TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technology

Department of Software Science

Andres Elliku 144009IVCM

TC70LT

SCALABLE COURSE ON CYBER ATTACK DETECTION

Master's thesis

Margus Ernits

MSc

Lecturer at Tallinn University of Technology

Tallinn 2018

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Andres Elliku

19.03.2018

Abstract

The aim of this thesis is to create a course teaching the basics of the complex topic of cyber attack detection, while at the same time providing freedom from constraints such as time and location and allowing massive scalability regarding the number of simultaneous learners. The course was designed, implemented and evaluated according to the ADDIE model. The ADDIE model is a popular instructional systems design framework used to develop courses. The course provides online hands-on labs. Including automatic feedback of objective completion.

Analysis was conducted to determine requirements that the learner should meet to successfully complete this course. Constraints were defined that would affect the course design, to maintain the objectives of the thesis.

The course learning objectives were developed. Based on the learning objectives, a course design was implemented along with the design of the learning process. On this basis suitable software was selected and a scalable architecture was designed. An architecture that would also allow further enhancements in the future if the need arises.

The first lab of the course was implemented as a proof of concept of the design. All lab objectives include automatic or dynamic feedback systems on objective completion. A pilot, in accordance to the ADDIE model, was conducted on a test group of people matching the student analysis. Their progression monitored, and feedback recorded for corrections and analysis of possible enhancements. During the pilot, the scalability of the course was verified.

Hereby the author would like to provide the readers the option of trying out the course in the pilot phase. To do this, sign up at https://rangeforce.com and use the promotional code: IPS-ait9La-IDS

Annotatsioon

Scalable Course on Cyber Attack Detection

Magistritöö eesmärgiks on luua e-õppe kursus õpetamaks küberrünnete tuvastamise baasteadmisi. Samal ajal vabastades õppurit aja ja koha piirangutest ning tagades väga hea mastabeeritavuse samaaegselt kursust läbivate õppurite arvus. Kursust kujundati, teostati ja testiti vastavalt ADDIE mudelile. ADDIE mudel on populaarne *instructional design systems* raamistik, mida kasutatakse e-õppe kursuste loomisel. Kursus ise pakub võrgus olevaid praktilisi laboreid, koos automaatsete tagasiside süsteemidega.

Õppurit analüüsiti, et kehtestada eeldused õppuri edukaks kursuse läbimiseks. Analüüsi teel leiti piirangud, mis mõjutavad kursuse disaini, et tagada töö eesmärkide saavutamine.

Loodi kursuse õpiobjektid ning nende põhjal teostati kursuse disain ning kujundati õppe protsess. Järgnevalt, arvestades eelnevate etappide tulemusi, valiti välja sobivad tehnoloogiad ning loodi sobilik arhitektuur. Arhitektuuri luues võeti arvesse ka võimalikku vajadust seda tulevikus täiendada.

Kontseptsiooni tõendusena teostati kursuse esimene labor, valideerimaks õpiobjekte ning kursuse disaini. Kõikide labori ülesannete juurde ehitati kaasnevad automaatsed või dünaamilised tagasiside süsteemid ülesande eduka täitmise kohta. Vastavalt ADDIE mudelile viidi järgnevalt läbi kursuse esimese labori piloteerimine test grupiga, kelle liikmed vastasid õppuri analüüsis seatud eeldustele. Nende labori läbimist jälgiti ning saadud tagasisidet kasutati paranduste ning juurde arenduste analüüsis. Lisaks kinnitati piloodi ajal ka kursuse mastabeeritavust.

Siinkohal kutsub autor lugejaid üles disaini ning piloteerimise faasis kursuse laboreid ka ise proovima ning tagasisidet andma. Selleks palun luua konto https://rangeforce.com lehel ning kasutada seal sooduskoodi: IPS-ait9La-IDS

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 70 leheküljel, 8 peatükki, 1 joonist, 1 tabelit.

Table of abbreviations and terms

IDS	Intrusion Detection System
IPS	Intrusion Prevention System
CERT-EE	Computer Emergency Response Team Estonia
S4A	Suricata for All. CERT-EE distributed network monitoring solution
Instructional Design	Methodology for course design [1]
ADDIE	Instructional systems design framework used to develop courses
VTA	Virtual Teaching Assistant
CTF	Capture the Flag
SCADA	Supervisory control and data acquisition
ICMP	Internet Control Message Protocol
DNS	Domain Name System
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
Rsyslog	Log processing software
Syslog-ng	Log processing software
Moloch database system	Large scale, open source, full packet capturing, indexing and
OISF	Open Information Security Foundation
NIST	National Institute of Standards and Technology
Microsoft ATA	Microsoft Advanced Threat Analytics
SEC	Simple Event Correlator
Azure cloud	Microsoft's Cloud computing Platform & Services
Azure Security Center advanced threat protection	Microsoft's offering to provide unified security management and
CLI	Command Line Interface
AlienVault OSSIM	Open Source Security Information and Event Management
VPN	Virtual Private Network. Form of remote access
IPTables	Firewall software
SID	Suricata rule keyword. Value uniquely identifies rule

Python	Programming language
Nfqueue	Userspace library providing an API to packets that have been queued by the kernel packet filter [2]
Sudo	Software giving the ability to granularly delegate root access [3]
Msg	Message
USR2	SIGUSR2 is a User-defined signal. Limited form of inter-process communication

Table of Contents

List of Diagrams

List of Tables

1. Introduction

The aim of the author's thesis "Scalable Course on Cyber Attack Detection", is to create a scalable and interactive hands-on course that does not require an instructor or classroom to teach complex cyber security topics. That can be taken regardless of time or location and allows for simultaneous use by any number of learners.

1.1. Problem statement

Due to the ever-increasing range and complexity of threats in cyberspace, detection has become an increasingly vital part of an organization's cyber security program. [4]

Even with well defended networks we see time and time again that only preventative measures are not enough to build an effective security program. In addition to that the annual FireEye M-Trends report has shown that time from compromise to detection is still too long. [5]

These threats coupled with the lack of trained cyber security specialists indicates the need for new highly effective training programs both for the people still studying and experienced professionals working in the field who want to expand and enhance their skill set. [6]

Time shortage on both the learners and teachers' sides, the decoupled nature of traditional learning materials from real world cases and up until recent years the difficulties of providing a dynamic and sufficiently complex technical infrastructure has limited how a course on cyber-attack detection might be provided.

Universities do teach cyber security specialists, to remediate this shortage. The issue here is providing practical hands-on training, that is needed to ensure sufficient qualification, is difficult because of the large number of students. Lectures and videos are suitable for large amounts of students, but do not provide practical hands-on training. Many already employed professionals need additional training as well. A scalable course, that does not include grading, could offer a possible solution to this.

From the author's personal work experience with the CERT-EE S4A project, one of the objectives when building the solution was to easily provide a way for our partners to add intrusion detection

system rules themselves. This in the hopes, that they will write rules specific to their environment, but also so that one client writing a rule, could benefit everybody else joined. The problem herein is, that a lot of partners lack the necessary skills to write these rules and time to acquire these skills in a traditional classroom setting.

Today's massively scalable and virtualized environments offer us new ways of teaching cyber security. Add to that the proven efficiency of simulations in training cyber security specialists and we have the perfect recipe for teaching cyber-attack detection at scale.

1.2. Objective of the Thesis

The author's master's thesis proposes a cloud based scalable course on cyber-attack detection, where teaching is done through hands-on labs with enough guidance to help the learner achieve the learning objectives. Scalable here meaning no time or location restrictions and scalable in the sense of simultaneous learner count being only limited by the available computing resources. Cyber attack detection i.e. the courses topic focusing on detecting unauthorized access of a computing environment, usually for malicious purposes by utilizing network traffic monitoring solutions combined with log collection and analysis.

This is provided in a format, where the learning process is divided into small practical tasks, each of which is accompanied by a guide to help the learner through the task. Objectives can start from simpler topics like setting up a basic intrusion detection system installation and progress up to detecting complex automated attacks. A close real-world approximation of the more complex tasks is the Cyber Olympics held at the Estonian Information Technology College.

By creating this course, it becomes possible to provide people novel ways how to enhance their skillset in cyber attack detection. Although, it does not provide the same level of support a traditional classroom learning environment, it frees the learner from time and location constraints placed on them and makes it possible to provide almost instantaneous feedback upon completion of a learning objective.

In regards, to a more traditional classroom setting, it can still provide value whereby it can free up time for the teacher, both on the preparation side and during the labs to give more support to the students that need it and letting everybody progress at their own pace.

1.3. Thesis outline

Chapter 1 states the problem of the thesis and gives an overview of the proposed solution. Chapter 2 will present background information regarding cyber attacks and their detection, including an overview of intrusion detection systems. In Chapter 3 related works and state of the art is described. The rest of the chapters cover the analysis and implementation following the instructional design methodology. Chapter 4 gives a background in instructional design and covers the analysis work necessary for the course. Including student analysis, defining constraints, selecting learning objectives and designing the course and architecture for the hands-on labs. In chapter 5 specific implementation of the labs is covered. Chapter 6 covers course evaluation and piloting, with feedback and corrections gleaned from the pilot. In chapter 7 future work is discussed. Mainly how to proceed with developing the course and what can be added or changed in the future. In conclusion, chapter 8 gives a summary.

1.4. Acknowledgements

The author would like to extend their gratitude, to all the RangeForce employees for their help in the courses development and to the participants of the courses pilot, for their invaluable feedback.

2. Introduction to Cyber Attack Detection

Before we define cyber attack detection, we should first explain what a cyber attack is. Simply put its unauthorized access of a computing environment, usually for malicious purposes. A more complete description of a cyber attack as defined by NIST states that it is: "An attack, via cyberspace, targeting an enterprise's use of cyberspace for the purpose of disrupting, disabling, destroying, or maliciously controlling a computing environment/infrastructure; or destroying the integrity of the data or stealing controlled information." [7]

By an obvious logical deduction, cyber attack detection is the process of discovering any and all cyber attacks conducted against the computing environment/infrastructure. Here the assumption being, that intrusive activities differ from normal usage. [8] This covers a very broad set of topics, including technical, organizational and policy. Creating a single course covering all these aspects, would not be very practical, because very often they also target different audiences. Thereby this thesis will focus only on the technical side. More specifically targeting people responsible for implementing technical mechanisms for cyber attack detection. From the technical side, there are also many approaches, some of them listed below, but creating a course covering all of them would not be sensible. Covering all these different technical approaches would introduce too much complexity in a single course and would also require different technical implementations. Thereby this thesis will focus on host and network based detection utilizing network monitoring, log collection and log analysis.

Delving deeper into the technical aspects of cyber attack detection, we can divide the different methodologies several ways. One popular way is to divide between host and network i.e. host based detection and network based detection.

- Host-based detection – Focused on collecting and aggregating data from hosts. Usually a sensor is placed on the machine, that will utilize audit trails provided by the operating system to track events on the host. [9] Some examples of these kinds of tools are the host based components of AlienVault-s OSSIM [10] and SEC [11] in combination with a log forwarding tool like rsyslog. [12]
- Network-based detection – Focused on collecting data from the network. This is done, by monitoring the stream of network traffic, from a central point. This allows for the analysis

of information as it traverses the network. The network traffic being analysed, is compared against rules or evaluated to be anomalous, to determine if an attack is detected. [9] Examples of these tools are Bro [13], Suricata [14] and Snort [15].

Additional methods of cyber attack detection include, but are not limited to:

- User awareness raising – Focused on training an organizations staff, to reinforce good security practices and educate them in detecting cyber attacks. [16]
- Honeypot – A technique where a resource or system, is designed to look attractive to an attacker, but not having any actual use. Thereby an access of this system or resource provides an indication that a possible cyber attack is happening. [7]

To reiterate, although many tools, tactics and procedures are used in cyber attack detection, combining them all in a single course would not be sensible. A course this large, would introduce a lot of complexity to the infrastructure, thereby negatively affecting scalability and portability. Would severely increase the number of requisites demanded from the learner for successful completion of the course. The approach taken in this thesis, is to focus on IDS/IPS solutions and log collection and analysis. A suitably sized range of topics that are closely related. From the analysis, design and implementation discussed in this thesis, a lot of the ideas, considerations and design decisions can form a basis for other people or the author to design separate courses covering these additional methods.

2.1. Intrusion Detection/Prevention Systems

Network based data collection is usually achieved with the usage of intrusion detection or intrusion prevention systems. These are systems designed to ingest raw packet data and automatically conduct rule based analysis of the packets. [9]

Moving on to specifically talk about IDS and IPS solutions. These solutions are designed either around anomaly based detection (Bro [17]) or rule based detection (Suricata and Snort). We will use these three open-source solutions as examples for describing how these network security monitoring solutions work.

As with any network monitoring solution they are designed to either receive a copy of the network traffic (IDS mode) or intercept the traffic (IPS mode). Some clarification here being, that Bro is not designed to be deployed inline, but it is possible to use the NetControl Framework to interface Bro with different network devices to modify the traffic. [13] In case of intrusion detection systems there are no automatic actions except reporting possible, but intrusion prevention systems are usually placed inline at a central point in the network and therefore allow direct manipulation or dropping of packets.

When in IDS mode i.e. receiving a copy of the network traffic, the traffic is evaluated and if either an anomaly is detected, or a signature gets matched an alert is generated. In the case of Bro, a Turing complete scripting language is offered for evaluating packets. For Suricata or Snort a declarative rule based approach is taken compared to Bro-s Turing complete imperative approach. Some advancements in bridging this difference have however been made, for example Lua scripting support for Suricata. [18] All of this still means, that if an alert is generated, human interaction, event correlation or some other system is usually still needed to verify the alert and take the necessary remediation steps.

In IPS mode, when the solution is placed inline i.e. it will receive and forward the traffic, new options become possible. Foremost among those being the possibility to drop traffic if the packet matches a rule that specifies dropping the packet as the action to take. In simplified terms this means that the IPS solution is like a firewall, but the rules that it uses can be extremely complex and granular. For example, targeting specific requests to a webserver where the network packet contains very specific elements, down to the byte level. This however brings the possibility of disrupting legitimate network traffic in case of false positives. Talking about firewalls, often next generation firewall products incorporate an IPS engine. The most widely known example of this being the inclusion of the Snort engine in multiple Cisco products [19]

In the last paragraph a term called false positive was mentioned. Since the terms false positive and false negative, are often used when talking about IDS solutions, a brief explanation is given here, to set a common background for the discussion. A false positive as defined by NIST is “An alert that incorrectly indicates that malicious activity is occurring.” [7] In more layman’s terms, an alert was generated by the IDS solution when it shouldn’t have. A false negative state is when an IDS

solution does not indicate the occurrence of malicious activity, when in actuality it is taking place i.e. it does not alert when it should. For the sake of completeness two other terms used are true positive i.e. when alert fires on malicious activity and true negative is when the IDS ignores valid activity. [20]

With the ever-increasing popularity of cloud services, some might start to wonder where IDS systems fit in this context. Although implementing such solutions might be dependent on the cloud solution used. For example, the Azure cloud provides the Azure Security Center. Even then they are still an excellent layer of defence in more traditional office networks, adding an extra layer protecting non-cloud services and workstations. Even in cloud environments, they can still be useful as an additional layer. Due to their wide range of use cases, for example a web server hosted in the cloud, can also have an IDS solution deployed on the same machine as the web application to add another layer of defence to that specific application.

As previously discussed, IDS systems operate by matching packet headers and content against rules. Thereby having well written rules becomes paramount in using IDS systems for detecting cyber attacks. Because without rules, there is nothing to determine if this or that traffic is malicious or not. Although for example both Suricata and Snort come with both free and paid rulesets, these can often, depending on the peculiarities of an organization, create false positives and furthermore most likely will not catch all malicious traffic. Due to this an operator of an IDS system should be able to create rules themselves as well. One good example being a custom signature designed to catch malicious activity targeted at a feature of a custom developed application of the organization, until the vulnerability can be fixed.

Of course, rules have their own limitations. Main ones being, that to write the rule, you must target a known thing. You can write more general rules targeting an anomaly, but these are very prone to creating false positive alerts. [21]

In recent times other central solutions like Microsoft ATA and accompanying products that send data to Azure, have enabled situations where incidents discovered at one organization to automatically be converted by machine learning into rules that can then be disseminated to other constituents. [22] These enable retrospective detection of cyber attacks that might have already happened but have remained undetected. The CERT-EE S4A projects also has this as one of its

goals. With the inclusion of Moloch [23] a discovered attack at one partner can be analysed at the network layer, to potentially come up with an IDS rule that can then be disseminated to all other partners from the central server.

The author as part of his daily job, has also been involved with the CERT-EE S4A project. The main goal of this project is to create a distributed network monitoring solution, that is built on Suricata. The main objective is to distribute rules across all the detector instances monitoring partner infrastructure and to centrally collect alerts generated by these detectors. [24] The benefits of this are multiple. CERT-EE unlike many of their partners, operates a 24/7 incident response team, thereby centrally collecting these alerts shortens response times and enables timely dissemination of new campaigns unfolding. On the other side, being able to distribute custom rules increases the chances of detecting cyber attacks happening in the networks of partner organizations.

To give the reader a better technical understanding of the S4A platform, it is based on a master slave architecture. The central side is responsible for distributing rules, collecting alerts and tracking the joined detector instances. On the detector side, the instances main function is to run the open source Suricata IDS with the ruleset provided from central and optionally custom rules added by the detectors local operator. Also includes EveBox, which is an open-source tool for viewing and triaging alerts generated by Suricata. [24]

Optionally the S4A platform allows to also install and manage an instance of Moloch on the same detector host. The full packet capture capabilities of Moloch enable further investigation of alerts generated by Suricata. Additionally, it enables hunting operations over the network to discover new attacks that do not yet have rules and thereby enables the creation of new rules. Or attacks discovered on endpoints can be analysed from the network traffic perspective to possibly come up with new rules that way.

3. Related Work

Over the years many have seen the practical value of exercise-based learning. Therefore, there are several examples of courses based on simulations carried out in test environments.

One such example is a course developed in 2006 in Towson University. The paper describes a laboratory-based capstone course that covers multiple tools and techniques used in defence of information systems. The course spans several weeks and 5 or 6 lab exercises with each exercise taking roughly two weeks. [25]

After initial implementation of the course and years of use, the author of the course has published a new article with lessons learned. [26] The course was originally taught in a more traditional format, however student feedback indicated that this format made it hard to both take notes and follow procedures in the lab. Over time, it was understood that giving the students all the materials and letting them progress at their own pace was a more efficient approach. This also freed up instructor time for questions and providing one-to-one help.

Another example is a distance education course developed in the Iowa State University in 2009. The course lasts for 18 weeks and consists of three phases: the planning & implementing phase, the defending & attacking phase and the infrastructure assessing phase. This course requires the students to plan and implement their own infrastructure in the first phase, complete with network plans and full implementation in a VMWare ESXi environment. The second phase has the students attack each other to capture flags while defending their own network. The last phase has the students create a comprehensive report and presentation about the first two phases. Including the tools used, attacks and weaknesses used and changes to make in the future. [27]

The United States Naval Academy has also developed an introductory cyber security course that includes multiple hands-on laboratories. In their paper they describe three laboratories developed for this course. The labs are conducted in a sandbox environment utilizing VMWare virtualization technologies. Serious development effort was also expended to create instructor notes, student instructions, worksheets and supporting software. The labs are accessed through a wireless connection in the classroom (the students are required to have a laptop). To handle 126 concurrent users, a four server system was used to host the virtual machines that make up the lab environment.

Master virtual machine images were created by instructors that were then cloned to provide six identical classrooms. Moving on to the labs themselves. The first lab covers network reconnaissance, the second one focuses on network attack and the third one focuses on network defence. All labs utilize worksheets, that the students must fill for tracking purposes. Finally, the authors present some lessons learned. Firstly, that the lab environment requires a lot of preparation from the instructors, not only from a previous skillset perspective but also how to setup and shutdown the virtual environment. They point out that lab assistants would have been helpful with answering student questions, but they did not have enough of them for every lab. It was difficult to track the progress of each student and instructor help usually started with the instructor trying to establish context. Since a fixed number of activities were present in the labs, students usually had to pair up thereby lessening the amount of hands-on experience each student received. Worksheets turned out to be a valuable tool to help with providing feedback and tracking student progression. To conclude the authors, reiterate that a lot of effort is required from the faculty and technical-support staff, both with development and ongoing maintenance but the student benefits are immense. [28]

One final example argues for the importance of basing the course on real-world scenarios to also teach about the main concerns of security in the real world and give experience in managing issues of real and usable services. This is done by holding an Internet role-game as laboratory exercises. In this course students are split into different teams covering roles such as internet service providers, certificate issuer and network infrastructure. Participants were also assigned adversarial tasks to allow everybody a chance to experience both sides. The course was split into two phases. The first phase entailed setting up the infrastructure. In the second phase, both participants and teachers simulated usage of the services built in the first phase and played out the assigned adversarial tasks. This approach according to the authors helps avoid issues like paranoid defence approaches, focusing only on initial systems installations and being able to anticipate attacks. These issues are countered by giving teams actual business-like missions, keeping the simulation alive by also including legitimate actions and tasking the teams to properly safeguard customer data. [29]

Although the examples given are effective in traditional learning environments with a course spanning several months, they do not scale very well, and a large portion of time is spent on setting up the infrastructure, thereby leaving less time to focus on the actual cyber attack detection portion.

Some lessons learned from these examples then are, that providing all the materials and labs to the learners at once and letting them progress at their own pace, has been shown to be an effective approach. Providing materials separately allows the learner to study them independently, allowing them to do additional research when need be. Handling more of the deployment of additional components for the learners, frees up their time to focus on the core content of the course.

The Virtual Teaching Assistant or for short VTA, is the core of the RangeForce platform. This solution is designed to provide an interactive learning environment for hands-on labs. It provides the lab guide, with the materials. Provides the learner, with feedback on the objectives being completed by automatic evaluation or dynamic questionnaires. The RangeForce platform itself provides deployment automation for the virtualized infrastructure and other labs covering different cyber security topics. From the course designers side, it provides an interface for course material development. Some similar alternatives include platforms like CTFd. [30] These CTF solutions, usually do not offer any kind of scripted feedback or grading system however. Instead completing a challenge provides a flag, usually in the form of a text string that can be submitted to the CTF platform for marking an objective completed.

Moodle, a learning platform also in use by the Tallinn University of Technology, offers some forms of automatic grading and feedback, but these are not suitable for these kinds of hands-on labs. In addition, the Moodle platform has a large and complex feature set, most of which is not needed for this course. [31]

Therefore, the VTA is the most suitable solution. It fully integrates with the hands-on labs. Provides scriptable automatic feedback, that is a large portion of how a course like this can scale. Provides a convenient development platform for the course designer and a suitable interface for the learner. This comprises the core functionality of needed features for the design, implementation and use of the course. At the same time, the solution remains small, easily portable and extendable. An added benefit is that this platform is free to use for students of Tallinn University of Technology and Estonian IT College.

4. Instructional Design

When designing courses aimed at teaching, a specific methodology should be chosen for development. This to ensure that different parts of the course come together to create a cohesive learning experience, that the course is validated against the target audience to be suitable and efficient, that set learning objectives are defined to ensure that the goals of the course are met and to properly define prerequisites of the learner. The key points of instructional design are:

- Instructional design is a technology for the development of learning experiences and environments which promote the acquisition of specific knowledge and skill by students. [1]
- Instructional design is a technology which incorporates known and verified learning strategies into instructional experiences which make the acquisition of knowledge and skill more efficient, effective, and appealing. [1]
- While instruction takes place in a larger organizational context, the technology of instructional design is concerned only with the development of learning experiences and environments, not with the broader concerns of systemic change, organizational behaviour, performance support, and other human resource problems. [1]
- Instruction involves directing students to appropriate learning activities; guiding students to appropriate knowledge; helping students rehearse, encode, and process information; monitoring student performance; and providing feedback as to the appropriateness of the student's learning activities and practice performance. Instructional design is the technology of creating learning experiences and learning environments which promote these instructional activities. [1]

The specific instructional design model chosen for this course is the ADDIE model. The ADDIE model has already found widespread use in courses for the Estonian IT College and Tallinn University of Technology. The ADDIE model is the generic process traditionally used by instructional designers and training developers. The five phases—Analysis, Design, Development, Implementation, and Evaluation—represent a dynamic, flexible guideline for building effective training and performance support tools. [32]

The next chapters of the thesis follow the phases of the ADDIE model. [32] A brief overview of these phases are now discussed.

The analysis phase is covered in the current chapter, where objectives, the learning environment and the student analysis including pre-requirements are defined. [32]

In the design phase lessons and specific learning objectives are defined. Exercises are planned, including content and media selection. [32]

Development phase will cover the actual content creation, including automatic checks for objective completion and setting up the base infrastructure for the labs. [32]

The implementation phase according to the ADDIE model traditionally deals with training the facilitators and learners. However, since this course is designed to be scalable in the sense of not requiring a classroom setting with a facilitator, in addition to technical differences, the author focuses on fine tuning the course content. [32]

Evaluation phase will cover final validation of the technical infrastructure and content, verifying learning outcomes and assessing the efficiency of the designed solution. Testing with people fitting the target audience, set out in the student analysis chapter. [32]

4.1. Student analysis

As per the design guidelines set out in the ADDIE model, in the analysis phase an assessment of the target audience must be conducted. Especially due to the complex nature of the topic, specific requirements for the learner must be set. These are set to ensure, that the learners current skill set, and knowledge base is sufficient to ensure successful completions of the labs and in extension that learning objectives are met. Although the course is also suitable for usage in a university setting, the main target audience are professionals already working in the information technology field. Due to this we should expect constraints on how much time can be allocated by the learner for any given lab, so minimizing the time usage is also something that will be considered in the chapter.

Defining the pre-existing minimum technical skillset:

- The learner is comfortable operating in a GNU/Linux based operating system environment. Due to the lab infrastructures heavy usage of Linux. The selection of the Linux operating system is explained in a following chapter. First and foremost, skills in operating in a CLI environment. Although a desktop environment is also used, previous experience with that is not important, since it mainly provides web browser and CLI functionalities. Main requirements for the CLI interface, are skills that include navigation of the file system, finding and modifying files, including their permissions. Using a text editor to modify the contents of configuration files. Running processes and services. Have a basic understanding of flows and redirections. Useful but not required in addition to the previous is familiarity with userland utilities like grep, find, cut etc. Courses like the operating systems course offered in Estonian Information Technology College, the systems administrations course offered by Tartu University or any other similar course fully cover these topics.
- Basic understanding of networking. Including but not limited to UDP and TCP traffic, familiarity with the HTTP protocol, how data is transferred over the network, specifically how network packets are constructed and what are the different components. Initial labs of the course focus almost solely on IDS solutions and the creation of rules that are utilized by these solutions. Since IDS solutions operate on the premise of analysing network traffic, the ability to create rules is predicated on understanding how networking works. The initial labs will focus on IPv4 traffic regarding traffic monitoring. Because the usage of IPv6 addressing is still not very widespread in corporate environments. A separate lab in the future can be added to address this. Courses built on the Cisco CCNA curriculum, like the ones offered in Estonian Information Technology College or Tallinn University of Technology cover these topics.
- Comfortable with regular expressions. Able to compose and utilize them. Required to be able to create certain kinds of IDS rules and when analysing logs. The Cyber Defense Monitoring Solutions course offered in the cyber security masters course of Tallinn University of Technology offers an excellent coverage of these topics. From freely available materials, sites like https://regexone.com and https://www.regular-expressions.info/tutorial.html offer an excellent introduction to the topic.

On the flipside the learners previous experience should also be evaluated to determine, if some of the labs might cover topics that are already familiar. In this case, specific recommendations can be given to the learner to focus on labs covering gaps in their skillset or those that enhance their knowledge. This allows to minimize the learners time spent on the course, but still meeting the learning objectives. As an example, maybe the learner is already familiar with deploying an IDS solution, so instead of spending time on rehashing known concepts the learner can directly proceed to learning how to create IDS rules.

Presenting a list of requirements for each lab along with a detailed description of learning objectives, presents the learner with an easy self-assessment option. The learner can then consider if the specific lab covers a topic they need additional training in. For more formal settings where accompanying evaluations might be used, another option is to use utilize some form of testing to help the learner verify that the topics covered in the lab provide no additional benefits to the learners existing knowledge base.

4.2. Platform constraints

As the thesis topic indicates a major part of this thesis focuses on the scalability of the course. Scalability from several different perspectives. Firstly, from the perspective of utilizing the massive increase of computing resources to design learning experiences that employ new and innovative ways of teaching complex topics. While at the same time attempting to minimize traditional time, location and resource constraints on the learning process. Secondly, designing hands-on learning experiences that scale to various environments and allow for maximum portability. Thirdly, designing new ways of offering automatic feedback and interactions as the learner progresses through the course.

Minimizing constraints on time, location and resources. A hands-on practical training course can be designed in such a way, that given sufficient motivation on the learner's side and properly developed and tested content, a traditional instructor led classroom setting is not a strict requirement. Offering the course in an always available online format, where the labs are hosted in a dedicated computing environment that is accessible to the learner with internet connectivity, allows the learner to choose a suitable time and place for the course. Breaking the course up into

several labs focusing on different topics, means that the learner does not have to complete the course in one go, but cover it in manageable chunks at suitable times. This also allows the learner to cover the topics at their own pace and work through additional learning materials as needed. The learner can also pick and choose the labs they want or need in accordance to their self-evaluation.

The scalability of the lab environment allows the course both to be deployed on the RangeForce platform and scale there. While at the same time allowing a wide range of other deployment scenarios including, but not limited to other lab environments like the IT College i-Tee remote lab environment, an organization using these materials for in house trainings or even at a person's home computer as a simple set of virtual machines. Although in other environments, small modifications might be required, especially on the presentation and accessibility side. Nevertheless, the basic principles, base infrastructure and defined learning objectives will remain valid and usable.

When an instructor is not available, proper care must be taken, that the learner is given sufficient information about the given tasks, the materials are sufficient and that timely information of the completion of the learning objectives is provided. These will help the learner to smoothly progress through the objectives, while still allowing them to stick to their own pace. From a scalability perspective, this will also help minimize the amount of computing resources spent per learner per task, automated feedback eliminates the need for a second party to validate that the lab is successfully completed and if necessary, hints can be provided where required.

4.3. Additional constraints

The course will be developed on the RangeForce platform and will be integrated into the offerings available there. This choice will be explained in a later chapter. However, different usage and deployment scenarios will be considered as explained in the previous chapter. We can use these as a basis to develop the principles of some technical aspects. Feedback gathered from RangeForce customers about other labs provided there also gives us some information to consider.

To make the course better accessible to already employed professionals the different learning objectives and labs should be designed in such a way to not take a significant amount of contiguous

time. This allows the learner flexibility in managing their time. It might also be a more eloquent solution to the employer as well, because the worker can invest smaller amounts of time over a longer period out of their working hours instead of leaving for multiple days. Restricting the planned length of the hands-on exercises also leaves us the possibility of utilizing this course during traditional synchronous classroom lectures. Thereby allowing us to cover several different forms of learning.

Based on these considerations, we can set the approximate time limit of each hands-on task or learning objective to roughly 1 academic hour aka 45 minutes. This timeframe shall be based on the predicate of the learner meeting all the prerequisites set out in the student analysis chapter. Additional theoretical materials, if needed, can be studied separately during any suitable time. Whether at home, in class or at work. Decoupling the hands-on tasks and practical materials this way further supports in providing this flexibility to the learner. Simultaneously, in a classroom setting, the lecturer might cover the theoretical materials in the first 45 minutes and the second 45 minutes can be dedicated to the hands-on lab. This given a fairly standard 90 minute lecture.

The practical lab exercises developed during the author's bachelor's thesis on intrusion detection systems [33], the bachelor thesis of Rene Juhanni on log event visualization [34], that was supervised by the author and Risto Vaarandi's cyber defence monitoring solutions course are a good indication that roughly 45 minutes is a reasonable time limit in which to operate.

Similarly experience and feedback gathered by RangeForce, also indicates that for organizations and employed professionals, this is a good approach to take.

4.4. Resource constraints

Modern cloud platforms and virtualization techniques offer us better resource handling and more resources in general. Despite this, to maintain scalability we should strive towards keeping the per learner resource usage of simulation environments small. This way we can better avoid the degradation of performance caused by large amounts of simultaneously conducted lab exercises. Maintain scalability if computing resources are limited or allow a very large number of simultaneous learners, with reasonable computing resources. The network traffic generation and automated attack machines can be separate dedicated machines that can be shared by the learners,

so we do not have to take them into consideration for planning resource usage. Specific designs for the lab infrastructure is covered in a dedicated chapter.

Because commercial solutions might impose prohibitively larger deployment fees, or in the case of RangeForce a large increase in per learner fees. The preferred approach should then strive for the use of open source and free components. The added benefit of this approach is the ready availability of open source software outside of the learning environment. Offering our learner, the ability to also utilize the same familiar solutions in their work. Even in the case of a constrained security budget. Since open source software often powers known commercial software used for attack detection [15], the student should be able to easily specialize in a commercial product later, if needed.

4.5. Course analysis

i.Defining learning outcomes

Since these components must ultimately serve the purpose of teaching cyber attack detection, the first step is defining the learning outcomes of the course. As also set out in the chosen ADDIE model for how to design such courses. Additionally, picking the right components is also predicated on knowing what the end goal is.

The purpose of cyber attack detection in an organization is to identify individuals who are using a computer system without authorization and those who have legitimate access to the system but are abusing their privileges. [7] This process involves gathering relevant data from multiple sources. Processing that data and extracting relevant information that can then be analysed either automatically and/or by a human. During analysis, anomalies and known attack vectors can be discovered either using a rule based approach or by comparisons against known good baselines.

This largely defines what the general outcomes of the course should be. That the student can setup the necessary technical infrastructure to collect, process and analyse the data available to him. During the analysis process, discover potential attack patterns using anomaly and/or rule based approaches from the information gleaned. Triaging and responding to possible attacks, is in itself a very broad and complex topic, that won't be covered here.

As a proof of concept, in the beginning the course will focus on IDS solutions as the main technical infrastructure that the learner will be able to deploy and manage by completing the course. Regarding IDS solutions, a large part of the management is centred around the rules utilized for analysing network traffic. Because of this a key skill to obtain is managing, creating and modifying the rules in use. Managing and modifying to reduce false positives created by existing rules and creation to target applications and behaviours specific to an organization or specific threats emerging. After being able to process network traffic and applying rules in the beginning stages of the analysis phase, alerts will be generated. Sometimes an excessive amount of them. Understanding and filtering the alerts generated is the final step. After that a feedback loop into the rule management and creation process begins and the process of responding to validated alerts.

An additional topic to be covered here, are premade rulesets made for IDS solutions. Examples being the Emerging Threats [35] and VRT [36] rulesets. These rulesets have both a free and commercial version. In the course, we can easily make use of the free versions, to teach the basics of managing these rulesets. They provide excellent value in an organization, by helping the IDS solution produce value with minimal effort. The downside being that often, these rules cannot account for the specifics of any given organization and therefore tend to have a higher rate of false positives. Here the leaners ability to manage them, becomes important. To be able to understand that a rule is creating false positive alerts and manage them appropriately. Mainly through enabling and disabling specific rules or rule collections. From the collections side, disabling some collections in the ruleset means a lot less rules to process by the IDS engine, thereby freeing up resources. Example here would be disabling rules targeting SCADA systems, if the organization does not have any systems falling under that classification.

Having covered the initial learning objectives focusing on IDS solutions, the course will move on to the log collection and analysis portion. With any log correlation solution, the first step is to collect them to a central point. Firstly, for structured storage to aid in incident response, performance and historical analysis. This usually involves deploying a software solution on the monitored hosts, that will send the logs generated by applications and the operating system to the central collection machine. Optionally doing local filtering, to reduce the number of logs sent. From the collector side, a solution that listens for incoming events, formats them to a structured fashion and stores them is needed. This defines the basic needs of our next learning objectives.

Firstly, that the learner knows how to deploy and configure a software component on the monitored host, responsible for sending the logs to the central collection and correlation server. Secondly, that they can deploy and configure software to receive, parse to a structured format and store these logs.

After covering log collection, due to the number of logs usually generated by this collection process, automated analysis and correlation becomes necessary simply because a human will not be able to track and correlate all this information. Also, because automated tools enable us to act much quicker. Thereby, our final learning objective is to teach how to do correlation and analysis on the logs. This involves adding a tool capable of log correlation to the central collection machine. A tool that can read the incoming logs and based on correlation rules take some form of action. Whether it is simply reporting the generation of an alert or something more complex, like adding an IPS rule or firewall rule. This kind of correlation software will also enable creating rules that do not act on a single piece of information but can also react to multiple events from multiple machines. An example here might be a correlation that a single source IP is doing brute force attacks against multiple endpoints over a period of time.

In the course as in real life, care should be taken, that when the learner creates rules, some guidelines should be followed. First and foremost, that these rules do not disrupt actual business processes. This usually applies to situations where IPS mode is utilized or log correlation software can take automatic blocking actions. In both cases, these can adversely affect the availability of business processes. Secondly that they generate a minimal number of false positives, to reduce the need for filtering the alerts and to not overload the security specialist who has to respond to these alerts.

Fortunately, in this kind of a lab environment, everything can be tightly controlled and monitored. Giving us the ability to incorporate these considerations into the feedback process and direct the learner's attention to them if need be.

Now that we have analysed, what the necessary base objectives for the course are. We can define them as skillsets that the learner should obtain and how we can provide evidence in the feedback process. Our full specification of the learning objectives, along with descriptions and evidence is:

Table 1. Learning objectives.

Learning objective	**Description**	**Evidence**
Learner can deploy an IDS solution	How to deploy an IDS solution.	Automated feedback ensures that the IDS is successfully installed
Learner can configure an IDS solution	Can configure the IDS solution to run properly. Knows how to configure IPS mode.	Automated feedback checks that the solution is running properly and in the correct mode.
Able to write Basic IDS rules	IDS rule structure. How to target different kinds of traffic. Available actions and keywords. Performance considerations.	Automated feedback checks that required fields are present in the rule, traffic directionality is correctly specified etc.
Can manage rules and rulesets	How to manage IDS rulesets. Disabling, enabling and updating rulesets.	Automated feedback checks, that the right ruleset is downloaded, and rules are modified according to the given task
Able to collect logs from one or more hosts	Collecting logs from hosts on the network, for analysis and storage.	Automated feedback checks that software is installed and running. Logs are reaching the collection point and are stored correctly
On a basic level able to analyse and correlate collected logs	Utilizing regular expressions and tools to analyse logs for cyber attack detection.	Automated feedback checks that alerts are firing. User must fill dynamic questionnaires answering questions about attacks they should detect.

This covers the initial set of learning objectives for the course. A minimum set to get the learner started. However future additions to this list are possible and will likely be implemented in some form or another. Possible topics include visualization and reporting. These additional topics will require additional analysis and considerations on the best design. Mainly due to additional resource requirements imposed with visualization software. Tools like Kibana [37] and Evebox [38] use Elasticsearch [39] as the backend database, which might require additional resources and thereby affect scalability. This must be carefully considered to ensure no scalability loss.

ii.Course design

After defining our learning outcomes, we can begin to design a course. While designing, we must also consider the constraints and student analysis conducted in the previous chapters. The course should also follow a logical progression where initial objectives cover the basics and progress to more complex topics. On the other hand, the hands-on tasks should be grouped into labs that cover a specific subject matter. This will help ensure scalability and permit the learners to skip certain labs that offer no benefit regarding the expansion of their skillset.

The first lab in the course will focus on the basics of IDS solutions. It will additionally, for this thesis act as an initial proof of concept to validate the design and implementation. IDS solutions were chosen, because they are widely used, are an excellent addition to a defence-in-depth approach and help cover a variety of different topics under the defined learning outcomes. [4] The first lab will cover deployment of an IDS solution. Familiarization with the configurations and the basics of creating rules. Also cover IPS mode and explain differences between the modes.

The second lab in the course will do a deeper dive into writing and managing IDS rules. First part of the lab will be to cover managing rules and rulesets. This includes updating and utilizing freely available rulesets and filtering out rules that create excessive amounts of false positives. Enabling and disabling them and a tool used for this. Second part will cover more specific targeting options, usage of different keywords used when writing rules and utilizing regular expressions for matching on packet headers and contents. Focus of this lab will be detecting attacks against the web platform placed behind the IDS solution. This will help tie together the first and second labs into a logical and cohesive progression in the topics covered. This way of segmenting the hands-on tasks then allows for students to skip the first lab, if they are already familiar with operating an IDS solution but have not yet learned to create rules for it.

After these two labs, a basic understanding of how to conduct network traffic based cyber attack detection is conducted. A solid foundation is laid from which to progress. From here we can proceed to collecting and analysing logs i.e. host based cyber attack detection. This usually involves logging different activities on the endpoint systems and forwarding the generated events for collection and processing to a central system. The collection portion enables correlation of events, including correlation over multiple systems and conducting post-mortem analysis in the

case of security incidents. This subject should cover the setup of necessary logging functions on the systems, forwarding relevant events to a central collection and analysis system and covering the automated analysis of these events.

The third lab will cover setting up a log collection solution, covering the learning objective of the same name. Covered topics are, deploying a log forwarding solution to the web and desktop hosts and setting up a log collection server on the IDS machine. Due to the amount of logging that most systems do, it is not rational to send everything for central processing. Instead local filtering can be utilized to only send relevant data, such as relevant firewall and authentication logs, to a central system for analysis. We achieve the defined outcome by teaching the student about the basics of storing the collected event data in a normalized and structured fashion that would allow correlation, visualization and analysis to be conducted during the next stage.

The fourth lab will cover the log analysis learning objective. How to correlate and alert on logs collected from the hosts. This will include deploying suitable software for the task on the collection machine. Configure it to use the available logs and finally to give basic understanding on how to create correlation rules.

In the final stages, the student should be nearing the completion of a framework that would cover the basic aspects of conducting cyber attack detection. Here the normalized and structured data is analysed and correlated. Involving the creation of rules that alert the user about anomalies and potential incidents in the infrastructure.

iii.Selection of software components

Previously set constraints on the course design, resource requirements and other specifications, provide us with some guidelines on the selection of software components for the lab infrastructure. Selection of software components will be based on insights gleaned from related work, the constraints previously set on both the architecture and learner. In suitable cases, multiple possible solutions will be compared to find the most suitable one given the constraints and learning objectives. Our first consideration will always be that the software suits the learning objectives.

After that in order of preference are open source solutions, then freeware, then solutions that offer free usage for educational purposes and if all else fails commercial solutions.

We will begin, with the operating system. When choosing the operating system, we must take into consideration first and foremost the learner themselves. Especially their pre-existing skillset. Considering what has been covered in the student analysis chapter, the most suitable would be a Linux distribution. This decision is also supported by the need to use an operating system that will help maintain portability, allow for as many usage scenarios as possible based on the licensing policy and help with fulfilling resource constraints.

Specifying a Linux distribution, Ubuntu Linux is among the most popular Linux distributions currently available. Therefore, it is safe to assume that for most of the learners going through the course, given that they meet the prerequisite of being familiar with Linux operating systems, the platform is already at least somewhat familiar. Additionally, Ubuntu Linux has a suitable licensing policy to allow free usage in both commercial, educational and personal use. The RangeForce platform already offers some premade components like the web server used in the labs. These components are also based on Ubuntu Linux, thereby using Ubuntu Linux for the other components as well, makes for a homogenous environment and reduces the complexity of the learning environment. Suricata, the IDS solution used for the labs, also offers a PPA for Ubuntu while also fully supporting more complex installations. That allows us to utilize the same base operating system for all the different labs that are provided with this course.

As with the operating system selection, we should consider the different constraints already set. For IDS solutions specifically, the allowed usage scenarios limit the selection. This because a lot of IDS solutions are commercial offerings. At the same time, since many commercial IDS offerings also utilize open source IDS solutions, the knowledge acquired with the use of an open source solution, can often be either directly applicable or require minimal readjustments when implementing a commercial solution. Based on this we can limit our initial selection to the three most popular open source IDS solutions. These being Suricata, Snort and Bro.

Although Bro-s licensing policy is suitable to us, since Bro utilizes its own scripting language for creating rules the learning curve tends to be much steeper. Due to this, a more traditional rule based approach makes more sense given the student analysis. This leaves us with Snort and

Suricata. Both are open source and offer a suitable licensing policy. The rule structure is very similar. Suricata supports using rules created for Snort like the VRT ruleset. [14] Conversely Snort support for Suricata rules is not that good, mainly because of additional keywords supported by Suricata. [40] Thereby the skill acquired practicing on one platform is almost directly applicable to the other. Although Snort will offer proper multithreading support with its 3.0 version, it is at the time of this writing still in alpha status. Suricata on the other hand has offered multithreaded support for years. This gives Suricata a slight advantage in scalability from small deployments to large scale installations. Effective hardware resource usage allows the beginner user to be slightly less concerned with the effectiveness of rulesets and custom rules created by them. A small additional consideration is also that the CERT-EE S4A project utilizes Suricata. Selecting Suricata then also helps the author in solving the side problem of CERT-EE-s partners being able to efficiently manage their S4A instance.

For managing rules and rulesets The OISF has recently introduced Suricata-Update. This is now the official way to update and manage rules for Suricata. [41] Since it is the official tool for Suricata, our chosen IDS solution, this will be used as our chosen software for rule management.

When picking solutions to be placed behind the IDS solution. In the context of this course, we should favour components that allow us to craft a wide range of different attacks which in turn allow us to present the learner a multitude of different tasks to solve regarding writing signatures. In this context web applications are a very good first target. We can run them on the same chosen operating system of Ubuntu Linux. We can easily modify the software stack of the application, for example exchanging the Apache web server for Nginx. From the rule creation side this allows us to present the learner with various scenarios when creating rules. The rules can be crafted to target packet headers, HTTP protocol headers, HTTP request body contents. Pre-existing skills about regular expressions, these skills can also be utilized in this context and allows the learner to refine their knowledge of regular expressions as well. This will also be suitable in the next labs, when dealing with subjects like log collection and analysis. All this combined allows us to present many different learning objectives, while still maintaining a homogenous and easily portable environment.

Moving on to log collection solutions. Probably the two most known and widely used log collection solutions are rsyslog and syslog-ng.

RSYSLOG [12] is advertised as the **r**ocket-fast **sys**tem for **log** processing. It has a modular design and has evolved over the years to include a wide range of functionalities. Among those functionalities are: multi-threading, support for TCP, SSL, TLS and RELP, offers database backends like MySQL, PostgreSQL, Oracle and others, can filter on any part of a syslog message, has a fully configurable output format and is also suitable for enterprise class usage. [12] It is also fully open source and free to use.

Syslog-ng [42] offers a very similar feature set. Additionally, provides a commercial solution with an extended feature set.

Due to rsyslog being by default included with our chosen Linux distribution Ubuntu and its syntax having a less steep learning curve when starting out, our labs will utilize rsyslog for the log collection labs.

For log processing and correlation, we can turn to SEC. SEC is described as an event correlation tool that allows advanced event processing for use in various situation like event log monitoring, fraud detection and many others. In the context of SEC, event correlation is defined as a procedure for processing a stream of events to detect and act on defined event groups happening inside a specific time frame. A key aspect setting SEC apart from other event correlation solutions, is that it is lightweight, platform independent and runs in a single process. Other benefits include running as a daemon, employ it in shell pipelines, interactive execution inside a terminal and running many SEC processes simultaneously for different tasks. [11]

As the description says, unlike many other similar solutions, SEC is lightweight and platform-independent, thereby fits our scalability constraints. It is easy to get up and running, while at the same time providing a lot of flexibility and advanced features, this provides us various usage scenarios in our hands-on labs. It also conforms to our constraint of being a free and open source solution. SEC is also used in the TTU cyber security masters course in the Cyber Defense Monitoring Solutions course taught by the tools author Risto Vaarandi.

iv.Designing the learning process

Having defined the learning objectives and general design of the course, with our software components selected, we can start to develop the learning process. The course will take a fully hands-on approach. Meaning that it will be fully conducted in the lab environment. To aid in completing the labs, guides will be provided for all objectives in the lab and additional materials will be referenced where suitable. This way the learner keeps the ability to progress at their own pace and is provided theoretical materials for additional reading. A hands-on labs approach is taken, because firstly practical exercises are a great way to obtain and cement the knowledge gained over the course. The practical application is something that we can automatically test and thereby provide feedback to the learner, that they have successfully completed an objective and thereby have gained an understanding of the topic. We're not asking the student, if they have completed a learning objective, but measure it. As an example, the student would be required to block a specific web attack using the IPS solution they deployed. In this case, we can measure that they created a new rule, if it blocks the attack and does not negatively impact the availability of the web service itself. This is also paramount in retaining the scalability of the course.

A different topic here is plagiarism and/or unfair help. Since this course is focused on teaching and especially highly motivated learners like already employed professionals wanting to expand their skillset, we therefore consider this to be out of scope in the current context. Where needed an additional separate exam or grading can be conducted for this.

In some cases, doing only automatic evaluation or feedback, might not be the most elegant solution. Another option is asking the student questions at the end of lab objective to verify that they now grasp the topic covered. Here an additional benefit of the RangeForce platform is the ability to customize the answers dynamically per learner. In the first lab for example, the student is asked to deploy an IDS rule detecting inbound ICMP traffic. This traffic is generated by the router machine using a randomly generated source address. In this case, we can ask the student to submit the source IP address of the traffic as an answer to the question, thereby validating objective completion. This makes validating the objective easier for the designer and introduces some more dynamics to the course regarding evaluation. This allows some flexibility on the design side, especially in cases where automatic checking might be cumbersome or complex to implement.

4.6. Architecture/Scalability

Now that we have defined the course outcomes, selected our software components and designed the learning process, a suitable base architecture for the labs shall be chosen. As defined in software selection all machines will be based on Ubuntu Linux. The tasks involving IDS will utilize Suricata. A custom web application will be used as the protected target. For log collection and analysis rsyslog and SEC is the chosen set of software for these topics. The architecture will have to scale from one learner hosting locally up until a reasonable number of learners simultaneously on a shared computing environment like RangeForce. The main constraint for the reasonable number being available computing resources. The lab architecture and course design should not affect this number. This does not only affect the computing resources, but also the network topology of the lab. The network topology as well, must be suitable in a wide range of scenarios to maintain portability.

From the previous chapters we can gather, that at a minimum we require:

- WWW - A target machine, chosen to be a server hosting a vulnerable web application. This will act as an application for our IPS solution to protect and provide additional logs for the second part of the course.
- IDS - Server running Suricata, our chosen IDS solution placed in front of the target machine. This can also be used as our log collection and correlation machine. This way helping to keep the architecture minimal.
- ROUTER - Server from where attacks and objective checks can be conducted. To keep the architecture minimal, this host will also serve as a router for the internal lab network. For conducting attacks and checks, multiple random additional IP addresses will be added to this machine. Some of these addresses are used for attacks and others for checks i.e. we will have red and white IP-s added to the machine.
- DESKTOP - A desktop machine for the learner, where materials and other servers can be accessed. To avoid potential connectivity losses that might happen if an incorrect rule is added to the IDS machine, this desktop will be placed in the same subnet as the IDS

machines external interface. Will also provide more logs for the log analysis and correlation portion of the course.

In total four machines per learner. This is a suitable amount, enough complexity for various scenarios, while maintaining scalability. Allowing use ranging from hosting locally up to RangeForce scale of tens or hundreds of learners going through the lab simultaneously.

Base network topology also allows for easily adding components later to adjust for different scenarios. Example being the addition of workstations for log collection objectives or creating IDS rules targeting traffic from them. The lab infrastructure will represent the infrastructure of the fictional enterprise Deeppacketcorp. Creating a fictional enterprise allows for a story to be added around the objectives, thereby emulating real world scenarios more closely. Also allows adding proper DNS names to hosts, which makes navigating the infrastructure easier for the learner and plays a role when doing log collection. Since a picture is often worth a thousand words, a diagram of the base architecture is presented here:

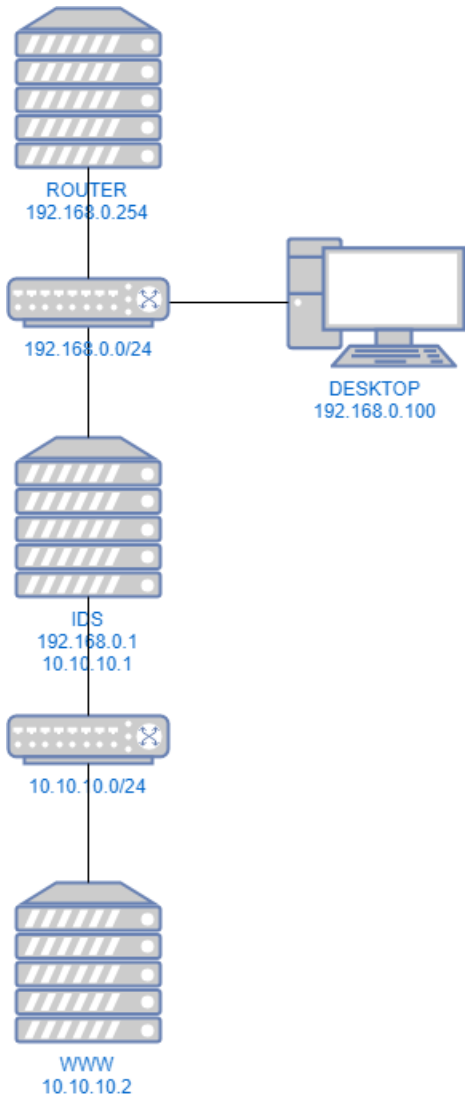

Figure 1. Diagram of base lab architecture

In the analysis phase in accordance with the ADDIE model, we defined our target audience of learners and the necessary pre-existing skillset they should have to be able to successfully complete the labs. We defined constraints affecting how the course should be designed and implemented. Learning objectives were defined, a course design was planned and a design for the learning process was conducted. Needed software components were picked. Finally, a base architecture considering all scalability and portability aspects was created. Having defined these, we can now proceed to the implementation phase of the course.

5. Implementation

The course will be implemented on the RangeForce platform. Although the core principles and most of the implementation is designed to be usable in a wide range of conditions. The RangeForce platform is selected, because it offers multiple benefits by providing a lot of base functionalities. Fully integrated solutions for lab infrastructure deployment and access. This includes automatic deployment of the entire lab infrastructure, that are specific for a given learner. Web based access with a remote desktop protocol implemented through a web based user interface. Learner credential management for authentication. Integration with the VTA to allow the designer convenient use of API-s when doing feedback and checks. Integrated Virtual Teaching assistant where materials can be developed and presented to the learner in a convenient way. All configurations, materials and checks are backed with git repository's allowing for version control and tracking of any changes.

Of course, implementing on other platforms will remain possible. Main things needed in that case, are a computing resource to run on, some version of automatically deploying the lab infrastructure or although cumbersome, manual setup is also possible. And of course, the VTA platform must be substituted for something else. Be it an instructor or some other software platform offering similar functionalities.

Delving deeper on the specific details, when using the RangeForce platform. Deployment starts with clean Ubuntu Linux machines deployed. From there, the router machine is the only machine where the learner does not have access to. On the router machine after booting, deploy time customizations are done. This is done, by refreshing all git repository's containing configurations and scripts. After this the initial bootstrap script is launched. With this, the web target is setup on the WWW machine, checks are deployed as continually running services on the router machine and any other lab specific functions are done.

A script utilizing the Python Faker library, is also used to randomly generate a set of red and white IP-s. Red IP-s are used for attacks and white IP-s for availability checks. These IP-s are attached to the router machine as additional interfaces. This way, the learner cannot predict what IP-s are used for which activities and makes it possible to ask questions from the learner, like which IP is conducting this or that attack.

Touching upon hardware requirements, as we consider scalability one of the key goals, the base architecture was kept as minimal as possible. Thanks to this, hardware requirements are modest. One virtual CPU per machine and 1GB of RAM per machine. Additionally, if possible having 2 virtual CPU-s for the IDS machine and adding additional RAM, might make the lab a bit smoother but should not be necessary. Given 4 machines, all with 1 GB of RAM, full memory usage per learner is 5 GB of RAM. The Estonian Information Technology College's instance of the RangeForce platform on the old server is 156GB of RAM and on the new server 512 GB of RAM, then a worst-case scenario there would be around 156 divided by 5 equalling around 31 simultaneously running labs. With the newer server, that count increases to around 100 labs.

Another topic in need of consideration in the implementation phase are the software requirements of the platform used. Although the VTA platform is used, a minimum set of defined software requirements are described for portability scenarios where other platforms might be used.

The course as previously discussed is being built on VTA. Specifically, version 2 of the platform. Based on the features needed for different parts of the course and its implementation, we will now define the software requirements for these.

The bare minimum software needed is a solution to display the lab guides to the learner. Advancing on from that, to provide scalability of the course a solution for automatic deployment of the lab infrastructure will be needed. Examples here might be cloud providers or configuration management tools like Puppet [43] or SaltStack [44]. Another aspect of this scalability is software functionality that will allow the learner to start the provisioning of the lab infrastructure to eliminate time constraints i.e. they can start the lab when it is suitable for them. To remove location constraints the software platform should provide remote access to the lab infrastructure. In VTA-s case, this is provided via a convenient web application. A bit more cumbersome alternative might be providing remote access through a VPN.

For the automatic feedback system to work, the solution used for the course guides should also include an API interface for the developed checks to use. For this, in the current course, the main API functionalities used are marking an objective done in the course guides and setting an answer for the question at the end of an objective. These covering the automatic and dynamic variants of

completing an objective. Having described the hardware and software requirements for providing this course, we can get back to specific implementation details.

All checks are deployed as services on the router machine. This ensures that they are continually running and will mark a lab objective done as soon as possible. Since the learner does not have access to the router machine, they will also be unable to figure out what and how is checked and are not able to game the system. The lab specific details tying it in with the learner's course is also placed on the router machine, making it the only machine from where lab objectives can be marked as completed. The checks utilize SSH when information needs to be gathered from a host to ensure that the lab objective is completed. Choosing a single protocol like this means, that there are fewer requirements imposed on the learner considering the lab environment. Checks are written in Python, to allow easy development and modifications. Since Python is very popular and easy to use, this also means that other people can easily modify the lab as well. The checking scripts are available at: https://github.com/haam3r/RF_IDS_Lab1_Scripts.

To give an idea of the logic and some possibilities of how the checks are implemented, we will now briefly describe the checks implemented in the first lab. All code examples given are snippets from the code developed for the course and are written in Python.

The first task of the lab is to install Suricata. Covering the learning objective that of the learner being able to deploy an IDS solution. The script running as a service will continually check if the Linux package manager has Suricata installed. An additional verification is that the package manager has a version above 4.0 installed. The checking command and evaluation logic in this case is simple. Important lines from the code are:

```
cmd = "dpkg -l | grep suricata | awk {'print $2,$3'}"
if result[0] == 'suricata' and result[1].startswith('4.'):
```

The full checking code can be found in appendix 1.

The second task revolves around configuring Suricata. To verify this learning objective, the checking code validates multiple things. Firstly, that the proper IPTables rule is in place, secondly that Suricata has the correct subnet defined as the HOME_NET parameters value. Thirdly that the Suricata daemon is running according to the system. Fourth check verifies that process listing also

has the Suricata process listed with the correct runtime arguments. Trivia piece with this check, are the square brackets surrounding the s in the beginning of the word suricata. This achieves querying the search term from the process listing without matching on the grep process itself. Ensures that if Suricata is not installed, the check would not return an exit code of 0. Final check confirms that Suricata has made a log entry of running in the correct mode and that the engine has started. These multiple checks help in avoiding corner cases and make sure that the configuration is in accordance with the given task. Full checking code included in appendix 2. The specific checks are the following:

```
'iptables -L FORWARD | grep -q "NFQUEUE num 0"; echo $?',
'suricata --dump-config | grep -qP "^vars\.address-
groups\.HOME_NET\s+=\s+\[10.10.10.0\/24\]"; echo $?',
'systemctl is-active --quiet suricata; echo $?',
'ps auxf | grep -qP "/usr/bin/[s]uricata -c
/etc/suricata/suricata.yaml --pidfile /var/run/suricata.pid
-q 0 -D -vvv"; echo $?',
'grep -iq -e "NFQ running in standard ACCEPT/DROP mode" -e
"engine started" /var/log/suricata/suricata.log; echo $?'
```

The third task asks the learner to write their first rule, which must fire against ICMP traffic destined to the internal subnet. The check verifies that the rule is placed in the correct file, on a single line and includes the necessary elements. These are the action keyword, protocol, directionality, SID field with a number chosen from the correct range and the message field. Full code in appendix 3. Relevant check command is:

```
"grep -qP
'^(alert|drop)\s+((?i)icmp)\s+\$\w+\s+[[:alnum:]]+\s+(-
>|<>)\s+\$\w+\s+[[:alnum:]]+\s+\((?=.*?msg\:\".+\"\;)(?=.*?
sid\:1\d{6}\;).*$' /etc/suricata/rules/custom.rules; echo
$?"
```

After creating their first rule, the learner is tasked to add the rule to Suricata and submit the source IP address of the generated alerts as the answer to mark the objective as completed. The code uses one of the randomly generated attacking IP addresses to do ping requests against the webserver and marks that same IP as the answer to the task. The full code for this can be found in appendix 4. Since the code is running as a service, additional logic is included here to only post the answer to the VTA API once, to not cause excessive unneeded requests against the API. Additionally, the

ping command is launched with arguments that limit packet count, so the check is faster and not too many logs are generated.

After installing and configuring Suricata, creating their first rule and using it, the final step for the learner is to convert the rule to a drop rule and verify that it is working, by viewing the logs. This check's code is in appendix 5. This check is twofold. On one side it checks that ping requests against the webserver are unsuccessful and that access to port 80 on the webserver is still available. This way of implementing the check, provides us a little more assurance that the rule is correctly designed by the learner, in that it affects only the unwanted traffic while not affecting allowed traffic. On the other side, checking that drop alerts are logged on the IDS machine. Relevant lines illustrating the checks are the following:

```
ping = subprocess.call(["ping", "-c", "1", "-I", src,
"www"])
nc = subprocess.call(["nc", "-z", "www", "80"])
cmd = 'grep -qP "Drop.+{src}" /var/log/suricata/fast.log;
echo $?'.format(src=src)
if ping != 0 and result.decode('UTF-8') == '0' and nc == 0:
```

6. Course evaluation

The final phase of the ADDIE model and instructional design in general, is the evaluation phase. It is also perhaps the most important phase, because here all the previous work is validated. In this phase, the designer makes sure, that the student analysis and course design were correct, the implementation works and most important, that the set learning objectives are valid and achieved by the course.

To conduct this evaluation, we must first define on the best course to do this. This starts with the questions of how and why. How to measure and why do I measure?

Starting with the question of how to measure. One of the simplest and most effective ways, is to test the at least a portion of the course on people matching the requirements set out in the student analysis phase. With this approach, we can measure, both time and difficulty by monitoring the test groups progression. Gathering feedback will help in determining whether any corrections in the infrastructure need to be made, if requisites for the student need to be adjusted and if the feedback automation needs adjustment. Feedback from learners participating ultimately helps to confirm that the course fulfils the learning objectives and has the intended effect.

Specifying in more detail the piloting of this course. Learners willing to participate in the piloting stage of the course, will be granted access to the first lab of the course on the RangeForce platform. There they will go through the lab. The time it takes for them to complete the lab will be measured to determine, if the lab meets the time requirements set out in the additional constraints chapter. If possible, the test groups participants will be monitored by the course designer when they are going through the lab. This will offer insight into their progression. On which objectives if any, they might struggle. Once completed they will be asked to offer feedback, what they felt was difficult, where maybe materials or guidelines where insufficient and where they feel that some topic was not covered sufficiently.

As said, the course evaluation must ultimately validate that the learning objectives are met by completing the course. To do that the learning objectives themselves must first be validated. Validated in the sense, that they cover the theme of the course to a sufficient degree. Put plainly, that the course ends up teaching what the course was designed to teach. Considering the very

complex topic the course deals with, it might be somewhat difficult to achieve a consensus on this. Asking experts in the field and comparing to already existing and similar courses however, allows us to confirm the designer's decision. In addition, it might offer additional ideas for future developments and/or adjustments to the course. Of course, with the caveat that the designer's decisions were valid.

Focusing on the scalability side, which given the current context must also be validated, the pilot group can help in this regard, both by taking the course at different times and places and by having multiple people taking it together. Thereby validating the independence of the course. From multiple people taking it together, that it scales architecturally, including the automated feedback. To also help in the scalability and validation efforts, the author hereby asks for outside help as well. With this code: IPS-ait9La-IDS, the first 100 readers can test at minimum a portion of the course themselves by signing up for the RangeForce platform at https://rangeforce.com. This code will be valid for the design and testing period of the course. After expiration, if interested to still try out the course, you can contact the author.

6.1. Pilot

For the pilot of the course, a few people matching the student requirements are asked to go through the lab, to validate the labs learning objectives, that the lab works properly and to gather feedback for fixes or enhancements needed.

Before the test group goes through the lab, they are asked, if they have ever done something similar and that they meet the requirements for the course. Main questions asked were:

- Have you installed Suricata?
- Are you familiar with IP networking and how a packet looks like?
- Are you familiar with regular expressions?
- Are you familiar with Ubuntu Linux?

This was done to ensure they match the target audience. The Suricata installation question, to understand that they fit the target profile of the first lab and the other three questions to establish that they meet the requirements set for the course. Lab completion time will be measured, to ensure

that the lab fits into the time constraints set out in the additional constraints chapter. The test groups progression through the lab is also monitored, to catch possible issues with any lab objectives, like an objectives description needing additional clarification or checks not accounting for all possible solutions. Interesting side note here is that with some testers the piloting of the first lab was done in a restaurant over the local Wi-Fi connection there. Serves as an illustrative example of the courses and platforms scalability. Specific feedback received will be covered in the next chapter.

Key takeaways from the pilot were the following:

- The first lab worked correctly. Although some minor issues with the web machine were discovered. While this does not seriously affect the first lab will become more important in the following labs and will have to be addressed regardless.
- All checks functioned properly, and no test group participants managed to confuse the checking system.
- Lab completion times averaged at around 40 minutes. Meaning that our lab fits in with the time constraints.

6.2. Feedback and corrections

During and after completing the pilot lab, the test group will be asked for feedback. The feedback will focus on understanding if the test group feel they have achieved the learning objectives of the lab, whether the guidance material was sufficient and logical. Were any of the tasks too confusing and if the automatic checks require refinement.

The other side is gathering telemetry from the platform. How long it takes to complete the lab i.e. does it meet the length requirements, how long does each objective take to glean whether it requires any fine tuning. Of course, some complications must be considered here. For example, if the learner takes a break during the lab, that will affect the overall completion time and therefore will skew the telemetry. To account for this during the pilot, with at least some pilot participants the author will monitor their progression in person.

Moving on to specific feedback gathered during the pilot. First off, from the telemetry side, the testers took an average of 40 minutes to complete the lab, thereby the first lab is suitably designed regarding the time constraints.

Everybody in the test group seemed to be happy with the general flow of the lab and structuring. One participant suggested to change the last drop rule objective to involve creating a completely new rule instead of changing the previous one. This would help clarify the structure of the rule files. One other suggestion was to describe how custom rules should be managed and backed up. For example, placing them under a version control solution such as git. This will be a topic in the second lab.

All the checks developed for the course worked. No pilot participants completed an objective in a way that a check failed to mark an objective as complete. Regarding the checks, one piece of feedback received was to enhance the regex checking of the first rule the learner creates. This will be considered, but the caveat with this being that the regex would have to cover many variations, thereby introducing more chances of failure to the check.

Rest of the feedback centred around the contents of the guide. Either typos discovered in the text or places where clarifications to objectives or background materials should be added. Some examples of these were:

- Clarifications should be added explaining what is nfqueue [2]
- Some commands given were missing the sudo statement in front of them
- Why should a USR2 signal be sent to the Suricata process instead of simply restarting when doing rule reloads
- Explanation should be added why the msg field in a Suricata rule is important

7. Future work

Having designed the course and piloted the first lab, by no means indicates that the course is done from the designer's perspective. Ongoing is the development of the following labs described in the course design. Each of these will need implementing, testing and piloting. Piloting each of them to validate for each lab that the learning objectives are met. Also, that they conform to the constraints set out in the analysis phase. Feedback from testers is taken into consideration and if necessary corrections are made to the lab.

Learning objectives were defined in the analysis chapter, but this list can be expanded in the future to cover additional requests. New labs can be developed for more advanced topics and more complex scenarios. Some good examples here might be labs covering different or more complex run modes of Suricata and labs covering network traffic and log analysis of Windows machines. Labs can be added to cover different tools. Future labs are also good candidates for creation by other people with the help and supervision of the author.

Although the checks code developed during the implementation of this course works as intended an additional idea for future work, especially as the course grows might be to restructure and enhance the codebase a bit. Some possible ideas here are to streamline the code used for deployment, with the possible addition of configuration management tools. Refactoring the code used for automatic feedback. This to reduce the current code reuse and to make it even more modular and portable.

Cyber security is a rapidly evolving topic therefore, the labs will need continuous updating and refinement in the future to respond to developments in the field.

8. Summary

The outcome of this thesis is the analysis, design and proof of concept implementation of a scalable course in cyber attack detection. This was done to address the problem of almost no available courses teaching complex cyber security topics, that are free of traditional time and location constraints and that impose limits to the number of simultaneous learners.

One of the main causes for needing a course such as this being a lack of time on both the learner and instructor side. Especially when talking about already employed professionals. From a second perspective, advancements made in computing technology in recent years, has made it possible to offer more dynamic and larger infrastructures to allow developing a course such as this one.

Aim of the thesis was to design and implement a course on cyber attack detection, that fulfils the following requirements:

- Free of time and location constraints on the learner
- Scales to any reasonable number of simultaneous learners depending on the available computing resources
- Teaches cyber attack detection through automated hands-on labs

A brief overview of related work was given. Mainly covering other courses developed on similar topics and a description of the Virtual Teaching Assistant that is a core component of the RangeForce platform. Secondly a general overview of cyber attacks and cyber attack detection was given, focusing specifically on intrusion detection systems to set the background of the course topic. To better explain the methodology of the design and implementation process, an overview of instructional design and the ADDIE model was given.

Following instructional design technology using the ADDIE model as our chosen methodology, the course constraints were analysed. The student pre-requirements were decided. A core set of learning objectives were developed. Based on the learning objectives, we selected software components to use and designed the learning process. Then the focus was turned on developing a scalable architecture that would fulfil our initial requirements. After the design phase, the first hands-on lab in the course was implemented as a proof of concept. This lab was tested on a pilot group to verify that the learning objectives and constraints were met. Based on the pilot, feedback for improvements and corrections was compiled and analysed.

Finally, future work on the course was discussed. First and foremost, of them the implementation and piloting of the other planned hands-on labs in the course. Not to mention the need for continuous updating of the course in the future.

In the end a course teaching cyber attack detection was designed, implemented and successfully piloted. The course was verified to be free of time and location constraints, that the deployment and feedback automation works well and that it scales well with the growth of simultaneous learners.

Mastabeeritav kursus küberrünnete tuvastamisest

Magistritöö (30 EAP)

Andres Elliku

Kokkuvõte

Käesoleva magistritöö väljundiks on küberrünnete tuvastamist käsitleva mastabeeritava kursuse analüüs, disain ja kontseptsiooni tõendus. Töös lahendatud probleem väljendus saadaolevate kursuste puudmist, kus õpetatakse keerukaid küberkaitsega seotud temaatikaid, mis oleksid vabad traditsioonilistest aja ja asukoha piirangutest, mis ühtlasi ei piiraks ka üheaegselt kursust läbivate õppurite arvu.

Peamisi põhjuseid selleks on aja puudus nii õppuri kui ja juhendaja poolt. Eriti väljendub see juba töötavate spetsialistide puhul. Teine põhjus on viimastel aastatel toimunud tehnoloogilised arengud, mille läbi on realiseerunud võimalus pakkuda palju suuremaid ja dünaamilisemaid taristuid kõnealuste kursuste arendamiseks.

Peamisteks eesmärkideks oli arendada ja teha kontseptsiooni tõendus küberrünnete tuvastamist käsitlevast kursusest, mis vastaks järgnevatele kriteeriumitele:

- Õppuri poolne vabadus aja ja koha suhtes
- Sõltuvalt saadaolevast arvutusressursist, mõistlikkuse piires mastabeeruv kursus
- Kursus õpetaks küberrünnete tuvastust kasutades automatiseeritud praktilisi laboreid

Töö käigus anti lühike ülevaade seonduvatest töödest. Käsitledes peamiselt sarnaseid kursuseid ning Virtual Teaching Assistant nimelist lahendust. Antud lahendus on RangeForce platvormi tuumik komponent. Järgnevalt anti ülevaade küberrünnetest ning küberrünnete tuvastamisest, koos spetsiifilise fookusega sissetungi tuvastuse süsteemide osas. Seeläbi pakkudes taustinfot magistritöö sisule. Paremini selgitamaks arenduse ajal kasutatavaid metodoloogiad anti ülevaade *instructional design* tehnoloogiast ning täpsemalt ADDIE mudelist.

Vastavalt ADDIE mudelile, uuriti kursusele ja õppurile rakenduvaid piiranguid. Arendati välja peamised õpiobjektid, arendati õpiprotsess ning valiti välja sobilikud tehnoloogiad. Edasi arendati välja mastabeeritav taristu mis vastaks algsetele nõuetele. Pärast analüüsi ja disaini faase teostati kursuse esimese laboriga kontseptsiooni tõendus. Teostatud laborit testiti piloot grupiga, kinnitamaks õpiobjektidele ja piirangutele vastavust. Piloodi tulemusi analüüsiti paranduste ja edasiarenduste tegemiseks.

Viimasena toodi välja järgnevad tegevused kursuse tuleviku osas. Peamisena neist kursuse ülejäänud laborite teostust ja testimist. Samuti vajadusest kursust pidevalt hooldada ja uuendada.

Lõpptulemusena disainiti, teostati ja testiti kursust, mis käsitleb küberrünnete tuvastamist. Läbi testimise tõestati, et kursuse läbimine ei ole piiratud aja ega kohaga. Kursuse laborite juurutus ning tagasiside automatiseerimine töötavad ning kursuse mastabeeritavus samaaegsete õppurite osas on tagatud.

List of References

[1] M. D. Merrill, L. Drake, M. J. Lacy, J. Pratt and t. I. R. Group, “Reclaiming Instructional Design,” *Educational Technology 1966, 36 (5), 5-7,* 1966.

[2] libnetfilter_queue, “libnetfilter_queue Documentation,” [Online]. Available: https://www.netfilter.org/projects/libnetfilter_queue/doxygen/index.html. [Accessed 22 April 2018].

[3] T. C. Miller, “Sudo in a Nutshell,” 2018. [Online]. Available: https://www.sudo.ws/intro.html. [Accessed 23 April 2018].

[4] B. Spitzberg, “Intrusion Prevention – Part of Your Defense in Depth Architecture?,” SANS Institute, 2003.

[5] FireEye, “M-Trends 2017,” 14 March 2017. [Online]. Available: https://www.fireeye.com/blog/threat-research/2017/03/m-trends-2017.html.

[6] B. K. Fite, “Simulating Cyber Operations: A Cyber Security Training Framework,” 2014. [Online]. Available: https://www.sans.org/reading-room/whitepapers/bestprac/simulating-cyber-operations-cyber-security-training-framework-34510.

[7] National Institute of Standards and Technology, “Glossary of Key Information Security,” National Institute of Standards and Technology, Gaithersburg, 2013.

[8] S. Singh and S. Silakari, “An Ensemble Approach for Cyber Attack Detection System: A Generic Framework,” *International Journal of Networked and Distributed Computing,* vol. II, no. 2, pp. 78-90, 2014.

[9] SANS Institute, “Host- vs. Network-Based Intrusion Detection Systems”.

[10] AlienVault Inc., “AlienVault OSSIM,” [Online]. Available: https://www.alienvault.com/products/ossim. [Accessed 22 04 2018].

[11] R. Vaarandi, “SEC - simple event correlator,” [Online]. Available: https://simple-evcorr.github.io/. [Accessed 21 April 2018].

[12] Adiscon GmbH, “RSYSLOG HOME,” [Online]. Available: https://www.rsyslog.com/. [Accessed 21 April 2018].

[13] The Bro Project, “NetControl Framework,” 03 April 2018. [Online]. Available: https://www.bro.org/sphinx/frameworks/netcontrol.html.

[14] Open Information Security Foundation, “Suricata Features,” [Online]. Available: https://suricata-ids.org/features/. [Accessed 15 April 2018].

[15] Cisco, “Snort FAQ,” 2018. [Online]. Available: https://www.snort.org/faq/does-cisco-sell-snort.

[16] A. Maqousi, T. Balikhina and M. Mackay, “AN EFFECTIVE METHOD FOR INFORMATION SECURITY AWARENESS RAISING INITIATIVES,” *International Journal of Computer Science & Information Technology,* vol. V, no. 2, pp. 63-72, 2013.

[17] The Bro Project, “The Bro Network Security Monitor,” 2014. [Online]. Available: https://www.bro.org/. [Accessed 21 April 2018].

[18] Open Information Security Foundation , “Lua Scripting,” 28 September 2016. [Online]. Available: http://suricata.readthedocs.io/en/suricata-4.0.4/rules/rule-lua-scripting.html.

[19] Cisco, “Does Cisco sell Snort?,” 2018. [Online]. Available: https://www.snort.org/faq/does-cisco-sell-snort.

[20] OWASP, “Intrusion Detection,” 27 March 2017. [Online]. Available: https://www.owasp.org/index.php/Intrusion_Detection. [Accessed 21 April 2018].

[21] N. Gupta, K. Srivastava and A. Sharma, “Reducing False Positive in Intrusion Detection System: A Survey,” *International Journal of Computer Science and Information Technologies,* vol. VII, no. 3, pp. 1600-1603, 2016.

[22] Microsoft, “Azure Advanced Threat Detection,” 21 November 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/security/azure-threat-detection. [Accessed 21 April 2018].

[23] Moloch, “Home,” [Online]. Available: https://molo.ch/. [Accessed 6 May 2018].

[24] CERT-EE, “S4A Documentation,” [Online]. Available: https://docs.s4a.cert.ee/index.html. [Accessed 21 April 2018].

[25] M. O'Leary, “A laboratory based capstone course in computer security for undergraduates,” in *SIGCSE '06 Proceedings of the 37th SIGCSE technical symposium on Computer science education*, Houston, 2006.

[26] M. O'Leary, “Innovative Pedagogical Approaches to a Capstone Laboratory Course in Cyber Operations,” in *SIGCSE'17*, Seattle, 2017.

[27] N. Evans, B. Blakely and D. Jacobson, “A security capstone course: An innovative practical approach to distance education,” in *Frontiers in Education Conference, 2009. FIE '09. 39th IEEE*, San Antonio, 2009.

[28] “Network Reconnaissance, Attack, and Defense Laboratories for an Introductory Cyber-Security Course,” *ACM Inroads,* vol. IV, no. 3, pp. 52-64, 2013.

[29] L. Catuogno and A. D. Santis, “An internet role-game for the laboratory of network security course,” in *ITiCSE '08 Proceedings of the 13th annual conference on Innovation and technology in computer science education*, Madrid, 2008.

[30] CTFd LLC, “CTFd,” 2017. [Online]. Available: https://ctfd.io/. [Accessed 15 04 2018].

[31] Moodle Pty Ltd, “Explore some of Moodle’s popular features,” [Online]. Available: https://moodle.com/features/. [Accessed 23 April 2018].

[32] R. Culatta, “ADDIE Model,” 2018. [Online]. Available: http://instructionaldesign.org/models/addie/.

[33] A. Elliku, “Creation of Online Course Materials on Intrusion Detection System for Estonian Information Technology College,” Tallinn, 2014.

[34] R. Juhanni, “Online Course Materials on Log Event Visualization for Estonian Information Technology College,” Tallinn, 2015.

[35] “Welcome to the Emerging Threats rule server,” [Online]. Available: https://rules.emergingthreats.net/. [Accessed 22 April 2018].

[36] Cisco, “Talos,” [Online]. Available: https://www.snort.org/talos. [Accessed 22 April 2018].

[37] Elasticsearch, “Kibana,” [Online]. Available: https://www.elastic.co/products/kibana. [Accessed 6 May 2018].

[38] J. Ish, “Evebox,” [Online]. Available: https://github.com/jasonish/evebox. [Accessed 6 May 2018].

[39] Elasticsearch, “Elasticsearch,” [Online]. Available: https://github.com/jasonish/evebox. [Accessed 6 May 2018].

[40] OISF, “Differences From Snort,” [Online]. Available: http://suricata.readthedocs.io/en/latest/rules/differences-from-snort.html. [Accessed 7 May 2018].

[41] OISF, “Rule Management with Suricata-Update,” [Online]. Available: http://suricata.readthedocs.io/en/suricata-4.0.4/rule-management/suricata-update.html. [Accessed 21 April 2018].

[42] Balabit Corp, “syslog-ng,” [Online]. Available: https://syslog-ng.com/log-management-software. [Accessed 22 April 2018].

[43] Puppet, “Puppet,” [Online]. Available: https://puppet.com/. [Accessed 22 April 2018].

[44] SaltStack, “SaltStack,” [Online]. Available: https://saltstack.com/. [Accessed 22 April 2018].

Appendix 1 – Check if Suricata is installed

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import logging
import subprocess
import sys

from objectiveschecks import check

logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s %(levelname)s
%(message)s',
                    filename='/root/running/checks.log',
filemode='a')

def main():
    '''
    Check if Suricata is installed and at least version 4.x
    '''


    vta_step = 'step-rx10'
    host = 'ids'
    cmd = "dpkg -l | grep suricata | awk {'print $2,$3'}"
    logging.debug('Starting check {}: Suricata installed
and version >= 4'.format(vta_step))
    ssh = subprocess.Popen(["ssh", "-o
StrictHostKeyChecking=no", host, cmd],
                           shell=False,
                           stdout=subprocess.PIPE,
                           stderr=subprocess.PIPE)
    try:
        result = ssh.stdout.readlines()[0].decode('UTF-
8').split(" ")
    except IndexError:
        logging.warning('Suricata not installed on host
{host}'.format(host=host))
        sys.exit(1)


    logging.debug('{0} is version {1}'.format(result[0],
result[1]))
```

```
    if result[0] == 'suricata' and
result[1].startswith('4.'):
        logging.info('Correct version of Suricata installed
on {host}'.format(host=host))
        post = check(vta_step, True)
        if post is False:
            logging.error('Setting objective {} completed
has failed'.format(vta_step))
            sys.exit(1)
        else:
            logging.info('Successfully set {step} as
completed'.format(step=vta_step))
    else:
        logging.info('Check failed on {}'.format(vta_step))
        sys.exit(1)

if __name__ == '__main__':
    main()
```

Appendix 2 – Check that Suricata is configured correctly

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import logging
import subprocess
import sys

from objectiveschecks import check

logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s %(levelname)s
%(message)s',
                    filename='/root/running/checks.log',
filemode='a')

def main():
    '''
    Check if Suricata is configured properly and running
    '''


    vta_step = 'step-rjo1'
    host = 'ids'
    cmds = [
        r"iptables -L FORWARD | grep -q 'NFQUEUE num 0';
echo $?",
        r"suricata --dump-config | grep -qP
'^vars\.address-groups\.HOME_NET\s+=\s+\[10.10.10.0\/24\]';
echo $?",
        r"systemctl is-active --quiet suricata; echo $?",
        r"ps auxf | grep -qP '/usr/bin/[s]uricata -c
/etc/suricata/suricata.yaml --pidfile /var/run/suricata.pid
-q 0 -D -vvv'; echo $?",
        r"grep -iq -e 'NFQ running in standard ACCEPT/DROP
mode' -e 'engine started' /var/log/suricata/suricata.log;
echo $?"
    ]
    success = 0


    logging.debug('Starting check {}'.format(vta_step))
    for cmd in cmds:
```

```
        ssh = subprocess.Popen(["ssh", "-o
StrictHostKeyChecking=no", host, cmd],
                               shell=False,
                               stdout=subprocess.PIPE,
                               stderr=subprocess.PIPE)
        try:
            result = ssh.stdout.readlines()[0].rstrip()
            logging.debug('Command {cmd} got result
{result}'.format(cmd=cmd, result=result))
            if result.decode('UTF-8') == '0':
                success += 1
        except IndexError:
            logging.warning('Suricata conf check failed for
cmd {cmd}'.format(cmd=cmd))
            sys.exit(1)


    if success == len(cmds):
        logging.info('All {nr} configuration checks passed
for {step}'.format(nr=len(cmds), step=vta_step))
        post = check(vta_step, True)
        if post is False:
            logging.error('Setting objective {} completed
has failed'.format(vta_step))
            sys.exit(1)
        else:
            logging.info('Successfully set {step} as
completed'.format(step=vta_step))
    else:
        logging.error('1 or more checks failed for
{step}'.format(step=vta_step))
        sys.exit(1)


if __name__ == '__main__':
    main()
```

Appendix 3 – Check for new rule with required keywords

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import logging
import subprocess
import sys

from objectiveschecks import check

logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s %(levelname)s
%(message)s',
                    filename='/root/running/checks.log',
filemode='a')

def main():
    '''
    Check if new rule is in custom.rules
    '''


    logging.debug('Starting check for step-3zc0')
    vta_step = 'step-3zc0'
    host = 'ids'
    cmds = [
        r"grep -qP
'^(alert|drop)\s+((?i)icmp)\s+\$\w+\s+[[:alnum:]]+\s+(-
>|<>)\s+\$\w+\s+[[:alnum:]]+\s+\((?=.*?msg\:\".+\"\;)(?=.*?
sid\:1\d{6}\;).*$' /etc/suricata/rules/custom.rules; echo
$?"
    ]
    success = 0


    for cmd in cmds:
        ssh = subprocess.Popen(["ssh", "-o
StrictHostKeyChecking=no", host, cmd],
                               shell=False,
                               stdout=subprocess.PIPE,
                               stderr=subprocess.PIPE)
        try:
            result = ssh.stdout.readlines()[0].rstrip()
```

```
            logging.debug('{cmd} got
{result}'.format(cmd=cmd, result=result))
            if result.decode('UTF-8') == '0':
                success += 1
        except IndexError as err:
            logging.error('Not okay: {}'.format(err))
            sys.exit(1)

    if success == len(cmds):
        logging.info('Marking {step}
done'.format(step=vta_step))
        post = check(vta_step, True)
        if post is False:
            logging.error('Setting {} completed has
failed'.format(vta_step))
            sys.exit(1)
        else:
            logging.info('Set {step} as
completed'.format(step=vta_step))
    else:
        logging.info('{step} failed'.format(step=vta_step))
        sys.exit(1)

if __name__ == '__main__':
    main()
```

Appendix 4 – Alert rule implemented

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import configparser
import logging
import subprocess
import sys

import requests
from requests.adapters import HTTPAdapter
from urllib3.util import Retry

logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s %(levelname)s
%(message)s',
                    filename='/root/running/checks.log',
filemode='a')

def main():
    '''
    Check if new rule fires
    '''


    vta_step = 'step-m071'
    config_filename = '/root/running/lab.ini'
    logging.debug('Reading:
{file}'.format(file=config_filename))
    config = configparser.ConfigParser()
    config.read(config_filename)


    try:
        ta_key = config.get('LAB', 'ta_key')
        vta_host = config.get('LAB', 'virtualta_hostname')
        lab_id = config.get('LAB', 'lab_id')
        uid = config.get('LAB', 'uid')
    except Exception:
        logging.error(" Exception: Parsing INI file
failed")
        sys.exit(1)
```

```
    logging.debug('Starting check for step-m071')
    with open("/root/running/red_ips.txt") as f:
        data = f.readlines()
    src = data[-1].rstrip()
    qa = list()
    qa.append(src)

    ping = subprocess.call(["ping", "-c", "1", "-I", src,
"www"])

    logging.debug('Exit code was {}'.format(ping))

    # curl -H "Content-Type: application/json"
    # -X PUT
https://portal2.rangeforce.com/api/v2/labuser_form/
    # -d '{"api_key":"<API key here>",
    # "labID":"<LAB ID here>","user":"<USER ID here>",
    # "qname":"question-qv7", "expected":["188.8.21.21"]}'
    payload = {
        "api_key": ta_key,
        "labID": lab_id,
        "user": uid,
        "qname": "question-qv7",
        "expected": qa
        }

    # Retry mechanism from
    #  https://www.peterbe.com/plog/best-practice-with-
retries-with-requests
    #  https://stackoverflow.com/questions/15431044/can-i-
set-max-retries-for-requests-request
    s = requests.Session()
    retries = Retry(total=5,
                    backoff_factor=0.1,
                    status_forcelist=[500, 502, 503, 504])

    s.mount('http://', HTTPAdapter(max_retries=retries))
    s.mount('https://', HTTPAdapter(max_retries=retries))

    # Send Obj data to VTA
    logging.debug(payload)
```

```
    try:
        file = open("/root/running/step-m071.txt", 'r')
        file.close()
        logging.debug("{step}: File found,
stopping".format(step=vta_step))
    except FileNotFoundError:
        r = s.put(vta_host + '/api/v2/labuser_form/',
json=payload)
        logging.debug("{step}: VTA response:
{r}".format(step=vta_step, r=r))
        with open("/root/running/step-m071.txt", 'w') as
m071:
            if r.status_code == requests.codes['ok']:
                m071.write(qa[0])
                logging.debug("step-m071: Wrote {qa} to
file".format(qa=qa))

if __name__ == '__main__':
    main()
```

Appendix 5 – Check for verifying correct drop rule implementation

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import logging
import subprocess
import sys

from objectiveschecks import check

logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s %(levelname)s
%(message)s',
                    filename='/root/running/checks.log',
filemode='a')

def main():
    '''
    Check if drop rule works
    '''


    vta_step = 'step-5v02'
    host = 'ids'
    with open("/root/running/red_ips.txt") as f:
        data = f.readlines()
    src = data[-1].rstrip()


    logging.debug('Starting check for
{step}'.format(step=vta_step))


    ping = subprocess.call(["ping", "-c", "1", "-I", src,
"www"])
    nc = subprocess.call(["nc", "-z", "www", "80"])
    logging.debug('Ping response code was {}'.format(ping))
    logging.debug('NC code was {}'.format(nc))


    cmd = 'grep -qP "Drop.+{src}"
/var/log/suricata/fast.log; echo $?'.format(src=src)
    ssh = subprocess.Popen(["ssh", "-o
StrictHostKeyChecking=no", host, cmd],
                           shell=False,
```

```
                            stdout=subprocess.PIPE,
                            stderr=subprocess.PIPE)
    try:
        result = ssh.stdout.readlines()[0].rstrip()
        print(result)
    except IndexError:
        logging.error('Grep for step {}
failed'.format(vta_step))
        sys.exit(1)


    if ping != 0 and result.decode('UTF-8') == '0' and nc
== 0:
        logging.debug("Marking {step} as
successful".format(step=vta_step))
        post = check(vta_step, True)
        if post is False:
            logging.error('Setting {} completed has
failed'.format(vta_step))
            sys.exit(1)
        else:
            logging.info('Set {step} as
completed'.format(step=vta_step))
    else:
        logging.info('{step} check
failed'.format(step=vta_step))
        sys.exit(1)


if __name__ == '__main__':
    main()
```

Appendix 6 – Function to mark objectives complete in VTA

```
#!/usr/bin/env python3

import configparser
import logging
import sys

import requests
import urllib3
from requests.adapters import HTTPAdapter
from urllib3.util import Retry

__author__ = 'Margus Ernits, Erki Naumanis'

logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s %(levelname)s
%(message)s',
                    filename='/root/running/checks.log',
filemode='a')



def check(step, objective):
    '''
        Mark an objective done or failed in VTA
    '''

urllib3.disable_warnings(urllib3.exceptions.InsecureRequest
Warning)

    config_filename = '/root/running/lab.ini'
    logging.debug('Reading configuration from:
{}'.format(config_filename))
    config = configparser.ConfigParser()
    config.read(config_filename)


    try:
        ta_key = config.get('LAB', 'ta_key')
        vta_host = config.get('LAB', 'virtualta_hostname')
        lab_id = config.get('LAB', 'lab_id')
        uid = config.get('LAB', 'uid')
    except Exception:
```

```
            logging.error("Exception: Failed parsing INI file")
            sys.exit(1)

        # example
        # curl -H "Content-Type: application/json"
        # -X PUT http://localhost:3013/api/v2/labuser_any/
        # -d '{"api_key":"<API key here>",
        # "labID":"<LAB ID here>", "userID":"<USER ID here>",
        # "oname":"asd", "score":100, "inc":true}'
        payload = {
            "api_key": ta_key,
            "labID": lab_id,
            "user": uid,
            "oname": step,
            "done": objective
            }

        # Retry mechanism from
        #  https://www.peterbe.com/plog/best-practice-with-
retries-with-requests
        #  https://stackoverflow.com/questions/15431044/can-i-
set-max-retries-for-requests-request
        s = requests.Session()
        retries = Retry(total=5,
                        backoff_factor=0.1,
                        status_forcelist=[500, 502, 503, 504])
        s.mount('http://', HTTPAdapter(max_retries=retries))
        s.mount('https://', HTTPAdapter(max_retries=retries))

        # Send Obj data to VTA
        logging.debug(payload)
        r = s.put(vta_host + '/api/v2/labuser_any',
json=payload, verify=False)
        if r.status_code == requests.codes['ok']:
            return True
        else:
            return False
```