Evaluating the response effectiveness of XDR technology in a scaled down environment

Olteanu, Ioana-Cristina

Award date:
2022

Eindhoven University of Technology
Department of Mathematics and Computer Science
Security Group

Evaluating the response effectiveness of XDR technology in a scaled down environment

Ioana-Cristina Olteanu

Supervisors:
Emmanuele Zambon (TU/e)
Rob Wiegertjes (KPMG)

Eindhoven, December 2022

Abstract

Extended Detection and Response solutions have emerged in recent years, and they promise to identify threats more efficiently than previous intrusion detection and response solutions. They combine existing technologies into a single one to provide broader infrastructure coverage, and they use machine learning and behavior analysis capabilities for threat detection and correlation. However, being relatively new solutions on the market, there is no public evaluation available for these solutions to assess whether their described capabilities are as good as advertised, especially the response ones. Hence, an organization might wonder whether it is worth investing in such a solution and, if so, which solution is better suited for them.

In this paper, we define a framework to test the response efficiency of XDR solutions, which considers testing it in a small-scaled environment. We propose four capabilities that our framework should support testing and suitable metrics to evaluate these capabilities. Moreover, we account for the bias created by testing the solution in a small scaled environment and propose a set of guidelines for building the testing environment and the attack dataset. These guidelines should minimize this bias and provide a consistent testing process. After defining the framework, we showcase an application of it, where we compare two XDR solutions. Finally, we present the computed results and the process of selecting the best-performing solution. By presenting an application of the framework, we show that it is feasible, especially for an organization with more resources than us. Finally, we argue its usefulness by showcasing its differences from existing solutions and its applicability, and by gathering experts' opinions on it.

Table of Contents

List of Tables

1. Introduction

Extended Detection and Response (XDR) represents a novel approach for threat detection and response that promises to identify highly sophisticated or hidden threats more efficiently and effectively than previous intrusion detection solutions. XDR products combine existing intrusion detection and response solutions into one system, as well as Machine Learning capabilities for analysis, correlation, and decision making. These features aim to facilitate faster detection so that analysts can respond quickly before the threat causes damage.

Numerous vendors created their XDR products by combining existing technologies and adding their features. Therefore, multiple products on the market promise to protect from cyberattacks. However, except for the vendors' promises, there is no structured evaluation available of how effective these tools are, especially concerning their response and automated features. The only information available is from websites that rank these products, but their testing methodology is not transparent. Moreover, the definition of XDR is still vague, as there is no unified definition to which all the products adhere. Some vendors created their definition, which characterizes their product. Other vendors referenced the definitions from market guides, such as Gartner [17], who provide a more consistent definition across products but still not one which we consider sufficient. Hence, a universal definition and a method of testing the performance of the XDR solution are needed to know whether these solutions can become helpful for an organization.

An evaluation of the performance of XDR solutions is useful for an organization that wants to invest in a security solution that claims to offer better protection compared to existing solutions. However, evaluating and testing an XDR solution cannot be done at the scale of an organization. As testing a solution requires launching attacks and observing how the tool handles them, an organization would not want to use its infrastructure for this purpose. Therefore, the testing is usually done in an isolated laboratory setup, which can be modeled according to the testing needs. However, such a setup is usually scaled down, as not many devices can be created, due to memory constraints, for instance. Furthermore, the devices in a lab setup do not mimic the same conditions as the ones in a real environment, as they are usually created from scratch, and they miss existing data, background traffic, and the long-time general usage of such a device. Hence, the performance of the XDR solutions under test in such an environment can be affected by this aspect.

Several frameworks for evaluating intrusion detection systems are available, but their focus lies on assessing their detection capabilities [5, 71, 13], compared to the response capabilities. Moreover, testing them has usually been done using publicly available datasets, such as DARPA [26], which we argue may not be suitable for a newer framework in this paper. Finally, all these frameworks do not consider the medium in which the intrusion detection solutions are tested. As we previously stated, evaluating the solution in a small-scaled environment compared to a real environment can create some bias in the results.

This paper proposes a framework for evaluating an XDR solution's response capabilities, which considers testing it in a small-scaled isolated environment. To do this, we first identify the main response capabilities of these solutions and provide a standard definition that comprises all of them. We continue by analyzing the metrics used for measuring the performance of intrusion detection solutions, security orchestration solutions, and incident management. We propose a new set of metrics that evaluates the identified response capabilities using these metrics as a base. We then investigate the possible sources of bias that can appear when testing the solution. After defining all these aspects, we define our framework by presenting guidelines for capability selection, benchmarking metrics selection, test environment setup, and attack dataset selection.

We start this paper by presenting the related work and the background information needed to understand our work comprehensively. In the background section, we also define XDR solutions, which should comprise all analyzed solutions. Next, we present the methodology, which consists of selecting the XDR tools capabilities, the capability benchmarking metrics, and the sources of bias in capability benchmarking experiments. We then present our framework and a practical application of it. Finally, we provide a discussion about our findings and present our conclusions.

1.1. Research questions

Our paper focus on answering the following research question, as well as the following sub-questions:

RQ: How is it possible to estimate the incident response capability an XDR solution would guarantee in a production environment by testing the solution in a scaled-down environment?

SRQ1: What suitable metrics can be used for evaluating the response capabilities of XDR solutions?

SRQ2: Which are the factors of the evaluation environment that can create a bias in the results of the evaluation of XDR solutions?

To answer the first sub-question, we first identify the capabilities that the framework should support testing. Then, based on these capabilities, we define suitable metrics for measuring them. In the framework, we propose guidelines for selecting the capabilities and the metrics. To answer the second sub-question, we collect and analyze the sources of bias affecting the metrics and the testing process. Using our findings, we propose guidelines for mitigating the bias, which are also part of the framework. Finally, we answer the main research question by answering these sub-questions and proposing the framework.

2. Related Work

2.1. Overview

The concept of Extended Detection and Response has not yet been tackled in the scientific literature. However, it can be considered an Intrusion Detection and Response solution. While the concept of Intrusion Response Systems is briefly tackled in scientific literature, Intrusion Detection Systems and their evaluation is a continuous research topic. Intrusion Detection Systems are continuously improving, and methods for evaluating and generating their best performance are frequently proposed. Furthermore, XDR represents an extension of Endpoint Detection and Response, and includes capabilities typical of Security Orchestration Automation and Response solutions. Therefore, we extend the scope of our analysis to evaluation frameworks for IDS, EDR and SOAR solutions.

We could group the related literature according to four main research directions, all of which are related to our work: (1) methodologies for testing security solutions, (2) metrics for evaluating security solutions, (3) sources of bias in the evaluation of security solutions and (4) datasets for testing security solutions. We now discuss each group in a separate subsection, and provide a summary of our original contribution at the end.

2.2. Methodology for testing security solutions

Different frameworks for evaluating intrusion detection systems (IDS) have been proposed in the scientific literature. Milenkoski et al. [31] present an overview of the methods used in the scientific literature for evaluating IDSs. This overview consists of metrics for evaluating the performance of IDSs, methods for generating and obtaining data sets for testing the solution, types of data sets, and techniques for measuring the tool's performance. The authors provide a comprehensive view of existing options for evaluating intrusion detection systems. We use their comparison of the different types of workloads to decide which one is the most suitable for our framework and their presentation of the metrics to contribute to our analysis of the advantages and disadvantages of existing metrics.

Considering the available frameworks for testing IDSs, we use the one by Cardenas et al. [5] as the basis for our methodology. Their paper analyzes metrics that can be used and provides their advantages and disadvantages. The framework proposes an optimization function calculating a trade-off among multiple metrics, including the Intrusion Detection Capability, the Positive Predictive Value, and the Negative Predictive Value. We use parts of their methodology, such as the analysis of metrics, using an attack model, and using multiple metrics to evaluate the solution, as inspiration for the framework defined in this paper. The metrics presented are also analyzed in our paper, especially the proposed method of computing the expected cost. The paper by Cardenas et al. [5] is relevant to our work because the approach and methodology for defining a framework for testing are similar to ours. We compare this to the papers of other authors, such as Gu et al. [71], Ficke et al. [1], Gaffney and Ulvila [16], and Elhamahmy et al. [13], which evaluate IDSs using other approaches, such as information theory or decision theory. We did not consider these approaches as we want to evaluate XDR solutions from an organization's perspective. For example, an organization would not benefit from metrics measuring the information [71] or from finding the optimal configuration of the solution [16]. For an organization, it is more interesting to observe whether the solution is cost-effective and provides adequate detection and response capabilities.

2.3. Metrics

Metrics for evaluating Intrusion Detection Systems and other security solutions have been proposed and tackled multiple times in the scientific literature. As previously mentioned, Milenkoski et al. [31] provide multiple metrics we can use as a starting point in the evaluation we carry out in this paper. However, we analyze some of these metrics in other papers presenting classification metrics to understand their purpose better. For example, we look at the papers by Gu. et al. [18], which explains the Intrusion Detection Capability, and the one by Axelsson [3], which presents the Bayesian Detection rate, as well as some problems encountered with evaluating intrusion detection systems. On the other hand, the basic classification metrics, such as the False Positive Rate or Detection Rate, do not require further investigation, as they are widely encountered in classification problems and are easily computed.

The metrics used for intrusion detection systems analyzed from the literature are usually used for evaluating the detection capabilities of these solutions. For this paper, they represent a basis for creating new similar metrics, which have the scope of measuring the capabilities of XDR solutions that influence the response.

We also use the paper by Islam et al. [19] to analyze metrics for evaluating SOAR solutions due to the similarity of XDR and SOAR. Their paper, which offers an overview of Security Orchestration tools, presents the quality attributes of these tools. However, they lack methods for measuring these attributes. We analyze these attributes and the corresponding metrics in our work and use some of them as a basis for the metrics used in our framework. These metrics have a broader focus than those used for IDSs. They also measure aspects such as the mean time to notification, remediation, and investigation and the time needed to analyze an attack. We use an adapted version of these metrics in our work and provide a method for computing them, which is not available in the original paper [19].

2.4. Sources of bias

To the best of our knowledge, the concept of bias created by testing environments for intrusion detection systems has not been extensively tackled in the scientific literature. However, Mell et al. [28] offer an overview of issues in testing intrusion detection systems. They offer insight into the ways the results of the evaluation can be affected by different environment configurations or settings, as well as general problems of the intrusion detection systems. However, their approach only concerns IDSs. Some of the presented problems do not apply to our work, such as the different requirements for testing signatures based vs. anomaly-based solutions, as an XDR comprises both. Nevertheless, some of the presented aspects, such as the use of background traffic and the difficulties in collecting attack scripts, represent a starting point for defining the sources of bias in our work.

To define the sources of bias, we analyze papers that identify biases in malware classification, benchmarking, and evaluation using machine learning. Pendlebury et al. [36] present two sources of bias encountered in the classification of Android malware: temporal and spatial. The temporal bias appears when a type of malware is classified by training the algorithm with newer malware. The spatial bias occurs when the percentage of malware in the testing and training datasets is not representative of an accurate world ratio between malware and goodware. Even if the temporal bias cannot occur when testing XDR solutions due to the nature of the deployment, the spatial bias can occur, depending on how the testing dataset is built. A notable reference in the paper by Pendlebury et al. [36] is the one by Kouwe et al. [72], which presents a series of benchmarking crimes in systems security, representing faults in benchmarking which are often encountered in publications. Some of the features presented in this list focus on how and what is measured when benchmarking the systems. In this list, some "crimes" concern testing conditions, which are useful to us. Finally, Arp. et al. [2] cover a series of pitfalls in evaluating computer security using machine learning. We use seven of the ten mentioned pitfalls as sources of bias or to support already defined sources. We adapt some of the biases collected to fit the problem at hand better.

2.5. Attacks

For testing the XDR solutions, we need to reason on what types of attacks could be used for benchmarking. Karantzas and Patsakis [21] present an assessment of multiple Endpoint Detection and Response solutions on the market. This study presents the attacks used for testing these solutions, which we use as inspiration for testing the XDR solution. However, they use Cobaltstrike [7], a framework that red teams use to launch attacks. Nevertheless, this framework is unavailable to us due to licensing. Hence, in our application of the framework, we use similar attacks but other frameworks or methods for launching them, such as Infection Monkey [46], and Metasploit [29].

We also examined existing efforts to define datasets for testing intrusion detection systems. Khraisat et. al. [23] analyzed the most used datasets and made a comparison between them: are DARPA/KDDCUP 99 [26], CAIDA [4], NSL-KDD [70], ISCX 2012 [67], ADFA-LD and ADFA-WD [9], and CICIDS 2017 [66]. In our paper, we use their results to dismiss publicly available datasets and enforce the idea of using a new dataset. However, we select some of the attacks from the newer datasets to be part of our dataset.

2.6. Our contribution

We use the existing frameworks for testing intrusion detection systems and the methods for doing it as a base of our framework. However, we redirect the scope of our framework to testing XDR solutions. Hence, instead of only focusing on measuring the detection capabilities of the solution, we broaden the scope to test the response capabilities of the solution and the efficiency it provides to the security operators. We use new metrics to evaluate these aspects, which we base on already existing metrics from testing IDS and SOAR, and for evaluating the incident management process.

All the existing frameworks we found do not consider the bias that can appear from testing a solution in a small-scale, isolated testing environment. Even though multiple papers tackle the difficulty of testing intrusion detection systems, none propose guidelines for setting up the environment and creating a suitable attack dataset to minimize this bias.

3. Background

3.1. XDR Solutions

Security products vendors and technology research companies have provided different definitions of XDR tools. Hence, there is no standard definition available for these tools, one to which all the products can refer to. An organization which would want to invest in such a solution may be confused from all the different definitions. It may be unclear to them what is the solution they want to acquire, and what capabilities does it have. Therefore, we consider important to have a unified definition of the XDR technology, which should cover all the available solutions.

We collected multiple definitions of XDR and extracted from them the parts that define the capabilities of the tools, the technology used by them and the data collected by them. We considered that these are the dimensions which need to be covered by a general definitions. The collected definitions can be seen in Table 3.1, grouped by each dimension.

Dimension	Definitions
Capabilities	“a new approach to threat detection and response that provides holistic protection against cyberattacks, unauthorized access and misuse” [52] “SaaS-based, vendor-specific, security threat detection and incident response tool” [17] “optimizes threat detection, investigation, response, and hunting in real time.”[15] "enables an enterprise to go beyond typical detective controls by providing a holistic and yet simpler view of threats [...]. XDR delivers real-time actionable threat information to security operations" [59] "XDR holds the promise of consolidating multiple products into a cohesive, unified security incident detection and response platform.” [59] “provides extended visibility, analysis, and response” [73] “collects and automatically correlates data across multiple security layers [. . .]. This allows for faster detection of threats and improved investigation and response times” [60] “collects threat data [. . .] for easier and faster investigation, threat hunting, and response.” [40] “enables security teams to rapidly and efficiently hunt and eliminate security threats [. . .] from one unified solution.” [40] “delivers visibility [. . .] to detect, analyze, hunt, and remediate today’s and tomorrow’s threats.” [6] “consolidate data from multiple layers [...] into a single console to help respond to threats faster.” [50] “leverages artificial intelligence (AI) and automation to automatically stop attacks, and remediate affected assets into a safe state.” [30]
Technology	“natively integrates multiple security products into a cohesive security operations system” [17] “unifies security-relevant endpoint detections with telemetry from security and business tools such as network analysis and visibility (NAV), email security, identity and access management, cloud security, and more.” [15] “holds the promise of consolidating multiple products into a cohesive, unified security incident detection and response platform.” [59] “is a consolidation of tools and data” [73] “collects and automatically correlates data across multiple security layers [...]. This allows for faster detection of threats and improved investigation and response times through security analysis” [60] “collects threat data from previously siloed security tools across an organization’s technology stack” [40]

	"delivers visibility into data across networks, clouds, endpoints, and applications while applying analytics and automation to detect, analyze, hunt, and remediate today's and tomorrow's threats." [6] "A cybersecurity platform that integrates different security products into a unified system."[50] "automatically collects, correlates, and analyzes signal, threat, and alert data [...]. It leverages artificial intelligence (AI) and automation to automatically stop attacks, and remediate affected assets into a safe state."[30]
Data	"enables an enterprise to go beyond typical detective controls by providing a holistic and yet simpler view of threats across the entire technology landscape." [59] "provides extended visibility, analysis, and response across networks and clouds in addition to apps and endpoints." [73] "extends those capabilities beyond endpoint to multiple security control points (including email, networks, server, and cloud)"[73] "collects and automatically correlates data across multiple security layers – email, endpoint, server, cloud workload, and network." [60] "can collect security telemetry from endpoints, cloud workloads, network email, and more."[40] "delivers visibility into data across networks, clouds, endpoints, and applications" [6] "consolidate data from multiple security layers (such as servers and endpoints) into a single console" [50] "collects, correlates, and analyzes signal, threat, and alert data from across your [...] environment, including endpoint, email, applications, and identities." [30]

Table 3.1: Collected definitions

XDR solutions are also often compared with Security Automation and Response (SOAR) solutions, as both aim to provide automation and response features. However, there are a few differences between the two, usually concerning their architecture and functionality. SOAR platforms can integrate with plenty of different tools, and they perform incident response based on manually created use-case-based playbooks. These playbooks work to orchestrate response actions across the environment, they assign tasks to personnel and incorporate user inputs to augment automated actions [63]. On the other hand, XDR solutions are usually made from tools produced by the same vendor, and they represent a unified solution of the security products of the vendor. Furthermore, the response actions and automated features do not have to be manually defined, but instead, they are embedded in the solution.

In some definitions, XDR is also considered the evolution of Endpoint Detection and Response (EDR). EDR is an endpoint security mechanism that does not cover the networking. Resources located on the endpoint, such as collected events, logs and binaries, are correlated and analyzed, to determine whether a suspicious activity occurs on the host. Usually EDR solutions are signature-based, but some approaches use machine learning capabilities to facilitate finding new patterns and correlations. Therefore, these tools are able to provide critical context to detect advanced threats and then run automated response activity, such as isolating in real time an endpoint from the network, to prevent further spreading. What XDR adds to EDR is the ability to collect data beyond the endpoint. It extends to components such as network, cloud and email.

Using the definitions from Table 3.1 and finding a common basis between them, we created our own definition of XDR solutions:

> Extended Detection and Response is a technology for threat detection and response, which unifies security products for detection and response and threat intelligence into a single platform. The solution provides broad coverage of the system, such as endpoints, servers, emails, cloud, and networks, and uses machine learning, correlation and analytics capabilities to enhance the response time and the efficiency of the security teams.

This definition is general enough to represent all the XDR solutions present on the market, as it comprises the main capabilities of the solutions, but at the same time accounts for the additional features and customization offered by the vendors.

Notation	Meaning
A	Alert event generated by the IDS
I	Intrusion event
FPR	False Positive Rate
TPR	True Positive Rate
FNR	False Negative Rate
TNR	True Negative Rate
B	Base Rate
PPV	Positive Predictive Value
NPV	Negative Predictive Value
C_α	Cost associated with the IDS generating an alert when an intrusion does not occur
C_β	Cost associated with an IDS which fails to detect an intrusion
C	Cost ratio = C_β/C_α
C_{exp}	Expected cost
X	IDS input modeled as a random variable
Y	IDS output modeled as a random variable
$H(X)$	Entropy/uncertainty of the input
$I(X;Y)$	Mutual information between the input and the output of the IDS

Table 3.2: Notations and Abbreviations

3.2. Security benchmarking metrics

Several metrics have been used and proposed in the scientific literature to evaluate the detection accuracy of intrusion detection systems. Milenkoski et al. [31] offer an overview of such metrics, which we analyze in this paper. We also investigated the metrics presented by them [31] in other scientific literature to understand each of them better. Furthermore, we also mention in this overview metrics which are not mentioned in [31] but are used in other scientific literature, if we can also consider them relevant for measuring the response effectiveness.

This overview presents the found metrics, how they are calculated, and their possible advantages and disadvantages. These metrics represent the source of inspiration for metrics we will use in the framework.

3.2.1. Intrusion Detection Systems metrics

A. Basic classification metrics

Classification metrics are basic metrics that are based on the classifications made by the intrusion detection system, which can be observed in Table 3.3. They measure the tool's capability to correctly classify an event based on a training model or signatures. These metrics are closely related to each other, as they can be computed using the number of True Positives, False Negatives, False Positives, and True Negatives.

	Alert	No alert
Intrusion	True positive (TP)	False negative (FN)
No intrusion	False positive (FP)	True negative (TN)

Table 3.3: Classifications made by an intrusion detection system

The False Negative Rate (FPR) measures the probability that in the case of an intrusion, the IDS does not generate an alert [31]. Its opposite is the True Positive Rate (TPR), also called Sensitivity, Detection Rate, or Recall, which consists of the probability that an alert generated by the IDS corresponds to an actual intrusion [31]. The False Positive Rate (FPR), also known as False Alarm Rate [3], quantifies the probability that an alert is generated even if an intrusion does not occur [31]. Then, its opposite, the True Negative Rate (TNR), measures the probability that the IDS does not generate an alert when no intrusion occurs [31]. The classification rate measures the probability that an event is correctly classified, i.e., the cases in which an alert corresponds to an intrusion, and no alert corresponds to no

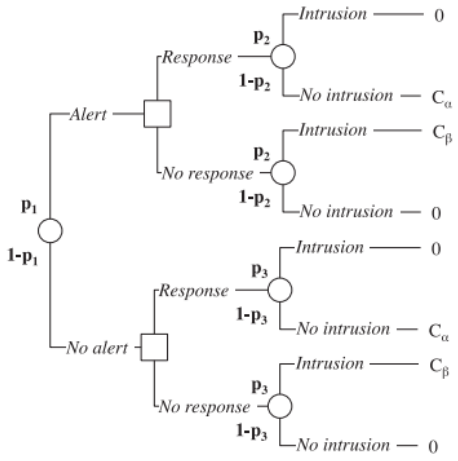

Figure 3.1: Decision tree for calculating expected cost [31]

intrusion [25]. Finally, the Precision represents the rate of the alerts generated, which correspond to an actual intrusion [25].The formulas for computing these metrics can be found in Table 3.4.

B. Positive Predictive Value and Negative Predictive Value

While the previous metrics were based on the probability of generating an alert, considering whether an intrusion takes place, the Bayesian Detection Rate focuses on measuring whether a generated alarm indicates that an intrusion occurs. This metric is also called the Positive Predictive Value and is associated with a similar metric, called the Negative Predictive Value, which measures the probability that an alarm's absence signifies no intrusion occurring [3]. Both these metrics are computed using Bayes' theorem to compute $P(I|A)$ and $P(\neg I|\neg A)$, and they depend on the base rate, also called the base fallacy rate (B). The base rate represents the probability that an intrusion event occurs. Computing the base rate is usually done by counting the number of packets representing intrusions in a system and comparing it to the total number of packets, as seen in [3]. Therefore, as intrusions usually do not occur very often, the rate is usually minimal and may not always be as accurate as possible. The complete formulas for computing these two metrics are found in Table 3.4.

C. Expected cost

Certain actions can be assigned costs that reflect the investment of an IDS in a given IT infrastructure. These costs can depend on the detection and false alarm rates, the hostility of the environment, the operational costs of the IDS, and the expected damage done by security breaches [5]. These costs may be relevant if, for instance, an action responding to an alert is costly. In [5], the authors propose assigning each classification made by the IDS a cost. For instance, the cost of an alert when an intrusion occurs will be denoted as $C(1,1)$, representing the cost of acting upon an intrusion when it is detected. In contrast, the cost of an alert when no intrusion occurs will be $C(0,1)$, representing the cost of responding when no intrusion occurs. Therefore, this mapping associates a classification with the cost of actions taken by the personnel or the system. Then, using these costs, the authors present the expected cost of an intrusion and the expected cost for a non-intrusion as trade-offs between the probability of a false alarm (α) and the probability of detection $(1-\beta)$. The expected cost for a non-intrusion is defined as

$$R(0,\alpha) \equiv C(0,0)(1-\alpha) + C(0,1)\alpha$$

where α represents the FPR. The expected cost for an intrusion is defined as

$$R(1,1-\beta) \equiv C(1,0)\beta + C(1,1)(1-\beta)$$

where $1-\beta$ is the Detection Rate (TPR). Using these, they define a trade-off between these expected costs to generalize an overall expected cost of the IDS as

$$E[C(I,A)] \equiv R(0,\alpha)(1-B) + R(1,1-\beta)B$$

$$\equiv C(0,0)(1-\alpha)(1-B) + C(0,1)\alpha(1-B) + C(1,0)\beta B + C(1,1)(1-\beta)B$$

Metric	Formula
False Negative Rate	$\beta = P(\neg A\|I) = \frac{FN}{FN+TP}$
True Positive Rate / Detection Rate / Recall	$1-\beta = P(A\|I) = \frac{TP}{TP+FN}$
False Positive Rate	$\alpha = P(A\|\neg I) = \frac{FP}{FP+TN}$
True Negative Rate	$1-\alpha = P(\neg A\|\neg I) = \frac{TN}{TN+FP}$
Classification Rate / Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
F-score	$2 * \frac{Precision*Recall}{Precision+Recall}$
Positive Predictive Value	$P(I\|A) = \frac{P(I)P(A\|I)}{P(I)P(A\|I)+P(\neg I)P(A\|\neg I)} = \frac{B(1-\beta)}{B(1-\beta)+(1-B)\alpha}$
Negative Predictive Value	$P(\neg I\|\neg A) = \frac{P(\neg I)P(\neg A\|\neg I)}{P(\neg I)P(\neg A\|\neg I)+P(I)P(\neg A\|I)} = \frac{(1-B)(1-\alpha)}{(1-B)(1-\alpha)+B\beta}$
Expected Cost	$Min\{C\beta B, (1-\alpha)(1-B)\} + Min\{C(1-\beta)B, \alpha(1-B)\}$
Intrusion Detection Capability	$C_{ID} = \frac{I(X;Y)}{H(X)}$

Table 3.4: Overview of metrics used in scientific literature

where B is the base rate previously defined. The scope is to minimize this expression, using α and $1-\beta$ as parameters.

This method of computing the expected cost is inspired from [16], where the expected cost is viewed from a decision theory perspective. The expected cost is here calculated using a decision tree and assigning probabilities for each outcome. This decision tree can be seen in Figure 3.1. Therefore, $p_1 = P(A) =$ the probability that the detector reports an alarm, $p_2 = P(I|A) = PPV$ = the conditional probability of an intrusion given that the detector reports an alarm and and $p_3 = P(I|\neg A) = 1-NPV$ = the conditional probability of an intrusion given that the detector reports an alarm.

The expected cost, C_{exp}, can be computed by rolling back this decision tree. The expected cost is computed for an operating point of the IDS, taking into consideration α and β. C_α represents the costs associated with taking action to respond when no intrusion occurs, while C_β are the costs of taking no action when an intrusion occurs. Hence, to compute it, the costs C_α and C_β should be assessed. Taking both the expected cost given no alert and the expected cost given an alert into consideration, the expected cost of an operating point is

$$C_{exp} = Min\{C\beta B, (1-\alpha)(1-B)\} + Min\{C(1-\beta)B, \alpha(1-B)\}$$

where B is the base rate and C is the cost ratio C_β/C_α.

D. Intrusion Detection Capability

In [18], the authors propose a new unified metric for evaluating IDSs, called Intrusion Detection Capability (C_{ID}). This metric is based on an information-theoretic approach, using entropy and mutual information for computations. The input is modeled as a stream of random variable *X*, where *X=1* represents an intrusion, and *X=0* a non-intrusive event. The output of the IDS consists of a random variable *Y*, where *Y=0* denotes no alert and *Y=1* represents an alert. The uncertainty of the input X and output Y will be denoted as *H(X)* and *H(Y)*. Using these, the authors model the number of the correct guesses made by the IDS as $I(X;Y)$, the mutual information between the input and the output. Using the mutual information, C_{ID} is computed as

$$C_{ID} = \frac{I(X;Y)}{H(X)}$$

C_{ID} incorporates the base rate B, TPR $(1-\beta)$, and FPR (α), as well as PPV and NPV. By the definition of entropy, H(X) can be computed as

$$H(X) = -Blog(B) - (1-B)log(1-B)$$

where B represents the probability of an intrusion occurring and 1-B the probability that an intrusion does not occur. Furthermore, the mutual information $I(X;Y)$ can be computed using the difference

Category	Metrics
SOAR metrics	Appropriate measures against attacks Accurate classifications and reliable taxonomies of threats Simplified user interface to control security tools Qualitative and quantitative information about the current incident Increase in Detection Rate Mean time to notification, remediation, and investigation Time to detect, triage attack, and remediation Time needed to analyze an attack Incident response capacity
Incident Management Metrics	Mean time to recovery Mean time to resolve Mean time to respond Mean time to acknowledge

Table 3.5: Overview of the metrics for evaluating SOAR and for incident management

$H(X) - H(X|Y)$. From [18], $H(X|Y)$ is computed in the following way

$$H(X|Y) = -B(1-\beta)logPPV - B\beta log(1-NPV) - (1-B)(1-\alpha)logNPV - (1-B)\alpha log(1-PPV)$$

Therefore, this metric consists of a combination of existing metrics and provides a trade-off between them. The aim is for an IDS to have a C_{ID} as high as possible for better detection performance.

E. F-score

The F-score [13] measures a trade-off between Precision and Recall, and its purpose is to measure the accuracy of classification. The F-score is computed as

$$\frac{2 * Precision * Recall}{Precision + Recall}$$

and therefore, it can be considered the harmonic mean between Precision and Recall.

3.2.2. Security Orchestration platforms metrics

XDR solutions use automation and response features, similar to Security Orchestration Platforms. In [19], the authors offer an overview of Security Orchestration platforms, where they define them as accurately as possible using multiple resources. In this overview, a section is dedicated to the quality attributes of such a platform, and each quality is accompanied by a metric that can measure it. Some of these metrics can also be useful for the evaluation of XDR solutions, such as: appropriate measures against attacks, accurate classifications and reliable taxonomies of threats, simplified user interface to control security tools, mean time to notification, remediation, and investigation, and incident response capacity. All of the selected metrics can be seen in Table 3.5. These metrics cover more dimensions in evaluating the performance of detection and response solutions compared to those used for evaluating IDSs, which mainly focus on detection capabilities. However, they lack a method for computing them, which means that if one wants to use them, they will have to provide their own method for computing them.

3.2.3. Incident management metrics

Responding to a threat using the XDR solution can also be viewed as an incident management technique. Several metrics can measure how a company resolves the issues that appear, especially in terms of time [37]. We analyzed these metrics and selected the ones which can be used for measuring the response of XDR solutions. The selected metrics can be found in Table 3.5. We omitted metrics such as the mean time to failure and the mean time to repair, as we considered that they are directly out of scope of the evaluation of XDR solution.

The mean time to recovery represents the average time it takes to recover from a system failure. Computing it is done by by adding up the total time spent on repairs during a given period and then dividing that time by the number of repairs [38]. The mean time to resolve is the average time it

takes to resolve a failure fully. Computing is is done by adding up the full resolution time during a given period and dividing it by the number of incidents [38]. The mean time to respond refers to the average time it takes to recover from a system failure from the time the team is alerted of that failure. To compute this metrics, add up the full response time from when the alert occurs to when the service is fully functional again, and then divide by the number of incidents [38]. Finally, the mean time to acknowledge is the average time it takes from when an alert is triggered to when work begins on the issue. To compute it, add up the time between alert and acknowledgment, and then divide by the number of incidents [38].

3.3. Similar intrusion detection solutions

3.3.1. EDR - Endpoint Detection and Response

Endpoint Detection and Response (EDR) represents an endpoint security mechanism that does not cover networking. Resources located on the endpoint, such as collected events, logs, and binaries, are correlated and analyzed to determine whether a suspicious activity occurs on the host. Usually, EDR solutions are signature-based, but some approaches use machine learning capabilities to facilitate finding new patterns and correlations. Therefore, these tools can provide critical context to detect advanced threats and then run automated response activity, such as isolating an endpoint from the network in real-time, to prevent further spreading [56].

3.3.2. NDR - Network Detection and Response

Network Detection and Response (solutions) are intrusion detection and response solutions focusing on suspicious traffic on the network. They usually use non-signatures based techniques, such as machine learning for detection [43]. Thee solutions monitor and analyze network traffic to generate a baseline profile. When collected network data deviates from this profile, then it marks it at suspicious and as an intrusion [39]. After detection, they also provide response capabilities that can create a more effective incident response [39]. The disadvantage of these solutions is that they miss intrusions occurring at the endpoint level, which can also be critical for an organization.

3.3.3. SOAR - Security Orchestration Automation and Response

According to Gartner, SOAR "refers to technologies that enable organizations to collect inputs monitored by the security operations teams." [42]. Moreover, in [19], a collection of definitions for security orchestration and automation is provided. One of these is from Forrester, which defines security automation and orchestration as "technology products that provide automated, coordinated and policy-based action of security processes across multiple technologies, making security operations faster, less error-prone and more efficient.". Hence, SOAR solutions manage existing intrusion detection tools in the system and coordinate them with the scope of simplifying and improving the response capabilities of the security team. They usually do this via playbooks, which specify the actions taken by each tool for different use cases.

4. Methodology

4.1. Overview

The process of defining our framework consisted of three steps: selecting the capabilities to be tested by the framework, defining the metrics which evaluate these capabilities, and determining the sources of bias that can influence the capability evaluation. We selected these steps because we considered that an XDR benchmarking testing framework consists of three steps: defining what to measure, measuring the selected aspects, and accounting for the differences that may appear due to different testing conditions. An overview of these steps can be found in Figure 4.1.

This section presents our process, which leads to the definition of our framework. We begin by describing the selection of XDR capabilities to be tested by the framework. Then, we define what we want to measure regarding these capabilities and the metrics necessary for measuring them. Finally, we investigate the possible sources of bias in testing and how they may influence the testing procedure and the defined metrics.

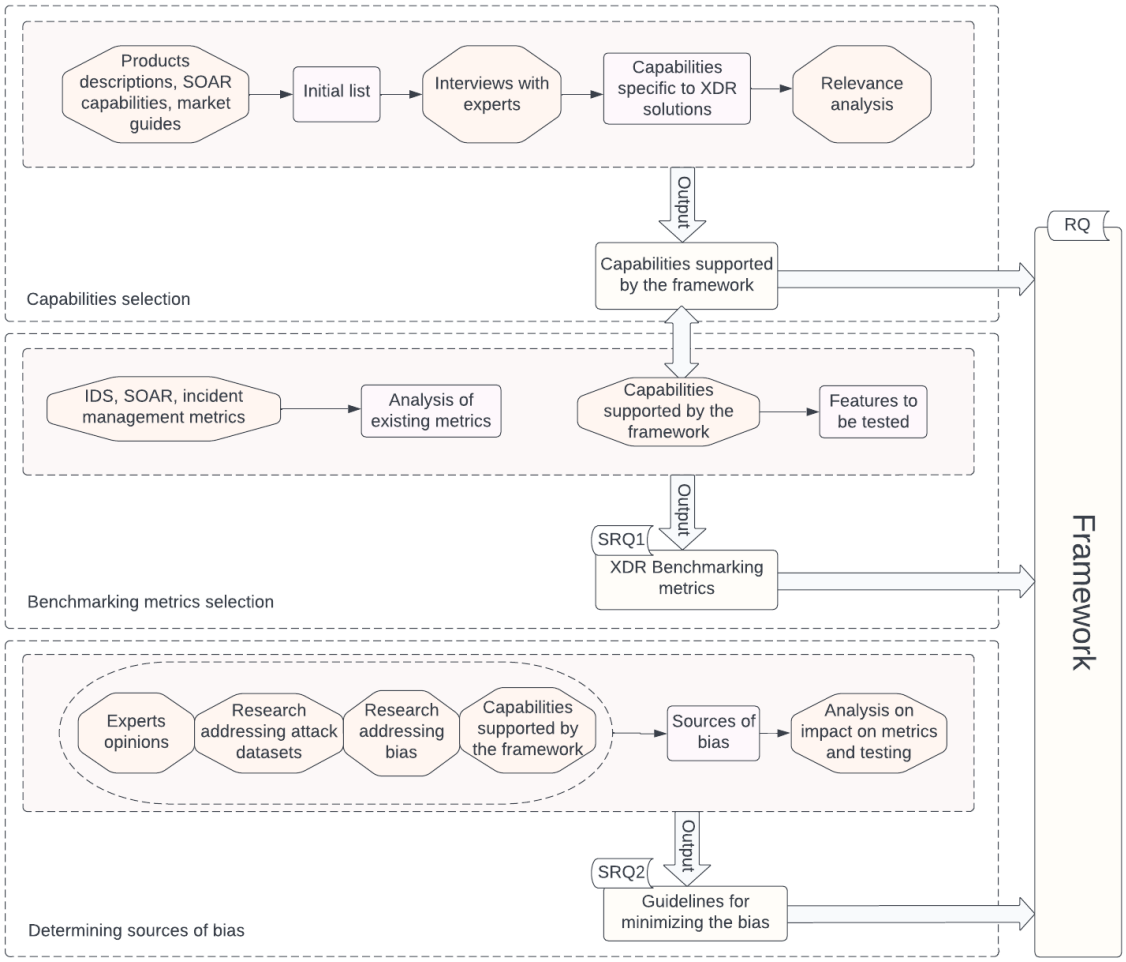

Figure 4.1: Overview of the methodology for defining the framework

4.2. Selection of relevant XDR tool capabilities

This section presents the process of selecting the capabilities that our framework should support testing. The main goal of this process is to limit the scope of the framework to the capabilities that are only characteristic of XDR solutions and not generally applicable to any intrusion detection, prevention, or response system. Frameworks for generally testing these solutions already exist, and we wanted to focus specifically on the capabilities of XDR solutions. This process consists of three steps: creating an initial list of capabilities, refining this list using interview results, and further reducing this list by evaluating the remaining capabilities.

We start with collecting capabilities from product descriptions, similar solutions, and market guides. Using these sources, we create an extensive list of general XDR solutions capabilities. Then, we conduct interviews with security experts and use the results to reduce the initial list to one consisting only of capabilities specific to XDR solutions. This step is necessary for removing the capabilities which can also apply generally to other security solutions. Finally, we perform our assessment on the resulting capabilities list to select only the ones related to the response effectiveness of the solution and to combine capabilities with similar goals into one. With this step, we only keep the capabilities concerning the response and ensure that each capability differs from another.

4.2.1. Initial identified capabilities

We gathered and analyzed the descriptions of the products of ten of the top XDR vendors [6, 14, 30, 52, 69, 60, 12, 27, 65, 11], and we extracted their featured capabilities. As there are plenty of similarities between XDR and SOAR solutions, we also extracted some capabilities representing SOAR solutions from Islam et al. [19], which should also apply to XDR solutions. Finally, we reviewed the Gartner market guide [17] for XDR products, as it contains an overview of features that should be present in any XDR solution. Therefore, we collected information from 12 representative sources.

After extracting all the capabilities, we analyzed them and grouped similar capabilities, which were the ones that referred to the same aspect or had the same purpose. We gave a representative name to each group of capabilities, and all the names made the list of the initially identified capabilities. In this way, we defined our first iteration of capabilities, which consisted of 15 items. Each capability and the number of occurrences of this capability in our sources can be seen in Table 4.1.

4.2.2. Conducting interviews

We interviewed five security specialists who have used XDR solutions. Four occupy high ranks in the KPMG cyber department (partner, senior manager, director, and manager) from multiple branches (The Netherlands, France, and Ireland), and one is the CTO of a security services and consulting company. Therefore, they have broad expertise in security solutions and products, having worked with them for several years. Hence, they could also provide answers regarding XDR solutions by comparing them with other security solutions, such as EDR, NDR and SOAR. The interview consisted of 3 questions:

1. What do you think defines an effective XDR solution? In other words, do you think that there is the main feature that defines such a solution?
2. I will mention one by one each of the 15 identified capabilities, and I would like to ask you whether you consider it relevant for the response effectiveness of an XDR solution. You can do this by assigning it a priority from 1 to 3. 1 means that the feature is not relevant for an XDR solution, 2 means that the capability is useful for an XDR solution, and 3 means that the feature is essential for an XDR solution.
3. Do you think XDR solutions have a capability missing from this list?

The purpose of the first question was to identify the essential capability of an XDR solution, one that should not be missing from the final capabilities list. We used the second question to refine the already created list of identified capabilities we previously presented. We wanted to observe whether some capabilities were relevant for an XDR solution or whether they could be more specific to already existing security solutions. It could have also been the case that some capabilities referred to the same feature, and we wanted to remove duplicates. The initial capabilities and the scores given by the interviewees can be seen in Table 4.2. The third question aimed to assess whether we missed something during our initial research that could be relevant for a solution. Besides these questions, we also had an additional question used for defining the sources of bias, which we will present in Section B.

#	Capability	Occ.
1	**Automate response actions**, such as blocking attacks, isolating endpoints, executing scripts, and eliminating malicious presence	9
2	**Use and create playbooks** to automate investigation and response tasks	3
3	**Aggregate and correlate events from multiple sources**, such as email, endpoints, servers, cloud environments, and networks	12
4	**Present all the collected events in a unified view**, to understand context and attacks and to identify the root cause and the scope of the attack	6
5	**Gather threat intelligence from various external sources** and use it to also provide contextual insight related to alerts or attacks to an analyst	6
6	**Present an overview of automated investigation and response actions** in one easy-to-use console, where analysts can also take remediation steps for stopping an attack	5
7	**Scan devices for gaps in coverage of the system, vulnerabilities, and other problems**	1
8	**Have a command line available** to initiate response actions, such as terminating processes, running scripts, installing, and uninstalling software and rebooting devices	1
9	**Have multiple integrations available**, which can enhance the capabilities of the tool	4
10	**Provide investigation and threat detection capabilities**, using security automation and analytics	9
11	**Provide alert grouping and scoring** using security automation and analytics, to let analysts focus on high confidence alerts and prioritize what is critical	5
12	**Use behavioral analysis and machine learning** to identify hidden threats, such as malware which has not been seen before, or malicious anomalies	6
13	**Have remote assistance available** for containing and eradicating threat components	2
14	**Offer predictive capabilities** for threat response	1
15	**Automate repetitive manual tasks and remove duplicate incidents**	3

Table 4.1: Number of sources that mention each capability

4.2.3. Refining the capabilities list using the results of the interviews

We first selected the capabilities for which the experts offered a score above and included 2 with at least an 80% agreement score. We considered that this approach would filter out the capabilities which are not seen as relevant by the experts. After this triage, we removed only two capabilities.

We then considered the experts' comments regarding some of the capabilities, especially for the ones they gave a score between 1 and 2, and further refined this list using these comments. We also used these comments to group some capabilities into one, considering similarities. Hence, the use and the creation of playbooks (2) were seen by some of them as not specific to an XDR solution but more to a SOAR solution. They considered that an effective XDR solution should have embedded the automated and response actions in their architecture. They also thought that automating repetitive tasks (15) is related to playbooks and could be removed from the initial list. Furthermore, we combined aggregating and correlating events from multiple sources (3) with multiple integrations (9), as the integrations usually offer more data sources. On the same note, the predictive capabilities (14) are usually done using machine learning algorithms, which are also the base of behavior analysis (12). Hence, we could group these two capabilities into one, focusing on machine learning.

Finally, we analyzed whether we needed to add any capabilities. According to the interviewees, no capabilities were missing from the given list. Therefore, we did not have to add any new capabilities to our initial list.

4.2.4. Assessing the remaining capabilities

The last step of refining the results was to analyze each of the remaining capabilities, to assess whether they influence the response in the face of a threat or are relevant to this framework. After this analysis, we kept only the relevant capabilities used throughout the framework.

A. Partially automate response actions. This is the capability of the tool to react automatically when it detects a threat. Some of the actions it can take consist of blocking attacks (traffic, downloads, or running of software), isolating endpoints, executing scripts, and eliminating malicious presence. This capability influences the response time of the security team by reducing the time the team needs to

#	Capability	Score				
1	Automate response actions	2.5	3	2	2	3
2	Use and create playbooks	3	1	3	3	2
3	Aggregate and correlate events from multiple sources	3	2	3	3	3
4	Present all the collected events in an unified view	3	2	2	2	3
5	Gather threat intelligence	2.5	3	2	2	2.5
6	Present automated investigation and response actions	2	3	2	2	3
7	Scan devices for gaps in coverage	1.5	1	2	2	2
8	Command line	1.5	3	3	3	2
9	Multiple integrations	3	3	3	3	3
10	Investigation and threat detection capabilities	3	2	2	2	3
11	Alert grouping and scoring	1.5	2	3	3	1.5
12	Identify hidden threats using behavior analysis	3	3	3	3	3
13	Remote assistance	3	3	2	2	2
14	Predictive capabilities	3	1	3	3	3
15	Automate repetitive tasks	2	3	2	2	2

Table 4.2: The score assigned to each capability by the interviewed experts

handle an attack. In addition, the remaining actions can be done faster by automating some of the actions they would need to take.

B. Aggregate and correlate events from multiple sources. The XDR solution should be able to collect logs and events from multiple sources, such as endpoints, emails, servers, cloud environments, and networks. All the events collected should then be aggregated and correlated to determine whether an attack occurs in the system. By having a broader coverage, the solution should be able to detect attacks that do not only affect the endpoint, and therefore, it should ensure broader coverage of detected attacks. This capability mainly affects the detection effectiveness of the solution. However, the response depends on the detection, and therefore, we can consider that a broader coverage of the system can also influence the response capability of the solution.

C. Present all the collected events in a dashboard or console. Having all the collected data in one dashboard allows the security team to view what happens in the system faster. In this way, they do not have to check all the assets. In addition, gathering everything in one view simplifies the process, as the analysts can see all the information in one place and take all the actions from the same place. Therefore, using this feature, the response time can be reduced.

D. Gather threat intelligence from various external sources. The capability of gathering threat intelligence can be useful for analysts to gather information about new or unknown attacks. Hence, it can decrease the response time, as the analysts will not have to look for new threats separately, and they will be able to make faster decisions regarding the response options.

E. Present an overview of automated investigation and response actions. Similar to presenting all the collected events in a single dashboard, having an overview of automated investigation and response actions can help the analysts observe possible remediation steps, without having to think about them independently. Furthermore, further investigation options can be suggested if they do not know what happens in the system. Therefore, this feature helps the security team to make decisions faster and decreases the response time. This capability can also be combined with the previous one, as threat intelligence helps decide which approach to take.

F. Provide various investigation and threat detection capabilities. Providing various investigation and threat detection capabilities is essential for any intrusion detection and response solution. This capability can be seen as a combination of the previous two when considered for evaluation. The threat detection and investigation capabilities are made available to the analysts in the overview for the automated investigation and response actions, combined with threat intelligence for threat detection.

G. Use behavior analysis and machine learning to identify hidden threats. This is an essential capability for threat detection and response, as it helps detect attacks that do not have a known signature

based on anomalies in user behavior and traffic. Therefore, it ensures a greater chance of detecting unknown and hidden threats, even if this capability is associated with plenty of false positives.

H. Have remote assistance available for eradicating threats. Even if this feature is considered helpful for an XDR solution, we can consider it out of the scope of this paper. This is due to not using this feature during the experiments and for this framework, as the framework focuses on assessing the internal capabilities of the tool.

The presented collected events in a single dashboard (C) are usually the events collected from multiple sources, defined in the second capability (B). Therefore, we can aggregate these two capabilities into one. Moreover, threat intelligence (D) is valuable when it is beside the overview of automated investigation and response actions (E), as the analysts can get the necessary knowledge on how the attacks work and how they can be solved. Hence, these two capabilities can also be combined, and as previously mentioned, they can represent the sixth capability. Considering these aspects and how each defined capability influences the response, we defined a shorter list of capabilities that the framework will measure. The list consists of the following four capabilities:

1. Automate parts of the response actions when a threat occurs
2. Aggregate and correlate events from multiple sources in a unified view
3. Present an overview of the automated investigation and response actions, as well as the threat intelligence required for responding to an attack
4. Use behavior analysis and machine learning to identify hidden and unknown threats

The selected capabilities are all the ones which had a score of 3, so our selection aligns with the opinions of the experts. These capabilities are all part of what we consider to be the response of an XDR solution. Therefore, responding to a threat does not mean that the tool blocks the threat itself but offers all the conditions for the security team to block it successfully and as fast as possible.

4.3. Capability benchmarking metrics

The traditional approach for evaluating the performance of intrusion detection systems consisted of measuring the detection capabilities. However, as XDR promises to offer capabilities that aim to help the analysts remediate threats faster and in a more automated way, this framework focuses on evaluating how the XDR solution aids the response and the analysts' efficiency.

We first analyzed the metrics used in the scientific literature for testing intrusion detection systems, the quality attributes for a security orchestration solution, and the incident management metrics. We looked for advantages and disadvantages of using them and for methods of computing them. Then, we took each XDR capability we selected and considered what we could measure regarding each. For each capability, we first summarized in free text, in the form of a question, what a user of our framework would deem desirable to evaluate. Then, we analyzed the existing IDS, SOAR, and incident management metrics and considered whether some of them could answer our free text questions. Using these metrics as a basis, we proposed a new set of metrics together with a method for defining them. The overview of the tested features, the questions, and the metrics used can be seen in Table 4.3.

Measuring these metrics is done after launching attacks against a system monitored by an XDR solution. Each attack comprises several steps, each adhering to a technique, as described for instance, by the MITRE ATT&CK framework [33]. When we launch an attack, each step, which we will refer to in the following sections as an incident or event, generates one or more alerts. In the following sections, we present and discuss each feature's metrics, their relevance, and how they answer our questions.

4.3.1. Features to be tested

We previously defined four capabilities that are characteristic of XDR solutions. We want our framework to enable one measuring these capabilities, but we want to do it in a way that accounts for the natural steps of handling an attack. These steps include detecting, triaging, investigating, responding to, and remediating the incidents, and we can map them to our capabilities. It is essential to mention that we do not individually evaluate the capability of user behavior analysis and machine learning but their evaluation becomes part of the evaluation of their related capabilities. Even if the response of the solution is depended on the detection of events, which is represented by this capability, we want to keep the scope of our framework limited to evaluating the features concerning directly the response.

Feature	Capabilities	Question
Investigating the events	Aggregate and correlate events from multiple sources Use behavior analysis and machine learning to identify hidden threats	How well do the tool's detection capabilities help an analyst investigate an attack while keeping the triage time at an acceptable level?
Suggested investigation and response actions	Present an overview of the automated investigation and response actions & the required threat intelligence	To what extent are the analysts helped by the tool to remediate the problem, as well as how much time is saved using these suggestions compared to a normal investigation?
Automation	Automate parts of the response actions when a threat occurs	How much does the tool simplify the remediation work of the analysts, especially in terms of time-saving?
Efficient response actions	Automate parts of the response actions when a threat occurs Use behavior analysis and machine learning to identify hidden threats	Considering multi-stage attacks, how early does the tool identify the attack and offer help in preventing the next steps from occurring and causing an impact on the system?

Table 4.3: Main questions to be answered by the metrics

A. Investigating the events

We associated *aggregating and correlating events from multiple sources* with the tool's detection capabilities and with investigating the events. XDR aims to offer the analysts more information regarding the detected incidents, compared to previous solutions, for a faster investigation of incidents. It also promises to have good detection capabilities. Hence, for this capability, we want to measure how well the tool's detection capabilities help an analyst investigate an attack while keeping the investigation time to an acceptable level. In this case, the detection capabilities also include the information offered by the tool regarding the detected incidents. Moreover, implicitly, by measuring the detection capabilities, we also measure the capability of *using behavior analysis and machine learning*. To measure these aspects, we are going to answer the following questions:

I1. How well does the tool identify the incidents which occur in the system?
I2. How much time does it take for an average analyst to investigate the events using the solution?

B. Suggested investigation and response actions

We consider that we should evaluate *presenting an overview of the automated investigation and response actions* by measuring how much these suggestions help the security operators to investigate and remediate an attack. XDR solutions claim to assist the security operators in their investigation and response by offering suggestions on how to proceed when facing a threat. Therefore, we want to measure these suggestions' usefulness, especially in decreasing response time. In order to do this, we are going to answer the following questions:

S1. How much time is needed to decide which of the suggestions are helpful and to consider additional steps which may be required to finish the incident response?
S2. How much time is needed to investigate and resolve the event using the suggestions offered by the solution?
S3. What is the distribution of time-saving using these options per attack?

For S3, the time needed to investigate and solve the event without using the suggestions offered by the solution is also required.

C. Automation

Automating parts of response actions should be evaluated by measuring how much time is saved in the security operators' work using the tool's automated features. Automated response actions' features help significantly solve an incident as they simplify the security oeprators' work by reducing their

tasks. Therefore, we want to measure how much time and effort is saved using automated features compared to solving the incident using only manual work. To do this, we answer the following questions:

A1. How much time is needed to solve the incidents using the automated features?
A2. What is the distribution of time-saving using the automated features per different types of events?

For A2, the time needed to solve the incidents manually is also required.

D. Efficient response actions

This feature is related to the *automation* capability and *using behavior analysis and machine learning to identify threats*. As many attacks targeting an organization are multi-staged, we want to observe whether the tool correctly detects different types of attacks in their early stages and automatically stops them before damage occurs in the system. To do this, we answer the following questions:

E1. How many of the attacks were successfully detected and blocked before an impact on the system?
E2. How many of the attacks did the tool not block?
E3. What is the rate of successful responses?

4.3.2. Analysis of existing metrics

We started by analyzing various IDS metrics which we encountered in research papers. We analyzed metrics such as the basic classification metrics (Detection rate, False Positive rate, True Positive rate, True Negative rate, Accuracy, and Precision), the F-score, the Positive Predictive Value (Bayesian detection rate) and Negative Predictive Value, the Intrusion Detection Capability, and the Expected cost. For each of them, we weighted the advantages and disadvantages of using them to measure the detection capabilities of the tool, which are part of our first question's answer. We continued by analyzing the metrics used to measure SOAR solutions' quality attributes and the incident management metrics. As they cover more aspects than the detection capabilities, we wanted to observe which of them we can use as a basis for the metrics measuring the remaining capabilities. We present in this section our analysis of the available metrics.

A. IDS metrics

Basic metrics. The primary classification metrics are the simple metrics computed by calculating the probability of an alert occurring or not and whether an event is malicious. Being too simple, these metrics cannot measure the detection accuracy of an IDS on their own and cannot be used to compare two IDSs. For instance, a tool can have a better Detection Rate than another and a higher False Positive Rate than the same one. In this case, using only the Detection Rate as a measurement can be misleading because it does not necessarily mean that the first tool performs better than the second. Therefore, in most cases, a trade-off or a combination of these metrics is required for these tasks. It will be seen that most of the metrics discussed in this section depend on one or more of these metrics and are influenced by changes in them.

F-score. The F-score is the harmonic mean of precision and recall (TPR) and is a metric that considers how the data is distributed. For example, if the data is highly imbalanced (e.g., 90% of the traffic is benign and 10% is malicious), then the F-score will provide a better assessment of the model performance [57]. However, its general disadvantage is that sometimes, it is harder to interpret.

PPV and NPV. The Positive Predictive Value (PPV) is dependent on the True Positive Rate (TPR) and False Positive Rate (FPR), while the Negative Predictive Value (NPV) depends on the True Negative Rate (TNR) and False Negative Rate (FNR). Intuitively, both metrics will increase if the FPR and FNR decrease, as a lower value for these metrics should yield better results. For example, an IDS with a low value for PPV indicates that the IDS often causes the triggering of attack responses when there are no attacks [31]. These metrics also depend on the Base-fallacy Rate (B), which is the probability that an intrusion occurs [3]. As seen in [3], this rate varies from one system to another, as in one environment, there may be a higher probability of an intrusion than in another. This aspect can influence the FPR and the FNR. PPV and NPV can be used for evaluating an IDS from a usability point of view, but there is no objective method for integrating both [18]. Hence, sometimes a trade-off between them can be required for better analyzing the detection performance of an IDS, or even a

combination of them with other metrics. Moreover, when using this metric, it should be considered how the base-rate fallacy affects it.

Expected cost. The expected cost [31][5] is a more robust metric, which uses the FNR and FPR, the base rate B, and the ratio C for its computation. C is the ratio between the costs associated with taking action to respond when no intrusion occurs C_α and the costs of taking no action when an intrusion occurs C_β. This metric aims to evaluate the costs associated with different operations of an organization. However, even if it is a more robust metric, it has its disadvantage: estimating C_α and C_β. There is no objective method of measuring these two variables, as costs vary in each organization and infrastructure. Moreover, such cost-analysis models, used for estimating costs, sometimes consider parameters that might not be easy to measure or might not be measurable at all, such as the system downtime [31]. However, these metrics can be helpful when the relationship between different attack detection costs can be estimated and when these estimations can be considered sufficiently accurate [31].

Intrusion Detection Capability. C_{ID} [18] is the metric that focuses on measuring the detection capabilities of an IDS from an information theory perspective. This metric is computed using the mutual information between the input and the output and the entropy of the input. The mutual information, the PPV, NPV, PPR, FNR, and B are all required for this computing. The main advantage of C_{ID} is that it unifies multiple basic metrics and considers all the essential aspects of detection capability when comparing two IDS. This metric is also objective, as it does not depend on any subjective measure and is sensitive to IDS operation parameters, which means that it can be used for fine-tuning the IDS. However, it is not as intuitive as other previously discussed metrics. The dependency of multiple metrics, which also need to be computed, makes it harder to compute than more specific metrics. Its dependency on the base rate also makes it subject to a bias created by the environment in which the metric is computed.

B. SOAR metrics

We analyzed the SOAR metrics presented in [19] and selected the ones which can be used or adapted to measure the defined capabilities. We found suitable metrics: simplified user interface to control security tools, qualitative and quantitative information about the current incident, mean time to remediation and investigation, time to detect, triage attacks and remediation, and time needed to analyze an attack. The first two metrics can be used for measuring the investigating capability, as they measure how valuable the information is to the analysts and how it can help them to faster investigate the events. In addition, the time to investigate and analyze the attacks is also relevant for this capability. Furthermore, we can use the (mean) time for remediation and investigation of the automated features and the suggestions for investigation and remediation capabilities. Finally, we can adapt them to consider the influence of the tool's processes (automated features and suggestions).

C. Incident management metrics

The incident management metrics measure how well an organization handles the issues that may arise. For example, in the case of evaluating XDR solutions, the alert can be considered the alert generated by the solution, while the failure can represent the system while under threat. Furthermore, recovering from a system failure would mean eradicating the threat from the system. Hence, we can use metrics such as the mean time to respond or to resolve, which are similar to the previously mentioned mean time to remediation metric. However, the mean time to recovery and the mean time to acknowledge are out of the scope of our framework.

Feature	Metrics
Investigating the events	F-score Time of investigation
Suggested investigation and response actions	Distribution of time saved by suggestions
Automation	Distribution of time saved by automated features
Efficient response actions	Successful responses rate

Table 4.4: Overview of the metrics required for each feature

4.3.3. Benchmarking metrics and their goals

A. F-score & Time of investigation

Based on our analysis, we consider that for the detection capabilities of the tool, a metric such as *F-score* can be used. We consider that this metric answers T1, as it provides a trade-off between the capability of detection (recall) and the credibility of the events (precision), provided that the data is representative of a real environment. It is also important to mention that we use the F1 score, as we consider the precision and recall equally important. Hence, we compute it as the harmonic mean between these two metrics. Moreover, this is a metric that is not influenced by the base fallacy rate [2]. Then, we combine the F-score with the *time to investigate an attack* to measure the first feature. The goal is to maximize the F-score while the average time to investigate the events is kept at an acceptable level for the organization. In other words, we aim for a tool with effective detection capabilities while offering the analysts the conditions to investigate the events in an acceptable time. Hence, we are aiming for the tool with the highest F-score, and the smallest time of investigation.

It may be that a solution has both the F-score and the time of investigation higher than another. Hence, we had to define a method for comparing the solutions based on these metrics, such that it accounts for all the scenarios. We decided to do this as a trade-off between the two metrics, where both are equally important. We wanted to define a product containing these two metrics, to look for the solution with the highest value for this product. For these, the product needs to be of two functions that we want to maximize.

As we aim for best detection, one of the components of the product is the F-score. However, we want the lowest value for the time of investigation, so we need a way to inverse it to obtain a function that we can maximize. We consider the function subtracting the investigation time from the maximum possible value. This function increases when the investigation time decreases. However, we do not have a threshold for the time of investigation, as it can vary from one incident to another and from one organization to another. We can consider the threshold as double the highest registered time of investigation, out of the tested solutions. If we had considered it without doubling, we would have assumed that the highest value is the worst-performing one, which is an unfair assumption. This way, we assume that the highest value is a medium performance.

Then, we observed that by building the second component in this way, the time of investigation has too much influence on the product: the F-score has only values between 0 and 1, while the time of investigation can be hundreds or thousands of seconds. Hence, we scale this product to take values between 0 and 1 by dividing all the elements by the double of the highest value. In this way, the second component will be $1 - scaled(timeofinvestigation)$, which is an increasing function with values between 0 and 1. In this way, both components of the product have an equal impact. As both components are subunitary, their product is also increasing and subunitary.

An alternative to the trade-off would have been a decision tree. We could have defined a threshold for the time of investigation, and then compare the solution based on the F-score. However, we consider both the time of investigation and the F-score equally important. This approach would have made only the F-score the primary metric of comparison.

B. Distribution of time saved by the suggestions

For the suggested investigation and response actions, we measure the *time needed to investigate and resolve an incident*. However, we compute it once for an investigation without the tools' suggestions

and once with the suggestions. Using these two computations, we can also compute the time saved by the suggestions offered by the tool. We also do not compute an average of these times. However, we consider it more beneficial to observe the *distribution of the time saved across different attacks or incidents*. The goal is to have a tool for which the distribution is balanced (close to uniform), as we aim for a tool that offers options for investigation and response for different types of events, not just specific ones. An effective tool should also present more events with a high percentage of time saved. We consider that the better-performing solution would have a higher mean and a smaller standard deviation. A higher mean represents a higher percentage of time saved on average. A smaller standard deviation represents a more uniform dataset, with few instances far from the mean.

Similar to the F-score and investigation time, we need a trade-off between the mean and the standard deviation, as it may be that one solution has both factors higher than another one. We use a similar approach, by crafting a product which we want to maximize.

As we want the highest value for the mean, one of the components of the product is the mean itself. Then, we want the lowest standard deviation, and therefore, as for the investigation time, we need to reverse it. The percentages of time saved, as we compute them, take values between 0 and 1. Then, we know that the mean will also take values between 0 and 1, as well as the standard deviation. To be more precise, the highest value of the standard deviation can be 0.5, which is the extreme case where half of the values are 0, and half are 1. This means that the double of the highest value of the standard deviation is 1, which allows us to apply a similar approach as for the previous metrics. The difference $1 - std(timessavedbysuggestions)$ will also be then subunitary and increasing, and the product $mean(timessavedbysuggestions) \cdot (1 - std(timessavedbysuggestions))$ is also subunitary and increasing.

An alternative approach would have been setting a threshold for the standard deviation, and compare the solutions based on the mean, or setting the threshold for the mean and comparing the standard deviations. Nevertheless, we consider both the standard deviation and the mean of equal importance, and this approach does not support this.

C. Distribution of time saved by the automated features

To measure the automation capabilities, we use the *time needed to resolve an incident*. We compute this time once for a resolution of the event done without using the automated features and once using them. Similar to the previous capability, we compute a difference between these times to observe the time saved using automated features. Then, we make a plot to observe *the distribution of time saved across different attacks or incidents*. Similarly, as for investigation and response suggestions, the goal is to have a tool that has a balanced distribution. The automated features should be available for different attacks and not be biased towards specific attacks. We use the same approach for comparing the distributions of the time saved, as the one described for the suggested investigation and response actions.

D. Successful responses rate

Finally, for measuring the efficient response actions, we use a metric similar to the detection rate, which we will call the *successful responses rate*. This metric evaluates how many attacks are efficiently stopped by the solution. The goal is the maximize the number of successful responses while keeping the number of no responses to a minimal level, which means having a highly successful response rate. The tool should be able to stop as many attacks as possible before any impact occurs on the organization. Hence, we are aiming for the one with the highest successful responses rate.

4.4. Sources of bias in capability benchmarking experiments

Testing an intrusion detection and response solution such as XDR can yield different results depending on the environment in which the solution is tested. Therefore, we consider that some factors can bias the specific XDR tool capabilities benchmark. These factors are generally some specific characteristics of the environment in which the solutions are deployed or of the testing approach. To determine these factors, we analyze multiple sources to help us identify them.

We first look at the capabilities of XDR solutions we previously defined and outline the natural factors that can influence them. Then, in our interviews mentioned in Section 4.2.2, we also asked the experts about the requirements necessary for the correct deployment of an XDR solution. We consider the responses to gather the experts' opinions regarding this problem and see whether we can identify

some new sources of bias. We continue by analyzing some research papers we found addressing the concept of bias. Finally, we analyze the research done on the composition of attack datasets used for testing purposes to observe whether some bias sources are related to the attack datasets used.

After identifying the possible sources of bias, we investigated whether and how they can influence the testing of XDR solutions. We also made some assumptions on how the sources of bias can influence some metrics. We support some assumptions by referring to papers and other documentation. We test some of the assumptions with experiments and argue why they should hold for the remaining ones.

4.4.1. Collecting sources of bias

A. Capability-driven analysis

As previously mentioned, the main capabilities of XDR solutions are automation, correlation and aggregation of events from multiple sources, automated investigation and response action suggestions, and behavior analysis and machine learning capabilities. External factors and conditions can influence the performance of these capabilities.

Behaviour analysis and Machine Learning. Many intrusion detection and response tools that cover anomaly detection have embedded a technology called UEBA, or UBA - User (and Entity) behavior analysis [51]. This technology analyzes specific data, also called baseline data, over a period to create a baseline profile. The baseline data usually consists of events and logs on the host, network activity logs, application data, and user context. The baseline profile outlines the expected behavior in the system, which the tool registers. Then, the solution trains its behavior analysis and anomaly detection capabilities with the observed data. After defining the expected behavior, it should mark everything not matching this profile as suspicious. We consider that the XDR solutions have a different performance depending on how realistic the training data is. Hence, we define a first feature that may cause bias: *the presence of baseline profile*.

Correlation and aggregation of events from multiple sources. XDR solutions can collect information regarding intrusions from all the hosts and network interfaces in an organization, process this information, and then present everything in a single point of view. An organization can have dozens or hundreds of devices, and all the logs and events from these devices are collected on a central agent, which then sends the collected data to be processed in the solution's cloud. We consider that there is a difference in the performance of the solution in a smaller organization or environment compared to a large scaled one. If there is a large number of targets, the analysts would have to triage the alerts from one place to all the targets. Moreover, an increase in the number of hosts would cause an increase in the number of displayed alerts, which may also cause a cluttered view, besides a more considerable time needed for triaging and investigating the events. As our framework focuses on measuring the efficiency enabled by the solution, we believe that *the complexity of the environment* is a factor that can cause bias in the performance of the solution.

Automation and suggestions features. Automated features include, but do not stop at, the detection of threats, triaging potential threats, deciding on the most appropriate action to contain or mitigate a threat, and executing mitigation actions [41]. These automated features can be embedded in the solution, but analysts can also create use case playbooks for automating specific tasks. Intuitively, the embedded features are also based on some internal use cases with predefined steps. Therefore, we can assume that these features are tied to specific attacks or techniques. For further references, and to outline the difference between an attack and a technique, we consider an attack as a standalone attack or an attack scenario, and the steps that are part of an attack scenario are achieved by different techniques.

We can also assume that the suggested actions are proposed similarly, based on the techniques, signatures, or attacks observed. Hence, it is very likely that if zero-days or even obfuscated attacks are detected, then no automated actions will be triggered. In this way, the attack dataset used for testing can also cause a bias in the solution's performance. Hence, in this case, we identify two sources of bias: *attack dataset coverage* and *presence of novel attacks*. Moreover, considering the metrics we selected for evaluating these two capabilities, there is another bias that can appear: *testers' skills*. If the test is conducted by highly skilled security experts, and in the real deployment the task is run by lower skilled security operators, then this can influence measuring to what extent the solution is helpful in saving time.

Category	Source of bias
Capability driven analysis	Presence of baseline profile The complexity of the environment Presence of novel attacks Attack dataset coverage Testers' skills
Experts opinions	Presence of baseline profile The complexity of the environment
Research addressing the concept of bias	Used security settings Configuration of the system Attack dataset coverage The complexity of the environment Distribution of testing dataset Quality of training data
Research addressing the composition of attack datasets	Presence of novel attacks Presence of stealthier attack techniques Attack dataset coverage

Table 4.5: Identified sources of bias

B. Experts opinions

In the interviews we conducted to refine the list of XDR capabilities, we also included a question designed to find the opinion of the experts on the requirements needed for an XDR solution to work correctly. The question we asked was:

Do you think any requirements need to be considered before or while installing the XDR solution? Such an example will be if you think the tool should be left in the system for training its behavior analysis capabilities.

According to the answers, properly training the solution's behavior analysis and machine learning capabilities is necessary for its good performance. For example, one of the interviews stated that proper training is done between six months and a year, which is more than we found in the documentation concerning UEBA. Moreover, for a good performance, some stated that resource management needs to be considered for extensive infrastructure. This is because by collecting data from multiple sources, the solution takes bandwidth from the infrastructure. If resources are not adequately managed, delays can occur in the collecting and processing of data, leading to delays in reporting the alerts and the incidents to the analysts. With these answers, we can support the two sources of bias we already defined: *the presence of the baseline profile*, and *the complexity of the environment*.

C. Research addressing the concept of bias

Banchmarking crimes. Kouwe et al. [72] present a series of benchmarking crimes in systems security, representing faults in benchmarking. Some of the features presented in this list focus on how and what is measured when benchmarking the systems. For instance, they present the problem of measuring only one feature when benchmarking while another feature of the system is omitted due to lower performance. However, some "crimes" also concern testing conditions, which can be relevant to us. These include "no proper baseline," "missing platform specification," and "false positives/negatives not tested." However, we adapted their original definition to better fit the problem at hand. Hence, we outlined four features that may cause bias in the performance of XDR solutions or the testing process: *used security settings, configuration of the system, attack dataset coverage, and missing platform specification*.

Used security settings refers to testing the solution in a system where, for instance, firewall rules are changed, compared to using the default settings. *Configuration of the system* represents the bias of testing the solution in a system that was just deployed, having only the default applications and settings available, compared to one that has been altered. These two features were inspired by the "no proper baseline" crime, which addressed the issue of using a system with changes in its baseline and not documenting them. We considered that to make this feature applicable to our case, we can refer to the security baselines and the system baselines as well.

Attack dataset coverage refers to testing a solution with a dataset that does not cover all the attacks,

compared to a complete dataset. This source of bias was inspired by the "false positives/negatives not tested" crime, which made us think of different cases not covered by the testing dataset.

Finally, *missing platform specification* refers to testing a solution that does not have system specifications available in an environment that may not be suitable for its performance. In the original paper, this crime referred to not mentioning the hardware setup used to perform the experiments. However, we considered that the missing requirements for the proper solution performance might be more relevant for us. However, by looking at the previously analyzed XDR solutions, system and installation requirements were found for all of them. Hence, this factor does not represent an issue, and we can remove it.

Base-rate fallacy. Axelsson [3] presents the bias induced by the *base-rate fallacy*, which is part of some metrics evaluating intrusion detection systems, such as the Bayesian Detection Rate and the Intrusion Detection Capability. The paper presents the problems that appear when trying to evaluate an IDS in a realistic setting correctly. These problems are caused by the base-rate fallacy, which influences the ability to measure the false positive rate correctly. The author explains how one should measure this rate by considering the number of intrusions they expect to detect and not using the maximum number of false positives observed, which means measuring the false positive rate by considering the real probability of intrusion, and not the observed one. Correctly measuring this rate is biased by the total number of events observed in the environment. Hence, one cannot correctly assess the impact of the false positive rate if the testing of the solution is done in a setting where fewer false positives can be detected. Moreover, the false negative rate influences metrics such as the Bayesian Detection Rate, which decreases with a large value of this rate. We can consider this bias related to *the complexity of the environment*, as the settings of the environment and the events used influence the base fallacy rate and the false positives rate.

Bias in machine learning. Arp et. al [2] cover ten pitfalls in the evaluation of computer security solutions using machine learning. Even if some of them are strongly related to the issues occurring in training the machine learning algorithms, the others can be applicable to the intrusion detection and response evaluation. These are the *sampling bias* and the *biased parameter selection* pitfalls, as well as all the pitfalls in the categories of *performance evaluation* and *deployment and operation*.

For the *performance evaluation*, the authors identify three pitfalls: *inappropriate baselines, inappropriate measures* and *the base rate fallacy*. The first one refers to not comparing the performance of an machine learning algorithm with the state of the art, inappropriate measures concerns using the wrong metrics to evaluate the algorithm and the base rate fallacy approaches the same problem discussed by Axelsson [3]. Adapting these to testing XDR solutions, we can consider that a solution has a good performance, but in comparison to other existing solutions, it does not perform so well. Moreover, using a set of metrics may show that a tool has a good performance, while others do not. Hence, as we already made an analysis on the existing metrics and chose suitable ones for our framework, and as we are comparing solutions between them, these do not consist sources of bias.

Concerning *deployment and operation*, the authors address two problems: *testing the solution only in a laboratory* and *using an inappropriate threat model*. The first aspect concerns our main problem concerning bias, and namely the fact that only testing the solutions in a laboratory is not representative of a real life setup. The other one concerns not taking into consideration how a real attacker operates, which is a part of our already defined biases concerning attacks.

The *sampling bias* addresses the problem of using data which does not sufficiently represent the true distribution of the underlying problem, and *biased parameter selection* presents the issue of not fixing the parameters of a learning-based method at the training time, and having them dependent on the test set. In our case, we observe the problem of testing the solution only with data from a lab setup where the balance between attacks and benign activity is radically different from the actual distribution. Moreover, by deploying the solution in small laboratory, with no other attacks occurring before the moment of testing, the solution may be trained to over detect, as it observes only simple, benign traffic. Hence, this paper supports some of our choices of bias, and introduces two new sources: *distribution of the testing dataset*, inspired from the sampling bias, and *quality of the training data*, inspired from biased parameter selection.

Malware classification bias. Pendlebury et al. [36] present two sources of bias encountered in the classification of Android malware: temporal and spatial. The *temporal bias* appears when a type of malware is classified by training the algorithm with newer malware. The *spatial bias* occurs when the

percentage of malware in the testing and training datasets is not representative of an accurate world ratio between malware and goodware. XDR solutions cannot train directly using a training dataset. Their machine learning and behavior analysis algorithms train using live data, which is ingested in real-time. Hence, due to the nature of this deployment, a strict temporal order between training and testing is created, so a temporal bias cannot appear. However, a spatial bias may occur. In a testing environment, the solution mostly observes normal traffic, as no malicious traffic occurs before testing. Hence, this can lead to overdetection, as any small deviation may be considered malicious. This spatial bias is similar to the *distribution of the testing dataset* bias, which we already defined.

D. Research addressing the composition of attack datasets

In general, testing intrusion detection systems and similar solutions is done using publicly available datasets, as building an attack set from scratch can be tedious. The most frequently used and analyzed datasets are DARPA/KDDCUP 99 [26], CAIDA [4], NSL-KDD [70], ISCX 2012 [67], ADFA-LD and ADFA-WD [9], and CICIDS 2017 [66]. Khraisat et al. [23] analyzed these datasets and outlined multiple comparisons between them. All these datasets comprise real network traffic, as they are generated by recording the traffic from a real environment. They usually consist of two sets: one for training the solution and one for testing the solution. All these datasets can be useful for testing the detection capabilities of intrusion detection systems and validating them. However, they have some limitations, which can cause some bias in the results.

Using a training dataset. In a real scenario, the security solution does not necessarily train with an existing dataset after deployment and usually trains by observing the system. However, attacks may occur even immediately after the installation of the tool, and in this case, the tool should have good detection and response capabilities immediately after deployment. Moreover, it may be that just using the testing dataset part of these sets can yield different results.

Missing novel attacks. The available datasets can quickly become outdated, as, in a period of a few years, new attacks, exploits, and malware appear, and they become new threats. Existing attacks in these datasets may also become irrelevant to the present infrastructures. A set with old and well-known attacks can bias the testing results, as most of these are covered by existing signatures. Using malware or exploits available for years should be easily detected and blocked, even in different forms, as there was sufficient time to create signatures for them. Nonetheless, the newer datasets are improvements to the old ones, as they contain newer attacks and zero days. For example, the CIDS 2017 dataset [66] contains attacks such as brute force for FTP and SSH, DoS, Heartbleed, and botnets. ADFA-LD and ADFA-WD [9] also contain Meterpreter, web, bind and reverse shells, exploits, data exfiltration, and back-door insertion [23]. However, newer attacks, such as ransomware and phishing, are still missing and are of high relevance nowadays.

Limited attacks and techniques. The limited number of attacks present in the datasets may create a bias in the results due to the lack of diversity. For instance, a tool may be more inclined to detect DoS and DDoS attacks, and if the tool is tested with the CAIDA dataset [4], which consists of only DDoS attacks, a good accuracy and detection rate would be observed. However, for other attacks, the tool may not perform so well. Furthermore, looking at how the attacks are generated, we can observe that for the CICIDS set, the authors used available tools and frameworks to launch the attacks against the targets, such as Metasploit [29], Nmap [35], and Selenium [64] [66]. Some of the newer intrusion detection and response solutions, such as XDR, may be able to flag the use of such tools by observing their fingerprints. Therefore, the use of evasion mechanisms and handcrafted attacks, which are widely used by attackers such as APT, is missing even from these datasets.

Missing multistage attacks. The use of multistage attacks, or targeted attacks, is also an aspect that is not considered in these datasets. However, as targeted attacks are the main threat concerning organizations [54], they should be the central part of an attack dataset simulating the threats an organization faces nowadays. Moreover, it would be interesting to observe whether the tool can detect the attack in its early stages and prevent a higher impact on the organization.

Missing host data. Finally, these datasets only cover network traffic, and therefore, they are limited in testing the capabilities of a hybrid intrusion detection system. XDR solutions analyze data from more than the network segment, and using these datasets will only provide a partial solution. However, there is a way of benchmarking host-based IDS solutions. The Virus Total dataset [62] contains a vast body of malware samples that spawn for five years. This dataset is not publicly available but can be obtained for a fee or for free for research purposes. However, it does not contain recent malware

Bias source	F-score	Investig. time	Distribution automation	Distribution suggestions	Successful responses	Testing process
B1. Used security settings						✓
B2. Configuration of the system						✓
B3. Presence of baseline profile	✓				✓	
B4. Testers' skills			✓	✓		
B5. Distribution of testing dataset	✓					
B6. Quality of training data	✓					
B7. Complexity of the environment	✓	✓				
B8. Attack dataset coverage	✓		✓	✓		
B9. Presence of novel attacks	✓		✓	✓	✓	
B10. Presence of stealthier attack techniques	✓		✓	✓	✓	

Table 4.6: Overview of the factors influenced by each source of bias

samples, as the dataset is only updated every six months.

Considering the presented aspects, we define the following bias sources: *presence of novel attacks, presence of stealthier attack techniques, and limited attack dataset*. However, the last source covers the same aspects as *attack dataset coverage*, and we can merge them into a single one.

4.4.2. Analysis of the selected sources

B1. Used security settings. Modifying security settings, such as the firewall, can change the impact of the attacks on the hosts. For example, when deploying a Windows machine, by default, there are very few rules allowing additional traffic on the host [47]. Most of the services are also blocked and unavailable. In such a case, there are fewer to no ports open, which leads to fewer points of exploitation. Furthermore, an antivirus solution on the endpoint is also by default enabled [49], which can block some attacks before they are processed by the tested solution.

Consequences: It is very likely that some attacks will not succeed and will be directly blocked by the system when the default settings are used. If an XDR solution is installed in such an environment, it will observe less malicious actions, as the firewall and the defender will block some attacks directly. Hence, when testing the solution, we would not observe how the tool handles some attacks, as we will not be able to launch them. However, when the rules are changed and additional traffic is allowed, and when the antivirus is disabled, more attacks can reach the target, and more exploits are possible.

Mitigation: When deploying hosts in a simulated environment created for testing, all of them will come with the default security settings. Hence, to allow all the attacks to be observed by the solution installed in such an environment, a specific configuration of the security settings is needed to allow the attacks to perform and reach the target, and the antivirus needs to be disabled. This way, we can monitor whether the solution detects, blocks, and remediates each attack we want to test.

B2. Configuration of the system. A modified system represents a system with additional applications installed and different services configured, which deviate from the system's default configuration. When a host is deployed in a new environment, it will come only to the default applications of the OS. These applications are usually patched for known exploits if the latest OS version is installed. This means that the only exploitable features would be services or misconfigured permissions. The attacks used for testing an XDR solution are more extensive and may require additional applications installed or services configured (i.e., SMB).

Consequences: If the system is in its default settings, then some attacks will not succeed, as they do not have their specific target. In this way, we will not be able to test the performance of the tool

altogether because the specific attacks cannot produce damage to the system.

Mitigation: In order to correctly measure how the solution evaluates the attacks, we need to enable the conditions necessary for the attacks to reach their goals. Therefore, installing vulnerable applications and services that correspond to the selected attacks is essential.

B3. Presence of baseline profile. An XDR solution has to observe the system and the present patterns to create a baseline profile and train its anomaly detection capabilities. When there is no data that the tool can ingest, and the tool is tested immediately after deployment, some parts of the attacks may go unnoticed due to the behavior not being considered suspicious. However, if the tool observes a pattern in the user behavior, it may flag some activity if it seems abnormal. Flagging this activity can help detect zero-day attacks and stealthier actions, which may pass unnoticed due to their resemblance with standard actions.

Consequences: We can assume that most of the detection and response is done using signatures when there is no baseline profile created. Based on this assumption, we consider that the solution will perform worse until the baseline profile is created. For instance, behavior that is very similar to typical behavior will likely go undetected. Moreover, an attack dataset also contains zero days for complete testing, and stealthier techniques are used for testing the solution. Testing the solution before the baseline profile is used can lead to a lower detection rate and, implicitly, a lower f-score and a lower successful response rate due to fewer attacks detected.

To support our claim, we look at the research done by Sommestad et al. [68], who determined that the detection of exploits is higher for publicly known attacks. Furthermore, there was an increase of 12% in the probability of detection for the dataset used by them. This means that the tool is less likely to detect zero-days by only using the signatures. Hence, by adding the solution's machine learning and behavior analysis capabilities, we add a new layer of detection which should increase the detection probability, as well as the detection rate and the metrics influenced by it.

Mitigation: To use the solution at its peak performance, a tester has to wait until the baseline profile is created. Hence, the solution must train in the environment for at least three months [51, 48], if not otherwise stated, and it should train with various traffic to ensure that a valuable baseline is created.

B4. Testers' skills. The distribution of time saved by automation and the distribution of time saved by suggestions are two metrics that depend on the opinion of security operators. These two metrics measure how much time is saved by the automated processes and, respectively, by the suggestions. Measuring this saved time is done by asking multiple security operators and computing an average. However, the opinions can vary substantially from a lower-skilled security operator to a higher-skilled security expert.

Consequences: A suggestion that may be obvious for a skilled analyst could be valuable for a less skilled one, thus saving time. On the other hand, a suggestion that would appear apparent to a skilled analyst and save time could be obscure to an unskilled one, thus having a lower influence on the metric. Hence, the computed metrics are influenced by these skills.

Mitigation: To ensure that the difference in the metrics between testing and actual deployment is as small as possible, we propose computing these metrics using the opinions of an equal number of high-skilled and lower-skilled security operators.

B5. Distribution of the testing dataset. The testing dataset contains benign and malicious samples when testing an intrusion detection and response solution. However, this dataset does not represent the actual distribution of the traffic encountered in a real setting, where the number of malicious data samples is much lower than that of regular, benign traffic.

Consequences: We can observe in the paper by Mimura [32] that the number of benign samples compared to the malicious samples can affect metrics such as the F-score. When performing binary classification, this metric decreases in value with the increase of the benign samples in the testing dataset. This decrease usually appears due to a higher number of false positives, which will likely increase if the benign sample size increases. This distribution also corresponds to an actual life deployment.

Mitigation: To ensure that the computed F-score in the experiments is close to the one measured in an actual deployment, the testing dataset should contain a significantly higher number of benign samples than malicious ones. Common ratios of benign samples to malicious samples are 8:2, 7:3, and 6:4 [32]. Hence, as we aim for a higher ratio, a ratio of 7:3 or 8:2 would be desirable. These ra-

tios are also not completely representative of a production environment, but ratios common used for benchmarking malware detection solutions. The number of benign samples is much larger compared to malicious ones, as incidents occur rarely or not at all in an organization. However, we consider difficult to reproduce the real ratio in a testing environment, so the proposed ratios should mitigate to some extent the unbalance between samples. These samples can be collected from a dataset, or the user interaction with the system can produce them. The malicious samples should also be mixed with the benign ones.

B6. Quality of training data. The environment where the solution is tested is usually isolated and controlled. The testers carefully select the data seen by the solution. Hence, the environment is not exposed to unwanted threats, only to the ones used for testing.

Consequences: If the solution is left to train in this environment for a longer period, which is required for setting up the baseline profile, it will likely observe only a limited set of data. The solution will consider the pattern of the user as the baseline, and other actions, which may be benign, might be considered malicious. Therefore, the tool will likely over-detect, which leads to a high increase in false positives and a lower f-score.

Mitigation: To minimize this overdetection, we consider training the solution with various and more realistic data.

B7. The complexity of the environment. The scale of the environment is one of the most apparent differences between a real environment and a testing or virtual one. An organization can have hundreds or thousands of devices connected in multiple VLANs and subnets. In addition, multiple users interact with the system, each one in a different way. In the case of an attack, several points need to be investigated and remediated. This is compared to a scaled-down environment containing only a few devices and generally having a much simpler structure. Hence, in an infrastructure with many devices, an analyst must triage and investigate the events from plenty of sources. Moreover, a large number of generated events can also cause scalability problems. For example, alerts from all the hosts are put together in a single platform and will likely create a cluttered view. This is a problem that would not occur in a simpler environment. Finally, in an environment designed for testing, the probability of an intrusion occurring is usually high because most of the total events presented in the solution are an intrusion. However, this probability is usually very low in a natural environment, as incidents do not occur very often.

Consequences: We assume that the investigation time increases with the number of targeted and affected devices. This is because investigating the events needs to be done for each device. Even if, after investigating the first devices, an analyst knows how to investigate the events for the following ones, it still takes some time to go through all of them. However, XDR solutions claim to help analysts in such cases and offer better visibility across multiple sources of data. Hence, we assume the tool's capabilities can reduce this bias.

Moreover, a cluttered view of the events can influence the investigation time because the analysts must investigate the affected hosts and the alerts. It may also take longer to collect the information from thousands of devices than from dozens or hundreds of them. Koutepas et. al [24] present common issues encountered in distributed intrusion detection systems. We look at their work as distributed IDSs have an architecture similar to an XDR solution covering many organizational devices. One of the issues they present is communication between components, where they highlight the problems of aggregating and analyzing all the information collected. Due to different configurations, extra traffic can be generated, slowing down the systems. The aggregated information may also not always be processed correctly. This supports our assumption that many devices can cause scalability issues.

Finally, the probability of intrusion, also known as the base fallacy rate, can influence several metrics for testing intrusion detection and response systems. Hence, these metrics can have different values depending on the environment in which they are computed.

Mitigation: To avoid the bias created by the base rate fallacy, we use metrics not affected by it, such as the F-score. To mitigate the bias created by scalability, we propose collecting the data from the small environment and then scaling it linearly to the real environment. We can reproduce the organization's network structure in the testing process but have only one or two devices in each segment. Each of these devices should represent a target for the attacks. Then, one can collect the alerts observed on one device and multiply them by the number of devices in the represented network fragment. Then, the metrics can be computed with the scaled numbers.

B8. Attack dataset coverage. This bias refers to not using an extensive set of attacks for testing the solution, compared to using a complete one. Ideally, a tester wants to observe how the tested solution evaluates all the attacks which can threaten an organization. A complete attack dataset would contain different techniques of an attack that lead to the same goal. It would contain stealthier approaches, diverse launching methods, and obfuscation techniques. Zero-day exploits are also essential in such a set, as they represent a real threat to an organization. However, only some attacks are used in natural test settings due to resources and time constraints. This attack set may have each attack type represented by one instance, but the solution may handle several variations and techniques differently. Zero days are also hard to acquire, solely for testing purposes.

Consequences: Missing a complete attack dataset would create a bias in the detection rate of the solution. For example, the selected dataset used for testing may be composed of attacks that the tool can easily recognize and knows how to handle. In this case, the tool will present a high detection rate. On the other hand, if the dataset contains only attacks that the tool does not handle properly, the tool will present a lower detection rate. We consider that the number of false positives should not be impacted by this aspect, as the benign traffic is not changed. Hence, the F-score will increase and decrease with the detection rate (recall). In both cases, the attack dataset does not represent the tool's performance due to a bias in the selected attacks. The dataset used can also influence the distribution of the time saved by automated features and suggestions for handling the attacks. For example, if fewer attacks are detected, fewer automated actions are taken, and fewer suggestions are proposed. As a result, we would observe less time saved for most of the attacks, suggesting that, in general, the tool makes the analysts' work more efficient. However, suppose the detection rate is high. In that case, it will likely be observed that plenty of time is saved across the attacks, possibly overestimating the efficiency created by the tool.

Mitigation: A dataset covering as many attack types and techniques is required to combat this bias as possible. To investigate and gather as many attack techniques as possible, we can use the MITRE ATT&CK [33] framework. Then, using these techniques, we can create an attack dataset that satisfies them.

B9. Presence of novel attacks. Attack datasets used for testing purposes are generally made of known attacks and exploits, which are available to testers. Only some of the newer publicly available datasets contain zero-day attacks, such as the ADFA-LD and ADFA-WD datasets. However, it is generally difficult to acquire or create instances of zero-day malware or exploits solely for testing purposes. Signatures should be able to detect most known attacks, especially if their release date is sufficiently far from the testing date. This is because, in such a case, signatures are very likely created to detect the attacks.

Consequences: According to Sommestad et al. [68], new exploits are less likely detected by these signatures. Therefore, we assume that using a dataset that only contains sufficiently known attacks increases the detection rate. Novel attacks, or zero-days, are more likely to be detected using the machine-learning capabilities of the solution, but there is no way to predict how well such attacks are detected in this way. However, some studies have proposed IDSs with higher detection rates, even with zero days samples. For example, Kaur et al. [22] propose a system that appears to achieve, in a worst-case scenario, a detection rate of 89%. Kanellopoulos et al. [20] also present a framework for zero-day vulnerability detection, which showed a detection rate of 96% in the experiments. The tests are done using zero days malware samples, but the types of malware used and how different they may be are not mentioned in the paper. We consider that some zero-day exploits and novel attacks may be sufficiently different compared to previously known attacks, and whether they will be detected is unpredictable. The machine learning algorithm used for detection can also vary from one solution to another. Moreover, looking at the analyzed solutions, there is no documentation regarding the technology used.

To test whether our assumption holds or not and possibly under what conditions, we calculated how the detection rate could be influenced by the zero days. We took the detection rate x of a dataset with n known attacks and the detection rate y of a dataset with m zero days. We analyzed whether combining these datasets yield a new detection rate and whether it is higher than the actual rate x. This means testing whether our assumption holds and discovering in what conditions it does. We used different ratios between the attack samples (known attacks to zero days), different numbers of attacks, and different detection rates for both types of attacks. We observed that the number of samples does not impact the results. We also discovered that the ratio between the number of known attacks and zero days does not influence the results. Our final observation was that the hypothesis

holds when the detection rate of the known attack dataset is higher than that of zero days. Hence, we can conclude that a set of both zero days and known attacks will present a higher detection rate only if the detection rate of the zero days is higher than that of known attacks. Although, as in most cases, this is an unlikely scenario, we consider that a combined dataset leads to a lower detection rate than one made solely of known attacks, which leads to a lower f-score.

Mitigation: The best method to reach a realistic detection rate is to integrate zero-days or novel attack samples in the testing dataset. However, we are aware that this is not always possible, as we mentioned that these samples are not easy to acquire. Therefore, we propose assessing how efficient the machine learning capabilities of the tool are in detecting novel attacks by looking at how the tested attacks are classified. This means observing whether they were detected by a specific signature, or by suspicious behavior. This is a sufficient resemblance of how novel attacks or zero days would be detected.

B10. Presence of stealthier attack techniques. In a real scenario, the attackers aim to be as stealthy as possible to avoid being detected by security products. Therefore, they use evasion methods, such as obfuscation, fragmentation, and encryption. However, for testing purposes, such techniques are rarely used. This is because handcrafting scripts and obfuscating known attacks or exploits is a tedious process, and sometimes it is out of the scope of the testing. However, we consider it essential to evaluate how the tool handles the attacks in such conditions. In general, the attacks are deployed using standard frameworks, such as Metasploit [29] and Cobalt Strike [7]. Such tools and the payloads they create may have a fingerprint that allows them to be detected by the solutions [45, 55]. They also use exploits that can be found easily in an exploit database, for which signatures are generally available.

Consequences: By only using such attacks, it is likely that more attacks will be detected compared to a dataset containing stealthier attacks. This would lead to a higher detection rate for the dataset without stealthier techniques.

Mitigation: To observe the real distribution of attacks detected, a dataset that contains at least some of the described techniques is required.

5. A framework for benchmarking XDR solutions

5.1. Capability selection guidelines

Our framework aims to evaluate XDR solutions' response capabilities from a cost perspective, to allow organizations to observe whether implementing such a solution would benefit them. Hence, we propose the following capabilities of the XDR solutions to be tested:

C1. Automate parts of the response actions
C2. Aggregate and correlate events from multiple sources in an unified view
C3. Present an overview of the automated investigation and response actions, as well as the threat intelligence required for responding to an attack
C4. Use behavior analysis and machine learning to identify hidden and unknown threats

Each capability covers a different aspect of an XDR solution's response. Hence, depending on what an organization wants to observe, it can select the relevant capabilities from the given list. We define four evaluation objectives, each containing a feature to be observed. These are the options the organization can select to be tested.

O1. Evaluate how the automated features make analysts' work more efficient
O2. Evaluate whether the solution can efficiently stop attacks before impact occurs using the automation features
O3. Evaluate how the triaging capabilities of the tool help an analyst
O4. Evaluate how the suggested investigation and response actions make the security operators' work more efficient

Each option corresponds to one or two of the defined capabilities. By choosing O1, then capability C1 is evaluated. By choosing O2, capabilities C1 and C4 are together evaluated. By choosing O3, C2 and C4 are evaluated. Finally, by choosing O4, C3 is evaluated. Different combinations of the presented options can be used, depending on the testing goals. Moreover, based on the selected options, different metrics are used for the evaluation, which are presented in the following section.

5.2. Benchmarking metrics selection guidelines

The metrics we use for our framework align with the capabilities we want to test and with how we want to evaluate them. We defined this metrics such that they evaluate the aspects of XDR solutions concerning response and efficiency. We present the metrics used, their definition, their formulas, and the methods needed to compute them in Table 5.1. For each of these metrics, we define the methods found in Table 5.3 for computing them. Along with the methods, we also present in the table which of the sub questions we previously defined does the method answer. Finally, the metrics corresponding to each feature and capability can be seen in Table 5.2. This table represents the guidelines for the metric selection, depending on the feature selection.

The **F-score** evaluates the detection capabilities of the tool, as the response is first dependent on the detected events. Using it, we want to observe how well the behavior analysis and machine learning capabilities (C4) and the signatures handle various threats. The **time of triage** measures the average time needed to triage an event using the given solution. Hence, it evaluates whether aggregating and correlating events from multiple sources (C2) leads to a time of triage sufficiently small for the standards of the organization. The **distribution of the time saved by automation** evaluates how much time is saved by the tool's automation features (C1) in handling attacks. It is a plot which outlines the percent of time saved per each incident, to observe how the distribution of time saved is across different types of attacks, and whether the tool has a bias towards specific attacks. The **distribution of time saved by the suggestions** evaluates how much time is saved by the investigation and response actions suggestions (C3) in handling attacks. This metric is similar to M3, as we plot the percent of time saved per each incident, but this time by the suggestions offered by the solution. We aim to observe the distribution of time saved across different types of attacks, and whether the tool has a bias towards specific attacks. The **rate of successful responses** measures the number of successfully blocked attacks. This metric evaluates a combination of C1 and C4, as we want to observe how many

Metric	Definition	Formula	Methods
M1. F-score	The harmonic mean between precision and recall	$\frac{2*Precision*Recall}{Precision+Recall}$	Method 1
M2. Time of investigation	The average time needed to investigate an event	$\frac{\sum_{i=1}^{n} triagetime(E_i)}{n}$	Method 2
M3. Rate of successful responses	The rate of successfully blocked attacks, compared to the total number of attacks	$\frac{successful\_responses}{\#ofattacks}$	Method 3
M5. Distribution of the time saved by automation	A plot of the percentages of time saved for each event by the automated features	Create a bar plot of $savedtimeautomation(E_i)$	Method 4
M4. Distribution of the time saved by suggestions	A plot of the percentages of time saved for each event by the automated features	Create a bar plot of $savedtimesuggestions(E_i)$	Method 5

Table 5.1: Overview of the metrics used in the framework

Feature	Capabilities	Metric
O1. Automated features	C1. Automate parts of the response actions	M4. Distribution of time saved by automation
O2. Efficient response actions	C1. Automate parts of the response actions C4. Use behavior analysis and machine learning for threat detection	M3. Rate of successful responses
O3. Investigating the events	C2. Aggregate and correlate events from multiple sources C4. Use behavior analysis and machine learning for threat detection	M1. F-score M2. Time of investigation
O4. Suggested investigation and response actions	C3. Present an overview of the automated investigation and response actions	M5. Distribution of the time saved by suggestions

Table 5.2: Mapping between options, capabilities and metrics

events are successfully blocked automatically by the tool, before impact occurs in the organization.

Depending on what features one wants to measure, the metric(s) associated with the feature need to be computed. For computing these metrics, various data needs to be collected. For the F-score, the testers should collect the number of False Positives, False Negatives, and True Positives. For the time of investigation, they should document the steps required to investigate each intrusion, and estimate the time needed to carry out each step. For computing the distribution of time saved by suggestions, they should define the steps required to investigate an intrusion and the ones needed to remediate it. Then, they should analyze the suggestions offered by the tool, to observe whether they can replace some of the defined steps, or some other steps would be needed. Then, they need to estimate the time needed to perform each step. For the distribution of time saved by automation, they should document the steps required to remediate an incident without the suggestions, and compute the time needed to carry them out. Then, they should observe which of these steps are automatically performed by the tool, and how much they take. Finally, for the rate of successful responses, they must see at what stage of an attack scenario the attack is stopped by the solution. For all the time estimates, using the opinions of different security operators and analysts is required.

Method	Procedure
Procedure 1. Compute the time needed to carry a step	1. Ask j analysts to estimate how long it would take them to perform this step. Without loss of generalization, let these times be $t_1, t_2, ..., t_j$. 2. Compute the average of these times: $t_{ik} = \frac{t_1+t_2+...+t_j}{j}$ 3. The estimate time to perform $step_{ik}$ is t_{ik}.
Method 1. Compute the F-score Q: I1	1. Count the number of total incidents launched against the systems. 2. Count the number of True Positives (TP). 3. Count the number of False Negatives (FN). 4. Count the number of False Positives (FP). 5. Compute the precision: $precision = \frac{TP}{TP+FP}$ 6. Compute the recall: $recall = \frac{TP}{TP+FN}$ 7. Compute the F-score: $\frac{2*Precision*Recall}{Precision+Recall}$
Method 2. Compute the time of investigation Q: I2	1. For each event E_i: i. Document the necessary steps for investigation E_i: $investigationsteps(E_i) = \{step_{i1}, step_{i2}, ..., step_{im}\}$ ii. For each step $step_{ik}$, assign an estimate of the time it takes on average for an analyst to carry it out, using Procedure 1. iii. Compute the total time to investigate E_i: $investigationtime(E_i) = \sum_{k=1}^{m} t_{ik}$ 2. Using this information, we compute an average of the time needed to investigate all the events: $t_{investigation} = \frac{\sum_{i=1}^{n} investigationtime(E_i)}{n}$
Method 3. Compute the rate of successful responses Q: E1, E2, E3	1. For each attack scenario A_i: i. Determine the steps which are part of the attack, considering the MITRE ATT&CK framework. ii. Identify the critical step $critical(A_i)$, which is the step where the attack needs to be blocked before an impact on the system. iii. Observe whether the tool stopped the attack before or at $critical(A_i)$. If this is the case, let $blocked(A_i) = 1$, otherwise let $blocked(A_i) = 0$. 2. Count the number of attacks which have $blocked(A_i) = 1$. These attacks were successfully blocked, and we call them $successful\_responses$. Then, the attacks with $blocked(A_i) = 0$ are the attacks which are not blocked in time, called $no\_responses$. 3. We can compute the rate of successful responses with the following formula: $successful\_rate = \frac{successful\_responses}{\#ofattacks}$

Method	Procedure
Method 4. Compute the distribution of time saved by the suggestions Q: S1, S2, S3	Step 1: Compute the time needed to look through the solutions proposed, decide which are relevant and which steps are additional needed. 1. For each event E_i: i. Ask the analysts to estimate the time needed to look through the proposed options and decide which are relevant. Compute an average of these times using Procedure 1, and call this time $analysistime(E_i)$. ii. Determine which k steps can be useful from the suggestions. Then the steps used to investigate and remediate E_i are: $$suggestedsteps(E_i) = \{step_{i1}, step_{i2}, ..., step_{ik}\}$$ iii. Determine which steps are additionally needed for E_i: $$additionalsteps(E_i) = \{step_{ik+1}, step_{ik+2}, ..., step_{ik+j}\}$$ iv. Using Procedure 1, document how much time following these steps would take. Then, the time to carry out $step_{il}$ is t_{il}. v. Compute the time needed to only investigate and remediate E_i: $$invresptime(E_i) = time(suggestedsteps(E_i)) + time(additionalsteps(E_i)) = \sum_{l=1}^{k+j} t_{il}$$ vi. Compute the total time needed to solve and remediate E_i, which takes into account the time needed for analysis: $$suggestionstime(E_i) = analysistime(E_i) + invresptime(E_i)$$ Step 2: Compute the time needed to investigate and remediate an incident, without the help of the tool. 1. For each event E_i: i. Document the steps needed to investigate and remediate E_i: $$originalsteps(E_i) = \{step_{i1}, step_{i2}, ..., step_{im}\}$$ ii. Using Procedure 1, document how much time following these steps would take. Then, the time to carry out $step_{ik}$ is t_{ik}. iii. Compute the total time needed to investigate and respond to E_i: $$originaltime(E_i) = \sum_{k=1}^{m} t_{ik}$$ Step 3: Plot the percentage of time saved by the suggestions. 1. For each event E_i: i. Compute the difference between the original time and the suggestions time: $$difsuggestions(E_i) = originaltime(E_i) - suggestionstime(E_i)$$ ii. Compute the percentage of time saved by suggestions: $$savedtimesuggestions(E_i) = \frac{difusggestions(E_i)}{originaltime(E_i)}$$ 2. Using the percentage of time saved for each event, create a bar plot to observe the distribution of time saved.

Method	Procedure
Method 5. Compute the distribution of time saved by automated features: A1, A2, A3	Step 1: Compute the time needed to solve incidents manually. 1. For each event E_i: i. Document the steps necessary to resolve E_i: $$E_i = \{step_{i1}, step_{i2}, ..., step_{im}\}$$ ii. Using Procedure 1, document how much time following these steps would take. Then, the time to carry out $step_{ik}$ is t_{ik}. iii. Compute the total time needed to solve E_i manually: $$manualtime(E_i) = \sum_{k=1}^{m} t_{ik}$$ Step 2: Compute the time needed to solve the incidents using the automated features. 1. For each event E_i: i. Consider that j steps were automatically performed by the solution, while m-j remain to be done manually. Measure the total time taken by the tool to perform the automated actions for remediating E_i. Measuring the time can be done by simply using a timer while the tool performs the actions. Let this time be $automatedtime(E_i)$. ii. Compute the time remaining to perform the manual actions, using the time estimates from Method 6. Without loss of generalization, let the first m-j steps in the set be the ones that are manually done. Then, the time to manually perform the steps is: $$additionalmanualtime(E_i) = \sum_{k=1}^{m-j} t_{ik}$$ iii. Compute the total time needed to remediate E_i using the automated features: $$remediationtime(E_i) = automatedtime(E_i) + additionalmanualtime(E_i)$$ Step 3: Plot the percentage of time saved by automation. 1. For each event E_i: i. Compute the difference $$difautomation(E_i) = manualtime(E_i) - remediation(E_i)$$ ii. Compute the percent of time saved: $$savedtimeautomation(E_i) = \frac{difautomation(E_i)}{manualtime(E_i)}$$ 2. Using the percent of time saved for each event, create a bar plot to observe the distribution of time saved.

Table 5.3: Overview of all the methods used to compute the metrics, and the sub-questions answered by each method. Procedure 1 is a sub procedure used in some of the methods to compute a time estimate. All the other methods are directly used for computing the metrics.

5.3. Comparing the tested solutions

Consider that we have multiple solutions that we tested: XDR_i. Then, for each XDR_i we registered: $F - score_i$, $t\_investigation_i$, $successful\_rate_i$, $timessuggestions_i$, and $timesautomation_i$, where the last two represent the list of the $savedtimesuggestions$ for each event, and of $savedtimeautomation$ respectively. In Table 5.4 can be seen that we define for each metric a formula, to be computed for each solution. We denote these formulas as f_1, f_2, f_3, and f_4. We then consider the better solution according to one metric the one which maximizes its formula. However, we are aware that the testers are computing multiple metrics for each solution. We consider that, for them, one metric can be more

Metric	Criteria
F-score & Time of investigation	$max\{f_{1i}\}$, where $f_{1i} = F - score_i \cdot (1 - \frac{t\_investigation_i}{max_j\{2 \cdot t\_investigation_j\}})$
Rate of successful responses	$max\{f_{2i}\}$, where $f_{2i} = successful\_rate_i$
Distribution of time saved by suggestions	$max\{f_{3i}\}$, where $f_{3i} = mean(timessuggestions_i) \cdot (1 - std(timessuggestions_i))$
Distribution of time saved by automation	$max\{f_{4i}\}$, where $f_{4i} = mean(timesautomation_i) \cdot (1 - std(timesautomation_i))$
All metrics	$max\{f_i\}$, where $f_i = \frac{w_1 \cdot f_{1i} + w_2 \cdot f_{2i} + w_3 \cdot f_{3i} + w_4 \cdot f_{4i}}{w_1 + w_2 + w_3 + w_4}$

Table 5.4: How the best performing solution can be selected

important than another, depending on what features are more relevant to them. Hence, they can assign to each computed formula a weight w_i, which is represented by a natural number. For the metrics that they do not compute, this weight will be zero. For the other metrics, they can assign a weight greater than 1, considering how much they want that metric to value. Then, the formula they use can also be seen in the table, under All metrics. The solution with the highest value of f is considered to perform the best.

5.4. Test environment setup guidelines

Setting up the test environment needs to be done such that the bias in the testing results is minimized. We need to account for the differences that occur between the testing environment and a real deployment in an organization. We previously identified ten sources of bias, which can affect the testing process or the used metrics. We can divide these sources of bias in two categories: one concerning the testing environment, and one the attack datasets. In this section, we propose some guidelines for setting up the testing environment and for conducting the experiments, to minimize the bias that occurs. These guidelines are independent of the features selected to be tested. Besides this lab, an attacker machine (or multiple) are needed for launching the attacks against the testing environment. Ideally, these machines should be completely separated from the testing environment. There are no guidelines for setting these machines up, as the necessary tools installed depend on the attacks used.

Bias Overview	Recommendations
G1. Used security settings. The security settings enforced in the testing environment need to resemble the ones which appear in a real deployment.	• Enable multiple services, and configure them to be usable. These services lead to open ports and stepping points. • Make the network between the hosts resemble the fragmentation of an organization, and set up the relevant networking rules. Optionally, configure some of the hosts to be accessible from the outside. Configure the others to only be accessed internally. • Create multiple accounts on the hosts, with different privileges.
G2. Configuration of the system. The devices used for testing should resemble the organization's assets.	• Configure some of these devices for regular employee use, and have different applications available, which are used for day to day work. • Create hosts simulating servers, as well as more critical assets. These should contain "sensitive" data which represent the target of the attacker. • Create directories and files on the hosts, to simulate a relevant directory structure.

G3. Presence of baseline profile & Quality of training data. The training period has to be considered before testing the solution, and the data used for training has to be useful and realistic.	• Deploy the solution for at least three months before testing, or at least the period needed for training mentioned in the documentation of the solution. In this period, the tool should monitor the normal behavior of the infrastructure. • In this period, produce realistic data to be ingested by the tool. The normal user behavior on the system needs to be reproduced, and it should be sufficient to create an useful baseline profile. This means that the behavior should not be simplistic, because otherwise the tool may classify everything as anomalous. • Use both regular users and administrative users, to differentiate between the actions they perform. • Ideally, some users should use the system in this period, and perform their daily activities on them, as if they were the employees using the system. Otherwise, use scripts which automate user behavior.
G4. The complexity of the environment. The testing environment needs to resemble as much as possible the complexity of the real organization's environment.	• As emulating the number of devices present in the organization is an almost impossible task, aim for an environment which at least represents the network segmentation of the organization, or a representative part of the network, and has one or two representative devices in each segment. • Make all the devices candidates to being targets for the attacks. All of them should also be monitored by the tested solution. • Scale the collected alerts from each segment to the actual number of devices in the organization's segment. Compute the metrics using these numbers.
G5. Testers' skills The security operators interviewed for computing the metrics need to have on average medium skills.	• Interview the same number of high skilled security operators as the one of low skilled to keep the experience at a medium level.

Table 5.5: The guidelines proposed for remediating the bias produced by the different sources

5.5. Attack dataset selection guidelines

Selecting the attack dataset used for benchmarking the solution is an essential part of the framework, as using the wrong dataset can yield biased results. In this section, we propose our guidelines for creating the attack dataset, considering the sources of bias we identified which concerns the datasets.

Bias	Guidelines
A1. Complete testing. An extensive attack dataset is required, to ensure that the XDR solution is tested against a broad range of attacks. Hence, the attack dataset needs to cover all the relevant attack scenarios and techniques.	• To determine all the attacks that should be part of the dataset, perform a reconnaissance on the current attack trends, goals and techniques. The relevant attacks can vary in time, as some attacks become obsolete, while new attacks emerge. Some of the attacks are also dependent on the target. • Look for different techniques for performing the collected attacks in the MITRE ATT&CK framework. • Outline the goals of these attacks also by analyzing the current trends. • Based on these findings, define a set of properties to be satisfied by the attack dataset. • Collect and/or create the attacks scripts and samples based on the defined properties. Additionally, use available testing frameworks, such as Cobalt Strike [7], Metasploit [29], and Infection Monkey [46].

A2. Novel attacks. Newer attacks, including zero days are also indispensable for a robust attack dataset, to determine the tool's behavior against unknown threats.	• Collect recent malware, exploits and techniques, which align with the attack properties previously defined. • For zero days, try to collect from experts as many handcrafted exploits or malware samples. • Combine the found samples with the already existing samples defined in A1. • If zero-days or novel sample attacks cannot be found, assess how the tested solution classifies the attacks. Consider all the attacks that were not detected by signatures as zero-days.
A3. Stealthier techniques. The attack techniques used should comprise both direct approaches of launching the attacks and stealthier approaches, as these are widely used by attackers in the wild.	• Investigate the techniques used for evasion from security products, such as obfuscation, fragmentation and encryption. • Incorporate samples which adhere to the found techniques in the attack dataset, and combine those with the samples defined in A1 and A2.
A4. Distribution of the testing dataset The ratio of benign samples should be significantly higher than the one of malicious samples	• Count the number of malicious samples collected in A1, A2 and A3. Collect from 3.5 to 4 times as many samples of benign workloads. • Combine the benign samples with the malicious samples.

Table 5.6: The guidelines proposed creating the attack dataset used for benchmarking

6. Practical application of the framework

6.1. Definition of the experiment goals

In this section, we present an application of our framework. The main goal is to showcase how following the procedure works on a real example, and to show the feasibility of the framework. In the experiments, we set up an Azure virtual laboratory and deploy two XDR solutions to be tested one by one. We then follow the framework for testing it, by choosing what to test, the needed metrics, and creating an attack dataset. However, due to resources constraints, we are unable to test all the capabilities, create an extensive attack dataset and to create a laboratory setup according to the guidelines. Hence, the goals of the experiments are the following:

1. Show how we select the capabilities we want to test, and implicitly the metrics to be computed, by referring to Table 5.2.
2. Deploy a testing laboratory by following the guidelines in Table 5.5. Due to resources constraints, we will only follow G1 and G2, and partially G4.
3. Create an attacker machine for launching the attacks.
4. Create a dataset of attacks by following the guidelines presented in Table 5.6. As we are unable to obtain novel attacks, we will only follow A1 and A2. The final result will not be the extensive dataset that we aim for, but a smaller example.
5. Deploy two XDR solutions.
6. Launch the attack dataset against the testing environment and collect the data needed for computing the metrics.
7. Show how we compute the metrics by following the methods described in Table 5.3.
8. Discuss the results of testing each solution.

6.2. Selection of the XDR solutions

We investigated which are some XDR solutions which have a trial period available, of at least 14 days. We looked for trial versions, as buying the license of the solution is expensive, and unnecessary for only the testing purposes. We found a few solutions which offer a trial period of 14 or 30 days. We applied to get the trial version from 5 solutions, but only 2 allowed us to get it. Some of the vendors did not answer, and one was reluctant to giving the solution to be tested for research purposes. Therefore, we were limited to testing only two solutions.

6.3. Practical application of the framework

6.3.1. Selecting capabilities and metrics

We assume that we are in the position of an organization which wants to acquire an XDR solution to replace an endpoint detection and response (EDR) solution we already had deployed. Our goal is to have a broader visibility on our infrastructure (beyond only endpoint), to make the security team's work more efficient, and to prevent an impact occurring on our assets. Based on this description, the features we wanted to test are: O2. Evaluate whether the solution can efficiently stop attacks before impact occurs and O3. Evaluate how the triaging capabilities of the tool help an analyst. Hence, we tested capabilities C1, C2, and C4: automate parts of the response actions, aggregate and correlate events from multiple sources and use behavior analysis and machine learning for threat detection. For testing these capabilities, we needed to compute the Rate of successful responses, the F-score, and the time of investigation.

6.3.2. Environment configuration

To test the solution, we created an Azure environment. As we had resource constraints, we deployed a laboratory with one VNET and two subnets, each subnet containing one device, each running Windows. We also deployed an attacker machine for running the attacks. Then, we performed the following changes on the host, to adhere to the G1 and G2 guidelines:

- modified some of the firewall rules of the targets, to have some services open, such as SMB and WMI, and to allow external traffic to reach the target
- disabled Windows Defender

- configured the enabled services (i.e., created SMB shares)
- enabled the communication between the two hosts
- made only one of the hosts accessible from the external network (the attacker)
- created an additional user account, beside the already existing administrator on each host
- created a directory structure with files, with some of the directories being visible to only the administrator

6.3.3. Attack dataset

The next step was creating the attack dataset. We started by performing an analysis on the attack trends and goals. We found that malware, and more exactly ransomware, are the most used attack vectors against an organization [54] [53]. These are followed by social engineering attacks, mostly represented by phishing [54], DDoS attacks [8] [10], and vulnerability exploits [34]. Therefore, a variety of these attack types should be part of the framework. However, going beyond these attack datasets would be ideal, to test even further the attack coverage of the tool.

Then, we looked at the current goals of the attacks. The organizations are most of the time hit by targeted attacks, and hence, it is beneficial to also look at their goals. According to TrendMicro, [61], targeted attacks' primary motives and goals are information theft, espionage, and sabotage. These include acquiring information owned by the target, monitoring the targets' activities and stealing information that these targets may have, and the destruction, defamation, or blackmail of the target. Moreover, concerning targeted attacks, we also looked at the most common techniques used for achieving these goals, by checking the MITRE framework [33]. Using it, we found that techniques such as initial access using phishing and shells, privilege escalation techniques, lateral movement, and C&C channels are all very common in multi stage attacks. Hence, we decided that these techniques should all be part of the attack dataset.

Based on all of our findings, we define the following properties to be satisfied by the dataset:

- Phishing
- Vulnerability exploitation
- Scanning
- Lateral movement
- Privilege escalation (User to Root)
- Data access and exfiltration
- Ransomware (Data encryption)
- Denial of service (System hindering)
- Bind/reverse shell (Remote operation)

The next step was finding and creating the attack samples. We found plenty of exploits to satisfy some of these properties in the Metasploit framework [29]. We created some scripts which selects the relevant exploits, and automatically launch them via Metasploit. Then, we found another open source framework for pentesting purposes, called Infection Monkey [46]. This framework uses attacks which follow the MITRE [33] framework. For each tactic, it contains different techniques that are launched on the target hosts. Besides infecting all the hosts on the targeted network, it can also be configured to perform encryption on folders, which is the goal of the ransomware. This framework can be launched automatically from the attacker machine, so it is simple to use, and it also satisfies most of our attack properties. For phishing purposes, we configured GoPhish [44] to send the emails with the malicious payloads. These payloads are usually needed by Metasploit to create the shells on the target. We also used Nessus [58] and WinPeas for an additional vulnerability scanning on the host. Finally, we also performed some manual attacks to simulate lateral movement, system enumeration, and a privilege escalation.

All the selected attacks can be seen in Table 6.1. We grouped the attacks by the corresponding MITRE technique. Moreover, as can be seen in the table, some of the attacks have sub components. These components are parts of the general attack which can be considered by themselves as standalone attacks. In the following sections, we will call the general attacks attacks, and all the attacks including their sub-components incidents.

Technique	Attack	Sub-component
Reconnaissance	Nessus Scan	
	Nmap TCP Scan	
	Exploit scan via Metasploit	
	WinPEAS scan	Enumerate domain information, Search system information, user information, processes information, services information, installed applications information, network information, windows credentials, browser information, generic files that can contain credentials, specific files that can contain credentials and for regexes inside files, Display interesting events information
Initial access	Phishing email link	
	Phishing email attachment	
Execution	CLI reverse shell	
	Meterpreter shell	
	PSExec authenticated shell	Powershell execution, Powershell execution via COMPSEC
	Execution through API and Powershell	Use WinAPI to aquire system singleton for monkey process, Run powershell to perform post breach actions
	Scripting	Use scripts for post breach actions
	Service execution	Create a service via MS-SCMR
Persistence	Create a scheduled task	
	Create account via net	
	Hidden files and directories	Create and modify hidden files and directories
	Modify powershell profile	
Defense evasion	Modified a file's modification timestamp	
Discovery	Data discovery	
	System enumeration	System owner user discovery, Hostname discovery, Discovery via registry, Account discovery, System discovery WMI reconnaissance , Password lookup using findstr, System discovery via sc, Ns lookup execution, Service information discovery via systeminfo
	Network scan via host	
	System info discovery	Network connections, Process discovery
	System network discovery	Network connections via netstat, Network interface info
Privilege escalation	Sticky keys process	
	Existing process migration	
Credential access	Credential dumping	Hash dump, Collect SMB hashes
	Bruteforce	SMB exploit using bruteforce
Lateral movement	PSExec	Remote execution of CLI via PSExec, PSEXESVC execution, Executable file in admin share
	Remote services	SMB exploit
	Remote file copy	Transfer file via share
Data exfiltration	File transfer	
	Exfiltration over C&C	
Impact	DDoS	
	Ransomware	Encrypt files via bitflip, Change files extension
C&C	C&C communication	Communication via port 5000

Table 6.1: Overview of the attacks used, grouped by MITRE technique

Attack scenario	Attacks
Attack 1	Nessus scan, Phishing email attachment, Meterpreter shell, Exploit scan via Metasploit, Data discovery, System enumeration, Privilege escalation with sticky keys, Lateral movement with PSExec, CLI reverse shell , System discovery, Data discovery, File transfer
Attack 2	Nmap scan, Phishing email link, Meterpreter shell, System discovery, WinPEAS scan, Privilege escalation with existing process migration, Hash dump, PSExec authenticated shell, Data disocvery, File transfer
Attack 3	Execution through API and powerhsell, Scripting, Service execution, Persistence actions, Defense evasion actions, System discovery 2, Collect SMB hashes, SMB exploit using bruteforce, Lateral movement via SMB - transfer malicious payload via share, Exfiltration over C&C, Ransomware
Attack 4	Nmap scan, DDoS

Table 6.2: Attack scenarios

Moreover, as we have in scope targeted attacks, we grouped the selected attacks in four attack scenarios. Two of them have the scope of data exfiltration, but using different techniques to achieve that. One of is made by all the steps from Infection Monkey, with the goal of ransomware, and the last one aims to create a service disruption via a DDoS attack. The components of each attack scenario can be observed in Table 6.2.

6.3.4. Launching the attacks and collecting results

We created a procedure first to launch the scripted attacks, including the ones using Metasploit, and then launch the attacks from the Infection Monkey platform. We called the first set of attacks the manual attacks and the other the monkey attacks. For the manual attacks, we created a script that launched a tmux session with four windows. In the first one, we could choose the script we wanted to run and follow the prompts, and in the second one, we could see the script's output, such as the choices for Metasploit exploits. We could navigate to the second window and run other commands in the msfconsole, which is the console used by Metasploit. We used the remaining two windows for the output generated when starting services, such as Nessus, GoPhish, and Infection Monkey. Using the windows helped us visualize the attacks and interact with them if some of the scrips did not work correctly.

To launch the monkey attacks, we had to install Infection Monkey on the attacker's machine and then configure its settings. We could select the used exploits, the targeted network and IPs, and the folder path to be encrypted by ransomware, and we could add credentials and passwords for bruteforcing. This configuration could be exported and imported after each reset of the environment. After running the manual attacks, we started the Infection Monkey server, imported the configuration file, and deployed the attacks.

We deployed the endpoint agents from the tested XDR solutions on the hosts. Luckily, they did not interfere, so we could test both solutions simultaneously. We then enabled all the features available on the tools to maximize the detection performance. For each solution, we had a dashboard to view the alerts, which we monitored during the attack dataset deployment.

For each incident, we collected all the generated alerts corresponding to them. All the alerts were displayed in a monitoring dashboard, and they were assigned one of the following security scores: Info, Low, Medium, High, and Critical. Besides the score, each alert was assigned a description of the possible attack and a mapping to a MITRE sub-technique. Moreover, from each alert, possibilities for further investigation were provided. We investigated these alerts to map them to the corresponding incident. Moreover, if there were duplicates of alert names and they were referring to the same incident, we counted all of them as one alert, as we wanted to count how many incidents raised alerts and not how many alerts were generated in total.

We considered only the alerts ranging from low to critical as True Positives and the ones only classified as info as False Negatives. We did this because the analysts usually focus on the higher severity alerts in a real environment with numerous alerts. In addition, info alerts are also usually associated with

Numbers	Solution 1	Solution 2
Total incidents	62	62
Total observed incidents (incl. info)	37	18
Total incidents that generated alerts	25	12
Total undetected incidents	25	44
Total events observed	73	64
True Positives	25	12
False Positives	11	2
False Negatives	37	50
Recall	0.403	0.193
Precision	0.694	0.857
F-Score	0.510	0.315
Average time of investigation	244s	250s
Successful responses	2	0
No responses	2	4
Rate of successful responses	0.5	0

Table 6.3: Collected results

system information. Moreover, the False Positives were the alerts displayed in the alerts dashboard that did not correspond to an intrusion. We also counted multiple alerts pointing to the same event as only one. Finally, the False Negatives were represented by the number of incidents not detected by the solution.

Then, we simulated an investigation process for each attack that generated alerts to estimate the investigation time. The investigation process was done for the attack and not for the incident. However, we used the incidents for correlation to conclude what attack was happening. We did this simulation by using the features available on each solution.

Finally, we monitored the actions taken by the solution to respond to the ongoing attacks. One solution proposed for some events actions to be taken directly from the console to block the attack. It did not perform the response actions automatically but required manual approval. However, it was straightforward to carry out these actions, as they required only a click in the console, so we considered them relevant for blocking the attacks. The other solution did not block any attacks, as it required external connectors to do that, which we did not have. We investigated those connectors, and they were able to provide automated responses such as blocking attacks, isolating endpoints, and quarantining files and e-mails. These were the same actions that were taken by the first solution to respond to attacks. Nevertheless, we could not evaluate their performance.

6.4. Benchmarking results

6.4.1. Comparing the tested solutions

The first step of comparing the performance of the two tested solutions is computing the targeted metrics: F-score, time of investigation, and rate of successful responses. To compute these, we used Method 1, Method 2 and respectively Method 3. Following Method 1 consisted of collecting all the True Positives, False Positives, and False Negatives, and then using the formulas for computing the F-score. However, for Method 2, we could not find in a short time analysts available to investigate the incidents using the tested solution, so we simulated an investigation ourselves, and estimated the times using our experience as a Tier 1 analyst. Hence, we documented the steps necessary for carrying out the investigation, and assigned to each step the time it took us to follow it. We then computed the investigation time for each event, and computed the average. For method 3, we determined the steps of each attack scenario when we built the the attack dataset. The next step was determining for scenario the critical step, and observing how many of those were successfully detected and blocked by the solution. The results and the metrics computed for each solution can be seen in Table 6.3.

The next step consists of comparing the solution using the computed metrics. Using the F-score and the time of investigation, we compute f_{11}, and f_{12}. The highest time of investigation for this experiment was the one of Solution 2, 250.9 seconds. We use the double of this value to scale the time of investigation for each solution, and compute their individual products:

$$f_{11} = 0.510 \cdot (1 - \frac{244.772}{2 * 250.9}) = 0.261$$

$$f_{12} = 0.315 \cdot (1 - \frac{250.9}{2 * 250.9}) = 0.157$$

Based on how efficient the tool is investigating the events, we can conclude that Solution 1 performs better than Solution 2. Regarding efficient response actions, one of the solution has a rate of successful responses equal to 0.5, and the other 0, which means that $f_{21} = 0.5$ and $f_{22} = 0$. Based on this evaluation objective, Solution 1 still performs better than Solution 2.

We can assume that for the organization, the efficient response actions is a more important aspect then efficiency of investigating the events, and they give the rate of successful responses a weight of 2, and to the F-score & Time of investigation a weight of 1. The other metrics are given a weight of 0, as they are not of importance to the organization. Then, we compute f_1 and f_2 the following way:

$$f_1 = \frac{1 \cdot f_{11} + 2 \cdot f_{12} + 0 \cdot f_{13} + 0 \cdot f_{14}}{1 + 2 + 0 + 0} = \frac{0.261 + 0.5 \cdot 2}{3} = 0.420$$

$$f_2 = \frac{1 \cdot f_{21} + 2 \cdot f_{22} + 0 \cdot f_{23} + 0 \cdot f_{24}}{1 + 2 + 0 + 0} = \frac{0.157 + 0}{3} = 0.052$$

Considering all of the computed metrics, we can conclude that Solution 1 performs better than Solution 2 for the organization, presenting a higher score.

6.4.2. Results validity

Even if we did not implement all the guidelines, we consider that we reduced some of the bias created by testing the solution in a small environment. We could launch all the attacks by configuring the firewall rules and disabling the antivirus on the hosts. Moreover, by configuring the WMI and SMB services, we allowed some attacks, such as the lateral movement ones, to be successful. These configurations allowed us to test all the attacks from our dataset, which reduced the *used security settings* and *configuration of the system* biases.

Concerning the attack dataset, we covered all the attacks we found to be relevant at this moment. Hence, we tested how the solutions handle all these attacks, mitigating the *attack dataset coverage* bias. We mostly used attack samples from testing frameworks [29, 46], but some of those used obfuscations techniques. Therefore, we could also observe whether our assumptions regarding the fingerprints left by these framework holds. For example, one of the solutions flagged the use of meterpreter, known to be used by Metasploit. It also partially flagged the Infection Monkey framework: it noticed that a framework was used, but it considered it CobaltStrike, a better-known framework. As Infection Monkey is newly developed, it may use behavior similar to CobaltStrike, which may have led to this result. Even so, only some of the incidents launched using these frameworks were detected, and even fewer were flagged to be part of the frameworks. The solutions also noticed the use of obfuscation techniques, such as the use of encrypted PowerShell script. These techniques allowed some of the attacks possibly not to be detected. Hence, we also partially mitigated the *presence of stealthier attack techniques* bias by using these attacks.

There are two biases concerning the testing dataset which we did not address with our application: *presence of novel attacks* and *distribution of the testing dataset*. We could not acquire novel attack samples, but we observed that the detection was done only with signatures at the moment of testing. Hence, we consider that even if we acquired such attacks, most likely, they would not have been detected, as the detection rules would not have captured them. Moreover, we could also not inject benign data, as we did not have any samples of benign data available. However, to measure the number of false positives, we performed some actions on the system during the testing time frame. Even so, the number of benign samples is much lower than in a production environment. Hence, the number of measured false positives may be smaller than the actual one observed in our testing process.

Finally, we did not address the biases of *presence of baseline profile*, *quality of training data*, and *complexity of environment*. We did not have time to train the solutions properly, but the trial versions did not seem to present any anomaly detection capabilities. All the detection was done using detection rules, and we do not know whether the solutions we tested do not have machine learning and behavioral analysis capabilities or were not available in the trial version. We consider that the detection rate could have been higher by having these capabilities. Furthermore, we did not scale our data to one of a larger environment because we only had two devices with almost the same number of registered events, and scaling them would have produced the same results when computing the metrics. We would have required more target devices in multiple VLANs and subnets to make the scale valuable. Hence, the computed time of investigation may be higher in the production environment. We acknowledge that all of these are limitations to our application.

Taking all these aspects into consideration, our results concerning the performance of the tested solutions are not entirely free of bias but should be suggestive of how one solution performs compared to the other. We could only implement some of our guidelines, so the results may be affected due to some sources of bias. However, this application should suggest how the framework is used in practice.

6.4.3. Results discussion

As seen in Table 6.3, the number of detected incidents is much lower than expected. However, even if we used only the endpoint agents for both solutions, numerous incidents should have been detected at the endpoint level but were not. For instance, the malicious payloads used for the meterpreter shell or the winPEAS scan were not detected. For the winPEAS scan, both solutions just detected some actions taken by them, such as searching for user or process information. Moreover, Solution 1 listed the enabled rules used to detect incidents. However, we observed that some of the incidents that these rules should have detected, according at least to their naming, were not. One example is the "TCP scan" rule, which should have detected scans made against the host but did not detect the Nmap or Nessus scans.

In Solution 1, several rules used by other connectors, including the network one, should have detected some undetected incidents. Hence, if we were to use these connectors, it is very likely that more incidents would have been detected. We also observed some rules which corresponded to other connectors. We investigated these connectors and found that they are endpoint agents made by the same vendors, requiring additional licensing. Those were not part of the main XDR product, which consisted of the endpoint and network agents, but rather some integrations. We concluded that additional payment beyond the license was required for better detection for this solution. However, these capabilities should be part of the core product, and one should not pay additionally for better performance.

Both solutions offered helpful information for investigating the events to facilitate this task. Solution 1 provided a description for each generated alert. It also highlighted the parts in the log that were considered malicious and triggered the alert. Moreover, an execution profile was available for each process and executed command. By looking at this execution profile, an analyst could observe how the process was started and its relation to other processes. This allows the analysts to find related alerts and get a bigger picture of the intrusion. Solution 2 offered a more detailed description of the alert, highlighting the potential intrusions that could cause it. In addition, a more detailed description of the event was available, which showed related events and processes. With these features, the investigation time was minimized, as searching for additional information was rarely required.

For instance, in Solution 1, we got the alert "Sensitive content access attempted using findstr", which referred to the "Password lookup using findstr" incident. Upon clicking on the alert, we saw the description "Detects usage of 'findstr' to find sensitive information from files", which already tells us that there was a search for sensitive information. Then, if we looked at the alert's details, we saw precisely the executed command: "findstr /si password *.xml". From this, we can conclude that a search for password files was performed. We could click on this command and show the execution profile. This graph showed that this command was run via a command line (cmd) process launched via a PowerShell script. We could also see that multiple findstr commands were executed and that a related alert to the one we are investigating was "Password searching in files via command prompt". Finally, we could also see that the PowerShell process via which this command was run also has an outbound connection to an IP, indicating a C&C connection. All this information was helpful in

getting an overview of the incident occurring in the system: a remote execution on the system from which a password lookup is performed.

The automation features, even if limited, helped block some incidents, at least for Solution 1. These automation features were not enabled by default, but a playbook enabling them was required. This playbook was easy to set up, as it already had a template, and we only had to check which automated actions the solution was allowed to take. All these automated actions had to be manually approved, which we considered useful. The offered actions were: adding the process/payload to the block list, disabling the affected user account, forcing the user to sign out, isolating the affected endpoint, and quarantining the malicious email. These actions were taken only for a subset of the detected events, and we do not know precisely how the solution selected which alerts required more attention. However, they were helpful, as they stopped two of the four attack scenarios in time. For instance, for Attack scenario 3, the tool detected the PowerShell execution and added the payload to the block list, which stopped its execution. Alternatively, one could have selected to isolate the endpoint, as this was the proposed alternative, which would have stopped all the connections to and from the target.

Solution 2 required additional connectors to enable its playbooks. Templates for isolating endpoints and blocking payloads were available, but they required connection to sources from other vendors. These sources required additional licensing, and we could not use them for testing. However, we investigated the solutions and learned that the vendor is developing its connectors. Hence, the playbooks can be configured in the future without needing products from other vendors.

7. Discussion

XDR solutions are advertised as very efficient intrusion detection and response tools. They claim to collect data from multiple sources, such as endpoint, network, e-mail, and cloud, which should make them better than standalone solutions dedicated to each data source, such as EDR and NDR. This data is collected and correlated in only one console, which should simplify the security operator's work. They previously needed to use multiple products to cover the whole attack surface and eventually orchestrate the response from each of them using SOAR solutions. Moreover, their detection capabilities include behavior analysis and machine learning techniques, which should enable better detection compared to existing security detection solutions. However, there is no standard evaluation of these solutions, allowing us to observe whether these solutions are as efficient as advertised.

7.1. Research questions

In this paper, we focused on creating a framework for testing the response effectiveness of an XDR solution while reducing the bias created by testing it in a small scaled environment. Using this framework, organizations should be able to compare XDR solutions and evaluate their main capabilities, to decided whether is it worth investing in one, and if so, which one to choose. We built this framework by answering the sub-questions defined at the beginning of the paper, which in turn, answer the main research question.

To determine how to compare the tested XDR solutions, we proposed the following question:

> **SRSQ:** What suitable metrics can be used for evaluating the response capabilities of XDR solutions?

To define the most suitable metrics, we first defined the capabilities characteristic of XDR solutions. Then, based on the selected capabilities, we defined a set of features to be tested: the efficiency of the investigation, the automated features, the response actions, and the suggestions proposed by the solution. For each feature, we defined a metric or combination of metrics that we considered would best evaluate it. The metrics were inspired by existing metrics for testing intrusion detection systems, security orchestration, automation, response solutions, and incident management metrics. The metrics we proposed were the F-score and time of investigation, the rate of successful responses, the distribution of time saved by the automated features, and the distribution of time saved by suggestions. These metrics evaluate how cost-efficient XDR solutions can be and how good their detection and response capabilities are, compared to previous solutions.

Then, to determine how to evaluate the solutions in a small-scaled environment, we proposed the following question:

> **SRSQ:** Which are the factors of the evaluation environment that can create a bias in the results of the evaluation of XDR solutions?

We investigated the sources of bias that can appear when testing an XDR solution, and determined how they affect some of the metrics or the testing process. We identified ten sources of bias: modified security settings, modified system, baseline data and baseline profile, difference in security operators skills, sampling bias, biased parameter selection, the complexity of the environment, missing complete testing, missing novel attacks, and missing stealthier approaches. Hence, based on these sources, the factors that can create bias are the configuration of the environment, the training process of the solution, the skills of the testers, and the composition of the testing dataset. These are the factors which can variate from a testing process to another one, and can affect its results. We proposed some mitigation actions, which represent the guidelines for setting up the testing environment and creating the testing dataset. However, we were not able to test how much these sources of bias influence the metrics, so we can only hope that our guidelines minimize the bias as much as possible.

Finally, using our results, we can answer the main research question:

> **RQ:** How is it possible to evaluate the incident response capability an XDR solution would guarantee in a production environment by testing the solution in a scaled-down environment?

We created a framework that provides guidelines for selecting the capabilities to be tested and the benchmarking metrics, and for setting up the testing environment and building the testing dataset.

We consider that the selected metrics evaluate the incident response capabilities, and the guidelines for creating the environment and the attack dataset should mitigate as much as possible the bias that can occur between testing. Using our guidelines, the difference between the computed metrics in a production environment and a testing environment should be as small as possible.

7.2. Application of the framework results

We also showcased an application of the framework, where we compared two XDR solutions and evaluated some of their capabilities. Testing the solutions allowed us to get insights into XDR solutions' inner workings. Our application focused on testing the detection, automated response, and investigation capabilities.

The solutions we tested did not present a high detection rate, with plenty of our attacks going unnoticed. However, we did not use all the available connectors, as we did not have access to them. Using them might increase the detection capabilities, as logs beyond the endpoint are analyzed, which may present a higher indicator of an intrusion. Even so, some of our undetected attacks should have been detected at the endpoint level, which indicates that the detection capabilities of the tested solutions are not accurate. Furthermore, we only observed detection made by signatures, and we did not observe any indicators of behavior analysis. This could have happened for two reasons: either the selected solutions did not have any machine learning and behavior analysis by themselves, and they required additional connectors, or the tested solutions required training time before using these capabilities in practice. The first option indicates that the solution only uses detection signatures, which is not what XDR solutions claim. The second one supports the theory that solutions only use detection rules before creating their baseline profile.

The tested solutions also did not present built-in automation features. Instead, playbooks had to be defined for taking automated actions. Nevertheless, the tools provided templates, which made their creation simple. Even so, for one of the solutions, the playbook templates required using additional agents, which also required licensing. As automation is one of the main capabilities of XDR solutions, it should be part of the basic product without needing to purchase external products. Moreover, the same solution also used outsourced connectors for collecting data, defeating XDR's purpose. Even if XDR solutions should have as many integrations as possible, the essential parts of the tool should be from the same vendor for a smoother connection between components and to eliminate the need for additional payment. The primary data sources, such as endpoint, network, email, and cloud, should be included in the basic product because this is a part of their definition. Otherwise, there is no difference between XDR solutions and products focused on a single source, such as EDR and NDR.

The automation capabilities of the solution for which we managed to create the automated response playbook were also limited. However, this limitation came from the detection capabilities of the solution. The number of incidents for which response actions were proposed or taken was smaller than that of detected incidents. This does not entirely simplify a security operator's work; for the remaining incidents, the user would have to perform the response actions manually. This aspect is accounted in the distribution of time saved by automation. If we were to compute it, we would observe time saved for only a limited number of events.

The solutions provide helpful insights for investigation, which should simplify analysts' work. For example, one of the solutions described the possible incident that caused the alert, the indicators which caused the alert, and an execution profile. It also provided a graph for some incidents that showcased how events are related. The other solution provided an extensive event description for each alert and event details. These features should be sufficient to determine whether an intrusion occurs in the system.

At a glance, XDR solutions are not much different from combined previous security solutions. They also require integrating external sources into their main product, and sometimes, multiple vendors' products are needed for the complete protection of the system. However, the main difference is that for XDR solutions, all the events are collected in a single dashboard, which makes the work of security operators more effective. On the other hand, machine learning, behavior analysis, and automated capabilities are less effective than claimed to be, at least at the beginning of the deployment period. The observed automated capabilities are limited and launched only for selected incidents. The automated actions also require manual deployment, which is not necessarily a disadvantage but

is not part of the solution description. Therefore, the image created about XDR solutions does not necessarily correspond with reality.

The detection, investigation, automation, and suggestions capabilities, are all evaluated with our framework. After testing the solution with it, an organization should be able to decide whether the tested solutions are effective compared to the existing solutions. Hence, they can measure the detection capabilities, accounting for false positives, which can create noise, the investigation capabilities, how effective the response capabilities are in stopping the attacks in time, and how much time is saved by automation and suggestions. These are all aspects that XDR solutions claim to be better compared to separate solutions.

7.3. Evaluation

7.3.1. Feasibility

The framework can be easily used in practice, as we could apply it ourselves even in limited conditions. For example, we could set up a smaller version of the targeted environment using only limited resources, and we could follow the guidelines regarding the environment's configuration and a limited network fragmentation. Moreover, we created a dataset covering all the relevant attacks at the moment of testing, and some stealthier techniques. Finally, we selected some of the metrics from our framework which corresponded to our evaluation objectives, and computed them successfully.

We assume that an organization that wants to deploy an XDR solution has the necessary resources to manage it, which includes having a dedicated security team, and the required people to perform the testing. Hence, an organization with numerous resources can more easily follow all the guidelines from the framework: they can create the necessary number of devices and have the personnel to configure them. They can also acquire the solutions for a period beyond the trial version to train the solution successfully. Finally, they can use the security operators from their team to compute the metrics that require their expertise.

7.3.2. Usefulness

Compared to existing frameworks for testing similar solutions, our framework evaluates not only the detection capabilities but also the efficiency of the response. It also focuses on a specific security solution rather than, in general, on intrusion detection and response solutions. Hence, it can be used to test whether the XDR performance is as effective as described. Furthermore, our guidelines for minimizing the bias also offer the conditions to produce more accurate results when comparing the solutions, which is a missing aspect in similar frameworks.

We showed the framework to experts in intrusion detection and response solutions. They agreed that our evaluation objectives and metrics align with what organizations aim to evaluate. For example, they want to measure how much time these solutions save and whether the generated alerts are valuable. These two parameters are captured by three of our metrics: distribution of time saved by automation, distribution of time saved by suggestions, time of investigation, and F-score. With the first two metrics, we measure how much time is saved for different incidents using the automated features and, respectively, the suggestions proposed by the solutions. Moreover, with the time of investigation, we can observe how short the time required to triage and investigate the alerts is, using the tool's features. Finally, the F-score accounts for both the detection rate of the events, which is the rate of detected events from the total number of launched incidents, and for the precision of classifying the events, which is the rate of true positives out of the total generated alerts. Hence, it measures how valuable the generated alerts are, considering the number of true positives, false positives, and true negatives.

They also considered that the performance of XDR solutions as a new technology emerging on the market needs to be evaluated to showcase whether their advertisement corresponds to reality. Furthermore, the guidelines for setting up the environment and building the testing dataset were considered practical and important, as not many evaluations of security solutions take account of the bias that can occur. Hence, our framework can be used in practice to evaluate the response effectiveness of XDR solutions and can also be extended to other similar solutions.

7.4. Limitations

We could not evaluate the extent to which the identified sources of bias influence the metrics or the testing process, due to resources constraints. By gathering these numbers, we would have been able to provide more accurate guidelines for the environment setup and the testing dataset composition. It would have also ensured that the bias between testing processes is also as little as possible. Moreover, we would have also observed whether there are additional metrics which are influenced by the sources of bias.

Our application of the framework is also a simpler variant of a real application. Due to time and resource constrains, we could not deploy a larger environment, and we could not let the solution train. We were also not able to gather benign traffic samples, especially 4 times as many as the malicious one. For detecting False Positives, we used the data generated while using the system before, during, and after the testing. Moreover, as we could not use full license products, we were limited to testing only two solutions, which were available to us for only two weeks.

Because we set up our environment in an Azure tenant, we could also not deploy the network sensor for one of the solutions. The other solution used an outsourced sensor, which was also behind a paywall. Hence, we tested the solutions only using their endpoint detection. By also using the network sensors, we consider that we may have had a better detection from the tool. The tested solutions did also not have any integrated automated features, or settings for tuning the detection capabilities. We were able to create some playbooks, but even those were limited in the nature of actions, as some of them required the use of other security products.

Finally, our attack dataset covered all the attack types that we required, but various techniques of them were missing, as well as novel attacks. The detection of the solution was also done only via signatures at the moment of testing, so we could not assess how novel attacks would have been detected by the solution.

7.5. Future work

Our research can be extended in several ways. First, the mentioned sources of bias can be further investigated and tested using experiments. These experiments would quantify the impact of the bias on the metrics or on the testing process. Using the found results, more accurate guidelines for setting up the environment and for building the tested dataset can be proposed. Moreover, it can be quantified how much the bias is minimized using the proposed guidelines.

Then, the framework can be extended to other security solutions emerging on the market, which similarly claim to perform better than previous ones. Our framework can also be extended to measuring additional capabilities, and to using additional or different metrics. Research can be further done in choosing even more suitable metrics, which are not biased by external factors which are not considered.

8. Conclusion

In this thesis, we created a framework that aims to evaluate XDR solutions, especially from a cost perspective, which also considers its application in a small scaled environment. In other words, we wanted to create a framework that helps organizations decide whether the tested solutions bring more effectiveness to the security operators' work.

To do this, we first gathered the capabilities characteristic of XDR solutions. Then, we derived some features from these capabilities to be tested by the framework. To evaluate each of these features, we proposed a set of metrics and a method of comparing the solutions based on these metrics. Finally, we wanted to account for the bias in the testing process. To do this, we first identified the sources of bias influencing some metrics and the testing process. Hence, to ensure that the testing results are as similar as possible from one testing process to another, we proposed a set of guidelines for setting up the environment and building the attack dataset. These guidelines aim to create consistent testing across different organizations.

After defining the framework, we also showcased an application of it. We deployed a testing environment and created a testing dataset by following some of our proposed guidelines. Then, we deployed two XDR solutions and presented how we collected the necessary data for computing the metrics corresponding to the evaluation objectives. We also presented the results and how we selected the best-performing solution. Testing two solutions allowed us to gain insights into how XDR solutions perform. We concluded that their advertised efficiency only sometimes corresponds to reality. The detection capabilities were less effective than described, the behavior analysis and machine learning capabilities and the automation capabilities were limited, and the integration of multiple security products encountered plenty of difficulties.

Our work also encountered some limitations due to resource constraints. We could not test the proposed biases and how much they affect the metrics and the testing process. Moreover, the framework's application was also limited: we could not follow all of the proposed guidelines and compute all of the metrics. The tested solutions also missed some components, which may have produced more valuable results.

The framework can be improved by defining more concrete guidelines for setting up the environment and creating the attack dataset. Experiments can be conducted in the future to determine to what extent the sources of bias influence the metrics and the testing process. Moreover, further investigation can identify new sources of bias, and additional metrics can also be used to evaluate the tested solutions better. Finally, the framework can be extended to evaluate the response effectiveness of other similar security solutions.

Bibliography

[1] M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh, "Evaluation of machine learning algorithms for intrusion detection system," in *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, 2017, pp. 000 277–000 282. 3

[2] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *USENIX Security Symposium 2022*, 01 2022. 4, 21, 25

[3] S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, pp. 186–205, 12 2002. 3, 8, 9, 19, 25

[4] CAIDA, "The CAIDA "DDoS Attack 2007" Dataset." [Online]. Available: https://www.caida.org/catalog/datasets/ddos-20070804_dataset/ 4, 26

[5] A. Cardenas, J. Baras, and K. Seamon, "A framework for the evaluation of intrusion detection systems," in *Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06)*, 06 2006, pp. 15 pp. – 77. 1, 3, 9, 20

[6] Cisco, "Cisco SecureX - Cisco SecureX Data sheet." [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/security/securex/secure-x-datasheet.html?CCID=cc001528&DTID=olgmcdc001463&OID=trlsc021059 6, 7, 14

[7] Cobaltstrike, "Cobaltstrike." [Online]. Available: https://www.cobaltstrike.com/ 4, 31, 38

[8] Comparitech, "20+ DDoS attack statistics and facts for 2018-2022." [Online]. Available: https://www.comparitech.com/blog/information-security/ddos-statistics-facts/ 41

[9] G. Creech and J. Hu, "Generation of a new ids test dataset: Time to retire the kdd collection," *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 4487–4492, 2013. 4, 26

[10] CSIS, "Significant Cyber Incidents." [Online]. Available: https://www.csis.org/programs/strategic-technologies-program/significant-cyber-incidents 41

[11] Cybereason, "Extended Detection and Response (XDR) | Cybereason XDR Platform." [Online]. Available: https://www.cybereason.com/platform/xdr 14

[12] Cynet, "Cynet XDR | Autonomous Breach Protection." [Online]. Available: https://www.cynet.com/ 14

[13] M. Elhamahmy, H. N. Elmahdy, and I. A. Saroit, "A new approach for evaluating intrusion detection system," *CiiT International Journal of Artificial Intelligent Systems and Machine Learning*, vol. 2, no. 11, pp. 290–298, 2010. 1, 3, 11

[14] FireEye, "FireEye Extended Detection and Response (XDR)." [Online]. Available: https://www.fireeye.com/products/xdr.html 14

[15] Forrester, "Introducing the Forrester new tech: Extended Detection and Response (XDR)." [Online]. Available: https://www.forrester.com/blogs/introducing-the-forrester-new-tech-extended-detection-and-response-xdr-a-battle-between\-precedent-and-innovation/ 6

[16] J. Gaffney and J. Ulvila, "Evaluation of intrusion detectors: a decision theory approach," in *Proceedings 2001 IEEE Symposium on Security and Privacy*, 2001, pp. 50–61. 3, 10

[17] Gartner, "Market guide for extended detection and response." [Online]. Available: https://www.gartner.com/en/documents/4007995 1, 6, 14

[18] G. Gu, P. Fogla, D. Dagon, W. Lee, and B. Skoric, "Measuring intrusion detection capability: an information-theoretic approach," in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2006, Taipei, Taiwan, March 21-24, 2006*, 01 2006, pp. 90–101. 3, 10, 11, 19, 20

[19] C. Islam, M. A. Babar, and S. Nepal, "A multi-vocal review of security orchestration," *ACM Comput. Surv.*, vol. 52, no. 2, apr 2019. 4, 11, 12, 14, 20

[20] D. Kanellopoulos, U. Singh, and C. Joshi, "A framework for zero-day vulnerabilities detection and prioritization," *Journal of Information Security and Applications*, vol. 46, pp. 164–172, 04 2019. 30

[21] G. Karantzas and C. Patsakis, "An empirical assessment of endpoint detection and response systems against advanced persistent threats attack vectors," *Journal of Cybersecurity and Privacy*, vol. 1, no. 3, pp. 387–421, 2021. 4

[22] R. Kaur and M. Singh, "A hybrid real-time zero-day attack detection and analysis system," *International Journal of Computer Network and Information Security*, vol. 7, pp. 19–31, 08 2015. 30

[23] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, 12 2019. 4, 26

[24] G. Koutepas, F. Stamatelopoulos, and V. Maglaris, "Efficiency and performance issues in distributed intrusion detection systems," in *Applied Telecommunication Symposium*, 04 2002. 29

[25] G. Kumar, "Evaluation metrics for intrusion detection systems - a study," in *International Journal of Computer Science and Mobile Applications, Vol.2 Issue. 11, November- 2014, pg. 11-17*, 2014. 9

[26] L. Laboratory MIT, "DARPA/KDDCUP 99 dataset." [Online]. Available: https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset 1, 4, 26

[27] McAfee, "MVISION XDR - Extended Detection & Response | McAfee." [Online]. Available: https://www.mcafee.com/enterprise/en-us/solutions/xdr.html 14

[28] P. Mell, V. Hu, R. Lippmann, J. Haines, and M. Zissman, "An overview of issues in testing intrusion detection systems1," *NIST Interagency/Internal Report (NISTIR)*, 01 2001. 4

[29] Metasploit, "Metasploit." [Online]. Available: https://www.metasploit.com/ 4, 26, 31, 38, 41, 45

[30] Microsoft, "Microsoft 365 Defender - Threat Protection | Microsoft Security." [Online]. Available: https://www.microsoft.com/en-us/security/business/threat-protection/microsoft-365-defender 6, 7, 14

[31] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, "Evaluating computer intrusion detection systems: A survey of common practices," *ACM Comput. Surv.*, vol. 48, no. 1, sep 2015. 3, 8, 9, 19, 20

[32] M. Mimura, "Impact of benign sample size on binary classification accuracy," *Expert Systems with Applications*, vol. 211, p. 118630, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422016773 28

[33] MITRE, "MITRE ATT&CK." [Online]. Available: https://attack.mitre.org/ 17, 30, 41

[34] NIST, "Cvss Severity Distribution Over Time." [Online]. Available: https://nvd.nist.gov/general/visualizations/vulnerability-visualizations/cvss-severity-distribution-over-time 41

[35] Nmap, "Nmap." [Online]. Available: https://nmap.org/ 26

[36] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "Tesseract: Eliminating experimental bias in malware classification across space and time," in *USENIX Security Symposium*, 2019. 4, 25

[37] Atlassian, "How to choose incident management KPIs and metrics." [Online]. Available: https://www.atlassian.com/incident-management/kpis 11

[38] ——, "MTBF, MTTR, MTTA, and MTTF." [Online]. Available: https://www.atlassian.com/incident-management/kpis/common-metrics 11, 12

[39] Cisco, "What is Network Detectio and Response." [Online]. Available: https://www.cisco.com/c/en/us/products/security/what-is-network-detection-response.html 12

[40] Crowdstrike, "What is XDR." [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/what-is-xdr/ 6, 7

[41] Cynet, "Security Automation: Tools, Processes and Best Practices." [Online]. Available: https://www.cynet.com/incident-response/security-automation-tools-process-and-best-practices/#heading- 23

[42] Gartner, "Security Orchestration Automation and Response." [Online]. Available: https://www.gartner.com/en/information-technology/glossary/security-orchestration-automation-response-soar 12

[43] ——, "What is Network Detection and Response." [Online]. Available: https://www.gartner.com/reviews/market/network-detection-and-response 12

[44] Gophish, "Gophish." [Online]. Available: https://getgophish.com/ 41

[45] Hatching, "Analyzing metasploit payload." [Online]. Available: https://hatching.io/blog/metasploit-payloads/ 31

[46] Infection Monkey, "Infection Monkey." [Online]. Available: https://www.akamai.com/infectionmonkey 4, 38, 41, 45

[47] Microsoft, "Best practices for configuring Windows Defender Firewall." [Online]. Available: https://learn.microsoft.com/en-us/windows/security/threat-protection/windows-firewall/best-practices-configuring 27

[48] ——, "Get behavioral analytics and anomaly detection." [Online]. Available: https://learn.microsoft.com/en-us/defender-cloud-apps/anomaly-detection-policy 28

[49] ——, "Microsoft Defender Antivirus in Windows." [Online]. Available: https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/microsoft-defender-antivirus-windows?view=o365-worldwide 27

[50] NordVPN, "XDR definition." [Online]. Available: https://nordvpn.com/cybersecurity/glossary/xdr/ 6, 7

[51] Palo Alto, "What is UEBA." [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-ueba 23, 28

[52] ——, "XDR - Extended Detection and Response - Palo Alto networks." [Online]. Available: https://www.paloaltonetworks.com/cortex/cortex-xdr 6, 14

[53] Panda Security, "73 ransomware statistics vital in security in 2022." [Online]. Available: https://www.pandasecurity.com/en/mediacenter/security/ransomware-statistics/#:~:text=The$%$20total$%$20cost$%$20of$%$20a,global$%$20average$%$20for$%$20all$%$20sectors. 41

[54] Purple Security, "Cybersecurity statistics." [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/ 26, 41

[55] Red Canary, "Cobalt Strike - Threat Detection Report." [Online]. Available: https://redcanary.com/threat-detection-report/threats/cobalt-strike/ 31

[56] SentinelOne, "About SentinelOne." [Online]. Available: https://nl.sentinelone.com/faq/ 12

[57] Statology, "F1 score vs. Accuracy." [Online]. Available: https://www.statology.org/f1-score-vs-accuracy/ 19

[58] Tenable, "Nessus - The Global Gold Standard in Vulnerability Assessment Built for the Modern Attack Surface." [Online]. Available: https://www.tenable.com/products/nessus 41

[59] Trellix, "What is XDR." [Online]. Available: https://www.trellix.com/en-us/security-awareness/endpoint/what-is-xdr.html 6, 7

[60] Trend Micro, "Advanced XDR Capabilities - Now with Vision One | Trend Micro." [Online]. Available: https://www.trendmicro.com/en_nl/business/products/detection-response/xdr.html 6, 7, 14

[61] ——, "Understanding targeted attacks: goals and motives." [Online]. Available: https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/understanding-targeted-attacks-goals-and-motives 41

[62] Virus Total, "Virus Total Intelligence." [Online]. Available: https://www.virustotal.com/gui/intelligence-overview 26

[63] D. Security, "Why XDR is not a replacement for SOAR." [Online]. Available: https://d3security.com/blog/why-xdr-is-not-a-replacement-for-soar/ 7

[64] Seleniun, "Selenium." [Online]. Available: https://www.selenium.dev/ 26

[65] SentinelOne, "SentinelOne." [Online]. Available: https://nl.sentinelone.com/ 14

[66] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*, 2018. 4, 26

[67] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012. 4, 26

[68] T. Sommestad, H. Holm, and D. Steinvall, "Variables influencing the effectiveness of signature-based network intrusion detection systems," *Information Security Journal: A Global Perspective*, vol. 0, no. 0, pp. 1–18, 2021. 28, 30

[69] Sophos, "Sophos XDR | Extended Detection and Response platform." [Online]. Available: https://www.sophos.com/en-us/products/endpoint-antivirus/xdr 14

[70] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6. 4, 26

[71] J. W. Ulvila and J. John E. Gaffney, "Evaluation of intrusion detection systems," *Volume 108, Number 6, November-December 2003, Journal of Research of the National Institute of Standards and Technology*, pp. 453–473, 2 2004. 1, 3

[72] E. van der Kouwe, G. Heiser, D. Andriesse, H. Bos, and C. Giuffrida, "SoK: Benchmarking Flaws in Systems Security," in *EuroS&P*, Jun. 2019. 4, 24

[73] VMware, "What is Extended Detection and Response? - VMware Glossary." [Online]. Available: https://www.vmware.com/topics/glossary/content/xdr-extended-detection-and-response.html 6, 7

A. Collected results

Technique	Incident	Subincident	Observed	Alerts	Response
Reconnaissance	Nessus Scan	Nessus Scan	N	N	N
Reconnaissance	Nmap TCP Scan	Nmap TCP Scan	N	N	N
Reconnaissance	Exploit scan via Metasploit	Exploit scan via Metasploit	Y	Y	N
Reconnaissance	WinPEAS scan	Enumerate domain information	Y	Y	N
		Search system information	N	N	N
		Search user information	Y	N	N
		Search processes information	Y	N	N
		Search services information	N	N	N
		Search installed applications information	Y	Y	N
		Search network information	Y	N	N
		Search windows credentials	Y	Y	N
		Search browser information	N	N	N
		Search generifc files that can contain credentials	N	N	N
		Search specific files that can contain credentials and for regexes inside files	N	N	N
		Display interesting events information	N	N	N
Initial access	Phishing email link	Phishing email link	N	N	N
Initial access	Phishing email attachment	Phishing email attachment	N	N	N
Execution	CLI reverse shell	CLI reverse shell	Y	Y	N
Execution/Defense evasion	Meterpreter shell	Meterpreter shell	Y	N	N
Execution	Psexec cred authentication	Powershell execution	Y	Y	N
		Powershell process execution via COMPSEC	Y	Y	Y
Discovery	Data discovery	Data discovery	Y	N	N
Discovery	System enumeration	System owner user discovery	Y	Y	N
		Hostname discovery	Y	Y	N
		Discovery via registry	Y	Y	Y
		Account discovery	Y	Y	N
		System discovery WMI reconnaissance	Y	Y	N
		Password lookup using findstr	Y	Y	Y
		System discovery via sc	Y	N	N
		Ns lookup execution	Y	N	N
		Service information discovery via systeminfo	Y	N	N
Privilege escalation	Priv esc with sticky keys process	Priv esc with sticky keys process	Y	Y	Y
Privilege escalation	Priv esc with existing process migration	Priv esc with existing process migration	Y	Y	N
Credential access	Hash dump	Hash dump	N	N	N
Lateral movement	Psexec lateral movement	Remote execution of CLI via PSExec	Y	Y	Y
		PSEXESVC execution	Y	Y	N
		Executable file in admin share	Y	Y	N
Data exfiltration	File transfer via download	File transfer via download	N	N	N
DDoS	Syn Flood	Syn Flood	N	N	N

Figure A.1: Results for manual attacks for solution 1

Execution	Execution through API & Powershell	Use WinAPI to aquire system singleton for monkey process	N	N	N
		Run powershell to perform post breach actions	Y	Y	Y
Execution	Scripting	Use scripts for post breach actions	N	N	N
Execution	Service execution	Create a service via MS-SCMR	N	N	N
Persistence	Scheduled task	Create a scheduled task	Y	Y	N
Persistence	Create account	Create account via net	Y	Y	N
Persistence	Hidden files and directories	Create hidden files and directories	Y	Y	N
		Modify attributes of hidden files and directories	Y	Y	N
Persistence	Powershell profile	Modify powershell profile	Y	N	N
Defense evasion	Timestomping	Modified a file's modification timestamp	Y	N	N
Credential access	Credential dumping	Hash dump	N	N	N
Credential access	Brute force	SMB exploit using bruteforce	N	N	N
Discovery	Account discovery	Account discovery	Y	N	N
Discovery	Remote system discovery	Network scan via host	Y	Y	N
Discovery	System info discovery	Network connections	N	N	N
		Process discovery	Y	N	N
Discovery	System network conf disc	Network connections via netstat	N	N	N
		Network interface info	N	N	N
Lateral movement	Remote services	SMB exploit	N	N	N
Lateral movement	Remote file copy	Transfer file via share	Y	Y	N
C3	Uncommonly used port	Communication via port 5000	N	N	N
Data exfiltration	Exfiltraiton over C&C channel	Exfiltraiton over C&C channel	N	N	N
Encryption	Folder encryption	Encrypt files via bitflip	N	N	N
		Change file extension	N	N	N

Figure A.2: Results for infection monkey attacks for solution 1

Technique	Incident	Subincident	Observed	Alerts	Blocked
Reconnaissance	Nessus Scan	Nessus Scan	Y	N	N
Reconnaissance	Nmap TCP Scan	Nmap TCP Scan	N	N	N
Reconnaissance	Exploit scan via Metasploit	Exploit scan via Metasploit	N	N	N
Reconnaissance	WinPEAS scan	Enumerate domain information	Y	Y	N
		Search system information	N	N	N
		Search user information	N	N	N
		Search processes information	N	N	N
		Search services information	N	N	N
		Search installed applications information	N	N	N
		Search network information	N	N	N
		Search windows credentials	N	N	N
		Search browser information	N	N	N
		Search generifc files that can contain credentials	N	N	N
		Search specific files that can contain credentials and for regexes inside files	N	N	N
		Display interesting events information	N	N	N
Initial access	Phishing email link	Phishing email link	N	N	N
Initial access	Phishing email attachment	Phishing email attachment	N	N	N
Execution	CLI reverse shell	CLI reverse shell	Y	Y	N
Execution/Defense evasion	Meterpreter shell	Meterpreter shell	Y	Y	N
Execution	Psexec cred authentication	Powershell execution	Y	Y	N
		Powershell process execution via COMPSEC	Y	Y	N
Discovery	Data discovery	Data discovery	N	N	N
Discovery	System enumeration	System owner user discovery	Y	N	N
		Hostname discovery	N	N	N
		Discovery via registry	Y	Y	N
		Account discovery	Y	Y	N
		System discovery WMI reconnaissance	N	N	N
		Password lookup using findstr	Y	Y	N
		System discovery via sc	N	N	N
		Ns lookup execution	N	N	N
		Service information discovery via systeminfo	N	N	N
Privilege escalation	Priv esc with sticky keys process	Priv esc with sticky keys process	Y	Y	N
Privilege escalation	Priv esc with existing process migration	Priv esc with existing process migration	N	N	N
Credential access	Hash dump	Hash dump	N	N	N
Lateral movement	Psexec lateral movement	Remote execution of CLI via PSExec	Y	N	N
		PSEXESVC execution	Y	N	N
		Executable file in admin share	Y	Y	N
Data exfiltration	File transfer via download	File transfer via download	N	N	N
DDoS	Syn Flood	Syn Flood	N	N	N

Figure A.3: Results for infection monkey attacks for solution 2

Execution	Execution through API & Powershell	Use WinAPI to aquire system singleton for monkey process	Y	Y	N
		Run powershell to perform post breach actions	Y	Y	N
Execution	Scripting	Use scripts for post breach actions	N	N	N
Execution	Service execution	Create a service via MS-SCMR	N	N	N
Persistence	Scheduled task	Create a scheduled task	N	N	N
Persistence	Create account	Create account via net	N	N	N
Persistence	Hidden files and directories	Create hidden files and directories	Y	N	N
		Modify attributes of hidden files and directories	N	N	N
Persistence	Powershell profile	Modify powershell profile	N	N	N
Defense evastion	Timestomping	Modified a file's modification timestamp	N	N	N
Credential access	Credential dumping	Hash dump	N	N	N
Credential access	Brute force	SMB exploit using bruteforce	N	N	N
Discovery	Account discovery	Account discovery	N	N	N
Discovery	Remote system discovery	Network scan via host	N	N	N
Discovery	System info discovery	Network connections	Y	N	N
Discovery	System network conf disc	Network connections via netstat	N	N	N
		Network interface info	N	N	N
Lateral movement	Remote services	SMB exploit	N	N	N
Lateral movement	Remote file copy	Copy file via share	N	N	N
C3	Uncommonly used port	Communication via port 5000	N	N	N
Data exfiltration	Exfiltraiton over C&C channel	Exfiltraiton over C&C channel	N	N	N
Encryption	Folder encryption	Encrypt files via bitflip	N	N	N
		Change file extension	N	N	N

Figure A.4: Results for infection monkey attacks for solution 2