S. Zeltyn, and L. Zhao, "Statistical analysis of a telephone call center: A queueing-science perspective," *Journal of the American statistical association*, vol. 100, no. 469, pp. 36–50, 2005.

[69] A. K. Erlang, "The theory of probabilities and telephone conversations," *Nyt Tidsskrift for Matematik B*, vol. 20, no. 16, pp. 33–39, 1909.

[70] C. Kim, S. Dudin, O. Taramin, and J. Baek, "Queueing system map| ph| n| n+ r with impatient heterogeneous customers as a model of call center," *Applied Mathematical Modelling*, vol. 37, no. 3, pp. 958–976, 2013.