HAW HAMBURG

MASTER THESIS
Henning Krause

An Information Security Advisory Risk Assessment Concept

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE
WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Henning Krause

An Information Security Advisory Risk Assessment Concept

Master thesis submitted for examination in Master's degree
in the study course *Master of Science Informatik*
at the Department Computer Science
at the Faculty of Engineering and Computer Science
at University of Applied Science Hamburg

Supervisor: Prof. Dr. Klaus-Peter Kossakowski
Supervisor: Prof. Dr. Bettina Buth

Submitted on: 31.08.2023

Henning Krause

Title of Thesis
An Information Security Advisory Risk Assessment Concept

Keywords
Information Security Advisory, Risk Assessment

Abstract English
The number of published information security advisories regarding vulnerabilities is constantly rising, resulting in a high workload for organizations which need to process the information to implement timely remediation measures to avoid the extensive costs of a successful exploitation. To aid organizations in the prioritization of advisories, this thesis develops an automated assessment concept for advisories. Building upon established concepts and open-source data, it recognises characteristics of advisories and improves current rating approaches. By using network topologies of an organization, the results can be adapted based on a local environment. The evaluation of the concept shows the prioritization of advisories which could serve as an entry point to the organization's network. By implementing an extension to the concept which regards the exploitation impacts of advisories, the assessment is improved to better reflect possible paths through the network. Without an organization's network topology, advisories with many, known-exploited or severe vulnerabilities are still highlighted, but a local contextualization and henceforth an improved assessment is lost. This work shows an assessment construct, that can serve as the foundation to built better and automatised security advisory ratings with or without topology knowledge.

Abstract German
Die Zahl der veröffentlichten Informationssicherheitswarnungen für Schwachstellen steigt stetig, müssen aber trotzdem zeitnah bearbeitet werden, um erhebliche Schadenskosten durch erfolgreiche Ausnutzungen der Schwachstellen zu vermeiden. Um Organisationen bei der Priorisierung von Sicherheitswarnungen zu unterstützen, wird in dieser Thesis ein automatisiertes Bewertungskonzept entwickelt. Es bezieht die Charakteristiken von Sicherheitswarnungen ein, um mithilfe von etablierten Konzepten und Open-Source-Informationen gängige Bewertungskonzepte zu verbessern. Weiterhin wird ein Verfahren zur Anpassung der Bewertung im lokalen Kontext einer Organisation entwickelt, indem die Netzwerktopologie betrachtet wird. Die Evaluierung des Konzepts zeigt, dass Sicherheitswarnungen, die als Zugangspunkt in das Netzwerk einer Organisation missbraucht werden können, priorisiert werden. Durch eine Erweiterung, in der die Auswirkungen der Schwachstellen zur verbesserten Darstellung möglicher Pfade durch das Netzwerk genutzt werden, wird das Verfahren verbessert und die Erweiterbarkeit gezeigt. Auch ohne die Netzwerktopologie einer Organisation werden immer noch Sicherheitswarnungen mit vielen, ausgenutzten oder schwerwiegenden Schwachstellen fokussiert. Allerdings geht in diesem Fall eine lokale Anpassung und dementsprechend eine bessere Bewertung verloren. Zusammenfassend zeigt diese Arbeit ein Bewertungskonzept, das als Grundlage für bessere und automatisierte Bewertungen von Informationssicherheitswarnungen genutzt werden kann, sowohl mit als auch ohne Informationen über die Netzwerktopologie.

Contents

List of Figures

List of Tables

Abbreviations

AHP Analytic Hierarchy Process.

ANML Advisory and Notification Markup Language.

API Application Programming Interface, *Glossary:* API

BSI Federal Office for Information Security.

CAIF Common Announcement Interchange Format.

CDF Cumulative Distribution Function.

CERT Computer Emergency Response Team.

CISA Cybersecurity & Infrastructure Security Agency.

CPE Common Platform Enumeration.

CSAF Common Security Advisory Framework.

CSIRT Computer Security Incident Response Team.

CTI Cyber Threat Intelligence.

CVE Common Vulnerabilities and Exposures.

CVRF Common Vulnerability Reporting Framework.

CVSS Common Vulnerability Scoring System.

CWE Common Weakness Enumeration.

DAF German Advisory Format.

DB Database.

DoS Denial-of-Service, *Glossary:* DoS

EISPP European Information Security Promotion Program.

EPSS Exploit Prediction Scoring System.

HTML HyperText Markup Language.

ICS Industrial Control System.

ML Machine Learning.

NLP Natural Language Processing.

NVD National Vulnerability Database.

OVAL Open Vulnerability and Assessment Language.

PDCA Plan, Do, Check, Act, *Glossary:* PDCA

PDF Probability Density Function.

PSIRT Product Security Incident Response Team.

ROLIE Resource-Oriented Lightweight Information Exchange.

RSS Rich Site Summary.

SBOM Software Bill of Materials.

TLP Traffic Light Protocol.

TTC Time To Compromise.

URI Uniform Resource Identifier.

1 Introduction

In February 2023, the vulnerability identified by *CVE-2021-21974* was actively exploited as an attack vector for a worldwide ransomware attack, having already encrypted thousands of *ESXi* servers [83]. This event is just one out of many. Each individual case costing the affected organization an average of US$4.35 million, according to a recent report [66]. The report further unveiled that out of the 550 organizations analysed, 83% were struck more than once. New attack opportunities for adversaries open up, as technology is introduced more and more into every aspect of our lives. The value in stolen data, ransom extortion and selling access to compromised networks is high, which reflects in the number of threat actor groups—over 200 tracked by CrowdStrike alone [41]. Furthermore, automation allows adversaries to execute attacks effectively and fast.

If cybersecurity incidents are happening at a high rate everywhere in the world, what makes the *ESXi* server event worth mentioning? On 23 February 2021, VMWare published a security advisory[1] regarding their *ESXi*, *vCenter Server* and *Cloud Foundation* products [101]. The advisory comprises three vulnerabilities that affected the products: *CVE-2021-21972*, *CVE-2021-21974* and *CVE-2021-21973*, with the CVSS scores 9.8, 8.8 and 5.3, respectively. In 2021, *CVE-2021-21972* has been exploited using automation concepts and allowed remote code execution [102]. Short-term remediation workarounds and a security patch have long been released. Yet, the new successful attack wave in 2023 exploited another vulnerability from the very same advisory. This raises the question of how this series of new incidents could happen, if all required information to perform necessary remediation measures were available long before.

The aforementioned advisory is just one example by a single organization. However, there are numerous security advisory publishing platforms releasing dozens of advisories each day. Most platforms focus on one or two vulnerability references, identified by the Common Vulnerabilities and Exposures (CVE) program, in each advisory [72]. Taking

[1] The terms *information security advisory*, *security advisory* and *advisory* are used interchangeably in this work.

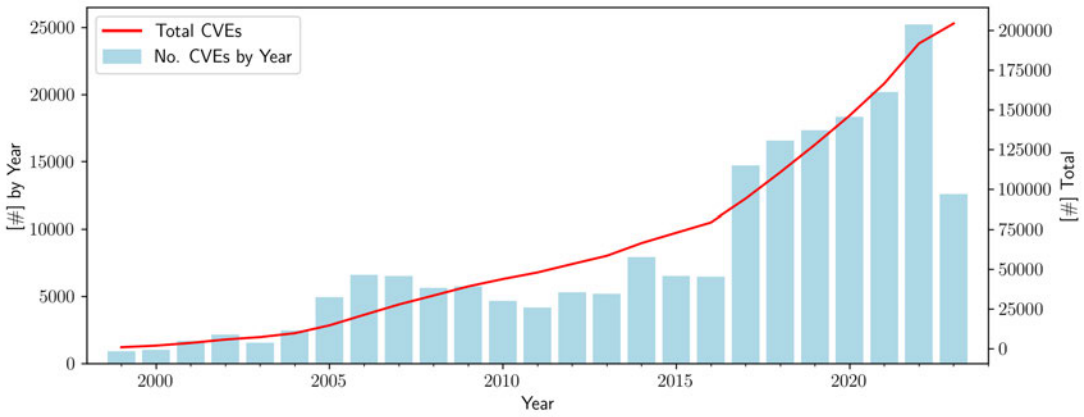


Figure 1.1: Visualisation of the published vulnerabilities in NVD. Note the secondary Y-axis. Adapted from https://www.cvedetails.com/browse-by-date.php, accessed on 09 June 2023.

Figure 1.1 into account, which shows the number of CVEs by year of public disclosure, it is quite evident that each year the number of published vulnerabilities increases. In June 2023, the annual amount already exceeds the total of 2016 by far. The problem of the amount of evaluation work required to handle this information stands out immediately. Furthermore, it does not seem as if there is going to be an end to the trend of increasing vulnerability discovery rates in the near future. With the European Union's ongoing work towards a regulation on cybersecurity requirements, known as the Cyber Resilience Act, manufacturers will be obligated to handle vulnerabilities responsibly [57]. This leads to a foreseeable rise in security advisory publications by manufacturers who did not before, even if the number of vulnerabilities does not increase. Additionally, this effect will scale along with the increasing number of vulnerabilities published, leading to a multiplication of the amount of advisories.

Recently, Ouzan [77] carried out research to identify public facing instances of vulnerabilities known to be exploited in the wild. He used open-source information, namely the known exploited vulnerabilities catalogue of the Cybersecurity & Infrastructure Security Agency (CISA) [22] to identify vulnerabilities and the search engine Shodan[2] to find vulnerable devices. The author showed that there are over 15 million vulnerable instances for which security advisories have been available. This highlights the fact that available advisories are not sufficiently used by system administrators. Additionally, there is no unification on the format of security advisories to date. They occur in various flavours

[2] see https://www.shodan.io/, accessed on 14 June 2023.

of structures and file formats. From JSON to HTML, each publisher chooses their own, creating the overhead of information extraction for the user. The above-mentioned problems are not specific for advisories. Being a rather young domain and focus of research, Cyber Threat Intelligence (CTI) in general faces the same issues [76]. However, with the release of the Common Security Advisory Framework (CSAF) standard in November 2022 [86], a new standardized format that has a focus on automation and uses industry best practices, progress in this matter may be achieved in the near future.

Problem Statement

Advisories can be of considerable importance for an organization's risk management process. They help to identify vulnerabilities in the system in a timely fashion, further providing guidance how to remediate these issues. As prevention has in most cases a lower cost than resolving the consequences of an incident, it is in the interest of an organization to handle vulnerabilities as efficiently as possible. The aforementioned high number of vulnerable devices contradicts with the recent research by de Smale et al. [30], which showed that all 22 interviewed organizations used security advisories published by a national CERT as sources for public vulnerability disclosures. If available information are generally used, but there are still numerous unattended vulnerabilities, one possible explanation is an ineffective processing of advisories by organisations operating vulnerable devices.

Organizations face two challenges regarding security advisories that prevent efficient processing. First, relevant advisories must be identified and collected. Second, the advisories must be assessed. The new CSAF standard can reduce the work required for the collection and processing of advisories, but does not assist in the evaluation of the information in the context of the organization. As will be shown in this work, practised ratings offered by publishers are not tailored for the information construct of advisories and hinder a prioritization due to their often discrete values. In an organization, a first assessment to derive a priority for advisories may be performed by security professionals. Nonetheless, even with a reasonable training, the process will require sufficient time as 'intuition only adds value after disciplined collection of objective information and disciplined scoring of separate traits' [54]. Especially small and medium-sized organizations struggle with the necessary capacity to process the amount of information due to limited personnel [30, 109]. Even aggregated and preprocessed information which offer a more sophisticated and comprehensive view on advisories—as for example offered by Computer Emergency

Response Teams (CERTs)—can not replace a local assessment. With CSAF just emerging, an automated collection is likely to be only scarcely implemented, which results in advisories only being periodically and not continuously monitored [109]. If an organization can not even keep up with tracking recently released advisories, local assessment is unlikely. Instead the organization would rely predominantly on ratings offered by the National CERTs (or any other publisher), even though those ratings should only be an aid that can not replace an independent, contextualized and structured assessment.

As a result, organizations are confronted with the emerging need to find an efficient assessment method for security advisories, to get hold of the increasing amount of information. A concept to assist in the assessment process, saving valuable labour cost, would be a great benefit. Therefore, the following research questions are addressed in this work:

RQ1 *Using only open-source information, can security advisories be assessed to allow a prioritization?*

RQ2 *How can the rating of a security advisory be adapted for a local environment, to better reflect the necessary remediation urgency in the context of an organization?*

For the development of a new assessment methodology, the focus lies in the use of easily accessible and free information. This allows small and medium size organizations, limited in their resources, to apply the concept without having the need to acquire new assets. Furthermore, an extension with additional data leading to an enhancement of the assessment results should be possible. To achieve this objective, relevant methods must be identified and used in a modular framework, allowing for easy replacement. Additionally, the calculation shall be processed automatically and result in a prioritization order of the advisories, providing a contextualized assessment that aids in the selection of what to address first. In conjunction with the CSAF standard, this work will allow organizations to reduce the work-load required to process security advisories and constitute a starting point for future research in the automated assessment of advisories.

The remainder of this thesis is structured as follows. In Section 2 relevant theoretical background information is presented, i.e. an introduction to security advisories, their context in the risk assessment process and related assessment approaches. Section 3.1 describes the selection of concepts and data sources used for the assessment process and Section 3.2 an implementation of the concept. In Section 3.3, the implementation will be

evaluated using two examples. Afterwards, the expandability of the assessment concept is shown by improving the initial concept in Section 4. Finally, the results are discussed in respect to the research questions and limitations and future perspectives are shown in Section 5.

2 Theoretical Background

In this section, an overview of the relevant theoretical background will be given. It will contribute to the development of an understanding of what security advisories are and in which contexts they are used for what purpose. The scope of this work constitutes a limitation on the amount of information that can be given. Especially regarding the risk management, there is an extensive amount of related concepts, methodologies and background information. Therefore, the focus will be on providing the reader a general overview and introduce the concepts relevant for the remainder of this work.

The necessity for security advisories descends from the existence of vulnerabilities and the threat with the emerging risk they pose to an organization. Thus, we will first introduce those terminologies to establish the context. To clarify the connection between the aforementioned, the illustration shown in Figure 2.1 will be used. It shows an office building next to an embankment dam, that holds back a substantial amount of water. If the dam, which already has a crack, breaks, the water will damage or even destroy the office building.

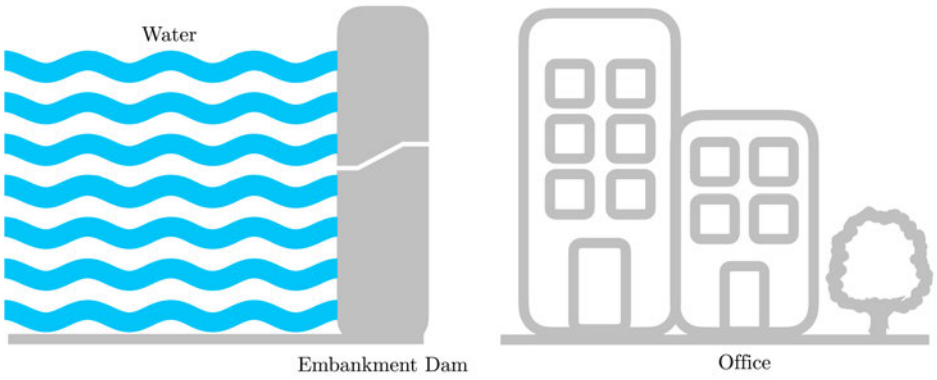


Figure 2.1: Illustrative example to show the relationship between vulnerability, threat and risk. Adapted from [80].

A general definition for vulnerabilities is given by the ISO 27000 standard [2], that is also frequently used throughout other work (e.g. [58, 59]): A vulnerability is a 'weakness of an asset [...] that can be exploited by one or more threats'. Considering the example, the vulnerability is represented by the crack in the dam, which induces a possible breaking point in the otherwise solid structure. Weaknesses in computer systems might for example be in procedures, design, implementation, system security procedures or internal controls [80]. As already apparent from the definition of a vulnerability, threats can exploit vulnerabilities. The water in the example could increase due to a heavy rain and cause the instable dam to collapse or someone could deliberately use the weak point of the crack to destroy the dam. In general, threats are a 'potential cause of an unwanted incident, that can result in harm to a system or organization', according to the ISO standard [2]. In other definitions, cause is further distinguished into events and circumstances [27, 80]. Risk, in a general sense as defined by the ISO standard, is the 'effect of uncertainty on objectives', with uncertainty being 'the state [...] of deficiency of information related to, understanding or knowledge of, an event, its consequence, or likelihood' and effect the 'deviation from the expected — positive or negative'. Broken down to the example, one does not know if the embankment dam will break and what the consequences will be for the objective of having a running factory. Hence, the concept of risk attempts to describe this problem. Another definition proposed by Könings [59, p. 12] describes risk as an evaluation of threat by probability and impact with regard to deviation from the expected system goals. This definition further highlights that risk is centred around threats. Also, the ISO definition indicates that the outcome of a risk can also be positive, which means that risks can lead to opportunities. However, in the context of this work, the effect of risk is considered to be negative, as the exploitation of vulnerabilities can never be desirable. Risks, that can only cause harm, are also referred to as hazards.

Now that basic contextual knowledge is established, we will turn to security advisories and their role in the context of a computer system's risk. Note that the concepts of CVE, Common Vulnerability Scoring System (CVSS), Common Weakness Enumeration (CWE) and Common Platform Enumeration (CPE) are used in this section. They are fundamental in the domain of this work and necessary to sufficiently describe and create an understanding of the background. Refer to Section 3.1.1 for an elaborate description of those.

2.1 Information Security Advisories

When new vulnerabilities are discovered, an organization needs timely information to implement suitable mitigation measures if it possesses an affected product, because a significant amount of vulnerabilities is already exploited on the disclosure date [94]. Security advisories are an established concept that allows vendors to distribute all necessary information required for the reaction of an organization [47]. Conversely, advisories also serve to inform that a certain product is not affected. This is seen less commonly, e.g. when there is a vulnerability that draws a lot of attention—like the *Log4Shell* vulnerabilities in *Log4j*[3]—, but can help to anticipate individual information requests from each customer. Another practised usage is to inform about end-of-life of products or specific versions. This does not directly imply existing vulnerabilities, but is still an important fact, as future vulnerabilities will not be fixed. Thus, the practised emerged to disclose this information in the same channel as vulnerabilities. In the following sections, an overview of how advisories are implemented and used will be given.

2.1.1 Content

Lacking a widely adopted standardization framework, the contents and structure of security advisories depend on the issuing organization. However, there are similar characteristics that allow for a generic description of what a consumer can expect from an advisory. They form a minimal set of information required so the advisory is useful for the reader. The following components were compiled from the sources [17, 21, 39, 86]:

Title A concise summary of the advisory. Usually includes the product and, if the advisory is published by an aggregator, the vendor. Version numbers and CVEs are not unusual as well.

ID The identifier of an advisory given by the publisher. Only guaranteed to be unique within the publications of a publisher. Often contains a vendor specific prefix.

Rating An assessment of the publisher of the advisory. The type of assessment varies among the publishers. Some use CVSS scores with the score of the most severe vulnerability referenced by the advisory. Others define arbitrary ratings, that use an ordinal scale which ranges from least to most severe.

[3] e.g. see https://www.sophos.com/en-us/security-advisories/sophos-sa-20211210-log4j-rce, accessed on 06 July 2023.

Description An exhaustive description of the threat. Technical details, timely information and possible impacts are described here.

Mitigation The mitigation measures depend on the availability of patches. If a patch is available, simple update instructions might be sufficient. Otherwise, alternative actions, such as system settings preventing the exploitation, are provided.

References Complete lists of CVEs involved with the advisory and of known affected CPEs is given. Furthermore, links to other useful resources in the context of the advisory may also be included.

2.1.2 Publication and Distribution

Advisories are published on platforms, which can be classified into sources, aggregators and hybrid source-aggregators of information [72]. Generally, sources are vendors that inform about vulnerabilities in their own products, e.g. by their Product Security Incident Response Team (PSIRT). Considering supply chains, which can be defined as 'consisting of two or more legally separated organizations collaborating in the generation of a product or service with the aim of improving the competitiveness of a supply chain as a whole' [98, p. 14], every organization may face the necessity of publishing an advisory to inform the next instance in the chain: manufacturer, integrator or operator. As a consequence, there may be multiple advisories on the same vulnerability. Aggregators, often offering a paid service, collect and aggregate advisories from multiple sources and redistribute them in a concise form to their customers. Their value lies in the work required to parse the various distribution types to assess all information. An example for a hybrid source-aggregator is the German Federal Office for Information Security (BSI), because it functions as an aggregator, but, if there is a very severe vulnerability, as for example in the Log4j incident[4], also publishes its own advisories.

There are various formats which are used for advisories and many vendors even offer different formats in which they publish. Ekelhart et al. [35] evaluated standards, that can, but may not be intended to, be used for sharing security advisories. In general, formats can be divided in three categories: (i) custom formats, (ii) general CTI standards that are known to have been used for advisories and (iii) specialized advisory standards. Custom formats are for example in plain text, PDF and HTML or build upon JSON or XML. The main problem is that extracting information from the custom formats is

[4]see https://www.bsi.bund.de/dok/log4j, accessed on 11 June 2023.

a tedious task, as each one requires a different parser and the structure may change without notice, breaking the parsers as a consequence. Standardized formats usually use JSON or XML and define a schema for it. An overview of specialized advisory formats is given in Table 2.1, which includes the base formats, release dates and comments for each format. It is clear that there has not been a lot of attention towards standardization and that only in recent years significant progress has been made to modernize. Meanwhile, the usage of other CTI standards, such as Open Vulnerability and Assessment Language (OVAL)[5] or STIX [29], has been established. However, they come with shortcomings when publishing advisories with them since they are not tailored to the specific needs of advisories [35].

Table 2.1: Overview of security advisory standards. Adapted and extended from Ekelhart et al. [35].

Name	**Basis**	**Release Date**	**Comment**
CAIF [42]	XML	2002-2005	Has been used by several CERTs. Issues: not enough elements to describe information which impedes automation.
EISPP [15]	XML	2002-2004	Used by the DAF. Issues: too many flexibilities which impedes automation.
ANML	XML	2003	Not known to be used. Issues: lacks automation capabilities, missing attributes.
CVRF [45]	XML	2017	Used by many organizations that publish in a standardized format.
CSAF [86]	JSON	2022	CVRF successor. Currently in transition phase and therefore sparsely used at the moment.

The distribution of advisories is implemented in different ways. First, publishers operate a dedicated web-page which lists all their publications. This page may only be accessible after a successful authentication. In order to notify interested parties of new advisories, email subscriptions or Rich Site Summary (RSS) feeds are offered. This service is occasionally extended by the option to restrict the notifications on a subset of the advisories,

[5] see https://github.com/CISecurity/OVALRepo, accessed on 12 June 2023.

e.g. by pre-filtering on specific products. More sophisticated concepts are usually accompanied with CTI formats and standards. For example, STIX uses the TAXII standard [52] and the new CSAF standard defines its own distribution method, which lists advisories on the publisher server either using a Resource-Oriented Lightweight Information Exchange (ROLIE) feed [38] or the combination of a listing and change file [86, see Section 7]. The used approaches then allow the automation of advisory retrieval including the acquisition of updated versions of existing advisories.

Despite the benefits of sharing CTI, there is a certain risk linked to it when the wrong stakeholders attain access to information at the wrong point in time [103]. To prevent such events, a common measure is the usage of the Traffic Light Protocol (TLP), which offers a method to restrict the circle of stakeholders who the information is allowed to be shared with (see Figure 2.2 for an overview). We note that there are also personal channels of distributing advisories, e.g. phone calls or in person meetings, but consider them out of the scope of this work, as they do not contribute to an automation concept.

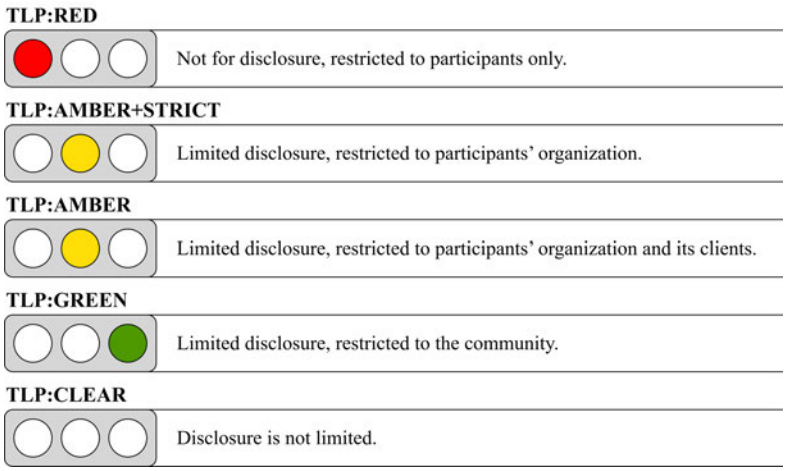


Figure 2.2: Overview of the Traffic Light Protocol (TLP) v2. Adapted from [23].

2.1.3 Information Security Advisories in the Vulnerability Life Cycle

The life cycle of a vulnerability follows a generic pattern, that varies according to the succession of events. *Zero-day-exploits* take advantage of vulnerabilities that have not been publicly disclosed or for which a patch is not yet available [5]. They are classified as exceptionally critical, because their exploitation is comparatively simple [82]. In contrast, within the *coordinated disclosure* process, the vulnerability is communicated to the vendor

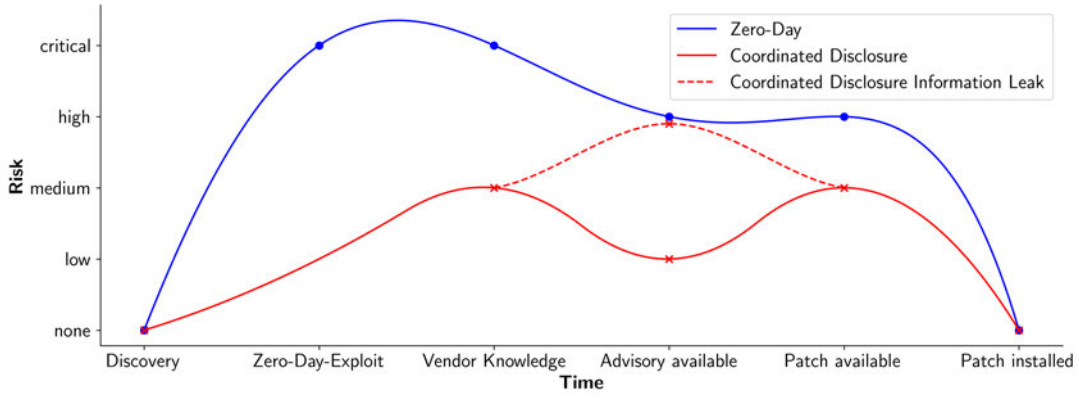


Figure 2.3: The graph shows the development of risk posed by the vulnerability throughout the events of the vulnerability's life cycle. Occurring events are marked on each graph. Adapted from [40, 82].

while few to no information is made public. This ensures that the vendor can develop a patch for the vulnerability before releasing information. However, there is always the possibility that information is leaked despite the effort of keeping it in secret. In Figure 2.3, the development of the generic risk posed by a vulnerability depending on its current state is shown. Naturally, those are merely examples to illustrate the issue, whereas in reality there are numerous variants. Security advisories are often published in conjunction with or shortly after the public disclosure of a vulnerability. As they include all relevant information for the evaluation of the risk posed by the vulnerability to an organization, they offer a valuable input for any risk management activity. Now, an overview of risk management is given and the role advisories take is described.

2.2 Risk Management

Every operation of an organization involves a certain risk. The question is not if an incident[6] will happen, but when it will happen. And if it occurs, what will be the impact and the cost associated with it? Will it mean the end of the company or can the consequences be handled? The question is whether the risk is acceptable. The threats an organization faces are incessantly changing as new vulnerabilities are discovered, new malware families are developed and new attack vectors are created. Security professionals

[6]The terms *information security incident*, *security incident* and *incident* are used interchangeably in this work.

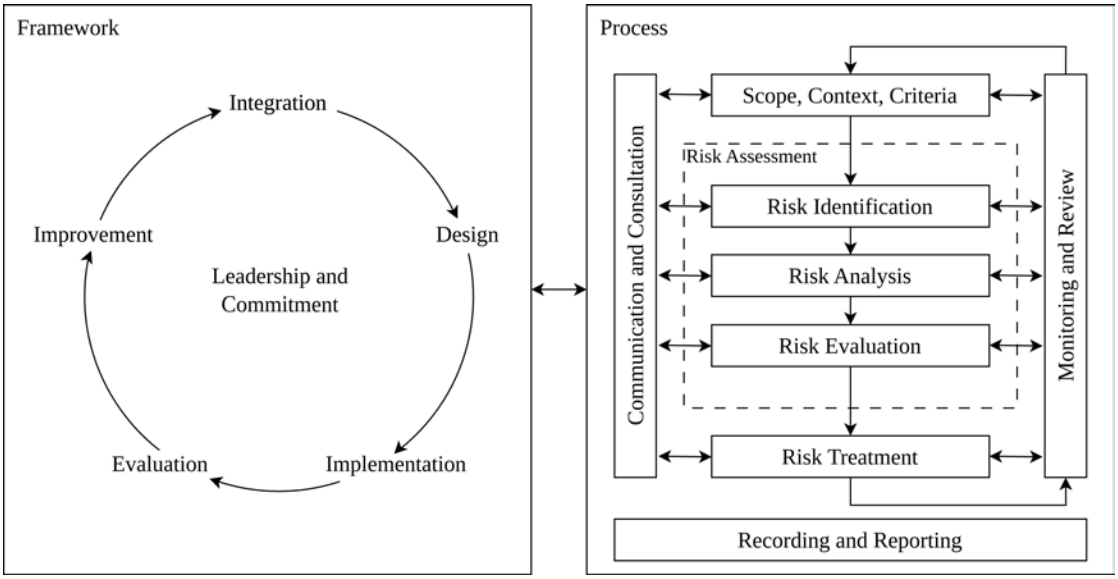


Figure 2.4: Risk Management Process as defined by the ISO 31000:2018 standard [3]. Adapted from [59, p. 46].

are at a constant race against adversaries. Thus, it appears that the assessment of risk must be a continuous rather than a sporadic process. Several different standards and methodologies have been published to help organizations achieve a coordinated process, e.g. the ISO 31000:2018 standard [3] or BSI-Standard 200-3 [1]. In general, standards form a generic guidance framework and can be applied across different business sectors, whereas methodologies have more concrete definitions and often a specific focus [34].

The risk management process is discussed based on the ISO 31000:2018 standard [3] hereafter, because it defines a general overview of the process and is widely accepted. In the standard, the management of risk is defined as a combination of three components: *Principles*, *Framework* and *Process*. The foundation is laid by the *Principles*, which define general considerations that should be regarded during the employment of the various tasks. For example, the *inclusive principle* suggests that risk management should involve stakeholders and be transparent so that it is easily understandable.

A fundamental element is that risk management shall be integrated in the organizations government and leadership process. To assist an organization in the achievement of the integration, the *Framework* offers an alignment of risk management activities to general management tasks. For that purpose, a slightly extended Plan, Do, Check, Act (PDCA) methodology is used. The PDCA cycle is a continuous improvement process consisting of

the eponymous steps. It has been successfully applied in quality management across numerous sectors for a long time [49]. Note that the ISO standard further split the *act* step in two sub-steps: *improvement* and *integration*. All steps involved in the superordinate *Framework* and the risk management *Process* are outlined in Figure 2.4. Risk management comprises several sub-processes which are performed in succession. They reoccur in circular progression, thus forming a continuous process. An exception is made by the sub-processes *Communication and Consultation*, *Recording and Reporting* and *Monitoring and Review*, which are constantly applied in parallel. In order to better understand the usage of security advisories in this context, each process will now be described based on the summaries of Königs [59]:

First, we will introduce the parallel processes. The *Communication and Consultation* process emphasizes a constant exchange between the different stakeholders including the consultation of domain experts. A communication concept to ensure a minimization of miscommunication is desirable. Effectiveness and efficiency of defined measures and processes are important to establish a capable risk management. Therefore, the *Monitoring and Review* activities advise a constant assessment, which may be complemented with external auditors and a maturity model. Introduced in the latest revision of the standard, *Recording and Reporting* underlines the importance of report creation and documentation of outcomes. This aids decision-making, communication of the risk and further aligns the standard with the organization's government process, in which reporting forms an essential part.

The previously introduced steps are rather managing activities. Now, the processes specific to the risk process will be introduced in the order in which they shall be completed. First, the *Scope, Context and Criteria* process creates the outline for the subsequent processes. Assets, constraints on events or time spans and external and internal influences shall be defined, narrowing down the scope considered in the next steps. This allows the application of the risk management process to regard only certain areas of an organization. Furthermore, the goals of the overall process, the types of risks taken into account and risk evaluation methods are determined. Overall, this process creates the preparation for the tasks to follow.

Risk Identification, *Risk Analysis* and *Risk Evaluation* constitute the process known as *Risk Assessment*. Before examining the process as a whole, the three sub-processes will be characterized. The goal of the *Risk Identification* process is to systematically identify all risks of the organization or the previously defined parts. In this methodology,

threats, vulnerabilities and eventually risks are mapped to assets. By using an asset specific assignment, it is possible to derive the impact from the assets' value, create dependencies between different assets and identify risk supporting assets. However, a critical prerequisite to enable this assignment is a functional asset management system. Such a system tracks a complete list of all the assets an organization possesses. Since 2015, the asset management landscape is dominated by the ISO 55000:2014 standard [4], which offers a generic suite of well known and extensively reviewed principles which is not restricted to a specific type of asset [60]. In software security, assets are often identified using the CPE or Software Bill of Materials (SBOM) concepts. Moreover, known threat and vulnerability lists, e.g. CWE and CVE, security standards and best practices aid the process. Standardized enumerations are a key enabler for the automation of this process.

Within the process of *Risk Analysis*, the identified risks are characterized to develop a deeper knowledge and understanding of them. Therefor, the impact and the probability of occurrence are determined, which are then used to assess a classification. Based on the requirements of the organization, the classification may be based on a qualitative, quantitative or semi-quantitative analysis. A common approach is the use of a risk matrix, which is shown in Figure 2.5. The adjustment based on individual requirements is advised as well.

Based on the analysis, the possible consequences of the risk for the previously defined goals and how they may be remediated are assessed in the *Risk Evaluation* process. Possible remedial measures are determined based on the evaluation of (i) the decision to reduce the probability, impact or both, (ii) the opportunity a risk may pose for the organization and (iii) the urgency to remediate the risk is aspired. Additionally, the *Risk Assessment* process is finalized by the documentation of the results.

There is a substantial amount of methods for the whole *Risk Assessment* process. The methods differ in complexity, uncertainty of the resulting outcome, resources required for the application and type of output, which can either be qualitative or quantitative [58, pp. 109-110]. A further distinction is possible when the concept of search is considered, namely forward or backward and bottom-up or top-down search [59, pp. 65–69]. Furthermore, not all methods deliver results for all sub-processes of the *Risk Assessment*. For example, structured and semi-structured interviews are only used for the *Risk Identification*, the root cause analysis focuses on the *Risk Evaluation* and *Risk Assessment* and the Delphi method encompasses the entire *Risk Assessment* process [58, pp. 114–117,

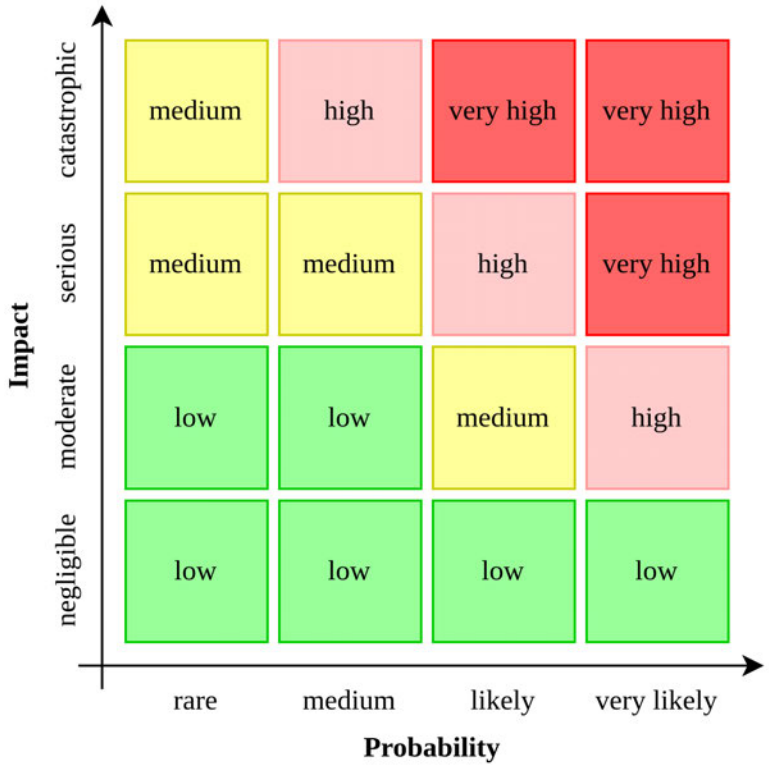


Figure 2.5: A generic risk matrix that maps probability of occurrence and impact to a risk value. Adapted from [1, p. 27].

124–125]. Another point of consideration is that the applicability of a method depends on the type of risk which will be assessed. Some methods are useful in many situations whereas others are specialized to certain use cases (see [87, pp. 60–61] Table 3.2 for an overview). In general, choosing the right assessment methods is an important element of the process. The quality of the output and the amount of resources used depend heavily on the set of methods applied. Using the wrong set of tools can amount to significant costs while the results have little impact.

Finally, the *Risk Treatment* process constitutes the planing phase for remediation steps, involving the planing, selection and implementation of countermeasures. Obviously, the cost is a critical factor at this point, both for the treatment of the risk and the cost of an incident itself. Considering this fact, acceptance of the risk, which means doing nothing, is a legitimate treatment method. Overall, some risks can be prevented, while others may only be minimized by appropriate measures, which is the reason why prevalent

actions include the creation of emergency and business continuity plans or insuring the risk [87].

Security advisories are of significant importance in risk management of software vulnerabilities. They are useful for all sub-processes of the *Risk Assessment* process due to their extensive contents. First, they map vulnerabilities to assets which can be used for the *Risk Identification*. Next, they contain information about the vulnerabilities, thus aiding the *Risk Analysis*. Finally, the remediation measures in the advisory present choices for the *Risk Evaluation*. Due to the number of advisories published daily and the demanding risk management process just presented, concepts to prioritize available information are necessary. Now, some existing approaches for that purpose will be presented.

2.3 Information Security Advisory Prioritization

To the best of our knowledge, there have not been any advancements in the prioritization of security advisories in academic literature in recent years. Therefore, current practices of advisory publishers, which may be used to directly infer a prioritization order, will also be presented here. As noted earlier, it is known that organizations might solely rely on this rating, which makes those methods relevant in the context of this work as well. Furthermore, the concepts of the two related tasks of vulnerability and patch prioritization may also be utilized for the assessment of security advisories. However, there are significant differences that have to be highlighted first. In the prioritization of vulnerabilities, the focus lies on assessing each one individually, but it is not uncommon that an advisory references multiple vulnerabilities or even none at all. A possible adaption may be to rate each vulnerability of an advisory and assign the rating of the most severe vulnerability to the advisory. Patch prioritization methods can involve the criticality of the patch which includes the severity of vulnerabilities remediated by it. Mitigation measures in advisories can include the installation of a patch, allowing a prioritization using such concepts. The shortcoming of this approach is that the mitigation may include other steps, such as network settings, especially when the vulnerability is new and a patch has not been developed yet. Both analogue methods, vulnerability and patch prioritization, are consequentially not suitable for all security advisories.

The ratings used by organizations publishing advisories will be detailed first. Some publishers make use of the fact that each vulnerability is rated with a CVSS base score. This score is then used as the rating of the security advisory. If there are multiple

vulnerabilities referenced by the advisory, the most severe score is used. For example, the Industrial Control System (ICS) advisories[7] of CISA use this method. The second approach is to use arbitrary ratings published along with the CVSS scores, e.g. applied by RedHat in its security advisories. Usually a motivation for the different scores and a general justification on how a score is chosen is given.[8] Finally, another concept applied is to derive a new rating based on the CVSS metric values. In the case of the *DAF-Score* used by the German 'CERT-Verbund' (an association of German CSIRTs and security teams)[9], it emphasizes the impact and probability of occurrence. The *DAF-Score* will be used as an evaluation criterion for this work and is therefore introduced in detail.

2.3.1 The DAF-Score

The score is a metric calculated based on the probability and the potential for damage, which are derived from a combination of CVSS values and characteristics set by the assessor.[10] This assessment concept also makes use of the practice to assign a CVSS vector to advisories. Here, in most cases the vector of the most severe vulnerability is chosen with occasional exceptions, according to an employee of the national CERT. A precondition for the *DAF-Score* is the availability of CVSS environmental metric group values and knowledge of the affected products of an advisory. The CERT maintains an applicable asset and vulnerability database, which makes the calculation possible for their work. Before the *DAF-Score* is illustrated with an example, the different calculation steps are presented.

First, the exploitation status of the advisory is determined based on the value set for *Exploit Code Maturity (E)* of CVSS. The different values are described as follows: *Theoretical* indicates that an individual discovered a flaw that might lead to a vulnerability. It becomes *usable*, when there is a proof of concept and turns *active* if a first sign of exploitation in the wild is discovered. A *public-exploit* assumes that the code required for the exploit is disclosed to the public. For the status, a straightforward mapping from

[7] e.g. https://www.cisa.gov/news-events/ics-advisories/icsa-23-138-04, accessed on 08 June 2023.
[8] see https://access.redhat.com/security/updates/classification, accessed on 08 June 2023.
[9] see https://www.bsi.bund.de/dok/9202084, accessed on 08 June 2023.
[10] The calculation depends heavily on CVSS. See Section 3.1.1 for an introduction.

the CVSS metric is possible.

$$Status = \begin{cases} theoretical, & \text{if } E = U \\ usable, & \text{if } E = P \\ active, & \text{if } E = F \\ public\text{-}exploit, & \text{if } E = H \end{cases} \tag{2.1}$$

Now the distribution method is derived using the following CVSS compositions based on the metrics *Attack Vector (AV)*, *Privileges Required (PR)*, *Integrity (I)*, *User Interaction (UI)* and *Attack Complexity (AC)*. It indicates the effort required to successfully exploit the vulnerability. The different methods are clearly described by their identifiers. *Manually* requires manual steps from the adversary, *automatic* indicates that the different steps can be automated and triggered on demand and *replicating* implies that the exploitation can be performed by bots.

$$Distribution = \begin{cases} replicating, & \text{if } AV = N \wedge PR = N \wedge I = H \wedge UI = N \wedge AC = L \\ manually, & \text{if } AV = L \vee AV = P \vee UI = R \\ automatic, & \text{otherwise} \end{cases} \tag{2.2}$$

An interpretation of the mapping concept expresses the following indications. The *replicating* method is only assumed, if the exploit can be performed over the network, does not require either privileges or user interaction, has a low complexity and a high integrity impact. If the exploit depends on user interaction, must be performed from the local network or even requires the adversary to physically touch or manipulate the vulnerable component, *manually* is set. For all other cases, an *automatic* distribution is anticipated. Using the determined status and distribution from Equation 2.1 and 2.2, respectively, an occurrence probability value is derived based on the matrix shown in Table 2.2. After the

Table 2.2: *DAF-Score* matrix which determines the occurrence probability based on the distribution method and exploit code status of a vulnerability.

		Distribution		
		manually	automatic	replicating
Status	theoretical	very low	low	medium
	usable	low	medium	high
	active	medium	high	high
	public-exploit	medium	high	very high

occurrence probability has been determined, the impact of a successful exploitation must be rated. The *DAF-Score* divides this assessment into two components: harm potential and current harm potential. Starting with the harm potential of the advisory, the evaluation connects the possible loss in case of an exploitation and the context to determine the value. The context contextualizes the loss and is chosen using the referenced CPEs, which have an assigned value in the database of the CERT. It can be *user*, *service*, *system* or *network*. There are two methods to determine the loss value. In the first place, the effect set by the author of the advisory will be utilized. This value is internally used by the CERT and not publicly accessible. For that reason, a complete listing of all possible scores will not be provided. It is comparable to the technical impact of CWE[11] and describes the result of an exploitation of the advisory. An example is 'Execute Admin', which leads to a loss of *take control*. If the effect is not set for the advisory, the following rules are used, which again make use of the CVSS vector.

$$Loss = \begin{cases} \text{take control}, & \text{if } I = H \\ \text{modification}, & \text{if } I = L \\ \text{disclosure}, & \text{if } \neg(C = N) \\ \text{availability}, & \text{if } \neg(A = N) \\ \text{circumvention}, & \text{if } A = N \wedge C = N \wedge I = N \end{cases} \tag{2.3}$$

Table 2.3 shows the selection matrix which uses the context and loss to select the harm potential for the advisory. Bringing it all together, Table 2.4 determines the current harm

Table 2.3: *DAF-Score* matrix which determines the harm potential based on the possible loss in case of an exploitation and the context of a vulnerability.

		Context			
		user	service	system	network
Loss	take control	high	high	very high	very high
	take partial control	medium	medium	high	high
	modification	low	medium	high	high
	disclosure	very low	low	medium	high
	availability	very low	low	medium	high
	circumvention	very low	low	medium	high

potential of the advisory. For that purpose, the previously deduced values of the general

[11] see Section 3.1.1 for an introduction to CWE.

harm potential and the occurrence probability are regarded. This mapping results in a final score that is used as the *DAF-Score* delivered along with an advisory.

Table 2.4: *DAF-Score* matrix which determines the risk based on the harm potential and occurrence probability of a vulnerability.

		Harm Potential				
		very low	low	medium	high	very high
Occurrence Probability	very low	very low	very low	low	low	medium
	low	very low	low	low	medium	high
	medium	low	low	medium	high	high
	high	low	medium	high	high	very high
	very high	medium	high	high	very high	very high

The ratings used in practice will be illustrated using a security advisory of RedHat concerning a bug fix and security update. For this example, the amount of information provided is limited to those necessary to determine the ratings. It becomes apparent that all ratings differ significantly. Even though the relative position on their individual scale is comparable, the values themselves and the respective metrics are not. Since this aspect also applies to other assessment methods, listing further examples does not offer additional knowledge.

Practised Advisory Rating Example

Content

ID: RHSA-2023:3771

URL: https://access.redhat.com/errata/RHSA-2023:3771

CVEs:

1. CVE-2023-20860
   Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N
   Base Score: 7.5
2. CVE-2023-20861
   Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L
   Base Score: 5.3

Ratings

RedHat: Important

Note: Taken as is from the RedHat advisory.

CVSS: 7.5

Note: The advisory has two CVE references with 7.5 being the highest score and therefore chosen as the advisory rating.

DAF-Score: High

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N/E:P/RL:O/RC:C

$E = P \rightarrow$ Status = *usable*

$AV = N, PR = N, I = H, UI = N, AC = L \rightarrow$ Distribution = *replicating*

Occurrence probability = *high* (see Table 2.2)

Context = *service* (set in database)

$I = H \rightarrow$ Loss = *take control*

Harm potential = *high* (see Table 2.3)

DAF-Score = *high* (see Table 2.4)

2.3.2 Related Literature

We now turn to related work found in academic literature as discussed above. What most works have in common is that they incorporate additional information in the rating to contextualize it to the assessing organization. This follows the theorem, that more information lead to better decisions. However, as argued in the introduction, obtaining and using more information involves additional labour costs which is in conflict with

the lacking personnel and the required work to process the data. Perhaps this is the reason why there are theoretical approaches, but practical application is rarely seen. The concepts differ in the kind of information and methods used for the rating calculation. Additional data may involve using known exploit catalogues [9, 25, 55, 85, 88, 111], the network topology of the organization [73, 85, 110], further metadata of the vulnerabilities [37, 104, 105], e.g. how long a vulnerability persisted in the assessed system or the age of the vulnerability, the criticality of the affected asset or an estimation of the patch effort [105].

The methods used for the rating calculation are widespread. Using a decision tree, the approach by CISA assigns vulnerabilities to the categories Track, Track*, Attend, Act based on the properties' exploitation status, technical impact, possibility of automation, mission prevalence and public well-being impact [24]. CVSS' temporal and environmental score, which complement the base score, offer a more meaningful rating as they extend the vulnerability rating with time-related and contextual information.[12] For that reason, Jung et al. [53] attempt to estimate the temporal score of vulnerabilities. Walkowski et al. [105] automate the calculation of the environmental score which they then use to rank vulnerabilities based on the local context. Another approach, also being used by advisory publishers, is the recombination of CVSS sub-scores to assign a new value to a vulnerability. Customization methods involve adding weights to individual scores to distinguish critical assets or deriving further priority scores from a subset of the CVSS metrics [7] or using an Analytic Hierarchy Process (AHP) [65]. Machine Learning (ML) approaches differ in the kind of model architecture, the set of data they use to train their models and in the kind of predictions made, e.g. estimating the exploit probability, as in FIRST's Exploit Prediction Scoring System (EPSS) [50, 51] or the Expected Exploitability model [99]. Also see [9, 16, 95, 111, 112] for more examples of proposed ML concepts. Furthermore, Karlsson et al. [55] build a recommender system in which they included CVE data, domain specific and user knowledge. Zhang and Li [112] incorporated the vulnerability assessment in the patch management to minimize the overall system risk.

If a vulnerability has only been published recently, the corresponding rating is often missing which poses an issue when trying to assess it. Thus, a field of study is to derive a rating solely from the textual description of the vulnerability using Natural Language Processing (NLP) methods [95, 97]. Another approach to overcome this problem is to use Twitter discussions about CVEs to predict the CVSS score [20].

[12] see Section 3.1.1 for an introduction to CVSS.

A concept that involves security advisories in the assessment for vulnerabilities is proposed by Miranda et al. [72]. The authors parameterize EPSS using the number of days since a CVE was published and the number of advisories referencing it. Even though this is a risk assessment for the vulnerabilities, the importance of an associated advisory could be directly inferred by prioritizing the advisories, that address vulnerabilities with a high score.

3 Assessing Information Security Advisories

We will now turn to the description of the security advisory assessment concept. First, fundamental concepts commonly used by all analysed methods are presented, followed by a discussion of possible approaches and the final selection used for this work in Section 3.1. In Section 3.2, the implementation is outlined, which is then evaluated in Section 3.3.

3.1 Methodology

3.1.1 Fundamental Concepts

In the domain of information systems, there are various industry standards and models that are widely used and well-known. Each one presented here has proven to be beneficial for threat information sharing and is regularly used in new methods presented in academic literature. They are also integrated throughout the concepts introduced in this section and are thus presumed to be understood. It is therefore important to first grasp the idea of them, to create a better understanding of their utilization in a specific context.

CWE

Weaknesses are conditions in a computer system that can be abused by vulnerabilities, as introduced in Section 2. There may be many vulnerabilities based on the same type of weakness, which induces a certain relation between them. Noticing and understanding this relation can help to discover patterns and prevent the emergence of new vulnerabilities. For that purpose, the Common Weakness Enumeration (CWE) offers a classification taxonomy for the identification and description of weaknesses [6].

CWE is a community driven effort, which focuses on documenting software and hardware weaknesses in a common language. Various characteristics of each weakness are described that help to understand the background, consequences and possible mitigations. For example, information about the introduction point in the product life cycle and common consequences, which indicate the impact if the weakness is exploited, are documented. Furthermore, connections between different weaknesses are established. They allow the illustration of abstraction levels and chain structures in weaknesses. Abstraction levels begin with a very conceptual, overarching description and can become detailed down to a product or technology specific point.[13] With the help of chain structures, an illustration of how multiple weaknesses can be combined for an exploitation can be given.[14]

Those concepts shall be highlighted with the following weakness, that has been the number one of the most dangerous software weaknesses in the past years.[15] *Out-of-bounds Write* is documented with the ID *CWE-787* and states that 'the product writes data past the end, or before the beginning, of the intended buffer.'[16] Among other things, it is the base abstraction of *Stack-based* (*CWE-121*) and *Heap-based Buffer Overflows* (*CWE-122*) and can result from *Untrusted Pointer Dereference* (*CWE-822*). This shows that using a value from an untrusted source as a pointer can result in a buffer overflow, which i.e. can induce a Denial-of-Service (DoS) as indicated in the common consequences. Relations of that nature help developers to understand implications and mitigate them in the first place.

Therefore, a standard practice is that a vulnerability is linked to the related CWEs which add further general knowledge to the vulnerability. However, to enable an unambiguous assignment, a method to identify vulnerabilities is necessary.

CVE

This demand is fulfilled by the Common Vulnerabilities and Exposures (CVE) [78]. The goal of this standard is to define a format in which information about disclosed vulnerabilities is published. It was introduced in 1999 and has since then developed into a program adopted around the world. Using a unique identifier for each vulnerability, knowledge

[13] https://cwe.mitre.org/documents/glossary/index.html, accessed on 10 July 2023.
[14] https://cwe.mitre.org/data/reports/chains_and_composites.html, accessed on 10 July 2023.
[15] https://cwe.mitre.org/top25/archive/2023/2023_top25_list.html, accessed on 10 July 2023.
[16] https://cwe.mitre.org/data/definitions/787.html, accessed on 10 July 2023.

can be consistently transferred. The IDs have the following format: *CVE-YEAR-XXXX*, e.g. resulting in *CVE-2023-2491*. The year indicates the year of publication, but not the year of discovery. With the increasing number of annually discovered vulnerabilities, the sequence counter—symbolized by the placeholder *XXXX*—has become an arbitrary number of four or more digits, allowing for an expansion on demand.

Assigning new CVEs follows a regulated process, which lies in the responsibility of a CVE Numbering Authority (CNA).[17] After the discovery of a new vulnerability, the person or organization who discovered it reports to a CNA, which then requests a unused CVE-ID from a root CNA. When the ID is reserved, all necessary information for the publication are collected before public disclosure. The CNA is accountable to ensure that all required information is provided.[18] Once the minimum requirements are met, the CVE is published by the CNA and can now be viewed publicly.

At the very least, the following information must be included in a CVE record: The CVE-ID, a description that gives the reader a reasonable understanding, which includes the vulnerability type, root cause and impact, and at least one public reference, ensuring the accessibility of the information. Furthermore, affected products, affected versions and fixed versions must be included. This information is essential for the reader to understand the relevance and context. An emerging problem is that determining if a product in a specific version exists in an organization is an laborious task. Automating that process saves time and cost. For that reason, an approach to systematizing product identification is proposed by CPE.

CPE

As just highlighted, an organization requires an identification mechanism for their assets, not only to relate vulnerabilities to products, but also to enforce configurations and policies. The Common Platform Enumeration (CPE) aims to satisfy that need. It is summarized in its documentation as a 'standardized method of describing and identifying classes of applications, operating systems, and hardware devices present among an enterprise's computing assets' [19]. In the current version 2.3, this method consists of a stack of concepts: the naming itself, name matching for comparing CPEs, a dictionary that forms a repository of CPE names and metadata, and the applicability language

[17] https://www.cve.org/About/Process, accessed on 14 July 2023.
[18] https://www.cve.org/ResourcesSupport/AllResources/CNARules, accessed on 14 July 2023.

which introduces logical expressions using CPE. For the scope of this work, the naming concept, which builds the foundation for the other concepts, will be introduced. Understanding this primary approach is sufficient to comprehend its usage and potential in the methods introduced throughout this section.

First, a CPE is used to identify product classes and not product instances. This means one can identify that a certain version of a product is installed on multiple different systems, but not distinguish the different installations and their user-defined configurations and licences. To be able to identify a product, specific attributes are necessary that in conjunction describe a product unambiguously. In CPE, these are collected in a logical, abstract construct called well-formed CPE name (WFN). A set of constraints ensure a comprehensive representation of WFNs. They include a set of allowed attributes, e.g. part, vendor, product etc., value restrictions for those attributes and logical operators. If a WFN complies to the specification, a product or a range of products—which usually means a range of versions—can be identified. Consider the following example given by the specification [19, p. 14]:

wfn:[part="a",vendor="microsoft",product="internet_ explorer",
version="8\.",update="sp?",edition=NA,language=ANY]*

It is interpreted as follows. Part *a* labels this CPE as a software identifier. The software is known as Internet Explorer produced by Microsoft and every minor version of the major version 8 is featured, because the version attribute uses an asterisk that is interpreted as *any*. Values for the update attribute are vendor-specific. Here, it indicates any update that starts with 'sp' and is followed by a single character, as indicated by the question mark wildcard. Edition and language use the logical values NA (i.e. "not applicable/not used") and ANY (i.e. "any value"). For all possible attributes that are not used in a WFN, the default value of ANY is assumed. In summary, this CPE identifies a software that is produced by Microsoft and called Internet Explorer, in any sub-version of 8 and any update starting with 'sp'.

The naming standard defines another representation of a CPE for machine-readable processing, the formatted string. It was introduced in the current version 2.3. Along with the binding syntax, algorithms to transform Uniform Resource Identifiers (URIs) into WFNs and vice versa are documented. In this representation, attributes are in a fixed order and separated by the colon character. The following example shows the previous WFN example in a formatted string syntax. The formatted string begins with the pre-

fix *cpe:2.3:*, includes all information of the WFN and additionally lists all unspecified attributes with the default value asterisk.

cpe:2.3:a:microsoft:internet_ explorer:8.:sp?::*:*:*:*:**

In former versions, CPE had a URI syntax which is similar to, but more strict than formatted strings. For legacy support it is also included in the current specification. As CPE identifiers are generally provided in the formatted string syntax, we refrain from introducing the URI syntax as well and refer to the specification. Formatted strings and URIs are clearly distinguishable by the prefix, which in case of the formatted strings includes *2.3* in addition to the *cpe* of the URI prefix.

CVSS

The specification document generally describes the Common Vulnerability Scoring System (CVSS) as 'an open framework for communicating the characteristics and severity of software vulnerabilities.' [26]. For that purpose, three different metric groups are defined: (i) base, (ii) temporal and (iii) environmental. Each group defines a set of metrics with predefined values, where each metric describes a characteristic of the vulnerability. The base group metrics are constant over time and usually chosen by the organization possessing the vulnerable product or a designated representative. It should reflect the worst case across different environments. In the temporal and environmental group, elements of the vulnerability that are changing over time and values that must be adjusted in a local context are represented. Based on the chosen metric values, a score is calculated that reflects the severity of the vulnerability. The temporal and environmental metrics alter the base score to adjust the severity in a specific environment.

Figure 3.1 shows an overview of the three groups and their metrics. Scoring CVSS metrics also produces a vector string, a textual representation of the metric values. This vector string is a specifically formatted text string that contains each value assigned to each metric, and should always be displayed with the vulnerability score. It follows a dedicated syntax that uses abbreviations of the metric names and values and separates them with a slash. The following is an example CVSS vector string. All base metric values are set, which result in a base score of 6.5 (see the specification for the exact calculation formulas of the score [26]):

CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:C/C:N/I:L/A:H

For example, the *Attack Vector (AV)* metric is set to *Network (N)*. The specification offers a precise description of each metric and guidance on how to rate each individual one, which reduces errors in the rating process. Furthermore, the framework defines a qualitative mapping for the score ranges: $0 \rightarrow None$, $[0.1, 3.9] \rightarrow Low$, $[4.0, 6.9] \rightarrow Medium$, $[7.0, 8.9] \rightarrow High$ and $[9.0, 10.0] \rightarrow Critical$. For the example vector, this mapping yields a qualitative severity of *Medium*.

A general consideration to keep in mind is the validity of vectors from different sources. When a vulnerability is disclosed to the public, it is assigned a CVSS vector. This vector reflects a very general assessment and can not reflect context characteristics of a product, such as special configurations, e.g. execution privileges, that can either increase or decrease the severity of the vulnerability. This means that if the vulnerability is discovered in a specific product, the vendor may release an updated version of the CVSS vector.[19] CVSS is currently available in version 3.1 [26]. At this juncture however, version 4.0 entered a public preview period and is expected to be released 13 October 2023.[20]

Now that the fundamental concepts are introduced, we will turn to the methods used for our security advisory evaluation concept.

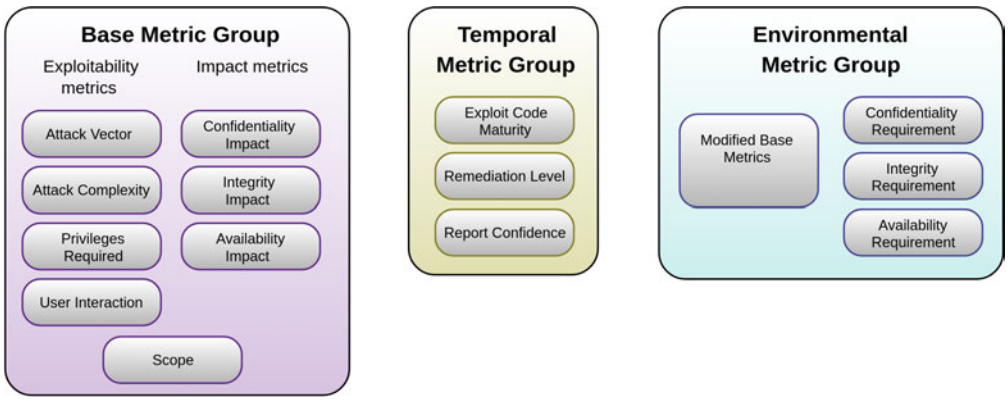


Figure 3.1: Overview of the CVSS metric groups. Owned by FIRST.Org, Inc. (FIRST), reprinted from [26] with permission.

[19] https://access.redhat.com/security/updates/classification/, accessed on 07 July 2023.
[20] https://www.first.org/cvss/v4-0/, accessed on 20 July 2023.

3.1.2 Time To Compromise

The Time To Compromise (TTC) methodology was proposed by McQueen et al. [69] and is centred around the idea, that in order to compromise a system, an adversary performs three processes which depend on the vulnerabilities contained in the system:

- Process 1: The adversary has knowledge of one or more vulnerabilities in the system for which they possess an exploit.
- Process 2: The adversary has knowledge of one or more vulnerabilities in the system, but does not possess an exploit. Thus, a new exploit must be developed in order to utilize the vulnerability.
- Process 3: The adversary does not have any knowledge of vulnerabilities in the system. Therefore, they can either wait for new vulnerabilities to be announced or attempt to discover ones themselves.

Process one and two are considered to be mutually exclusive; either the exploit is known or it is not for a vulnerability. The third process on the other hand occurs in parallel to the other two. For each process, the probability that the adversary is in it, P_x, and the expected time required to perform it, t_x, is determined. The Time To Compromise (TTC) is calculated by the following formula, where u is the probability that process one is unsuccessful:

$$T = t_1 * P_1 + t_2 * (1 - P_1) * (1 - u) + t_3 * u * (1 - P_1) \tag{3.1}$$

The calculation can be summarized as follows. For each process, the time required to complete it and the probability that the adversary is in the process are determined. To calculate the different probabilities and required times, McQueen et al. make use of heuristics and static values in their work. One major component is the distinction of four different adversaries' skill levels—*novice*, *beginner*, *intermediate* and *expert*—which must be chosen by the assessor. The authors argue that the skill level influences the estimated time required to complete a step and must therefore be included in the assessment.

Since the publication of the original TTC methodology in 2006, various extensions have been proposed to either enhance the calculation, e.g. the mean time-to-compromise (MTTC) [64], or show the application for a specific assessment, e.g. calculating the

TTC of attack paths in a network [113]. A common extension is the introduction of knowledge of vulnerabilities to distinguish the exploitation difficulty. For that purpose, especially CVSS is an easily implemented approach as shown in numerous works [75, 84, 113, 114].

A number of improvements to the original model were proposed by Zieger et al. [114]. The authors showed that the original formula contains a mathematical flaw and they therefore proposed an adjustment to fix it in their β-TTC model. Also, they argue that skill must be represented as a continuous variable rather than a discrete one, because in the real world every possible skill is present. The levels are differentiated by the share they take in the total population. To implement that idea into the model, the authors introduced skill as a beta distribution in their calculation. Furthermore, they differentiate the type of vulnerabilities to calculate distinct TTCs. Vulnerability types are distinguished using the CIA triad—confidentiality, integrity, and availability. This allows to highlight different views of the TTC.

The fundamental idea of TTC can be transferred to the evaluation and prioritization of security advisories. However, before applying the concept to a prioritization task, the calculation must be adjusted. A major problem is that TTC and even enhanced versions, such has the β-TTC, essentially reflect the amount of vulnerabilities, which is not desirable for advisories. For example, an advisory including one vulnerability that is actively and automatically exploited should be ranked above an advisory with several vulnerabilities of low severity and no known exploits. Another component of TTC is the estimation of available vulnerabilities.

For the scope of this information security advisory assessment, the viewpoint is different. There is definitive knowledge on the number of advisories that shall be ranked and the number of vulnerabilities covered by them. Furthermore, for the prioritization of advisories, the concept of process three does not make sense. New vulnerabilities may be discovered in a system covered by the advisory, but this only becomes relevant to the prioritization once there is an advisory for it which must be considered in the ranking. Thus, only process one and two must be regarded.

To assess the values of probability and time for process one and two, concepts for the assessment must be determined which will be used for implementation of the proposed approach. The following sections introduce the different components that are necessary and also outlines practical approaches which can be considered for each component.

3.1.3 Vulnerability Exploit Probability

In order to assess process one and two, the probability that an exploit for a vulnerability exists has to be determined. In the original TTC model and various extensions, the concept is based on the assumption that existing exploits are evenly distributed among all vulnerabilities. However, as there is more information available for each vulnerability, this information can be included to create a more precise assessment.

Definite information can be obtained from data sources that publish lists with vulnerabilities known to be exploited, which conversely indicates that an exploit must exist. An open-source example is the *Known Exploited Vulnerabilities Catalog* of CISA which aggregates a list of CVEs. There are commercial providers who offer such data as well. They can, if available to an organization, complement the list with additional information. Further sources of information can be exploitation tools and providers that define the temporal metric group of CVSS. If an exploitation tool can exploit a vulnerability, there must be an exploit available. Included in the temporal metric group is the *Exploit Code Maturity* value indicating how well-developed an exploit of this vulnerability is. Two values for this metric indicate a definite existence of an exploit. In summary, all information available to an organization should be used to identify knowledge of known exploited vulnerabilities.

Even when there is no definite information available, it does not imply that an exploit does not exist. Some exploits may not be publicly known while others might not be in the data sources used to create the ground truth. Therefore, a concept is required to estimate the probability that an exploit exists when a vulnerability is not included in the ground truth. Using statistical inference, features of the vulnerability can be utilized to derive the underlying probability. One apparent approach is to base the inference on CVSS scores, as they are easily available, widely used and indicate the severity of a vulnerability. Furthermore, in recent years more work has focused on predicting exploits using ML. Usually the models are trained with a feature set originated from the data available for the research. This can pose a limitation, because most of the time the authors also use proprietary sources that might not be available for every organization, which hinders a local implementation. One could also decide to use a pre-trained model, but they might suffer from data biases, i.e. geographical characteristics. The prediction output of the models varies between forecasting the existence of exploits and the actual exploitation in the wild. Example models are *EPSS* [51] and the *Expected Exploitability* model [99].

In this work, a heuristic approach will be implemented using known exploit lists to derive a probability that an exploit exists based on the CVSS base score. This decision was made based on several factors: First, the quality of ML models highly depends on the data used to train them. As this work focuses on offering a concept using solely open source data, recent approaches will not result in the same efficacy when limiting the training data to freely available sources. Second, even when proprietary data sources were used, the model would be biased by the kind of providers, e.g. their geographic location and the coverage of their network collection. Third, the effort required for the data collection would exceed the scope of this work. However, we encourage the user to implement an alternative approach to predict exploits that is suitable for the context of the organization.

Now that process one and two can be assigned a probability, the effort required to fine-tune an exploit and to develop a new exploit has to be determined. Both operations largely depend on the adversary profile and on the complexity of exploiting the vulnerability.

3.1.4 Adversary Profile

Adversary profiles are based on common characteristics, such as motivation, capability factor, attack signature, demographics, age and attack method [13, 107], that an individual or group share. Understanding adversaries and the way they act can help to identify measures to prevent successful attacks [14]. The idea of defining adversary profiles in the context of this work is that each organization attracts different kind of adversaries with different capabilities. Especially relevant are the skill level and the resources possessed. The skill directly influences the time it takes to perform or develop a new exploit for a vulnerability. McQueen et al. [69] used discrete values to distinguish different skill levels which may be chosen by the assessor (see Section 3.1.2). However, as Zieger et al. [114] pointed out, this approach does not sufficiently reflect reality. There are different skill levels, but each skill level is present in the world, only with a different probability. Furthermore, a discrete distinction disregards the fact that one may be in any stage between two different levels, which can only be represented by a continuous skill distribution. Following the proposition of Zieger et al., we require skill to be a distribution function $F : [0, 1] \rightarrow [0, 1]$ given by $F(x) = P(X \leq x)$. Note that we limit the domain of the function $[0, 1]$ in contrast to generic distribution functions using $\mathbb{R}$ [44]. The function has the following properties: $\lim_{x \rightarrow 0} F(x)$=0 and $\lim_{x \rightarrow 1} F(x) = 1$. Using this representation

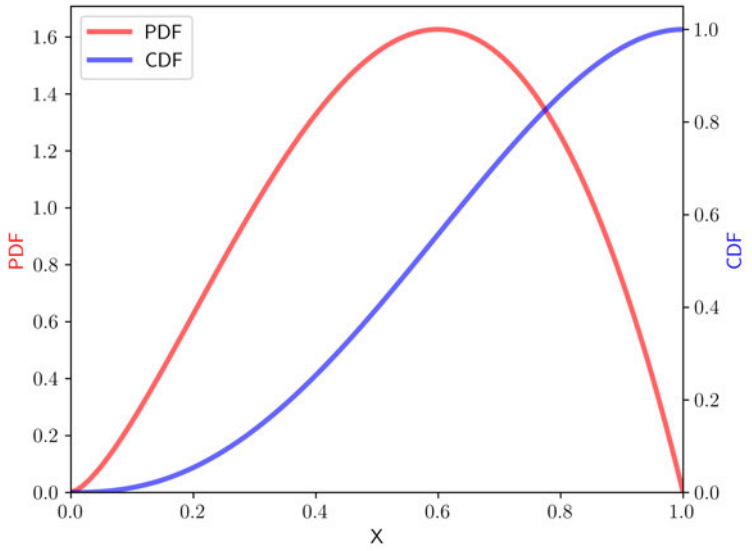


Figure 3.2: A sample skill distribution function is shown. The Probability Density Function (PDF) and Cumulative Distribution Function (CDF) are plotted, using different Y-axis.

of skill, the function is interpreted as follows. The minimum 0 represents the lowest skill and the maximum 1 the highest skill possible. Consider Figure 3.2, which illustrates a sample function. The Probability Density Function (PDF) illustrates the probability that the skill falls in specific range. In this example, most skill is assumed to be in the middle of the range and few in the lower and upper range. The Cumulative Distribution Function (CDF) displays the value of $F(x)$ which indicates the probability that the skill is lower or equal to a specific value.

Sufficient resources can help adversaries in at least two ways: first, networking is an important factor in the organization of criminality. Through former acquaintances and dedicated online forums criminal networks are formed for a joined activity. Usually, there are few *core* members which cooperate for an extended period. Additional *enablers* are recruited who offer specialized services such as digital tools, hacking, money laundering etc. [63]. It is obvious that for the acquisition of those services the adversary has to have and advance the necessary funds. Second, when the adversary does not possess an exploit for a vulnerability, they might be able to purchase one from an underground market or even hire another individual who performs the exploit. Hacking forums where code can be purchased are frequently used by cyber criminals, thus there is a great chance that an exploit is available for purchase if it does exist [79]. Finally, if the adversary has the

resources to operate a team that works on the exploitation, work might be parallelized and thus completed faster. This is not relevant in all scenarios. If a security advisory includes one vulnerability, parallelization will not play a factor in the assessment, as different adversaries can not be distinguished. It does not matter how many individuals work on the same vulnerability, because the estimated effort is identical for each one. However, if there are multiple vulnerabilities, the adversary might attempt to exploit several different ones in parallel which might lead to a success faster.

Profiling adversaries is a steadily growing, but rather young field of research, which still needs advancements in certain areas, such as common definitions and approaches, or improvements to the lack of data which is difficult to collect [14]. However, using the available literature, there is already enough groundwork to determine certain profiles. As stated before, the adversary characteristics that are relevant for this work are restrained to the skill and resources. For that reason, the profiles are not given a meaningful name, which would allow further inferences about other traits of the adversary. Thus, each profile may encompass multiple types as defined in other work, such as the *Threat Agent Library* [18]. To provide an idea for the reader which commonly known type would fit the profile, an example from the ontology proposed by [68] will be given as well. Here, a description of each profile is given, and later in Section 3.2, the respective skill distribution functions and resource level factors are determined.

Profile 1 This profile describes a single individual, that tries to cause harm. The focus might not be on a specific organization, but to be destructive in general. A technical background is not necessarily assumed, which indicates that the individual has little to no technical expertise and makes use of pre-existing automated tools. Thus, the skill is considered to be distributed slightly on the lower side of the scale. There are no resources to spend. Example types that fit this profile are *Irrational Individual* or *Thief*.

Profile 2 Here, a single person or a small group with the same interest, who exchange knowledge, is assumed. The skill should be distributed as it is expected in the whole population. Examples are *Disgruntled Employee* and *Anarchist*.

Profile 3 In this profile, the adversary is organized in a group that has organizational structures. The assessed organization is in particular focus of the adversary. The members of the group have technical background or significant knowledge of the domain. The group has some resources to spend, but are limited in the amount. Examples are *Radical* or *Civil Activists*.

Profile 4 This profile assumes the most threatening type of group. The members have an extensive expertise and are organized in a group, that works with a dedicated focus. They possess essentially endless resources and may acquire any service that can help them in achieving their goal. An example is the *Government Cyberwarrior*, who are also known as nation-state actors.

In the end, each organization must determine the type of adversaries which might have an interest in attacking the organization for itself, as the context specific features that draw attention are only known to local assessors. With the profiles proposed in this section, a first set for the evaluation of the concept and a general guidance on which aspects can be considered is offered. Using the adversary profile factors, the calculation of the assessment shall be weighted accordingly. Of course, determining a single profile can not represent the issue that an organization can become a target solely by coincidence, for example when an adversary confuses two different organizations. Thus, definite knowledge on who will or will not target a certain organization does not exist. A possible solution to that issue is discussed in Section 5.2. At this point, the most likely profile shall be chosen.

We now turn to the estimation of the effort required to fine-tune an existing exploit or create a new exploit, to finalize the assessment concept for different vulnerabilities.

3.1.5 Vulnerability Exploitation Effort

When estimating the vulnerability exploitation effort, the two cases—case one: an exploit already exists which can be fine-tuned to the current situation and case two: a new exploit has to be created—have to be distinguished. The TTC model uses time as the score for a rating. For the first case, in which an exploit exists, McQueen et al. [69] base their chosen value of 8 hours or one working day on an experiment, which simulated an attack of adversaries trying to break into a system and measured the time taken to successfully exploit vulnerabilities. The time it takes to create a new exploit is set to the average time of 5.8 days from vulnerability announcement to exploit code availability. In the calculation of TTC, this value is scaled based on the estimated number of tries needed by the adversary to find a vulnerability.

The approach of McQueen et al. does not differentiate between different vulnerabilities and estimates the same amount of time for each one regardless of its features. This concept neglects the fact that there can be significant differences. For example, if the exploitation requires user interaction and authentication in order to be successful, the

effort can be considered considerably higher in comparison to a vulnerability which can be automatically exploited without any form of authentication. Zieger et al. [114] take this fact into account and improve the calculation by introducing the utilization of available vulnerability information which indicates the difficulty of the exploit. The authors leverage the base score of CVSS, which includes three metrics that are useful for that purpose. First, the *access complexity (Ac)* metric indicates the complexity required for the exploit once the adversary has gained access to a system. Second, *authentication (Au)* measures how many times the adversary is required to authenticate until a successful exploitation is possible. Third, *exploitability (Ex)* represents the current state of the exploit code maturity. All factors are included in the calculation of the *exploit complexity* score as shown in Equation 3.3, that is used to scale the amount of usable vulnerabilities and consequently the probability for process one.

For this concept, scaling the exploitation time by an *exploit complexity* score as defined by Zieger et al. [114] will be included. CVSS data is available for most of the vulnerabilities and can easily be made use of. At the point in the vulnerability life cycle when an advisory is published, it can be assumed that the CVSS rating is available as well. When the temporal metric *exploitability (Ex)* is not available, the default value for *not defined* in CVSS reflects the most critical assumption. The possibility of missing values can therefore be disregarded.

Using different base values other than the ones of McQueen et al., is often seen in other works. For example, Rencelj Ling and Ekstedt [84] define a TTC model for industrial control systems, using the mean time to exploitation discovered in the work of Ablon and Bogart [5], who analysed 207 zero-day exploits. However, the values for this work are going to be left as defined by McQueen et al. This decision was made on the following considerations: The values are based on observations derived from reports that analysed a subset of all exploited vulnerabilities. They are prone to be biased by the selection of vulnerabilities included in the data of the analysis. It influences the derived heuristics which makes the generalization potential questionable. Newer estimates have the same issues, which prevents improvements in that aspect. Furthermore, the goal of this assessment concept is not to estimate an accurate time, implying that the ratio between the different exploit cases is considerably more important than the actual values. Also, as stated before, the significantly more important aspect is that the features of the vulnerability are included as a scaling factor, which will also be included in this concept. For those reasons we leave the evaluation of other base values for future work.

With the methods presented to this point, vulnerabilities can be assessed in consideration of the existence of an exploit and the time it will take an adversary to fine-tune or create a new exploit. Using them will result in a rating of vulnerabilities. This neglects the fact that advisories can have one, two or even many vulnerabilities. Therefore, we will now address this issue by regarding advisories as a construct of multiple vulnerabilities.

3.1.6 Advisory Assessment

Advisories must be regarded as a construct of multiple vulnerabilities. Consequently, a single rating based on the ratings of each vulnerability must be created. Here, possible solutions are outlined which aim to achieve that goal.

In the TTC model, vulnerabilities are not differentiated. Only the difference in the number of vulnerabilities influences the final result. The calculation regards the generic probability of a successful exploitation. Essentially this results in a higher risk (represented as a shorter time to compromise in TTC) with a greater number of vulnerabilities. Thus, this approach can not be used. Even though Zieger et al. [114] introduced vulnerability characteristics with CVSS values, their final result suffers the same symptom.

In general, the following considerations shall be reflected by an assessment method. The number of vulnerabilities of an advisory shall have an influence, but not be the deciding factor. The vulnerability's impact on the final rating shall be adjusted based on its estimated severity. This requirement is already fulfilled by the assessment methods presented in the previous sections, if they are summed up. Therefore, we propose the following method. All vulnerability ratings, as determined by the previously presented methods, shall be added together. This total score shall be adjusted using a saturation function which fulfils $lim_{x\to\infty} f(x) = 1$ and $f(0) = 0$. Utilizing a saturation function has the advantage that the influence of a high number of vulnerabilities is reduced. Of course, more vulnerabilities will still lead to a higher score, but the score is normalized to $[0, 1)$ independent of the total. Another advantage is that the user is able to adjust how fast the saturation is reached by setting a custom function.

3.1.7 Advisory Contextualization

In the last step, the goal is to adapt the rating of the advisories to the local settings of an organization. To achieve that goal, the advisories shall be examined in the context of

the organization's network, as the localization of the affected asset plays an important role for the urgency with which the advisory should be regarded. Consider this minimal example. There are two hosts. Both hosts can be mapped to exactly one security advisory, which have a similar rating as calculated by the previous steps. However, host one is a work station that is not connected to the network and can only be used by a local operator. In contrast, host two is connected to the internet and is reachable from outside the organization. Which advisory would you prioritize? Of course, host two is the more severe threat, because the vulnerabilities of the advisory may be remotely exploited and the host can afterwards be abused to further penetrate the network of the organization. This movement of adversaries is also known as lateral movement, that describes the process when an adversary moves from one host to the next in an organization's environment [12]. Essentially this describes the assumed perception of advisory exploitation paths. Before one can exploit advisories on an internal host, an accessible host which can access the internal one must be infiltrated first.

Attack graphs and attack trees are the most commonly used methods of representing cyberattacks using attack modelling techniques. They share the general idea that cyberattacks are sets of exploits that may require preconditions, create post-conditions and eventually lead to the goal [62]. Both have been used for a long time to model and analyse network vulnerabilities [10, 81]. There has been a lot of work on the generation and analysis of those methods [74]. For example, the TTC model has been used to estimate a network's mean time to compromise [75], or the work of Aksu et al. [8] defines a vulnerability and asset centred risk assessment model and possible metrics.

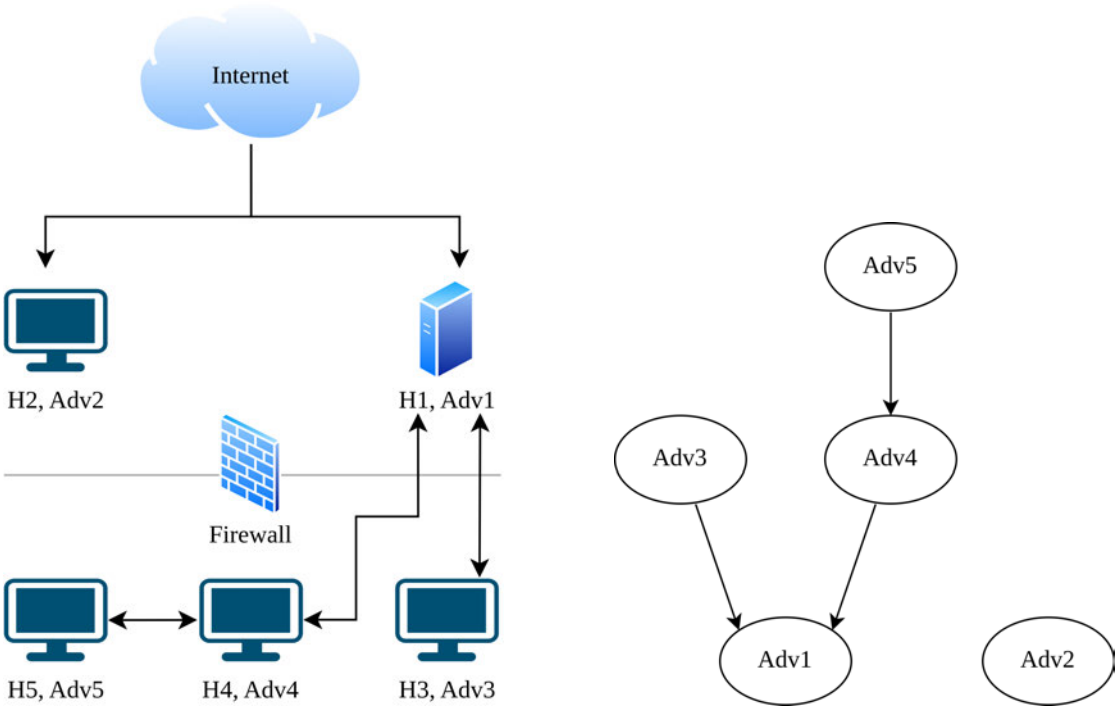
For the assessment of advisories, the idea proposed in the work of Sawilla and Ou [91] is adapted. The authors introduce an algorithm for the analysis of a dependency attack graph. In a dependency attack graph, a vertex represents an asset in the network which can be reached by traversing the graph through other vertices. Consequently, the goal of reaching that vertex depends on the other vertices. This can be illustrated by two hosts, a and b, that are connected, but where only host a can be reached by the adversary. Therefore, in order to achieve the goal of compromising host b, host a must be compromised first, which shows a *dependency* among the two exploits. Applying this concept to the prioritization of advisories, one can derive that to exploit advisories on host b, any advisory of host a must be exploited first. As a result, advisories of host a should be prioritized.

Typically, the flow of an organization's network is protected and restricted using a firewall. There are two different kinds of firewalls. The first one, called a host-based or distributed firewall, operates as a software application on a single network endpoint, i.e. hosts, and thus only protects the hardware running the application. Network-based firewalls are a collection of components between two networks and thus apply the filtering for a whole network topology [48]. For this work, an export of all firewall rules is assumed, which must further be transformed to a list that indicates which hosts can connect to each other. Protocol or port specific connection policies will be ignored for the demonstration. This reduces the complexity in a way that the applicability of the assessment concept can be shown. Assuming the starting point of the adversary is another basis for the implementation that must be regarded. Of course, if the adversary starts the attack from a host already within the network, the dependency graph changes, as other connections are possible.

In Figure 3.3 a simple example is shown to demonstrate the idea of the transformation process from a network graph to an advisory dependency attack graph. The basic network setup is illustrated in Figure 3.3a. As aforementioned, the network configuration is simplified to the point where solely the possible direct connection between two hosts is indicated. For this example, host one can connect to both host three and four, but not to host five. Host five can only be reached through host four. Note that the arcs are directed to illustrate the possibility to establish a connection. Even though not included in this example, this differentiates the possibility that host a can establish a connection to host b, but not vice versa.

3.1.8 Concept Summary

In summary, the proposed security advisory assessment concept consists of three steps. First, the different vulnerabilities of each advisory are enriched with the general considerations defined by the TTC. Each vulnerability is rated regarding the aspects of existence of an exploit and assumed exploitation effort in relation to the adversary's skill and resources. In the second step, advisories are viewed as a construct that consists of multiple vulnerabilities to assign a rating based on the different vulnerabilities addressed by the advisory. Finally, for the third step network information provided by the user is required. Using that information, each advisory is assumed to apply to a set of assets in the organization's system architecture. This knowledge is represented in a dependency



(a) Network graph (b) Advisory dependency graph

Figure 3.3: Visualization of the dependency attack graph creation. Figure 3.3a illustrates the network setup of the hosts. Arrows indicate that a host can connect to another host. The labels list the host and advisory ID. In Figure 3.3b the resulting dependency attack graph is shown.

attack graph to show how an exploitation of one advisory can lead to the exploitation of another. The ratings of an advisory are adjusted based on this knowledge.

3.1.9 Data Description

In order to implement and evaluate the concept, several data sources were used. For the implementation itself only open-source data were considered. The advisories necessary for the evaluation of the model were provided by a national CERT. Since they are not required for the implementation of the concept, the applicability is not hindered by using them for the evaluation. Table 3.1 shows an overview of all data sources and assigns a unique name for each source which will be utilized for the remainder of this work. Now, a description and characterization of each data source will be given. They are referred to as Databases (DBs), because each one forms an organized collection of data.

Table 3.1: Overview of the data sources used for this work.

Name	Summary
Advisory DB	The advisory DB provided by the national CERT, including all advisories ever published.
Vulnerability DB	Snapshot of the National Vulnerability Database (NVD). A vulnerability listing that defines the CVE ID, a description and related CWEs.
Weakness DB	Snapshot of the Common Weakness Enumeration (CWE). A weakness listing that defines the CWE ID, a description and exploitation consequences.
Exploited CVE DB	Joined set of CISA's known exploited catalogue [22] and Suciu et al. [99] identified exploited CVEs.

Some publishers may already include information about associated vulnerabilities, but this is not the case for every advisory. Thus, a vulnerability database is necessary to complete missing data. In the enumeration of vulnerabilities, the **National Vulnerability Database (NVD)**[21] plays an essential role. New CVE entries are requested at authorized organizations and reviewed before being published to this database. It is the industry standard for the identification of vulnerabilities and typically used in security advisories for referencing. The data can be accessed via an Application Programming Interface (API) or be downloaded as a JSON feed of the entire data. For this work, a JSON snapshot was downloaded on 02 January 2023 18:10pm. Using this data allows to enhance the information in the advisory. Data points can be mapped using CVE-ID references, introducing additional knowledge of each vulnerability, which includes CVSS base vectors, descriptions and CWE references. This database poses two downsides that are relevant for this work: the temporal scores of the CVSS vectors are not provided and the database is known to occasionally have a delay completing their information, which may lead to problems when assessing advisories that include recently published vulnerabilities [11, 36, 89]. Efforts have been made to predict missing information [33, 36], but we consider this out of the scope of this work and therefore encourage the use of a more complete database, if available, and deem NVD sufficient to implement the concept.

[21] https://nvd.nist.gov/, accessed on 16 June 2023.

The references provided with the vulnerabilities can further be used to derive information from the **Common Weakness Enumeration (CWE)**[22] maintained by MITRE. Again, the unique IDs can be utilized to directly map related CWEs to CVEs. The relation between the two is that a weakness is a deficit in the concept of software or hardware that could lead to a vulnerability [6], as introduced in Section 3.1.1. Thus, a vulnerability could be considered an instantiation of one or more weaknesses. Every weakness in the database comprises i.a. of a description, its relations to other weaknesses and a list of common consequences. Especially the consequences are of interest, because they are a concise set of values that define the impact if a weakness is exploited, e.g. *Bypass Protection Mechanism*. In the vulnerability data, such impacts are only described in textual form, consequently requiring NLP to extract the specific values. By mapping all CWEs to a CVE and creating a set of unique impacts, the text processing can be avoided and an exact result be build for most cases.

The snapshot of the national CERT's advisory database consists of all security advisories published by the CERT from 2013 until 03 June 2022. It contains 20243 unique advisories and will be used to evaluate the implementation of the concept. Each advisory has an ID, the publishing date, the last modification date, a description, CVE references, an assigned CVSS vector and a DAF-score. Generally, the assigned CVSS vector of the most severe vulnerability is used. They may be altered by the author of the advisory. However, this is only rarely done, according to an employee of the CERT. Based on the chosen vector, the DAF-score is calculated for the advisory, which defines a rating resulting in an ordinal value (see Section 2.3).

To create a ground truth of known exploited vulnerabilities, two sources are used. First, in the recent work of Suciu et al. [99], the authors identified 32093 known exploited vulnerabilities. They used various sources for the identification, i.a. the temporal score values *Functional* and *High* of CVSS provided by IBM XForce Exchange[23] and Tenable Nessus[24], data of commercial tools such as Metasploit[25] and Canvas[26], and scrapes of open source data, e.g. AlienVault OTX[27]. The collection is provided in their GitHub repository.[28] Even though the authors declare in their paper that they identified 32093 vulnerabilities, the list they published only includes 16503. However, as the authors

[22] https://cwe.mitre.org/data/downloads.html, accessed on 16 June 2023.
[23] https://exchange.xforce.ibmcloud.com/, accessed on 20 June 2023.
[24] https://www.tenable.com/products/nessus, accessed on 20 June 2023.
[25] https://www.metasploit.com/, accessed on 20 June 2023.
[26] https://www.immunityinc.com/products/canvas/, accessed on 20 June 2023.
[27] https://otx.alienvault.com/, accessed on 20 June 2023.
[28] https://github.com/sdsatumd/exploitability-tools, accessed on 20 June 2023.

used an extensive amount of data sources that would exceed the scope of this work, the reduced list is used either way. Furthermore, CISA started to publish a catalogue of known exploited vulnerabilities in November 2021 [22]. This offers a convenient extension, because it also includes newer vulnerabilities, whereas the data collection of Suciu et al. [99] stops in 2021. Both sources are combined and duplicates are removed by matching the CVE-IDs, yielding a set of 17371 unique known exploited CVEs.

3.2 Implementation

Throughout the previous section, possible methods and reasoning for the final choice were described. Now that the assessment process is conceptually elaborated, the implementation for an evaluation of the concept will be presented.

Essentially, the process of advisory ranking consists of three sub-processes:

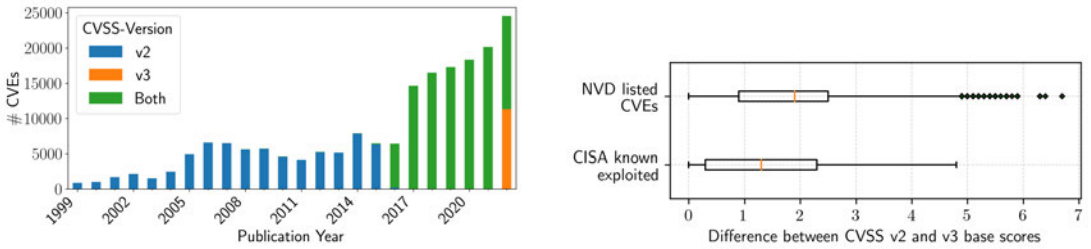
1. Assess each vulnerability.
2. Assess each advisory.
3. Apply a network contextualized scaling of the ratings.

The introduction of the implementation follows the sequential order of the different steps.

3.2.1 Preliminaries

Differentiating between CVSS v2 and v3. Different CVSS metrics will be used for the implementation and the base scores for the evaluation. As aforementioned, the current version is v3.1 and will soon be v4. However, since the *vulnerability DB* is used from NVD, the handling of obsolete versions has to be defined. NVD does not update old CVSS versions to newer ones, as can be seen in Figure 3.4a. Since the release of v3 in 2015, both versions have been maintained for a couple of years. With the final deprecation of v2 in 2022, the first entries with only CVSS v3 appear in that year. In total, there are 11334, 73333 and 106754 with CVSS versions v3, v2 and both versions, respectively.

The heterogeneity is a problem due to the difference of CVSS metrics and scores. Figure 3.4b illustrates this issue for NVD and CISA's known exploited catalogue. Obviously, only CVEs with both versions could be included in the calculation. The boxplot clearly shows for both datasets, that the median of the difference is above one and the 75% quantile even above two. A difference of 15-20% would result in a significant impact in the rating. Therefore, whenever CVSS values are used, care has to be taken for possible issues.



(a) CVSS versions in the NVD database.

(b) Difference in CVSS base scores.

Figure 3.4: Analysis of the NVD CVSS versions. Figure 3.4b shows the difference between both version's base score, for the NVD DB and the CISA known exploited catalogue. All CVEs with both versions were included in the calculation.

Missing CVSS values. During the implementation, missing CVSS have been noticed in the *advisory DB*. This is the case when other identifiers than CVE are used for the vulnerabilities, which as a consequence prevents the retrieval of according CVSS vectors from the NVD DB. As CVSS is the most prevalent method to identify vulnerabilities, the assumption that most organizations rely on it and do not have additional sources for other identifiers, is applicable. To cope with the issue of other IDs, a default value will be used for all vulnerabilities that do not have a valid CVSS vector. It should approximately match the mean base score. In our *vulnerability DB* snapshot, all CVEs with a v3 vector had an average base score of 7.21 with a standard deviation 1.66 and a variance of 2.76. Therefore, the following vector has been chosen, which results in a base score of 7.0 and also in a medium exploit complexity, which will be introduced later in this section:

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:L/A:L

Adversary profiles. Throughout the rest of the calculation both the skill level and the resource level will be used. Therefore, those have to be defined first. The profile descriptions in Section 3.1.4 give four different profiles, where the resources and skill rise gradually by each profile. In Figure 3.5 the proposed skill distributions are shown. Fol-

Table 3.2: Adversary profile scaling factors.

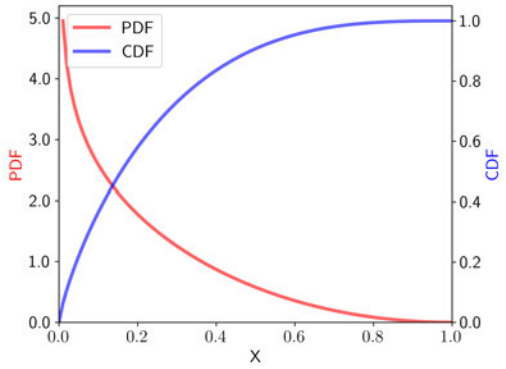
	profile one	profile two	profile three	profile four
exploit factor (ef)	1.0	1.1	1.1	1.2
advisory factor (af)	1.0	1.2	1.5	2.0

lowing the PDFs, it is clearly visible that with each profile the concentration of assumed skill shifts towards a higher level with each profile. Profile scaling will be applied to (i) the probability if an exploit can be used (*exploit factor*) and (ii) to the advisory as a whole to allow the assumption of working on multiple vulnerabilities in parallel (*advisory factor*). For each case an individual factor is used. For the defined profiles the values given in Table 3.2 are proposed.

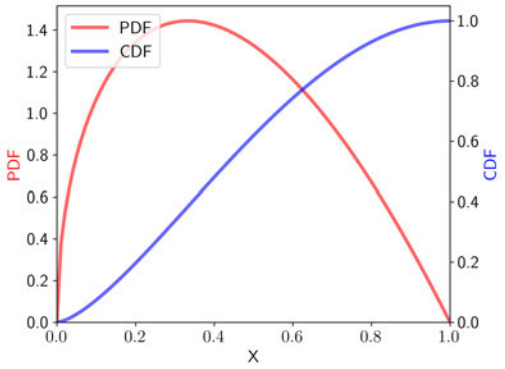
3.2.2 Vulnerability Assessment

The first step for the assessment is to determine the probability that a vulnerability exists. This can be done using either of the following methods: (i) vulnerability is included in the known exploit list, (ii) exploit existence is indicated by the CVSS score, (iii) probability based on heuristics values derived from the CVSS base, or if available, temporal score or (iv) assuming a general distribution as used in the TTC. Choosing a method follows a logical order. Definite information used by methods *i* and *ii* is favoured over heuristic estimation in method *iii*. Method *iv* serves as a backup purpose, if no other information is available. To create the heuristic for method *iii*, CISA's known exploited catalogue [22] is used, as it provides an up-to-date view on exploited vulnerabilities. The following procedure was performed for CVE version 2 and 3 individually by filtering the data for available base scores. Out of 868 known exploited CVEs, 707 had a v3 and 829 a v2 vector.

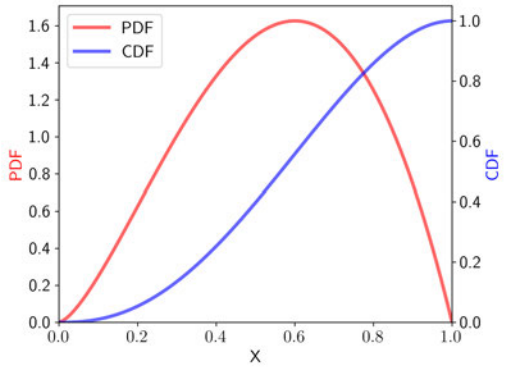
Each vulnerability was assigned to a base score range and normalized so that each range contains the relative frequencies of unique values. Then the probabilities are multiplied with the total number of exploited CVEs, 32093 identified by Suciu et al. [99], to estimate the total number of exploited ones in each range. Finally, the number of exploited vulnerabilities is divided by the total number of vulnerabilities in each range. Table 3.3 shows the resulting share of exploited vulnerabilities for each range. The lower ranges in both versions had no known exploited vulnerabilities. However, it is assumed that the probability that an exploit exists is never zero, because there is no definite knowledge



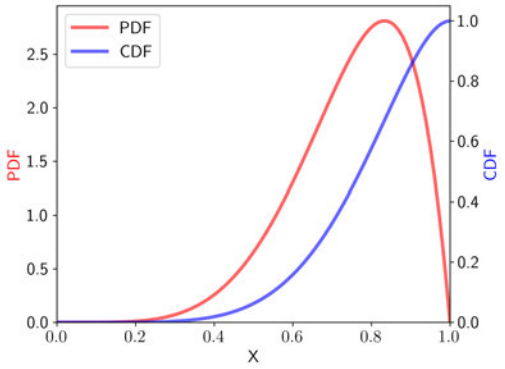
(a) Profile 1, $\alpha = 0.8$, $\beta = 3$



(b) Profile 2, $\alpha = 1.5$, $\beta = 2$



(c) Profile 3, $\alpha = 2.5$, $\beta = 2$



(d) Profile 4, $\alpha = 6$, $\beta = 2$

Figure 3.5: Skill distributions for the adversary profiles. A beta distribution function with different α and β parameters were used to create the skill distributions.

about that fact. Therefore, we assigned a default value of 1.00% for those ranges. This heuristic follows the expected trend that with a higher severity the number of exploits increases. With the heuristic values determined, now the *exploit probability (ep)* for a *vulnerability (v)* can be formally defined as follows.

$$ep = \begin{cases} 1, & \text{if } v \in ke \\ 1, & \text{if } E = H \vee E = F \\ eh(v), & \exists CVSS \\ 0.1676, & \text{else} \end{cases} \tag{3.2}$$

Table 3.3: Exploit probabilities by CVSS base score range. Both CVSS versions, v2 and v3, have been calculated separately and are therefore listed individually.

Range	[0, 1)	[1, 2)	[2, 3)	[3, 4)	[4, 5)	[5, 6)	[6, 7)	[7, 8)	[8, 9)	[9, 10]
v2 (%)	1.00	3.80	4.38	3.37	9.24	7.37	22.55	31.34	30.43	65.72
v3 (%)	1.00	1.00	1.00	2.70	5.79	7.36	7.59	33.17	43.04	58.00

where *ke* is the known exploited list, *eh* the exploit heuristic function that returns the according value of Table 3.3 and 0.1676 the general exploit probability. Because it follows the assumption that each vulnerability has the same probability to have an exploit, we chose to refer to it as a general probability. It was determined by dividing the number of exploited CVEs by the total of number of vulnerabilities in the NVD DB.

Next, the probability that the adversary can use the exploit has to be determined. It depends on three factors. First, the complexity of the exploitation process differs between vulnerabilities. In the work of Zieger et al. [114], the authors proposed an *exploit complexity* calculation using CVSS metrics. They argue to use *access complexity (Ac)*, *authentication (Au)* and *exploitability (Ex)*, which all indicate different aspects of the vulnerability exploitation complexity. As we saw in Section 3.2.1, a differentiation of both CVSS versions is necessary for the implementation. Due to the fact that both versions will be supported, the *exploit complexity* score of Zieger et al. [114] must be adapted for the new version as well. For CVSS v2, the original score as proposed by the authors will be used:

$$exc_2(y) = 1 - \left(Ac(y) * Au(y) * \frac{Ex(y) - c_{2_1}}{c_{2_2}} - c_{2_3} \right) * c_{2_4} \tag{3.3}$$

with $c_{2_1} = 0.6$, $c_{2_2} = 0.45$, $c_{2_3} = 0.0875$, $c_{2_4} = \frac{1}{0.3568}$.

With version three of CVSS, the metrics in the base groups were adjusted. A new metric that is relevant for the exploitation complexity is *user interaction (Ui)*, which can either be none or required, and represents if a human interaction with another person other than the adversary, is necessary for a successful exploitation. Clearly, it becomes significantly more difficult to perform an exploit if such an interaction is required. Therefore, we propose to introduce the metric to the exploit complexity calculation as follows:

$$exc_3(y) = 1 - \left(Ac(y) * Pr(y) * Ui(y) * \frac{E(y) - c_{3_1}}{c_{3_2}} - c_{3_3} \right) * c_{3_4} \tag{3.4}$$

with $c_{3_1} = 0.6$, $c_{3_2} = 0.5$, $c_{3_3} = 0.0457$, $c_{3_4} = \frac{1}{0.3994}$.

The scaling values $c_{3_{1-4}}$ have been adjusted so that the new equation meets the requirement $exc \rightarrow \{x \in \mathbb{R} : 0 \leq x \leq 1\}$ and can be used interchangeably with Equation 3.3 of exc_2 for CVSS v2.

Second, a scaling based on the assumed skill distribution must be introduced. A skill distribution *sf* should return the probability that the adversary has a skill level $x \in [0, 1]$. Thus, to scale the exploit complexity, the integral between 0 and 1 of the skill times the exploit complexity has to be calculated.

$$P_{\text{can use}} = 1 - \int_0^1 sf(x) * exc_x \, dx \tag{3.5}$$

By subtracting the result from 1, a lower exploit complexity will result in a higher probability, while a higher complexity results in a lower probability.

Third, the advisory *exploit factor ef* has to be respected. If the adversary has the resources to either acquire the exploit itself or the expertise to use it, it effectively raises the probability that the exploit can be used. Therefore, we propose to multiply the calculated function with the *exploit factor ef* set in the profile: $P'_{\text{can use}} = P_{\text{can use}} * ef$. Results greater than one will take one as the value.

Altogether, a probability that an exploit exists and the adversary is able to use it, can be calculated which represents the probability for process one:

$$P_{\text{one}} = ep * P'_{\text{can use}} \tag{3.6}$$

The second process represents the creation of a new exploit. This must be done either when no exploit exists or the adversary can not use it. For that purpose, the probability can easily be determined by the inverse probability of process one:

$$P_{\text{two}} = 1 - P_{\text{one}} \tag{3.7}$$

Using both probabilities, the final *vulnerability rating vr* can now be determined by scaling the base values for each process with the probability of it:

$$vr = P_{\text{one}} * 1 + P_{\text{two}} * 5.8 \tag{3.8}$$

As argued during the introduction of TTC, the base values were taken from the original paper of McQueen et al. [69].

3.2.3 Advisory Assessment

We now turn to rating advisories as a hole. A security advisory consists of multiple vulnerabilities, which we formalize as follows. Let $A := \{a_1, a_2, ..., a_n\}$ be the set of all advisories that must be rated and $V := \{v_1, v_2, ..., v_n\}$ the set of all vulnerabilities of all advisories with a rating vr_x from the previous calculation. For each subset of vulnerabilities that belong to an advisory Va_n applies $Va_n \subset V$. Due to the fact that the current assessment relies on the assumed timings required to run the different processes, higher ratings currently indicate a lower priority. To change that, the ratings of the vulnerabilities are now going to be normalized first. The minimum $vr_{min} := min\{vr_x | vr \in V\}$ and maximum $vr_{max} := max\{vr_x | vr \in V\}$ ratings are used to inverse and normalize with

$$vr_{x\_new} := 1 - \frac{vr_x - rv_{min}}{vr_{max} - vr_{min}} \tag{3.9}$$

for each vulnerability rating. A higher rating now indicates a higher priority. Before the saturation function is applied, the sum of all vulnerability ratings is calculated for each advisory.

$$asr_n := \sum_{x \in Va_n} vr_{x\_new} \tag{3.10}$$

Another aspect that has to be regarded is the possibility of performing exploitation efforts in parallel. If an advisory has more than one vulnerability and the adversary the necessary resources to work on two exploits at the same time, the rating shall be adjusted. For that purpose, we scale the summed ratings with the *advisory factor (af)* defined in each resource profile $asr'_n = asr_n * af$. Finally, a saturation function is applied to the sum of all vulnerability ratings to calculate the final set of advisory ratings $AR := \{ar_1, ar_2, ..., ar_n\}$:

$$ar_n := \frac{asr'_n}{asr'_n + s} \tag{3.11}$$

with the function scaling value $s = 2$.

By using a saturation function, the domain of the result is restricted to $[0, 1)$, which prevents extreme results for advisories with many vulnerabilities. Furthermore, by choosing an entirely different saturation function or altering the parameter s, the assessor can influence how fast the saturation is achieved or even introduce a maximum value. At this point, each advisory has a single rating and can now be considered in the context of the assessors network configuration.

3.2.4 Advisory Network Attack Graph

The minimal requirement for this concept to work is that the user provides basic network configuration information. This must include a list of hosts present in the network, which should be distinguishable by a unique identifier $H := \{h_1, h_2, ..., h_n\}$. Additionally, for each host all possible connections must be specified $HC := \{hc_1, hc_2, ..., hc_n\}$ with $HC \subseteq H \times H$. At this point, it is enough to state that a host can connect to another host without further details. The input of the rated advisories from step one and two must be prepared for this step. As stated in Section 3.1.1, using concepts like CPEs and SBOMs, each advisory can be mapped to one or more hosts that are affected by the advisory. However, automation of this step is left out of the scope of this work. The result shall be a set of mappings $AH \subseteq A \times H$ with $(a, h) \in AH$ indicating that advisory a applies to host h.

For this work, an attack graph is defined as a tuple $G = (A, C, r)$, where A is the set of host-specific advisories (vertices), C is the set of connections (arcs), which indicate that advisory u depends on advisory v due to the fact that the host, to which advisory u applies to, can only be reached by the adversary if the advisory on host v is exploited first, and r is a mapping of ratings to the connections. A special case for the advisory graph is that one host can have multiple advisories, which of course allows a 'connection' between those two advisories.

To perform an analysis from a specific vantage point, a start vertex has to be inserted. It should also be provided by the assessor and will be added to the graph as a single vertex with all possible connections assumed for that location in the network. Hosts that are accessible by the internet are always considered to be reachable by an adversary. Including the starting vertex can be either done before or after generating the advisory attack graph. If done before, it is added to H with a specific start ID and all connections to HC. Otherwise, all possible connections and the vertex have to be added to the graph itself, but heed has to be taken to consider the advisories as vertices and not the hosts.

Now, a dependency attack graph can be derived from the advisory ratings AR (Equation 3.11) and the aforementioned user inputs HC and AH. The process is shown in Algorithm 3.1.

Algorithm 3.1: Advisory network attack graph generation.

```
1  input: HC, AH, AR
2  output: G
3  A ← new list
4  C ← new list
5  r ← new hash table
6  for advisory, rating in AR
7    insert advisory into A
8    for (a, h) in AH where a = advisory
9      for (fromH, toH) in HC where h = fromH
10       for (a2, h2) in AH where h2 = toH
11         or h2 = h  // include other advisories from current host
12         and a2 != advisory
13           connection ← (advisory, a2)
14           insert connection into C
15           r[connection] ← rating
16       endfor
17     endfor
18   endfor
19 endfor
20 return G(A,C,r)
```

To reduce graph complexity, prevent cyclic value propagation and only regard possible paths from the start point for updating the ratings, the graph is going to be filtered for shortest paths. The starting vertex is the source, the ratings are the weights for each connection and the goal is to find the shortest paths from the source to all possible destination vertices. Note that there might be vertices that can not be reached from the chosen starting point. A possible solution to solve that problem is Dijkstra's algorithm [32].[29] Shortest path algorithms usually choose those paths with the least weight. Since the rating indicators are exactly the opposite—higher ratings mean a higher exploitation risk and thus indicate the shorter path—, an according algorithm that accounts for that fact has to be chosen or the weights have to be inverted solely for the path finding process.

Now that the graph solely includes the shortest paths, the ratings can be adjusted. As a reminder, keep in mind that the graph is directed and a connection indicates that the out-vertex depends on the in-vertex, because the host of the in-vertex can establish a connection to the host of the out-vertex. Also, the weight of an arc represents the rating of the advisory. To emphasize the dependencies, we propose to update the weight of each

[29] As shortest paths algorithms are commonly implemented in software libraries, see for example https://networkx.org/documentation/stable/reference/algorithms/shortest_paths.html, and there are many alternatives, we will not reprint the algorithm at this point. Also, a survey on other alternative algorithms can be found in the work of Madkour et al. [67].

vertex with the weights of all connected out-vertex. That way, the advisories pass on their weight to the advisory that must be exploited first due to networking restrictions. Each vertex should only pass on their weight a single time and only when all incoming weights have been received so that the updated rating will be propagated.

This work follows the fundamental idea of vulnerability dependency attack graphs proposed by Sawilla and Ou [91]. However, due to the changes made, the rating adjustment changes significantly. Despite the changes, an adjusted rating propagation algorithm using an adjacency matrix will be shown, helping future work to relate back to the original work and improving it by incorporating further concepts. Additionally, another approach based on depth-first search is shown, which offers a clearer view on the approach used in this work, but which can not be applied to the calculation of Sawilla and Ou [91].

The calculation with an adjacency matrix can be achieved in an iterative algorithm. For that purpose, the attack graph is represented as an adjacency matrix $AG = (ag_{jk})$, which is given by

$$ag_{jk} = \begin{cases} r_j, & \text{if } \{j,k\} \in C \\ 0, & else \end{cases} \tag{3.12}$$

where r_j is the j rating in the vector $\vec{r}$ that consists of all vertex ratings. To ensure that each rating is only passed once, a memory vector $\vec{m}$ is created with

$$m_j = \begin{cases} 1, & \text{if } \sum_{k=1}^{n} ag_{jk} > 0 \\ 0, & else \end{cases} \tag{3.13}$$

which sets m_j to zero if no other vertex depends on it, i.e. it never has to pass on its rating, and 1 otherwise. Then, for each iteration step, another filter is necessary to ensure that a vertex does not have any incoming arcs and can safely pass on its rating. We define this as the dependency filter $\vec{d}$ with

$$d_k = \begin{cases} 1, & \text{if } \sum_{j=1}^{n} ag_{jk} = 0 \\ 0, & else \end{cases} \tag{3.14}$$

so that one is set when a vertex has no incoming arcs and zero otherwise. Using both the memory and the dependency filter, the passed on rating will be neutralized if either is zero.

Each iteration step will update the dependency filter (Equation 3.14), update the ratings and finally update the memory filter. This is performed until $|\vec{m}| = 0$, meaning that all ratings have been propagated through the graph. This can be formalized as shown in Algorithm 3.2.

Algorithm 3.2: Graph rating propagation.

```
1  input: AG, m⃗, r⃗
2  output: scaled ratings
3  while |m⃗| > 0
4    d⃗ ← see Equation 3.14
5    s⃗ ← m⃗ ∘ d⃗ // step filter
6    AGT ← AG^T * s⃗ // passed ratings this iteration
7    r⃗ ← AGT // add additional ratings to previous ones
8    m⃗ ← m⃗ ∘ (1⃗ − d⃗)
9    AG ← update with r⃗
10 end
11 return r⃗
```

The output eventually returns a vector of scaled ratings, that have to be mapped to their advisories.

For the depth-first search, the dependency tree representation is used as input. The search begins from the start vertex, updating all ratings in reverse order. That means, each leave of the tree passes on its rating first and afterwards vertices with the highest depth. Here, the output is the list of advisories with updated ratings.

Algorithm 3.3: Depth-first search rating propagation.

```
1  input: G
2  output: scaled ratings
3  function dfs_update(G, vertex)
4    in_edges := {(a1,a2) ∈ G.C | a2 = vertex}
5    out_edges := {(a1,a2) ∈ G.C | a1 = vertex}
6    if |in_edges| > 0
7      dfs_update(G, in_edge) forall in_edges
8    endif
9    for (a1, a2) in out_edges
10     advisory := x ∈ G.A | x = a2
11     new_rating = vertex.rating + advisory.rating
12     advisory.rating ← new_rating
13   endfor
14 end
15 dfs_update(G, start_vertex)
16 return G.A
```

One issue in the generation of the advisory attack graph is that an advisory may affect multiple hosts. This can either be the case when there are multiple hosts with a similar configuration, e.g. work stations with a defined setup, or when the advisory references multiple products that are present in the assets of an organization. This fact raises the question, how the rating shall be influenced by multiple occurrences. To generally cope with such cases, identical advisories need to be distinguishable in the attack graph. A new ID is assigned to each advisory which is a combination of both host and advisory ID. As an advisory is unique per host and host IDs are required to be unique, the new identifier will be unique as well. Then the steps for the calculation are performed as described above. Afterwards, one solution is to group the results by advisory ID and then calculate the sum of the ratings per group. This approach has the advantage, that the number of affected hosts is now also reflected in the rating and not only the network setup. However, this might lead to advisories with a very high rating, e.g. a host type exists very often in a network, as it for example can be the case for employee clients. If that outcome is not desired, another solution is to take the maximum value for each advisory and use that as the final rating.

3.3 Evaluation

3.3.1 Network Description

The implementation is going to be evaluated with two different network examples. The first example is a minimal, invented network. It is specifically tailored to showcase the possibilities of the concept. Using an abstraction level which narrows down the provided information will help to focus on the important aspects. It is shown in Figure 3.6 and will from hereon be referenced to as *network 1*. In *network 1*, there are five different hosts, of which two, *H1* and *H2*, are connected to the internet. Hosts *H3* and *H4* can only be reached through *H1* and have neither incoming nor outgoing direct connections to the internet. The path to *H5*, which also is not connected to the internet, leads trough host *H4* and consequently also through *H1*.

To show the application of the concept in a real-world scenario, a network configuration provided by the national CERT is used for the second evaluation. It will be referred to as *network 2*. A complete network architecture of an organization scales with the size of the organization and becomes humongous in the number of hosts, services, software and

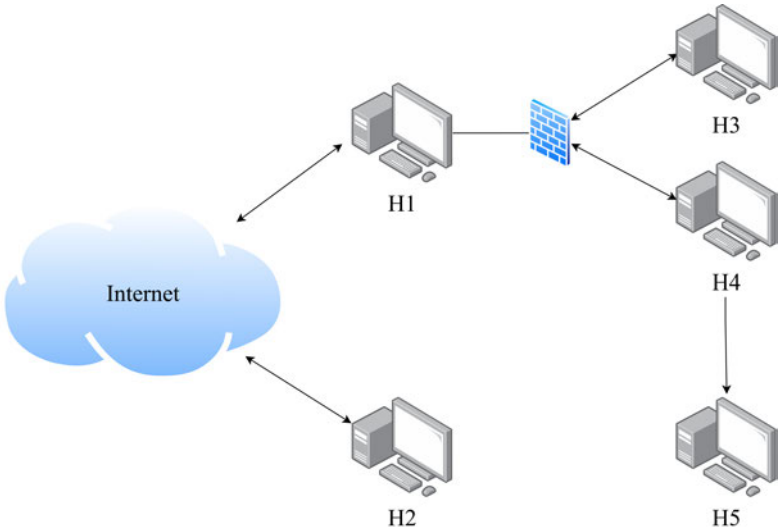


Figure 3.6: The different hosts in the minimal working example network 1 and their communication possibilities.

firewalls. This diminishes the confirmability of the calculated result. For that reason, the set of hosts were reduced to those necessary for a specific service the CERT provides. They are distributed among different layers in the network and their communication is restricted by multiple firewalls, both network- and host-based. A schematic visualization is shown in Figure 3.7.

It is apparent, that already at this reduced example, regarding both hosts and network configuration, the interactions can become unclear. We will highlight some aspects that are going to become of interest when the advisory evaluation concept is applied. The hosts are distributed among two different locations. Location one has a three zone concept, in which layer two and three are protected by a firewall. The first layer has a direct connection to the internet. In location two, there is a single zone which is protected by another firewall. However, here a direct connection to *H1*, *H2* and *H3* is allowed. In general, most hosts which are behind a firewall can not be connected to from outside the firewall zone. An exception are the aforementioned hosts in location two, and hosts *H4* and *H7-8*. Even though hosts *H7-9* can theoretically be reached, there is the requirement to establish the connection from a set of other hosts. In case of *8*, it is possible from a customer device or, which also applies for hosts *H7* and *H9*, from host *H6*.

For both examples, the adversary is assumed to be located outside of the organization's network, which is equivalent to assuming an attack over the internet. This means, that at

the start of the attack, all advisories that are reachable from the internet are exploitable for the adversary.

For both example networks, a number of security advisories were chosen. Each host got a random number between one and three advisories, which were selected randomly from the *vulnerability DB*. The assessed prioritization order is going to be compared against two other concepts applied in practice: the DAF-Score and the maximum CVSS base score from all referenced vulnerabilities of an advisory. Both values are taken from the databases used for this work. While the DAF-Score is already included in the *advisory DB* and can be directly imported, the CVSS base scores had to be mapped from the *vulnerability DB*. An overview of the advisories in *network 1* is given in Table 3.4 and for *network 2* in Table 3.5. Note that in *network 2* the advisory with the ID *Advisory_15534* applies to two different hosts, *H1* and *H6*.

Table 3.4: Network 1 advisories by host.

Host	**ID**	**DAF-Score**	**Max CVSS**	**# Vulnerabilities**
H1	Advisory_16824	high	8.1	2
	Advisory_20128	medium	9.8	1
	Advisory_9081	low	8.1	6
H2	Advisory_16358	low	7.8	1
	Advisory_16437	low	5.9	1
	Advisory_493	high	8.8	1
H3	Advisory_16585	high	7.5	4
H4	Advisory_20242	low	7.8	2
	Advisory_17486	low	7.5	1
H5	Advisory_18840	high	9.8	36
	Advisory_2240	low	7.5	2
	Advisory_1906	low	5.9	2

3.3.2 Results

The advisory assessment concept was applied to both examples. Now, we will discuss the outcome for both networks, by highlighting the important and interesting aspects. The full list of results can be found in Appendix A. There, all result values of the advisories are listed for both network examples, and additionally, an overview of the

Table 3.5: Network 2 advisories by host.

Host	ID	DAF-Score	Max CVSS	# Vulnerabilities
H1	Advisory_15534	high	8.8	2
H2	Advisory_2490	high	7.5	3
H3	Advisory_2016	high	9.8	1
	Advisory_322	very_high	8.8	9
	Advisory_15904	very_high	9.0	28
H4	Advisory_2080	high	8.8	13
H5	Advisory_17032	high	7.8	25
	Advisory_1849	high	8.1	3
H6	Advisory_17868	high	8.8	11
	Advisory_15534	high	8.8	2
	Advisory_462	high	8.1	4
H7	Advisory_16830	very_high	9.9	23
	Advisory_19489	very_high	8.8	2
H8	Advisory_19765	high	7.5	1
	Advisory_416	high	8.1	4
	Advisory_1029	very_high	9.8	20
H9	Advisory_16620	high	9.6	2
	Advisory_17314	high	8.8	11
	Advisory_16212	very_high	8.1	3
H10	Advisory_15560	very_high	7.8	5
H11	Advisory_16809	very_high	9.1	6

derived prioritization rank based on the different concepts is given. Note that due to the fact that DAF-score has numerous examples per ordinal score value, a meaningful ranking could not be derived, as an order within one group can not be determined, and was therefore left out. When two CVSS scores were equivalent, the initial list order was used as a second factor.

Starting with *network 1*, the initial scores without network contextualization will be inspected first. The advisory with the highest rating is *Advisory_18840* with a maximum CVSS value of 9.8 and 36 different vulnerabilities. What is noteworthy is the fact that when regarding solely the CVSS score or the DAF-Score, there are multiple other advisories that could be ranked on the same position. The actual rating value of the advisory

assessment concept, even though it is normalized, has a comparably large distance to the other scores which shows that it initially received a lot higher ranking—more than triple of the succeeding advisory. This can be explained by the fact that the number of vulnerabilities is significantly higher than in any other advisory, and it is also the only advisory which has known exploited vulnerabilities. On the other hand, the associated host is rather isolated in the network. Consequently, when applying the contextualized assessment, the score of this advisory is passed on to advisories which would enable the adversary to access the corresponding host. That leads to a prioritization of the latter. It is quite obvious that host *H1* is the gateway to the secured network zone. Thus, advisories on this host should be regarded with a higher priority, as a successful exploitation opens the path. In this example, three advisories apply to host *H1*. First, the advisory ratings without network contextualization will be inspected. The DAF-Scores are different—low, medium and high—, but all maximum CVSS base score values are quite high with 8.1 twice and 9.8 once. With this assessment concept they got the following ranking orders (p1, p4): *Advisory_16824* (2, 4), *Advisory_20128* (4, 5) and *Advisory_9081* (3, 2). Interestingly, the different profiles lead to different ranking orders for *Advisory_16824* and *Advisory_9081*, even with a significant difference in the actual ratings in case of the latter. *Advisory_9081* has an average CVSS base score of 6.75 and 6 vulnerabilities, which are emphasized by profile four, as lower rated and overall more vulnerabilities are assigned a higher significance due to a higher skill and resources of an adversary. That change in ranking order leads to the fact that the final network adjusted ranking is also different for each profile: one time *Advisory_16824* ranks first and one time *Advisory_9081*. Here, we see the influence of the shortest path algorithm. Only the advisory with the higher rating will receive the passed on ratings of dependent advisories.

In summary, this example confirmed several expected aspects of the assessment. Advisories which open a path to further exploitable advisories will be prioritized. If there are multiple advisories that open that possibility, the one with the highest rating will be favoured. Before the ratings are adjusted based on the network knowledge, the advisory with an immense amount of vulnerabilities and known exploited ones was prioritized. Next, we saw that advisories with either severe or several vulnerabilities received a higher ranking. Finally, the assumed adversary profile influenced the ranking and shifted the attention based on the set skill and resource level.

In *network 2* the same effects for the assessment can be observed. Only host *H4* and *H6* can open a path to further advisories. Since the possibilities of *H6* are a subset of *H4* and the latter has the advisory with the higher rating, the network adjustment changes

the rating just for a single advisory. This fact makes sense, as one would want to address the most critical advisory, which allows the same lateral movement in the network as less critical ones, first. Coincidentally, the random selection of advisories resulted in a set of very critical ones. Thus, for the rating without network contextualization, especially advisories with a lot and exploited vulnerabilities are prioritized. Consider the top five:

1. **Advisory_15904** 3 of 28 CVEs known exploited. Mean CVSS base score: 7.66
2. **Advisory_17032** 3 of 25 CVEs known exploited. Mean CVSS base score: 7.15
3. **Advisory_16830** 0 of 23 CVEs known exploited. Mean CVSS base score: 7.46
4. **Advisory_322** 4 of 9 CVEs known exploited. Mean CVSS base score: 6.84
5. **Advisory_1029** 0 of 20 CVEs known exploited. Mean CVSS base score: 7.29

This highlights several aspects of the assessment. First, a higher amount of referenced vulnerabilities in an advisory lead to a higher rating. This is also due to the fact that the mean base score for all advisories in *network 2* is never below seven. Second, advisories which have known exploited vulnerabilities rank higher than other advisories with a similar amount of vulnerabilities. Third, known exploited vulnerabilities are so influential, that even advisories with considerably more referenced vulnerabilities are ranked below them.

3.4 Takeaways

For both examples we saw that advisories, which would enable the adversary to use the affected host as a gateway to the network of an organization on successful exploitation, are prioritized. This ranking makes sense since possible entry points for adversaries should be handled as soon as possible. The resulting values before the network adjustment lay an emphasis on advisories with many, severe or exploited vulnerabilities. Regarding the adaption of TTC components, it reflects a focus on advisories with a higher probability of being exploited sooner than others. Generally, choosing a profile with a higher skill level and higher resources will lead to favouring advisories with more vulnerabilities, but also allow vulnerabilities with a higher exploit complexity to receive a slightly higher score. In *network 1*, the impact was so significant that a different path through the network was favoured in comparison to the other profile, which eventually led to a different prioritization order. With the shortest path filtering on the advisory network graph

currently used, only a single advisory of a host will get passed on weights from dependent advisories. However, as we have seen in example *network 1*, both *Advisory_ 9081* and *Advisory_ 16824* on host *H1* are a possible selection for moving on to the next host.

When comparing the different prioritization approaches, the advisory assessment concept proposed in this work can eliminate a shortcoming of the other ones currently used in practice: the significant problem of ambiguous results. In *network 2*, the DAF-score is only able to categorize the advisories in two different groups. The CVSS base score approach is slightly better, but there are nevertheless many duplicate values. Of course, they still offer a first grasp on the importance of an advisory, but are essentially not usable for determining an order due to that fact. In *network 1*, where the DAF-score is diverse, the order within the score, $low \rightarrow medium \rightarrow high$, is almost equivalent to the not contextualized ranking of this concept. One main difference is in advisory *Advisory_ 9081*. A manual inspection of the advisory unveiled a problem of the DAF-score. It is calculated based on the most severe vulnerability or on an adjusted score if the author of the advisory decides to change it. By limiting the calculation to one score, not all vulnerabilities can be adequately represented. For example, the textual description states that only some vulnerabilities require user interaction, yet the CVSS metric is set to *required* for the entire advisory.

Another aspect that caught our attention during the manual inspection of the advisories is the possible impacts of a successful exploitation. Usually the description of an advisory outlines what they are. Especially interesting are those advisories, that have limited impacts of either DoS or information disclosure. If we consider that aspect in respect to the last assessment step, the contextualization of advisories by a network attack graph, it shows a clear limitation of the current approach. Hosts with advisories that for example only impact their availability are not usable for an adversary to perform lateral movement and exploit the next advisory in the network graph. For that reason, we propose to extend the assessment graph by incorporating this knowledge in the following section.

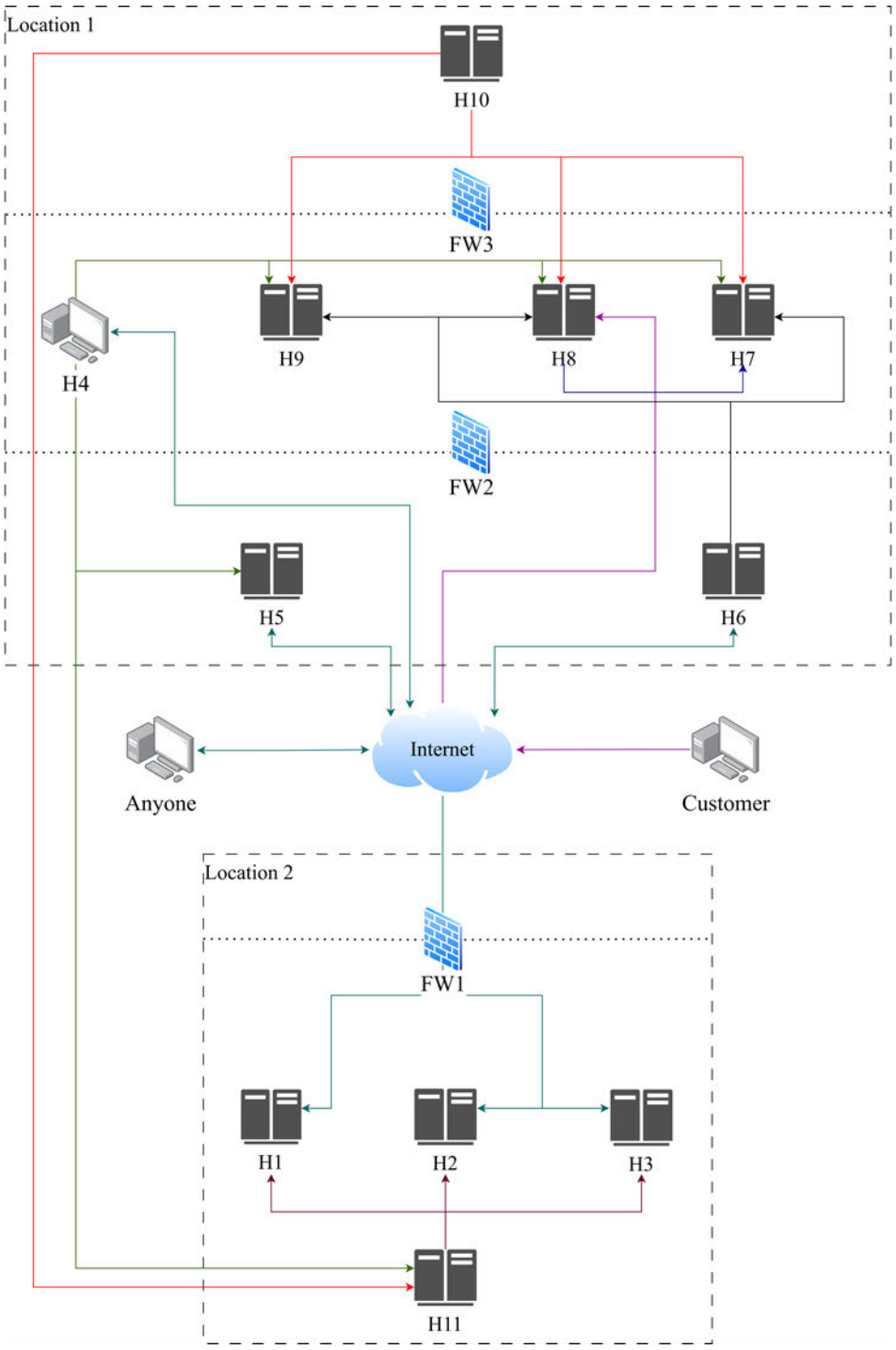


Figure 3.7: The different hosts in the network architecture of a service provided by a national CERT. The different connections are colour coded, because several hosts can establish a connection to another host, but not vice versa.

4 Extension

One of the most important and yet most work demanding assessments is understanding the risk of an advisory in the local context. As highlighted in the evaluation of the first implementation, the current approach does not account for the fact that vulnerabilities of the advisories may not enable lateral movement for the adversary from the host it applies to. For that reason, we propose to extend the concept to diminish this shortcoming and highlight how an improvement with the use of freely available data can be implemented in the assessment process easily. The extension regards the third sub-process of this ranking concept, which was presented in Section 3.1.7. In this assessment step, the user provided network information are associated with the advisories to derive which ones are directly accessible by an adversary and could function as a gateway to the inside of the organization's network. In this section, the process of the calculation will be revised to implement the extension.

4.1 Extending the Concept

A successful exploitation of a vulnerability referenced by an advisory can have various different impacts, which are usually reported in its description. They range from data disclosure to a complete takeover of control by the adversary, if they gain admin privileges. Based on the impact, the adversary has different options to proceed. Considering this fact in the context of the advisory network graph, it influences the possible paths an adversary can take to move through the organization's network. For example, if the exploitation of an advisory impacts solely the availability of the host, adversaries might not be able to move to the next host due to limited privileges and possibilities. This knowledge can be considered in the creation process of the dependency attack graph. However, first a clear set of possible impacts must be determined for each advisory. This task is challenging because the impacts are usually only available in free text form. Therefore, a concept to create the set of possible impacts is necessary.

A related value is used by the *DAF-Score*, the *Loss* value, to adapt the risk of the advisory based on the impact. We refrain from using this value, due to the following limitations. The CERT sets only a single loss value for each advisory. When there are multiple vulnerabilities, which is often the case, there can be various possible impacts. Even a single vulnerability can result in multiple impacts. This can not be represented by a single value. Also, the value is not publicly available, which is in conflict with the aim of this work to make the concept easily usable. An alternative is offered by the CWE methodology, which was introduced in Section 3.1.1. Each vulnerability with a CVE identifier is mapped to the corresponding CWEs in the NVD DB, which is used as the vulnerability information source in this work anyway. A number of impacts are associated with every CWE to indicate the typical effects if the weakness is exploited. Using this information, a set of impacts for advisories can be derived in the following way. For an advisory, each vulnerability is mapped to its weaknesses as defined by CWE. Then, all impacts of each CWE are extracted to form a list of impacts for the advisory. Finally, the list is reduced to set of unique impact values, which disregards the differentiation between the CVEs and emphasizes the advisory as a whole construct.

One issue that arose during the mapping process is that there is a substantial amount of security advisories that could only be associated with the CWE 'values' *NVD-CWE-noinfo* or *NVD-CWE-Other*. This is a result of the CVEs referenced by the advisories and the NVD vulnerability DB used for this work. NVD defines those values as follows:[30]

NVD-CWE-noinfo There is insufficient information about the issue to classify it; details are unknown or unspecified.

NVD-CWE-Other NVD is only using a subset of CWE for mapping instead of the entire CWE, and the weakness type is not covered by that subset.

If all references in an advisory point to CVEs, which have no other information than those two values, no CWE values can be derived for them. This circumstance applies to 1062 of 20243 (5.23%) advisories in our *advisory DB*. However, all advisories have descriptions and those include a textual description of the impact. Taking the following advisory description as an example (translated from German):

> *A remote adversary can exploit a vulnerability in MySQL to execute a Denial-of-Service (DoS) attack. The vendor states that the vulnerability also applies to MariaDB and publishes security updates.*

[30] https://nvd.nist.gov/vuln/categories, accessed on 20 July 2023.

Thus the description clearly indicates that the impact of a successful exploitation is a DoS. We propose to make use of this fact and derive missing CWE information from the advisory descriptions. This problem can be categorized in the field of NLP and more specifically as a multi label text classification task.

As a final adjustment, the CWE scopes will be mapped to the STRIDE threat model. STRIDE is a well known model that is commonly seen in risk assessment tasks, e.g. in the work of [106]. Thus, we consider it a suitable approach to model the impacts in a clear and structured way.

4.2 Further Background Information

For the proposed extension concept additional methods are necessary to achieve the described goal, i.e. text classification and the STRIDE model. In this section, the new methods will be introduced before the concrete implementation will be presented in following section.

4.2.1 ML: Text Classification

In recent years, especially deep-learning approaches have shown outstanding success in classification tasks [61]. Therefore, the focus has been put on those models. Deep learning describes a kind of ML models that learn the representation of data by breaking it down to smaller, simpler representations. A representation consists of all pieces of information, called features, given to the model. Important features of the data are identified by the deep learning model itself. For example, when classifying an image, the model might first extract edges, then corners and colours and finally object parts to output the object type [43].[31] This solves the problem of simpler machine learning algorithms which heavily depended on the representation of input data.

The high attention towards ML has resulted in a tremendous number of models and architectures published in academic literature. An extensive review and analysis was

[31] In the scope of this work, ML is regarded as a tool to achieve a goal rather than the focus of research. Therefore, the introduction is reduced to provide an overview and general understanding without going in too much detail while still arguing the decision process for a model selection. We kindly refer the interested reader to the free online available book of Goodfellow et al. [43] for an in-depth introduction to deep learning.

performed in the work by Minaee et al. [71] and showed that especially BERT [31] and its successors result in good performances, i.a. for text classification. For that reason, we will briefly outline the innovative concept of BERT and the rationale for the final choice of model.

The BERT model is based on the transformer architecture, which will therefore be outlined first. Published in 2017 by Google [100], the new architecture improved the required computational resources significantly by reducing sequential processing and allowing more parallelization while still achieving better results than other state-of-the-art models at that time. This is mainly achieved with two novel concepts: positional encoding and self-attention. Positional encoding assigns a number to each word of a sentence so that the model can learn the positional importance even when not being trained sequentially. Self-attention introduces the ability to the model to understand words in context of other words surrounding it. Now, the architecture of BERT adapts the implementation of transformers [31]. Its training is split in two steps: pre-training and fine-tuning. The pre-training phase initializes the model with a general understanding of natural language. Afterwards, the fine-tuning can be done for a specific task in an efficient manner. Even though for operational purposes only the fine-tuning has to be done, it turned out that the processing time would be unacceptably long with the available hardware resources. Therefore, for this work the DistilBERT model will be used. It is a concept to reduce the size of the final model and improve fine-tuning time, while still achieving a good performance [90].

4.2.2 STRIDE

STRIDE is an acronym that stands for **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service and **E**elevation of privilege. This acronym was created by Microsoft to be a mnemonic for developers to memorize possible types of threats that are commonly seen and simplify conversations by a defined wording. It has been used for a long time and is part of Microsoft's threat modelling tool which is a core element of their Security Development Lifecycle (SDL) [70]. An extensive study also showed that it is lightweight and easy to understand [92]. Even though the model lacks a standard methodology [56], propositions on how to apply the model can be found in literature, e.g. in the work of Khan et al. [56] or Shostack [96]. Table 4.1 lists the attributes of STRIDE along with a description to give an overview of the different aspects. There are two variants of STRIDE that extend the basic idea [96]. First, STRIDE-per-

Table 4.1: Overview of the threats included in the STRIDE model. Definitions adapted from [46, 70, 96].

Threat	**Definition**
Spoofing	Pretending to be someone else, e.g. by using another user's authentication information.
Tampering	Altering data on disk, on a network or in memory.
Repudiation	Denying having performed an action. Usually associated with a lack of proof to be confident on who did it.
Information Disclosure	Information is exposed to a party that is not authorized to have access to it.
Denial of Service	Denying the access to a service to a valid using by occupying needed resources.
Elevation of Privilege	Some user, malicious or rightful, attains more privileges than intended.

element emphasizes the fact that certain threats rarely apply to some elements. For example, data flows are unlikely to spoof other data flows. This variant helps to make finding threats easier by reducing the set of applicable STRIDE elements. The second variant is STRIDE-per-interaction, which aims to make threats easier to understand. Every interaction between two elements is assessed separately to identify possible threats. That way, the focus is narrowed down and the process easier to follow. Those variants are helpful when STRIDE is used for threat modelling and were shortly introduced to provide a complete picture of the framework. However, in this work STRIDE is used as a descriptive framework, which is why the differentiation of the different threat types is sufficient.

4.3 Implementation of the Extension

We will now introduce the implementation of the extension. The preliminary methods of the ML model and STRIDE mapping will be presented, followed by the adaption of the advisory assessment concept.

4.3.1 Preliminary Methods

First, the used fine-tuning process for the DistilBERT ML will be described. The model was trained with the *vulnerability DB* using the open-source library of Huggingface [108]. The dataset of all advisories was split into train (90%), test (5%) and validation (5%). In the DB there is a significant imbalance in the distribution of classes due to different scope combinations. Thus, stratified sampling was applied which preserves the relative class frequencies during the splitting. Title and description of the advisory were joined and used as input for the model. Padding and truncation to 512 words were applied so the input matches the model's requirements. For the pre-processing and creation of tokens the available class of Huggingface was used. The model was trained for 10 epochs, with a weight decay of 0.1 and an initial learning rate of $2 * 10^{-5}$. The model will be evaluated with precision, recall and F1 scores which are calculated as follows:

$$precision = \frac{tp}{tp + fp}$$
$$recall = \frac{tp}{tp + fn}$$
$$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

with $tp :=$ true positive, i.a. correct result, $fp :=$ false positive, i.a. unexpected result and $fn :=$ false negative, i.a. missing result.

Applied to the validation dataset, the model achieves a precision score of 81.49% and a recall of 74.83%, resulting in a F1-score of 78.02%. Since the classification task is a multi label problem, an average for the scores has to be chosen. We used a micro average, that calculates the score globally instead of choosing a macro average to distinguish each label. The outcome is considered sufficient to illustrate the concept and the comparison of other models or fine-tuning the current one is left for future work.

With each advisory having a CWE scope or receiving one from the ML model, the derivation of STRIDE threats can be performed. Possible mappings from CWE's technical impact and scope to the STRIDE framework have previously been elaborated. Therefore, the mapping concept is based and adapted from the prior works of Honkaranta et al. [46] and Schaad and Binder [93]. It is shown in Table 4.2. Note that a STRIDE threat may require one scope or several in combination. Also, the elevation of privilege threat is indicated by two possible scope combinations.

Table 4.2: CWE-STRIDE mapping. Adapted from [46, 93]

STRIDE Threat	CWE Scope
Spoofing	Access Control ∧ Authentication
Tampering	Integrity
Repudiation	Non-Repudiation ∨ Accountability
Information disclosure	Confidentiality
Denial of service	Availability
Elevation of privilege	(Access Control ∧ Authentication) ∨ (Confidentiality ∧ Integrity ∧ Availability ∧ Access control)

Now that each advisory is mapped to all applicable STRIDE threats, a filtering of possible paths can be applied. Lateral movement has three main stages: (i) reconnaissance, (ii) credential/privilege gathering and (iii) gaining access to other hosts [28]. For the prioritization of advisories especially stage two is of interest, which can be achieved by the exploitation of an advisory. However, not all types of impacts allow an adversary to make further use of it in terms of lateral movement. For example, if an exploited vulnerability leads solely to a DoS, it is in most cases impossible to abuse that host as an enabler of further network penetration.[32] Using the knowledge offered by the threat types of STRIDE, which each advisory has just been assigned, possible paths in the network advisory graph can be filtered. We propose to use the threat types *spoofing*, *elevation of privilege* and *tampering* as possible enablers for lateral movement. They are straightforward to associate with the goal: *spoofing* and *elevation of privileges* allow an adversary to assume the identity of a legitimate user and *tampering* is often associated with the execution of arbitrary code. The other threats might also be used within an attack chain to achieve that goal, but can not allow to achieve it directly. For example, if a user had a text file containing all passwords on their computer which would be made accessible by an advisory that impacts *information disclosure*, access to new hosts might be possible with the found credentials, if they are associated with the necessary privileges. Due to the many uncertainties usually associated with such a reasoning, we remain with the three aforementioned threats.

[32] If the DoS is caused by a system crash due to a buffer overflow, often the possibility of further exploits, e.g. privilege escalation, exists. At this point, this link is left aside.

4.3.2 Assessment Adjustment

To implement the concept in the assessment, Algorithm 3.1 has to be adjusted. It is as simple as adding an if-clause after line 7, which states that only advisories that have one of the enabling threats will be inserted in the connection list, resulting in:

Algorithm 4.1: Extended advisory network attack graph generation.

```
1  input: HC, AH, AR
2  output: G
3  A ← new list
4  C ← new list
5  r ← new hash table
6  move_threats ← [spoofing, elevation of privileges, tampering]
7  for advisory, rating in AR
8    insert advisory into A
9    if any move_threats in advisory
10     for (a, h) in AH where a = advisory
11       for (fromH, toH) in HC where h = fromH
12         for (a2, h2) in AH where h2 = toH
13           or h2 = h  // include other advisories from current host
14           and a2 != advisory
15             connection ← (advisory, a2)
16             insert connection into C
17             r[connection] ← rating
18         endfor
19       endfor
20     endfor
21   endif
22 endfor
23 return G(A,C,r)
```

4.4 Evaluation of the Improved Concept

For the evaluation, the same examples introduced in Section 3.3 will be used. Only the effect of the extension will be discussed here, to highlight the impact of the changes. The full result values and the derived prioritization order are given in Appendix B.

In *network 1*, host *H1* is the gateway for an adversary to the organization's network. Hence, advisories applying to this host were prioritized by the assessment concept. Among the three advisories, two were chosen based on the assumed adversary profile in the initial implementation: *Advisory_9081* and *Advisory_16824*. With the extension

implemented, the prioritization changed for profile one. Initially advisory *Advisory_-16824* was selected as the shortest path to the inner network, due to the fact that it had the highest rating on *H1* when assessing using profile one. However, with the extension in place, it does not match the STRIDE threat criteria to be regarded as a possible path, resulting in the second-highest rated advisory of *H1* being chosen. Because the only other advisory that receives an adjustment is one of host *H4* and the formerly selected one does enable lateral movement, the second place stays the same. For profile four, the extension did not have any effect and the order remained unchanged. This is a result of all advisories that were selected as path enablers fulfilled the STRIDE criteria. Hence, nothing needed to be adjusted.

The same can be observed in *network 2*. Because all advisories that were used for the shortest paths did enable lateral movement according to the STRIDE criteria, the prioritization order stayed exactly the same for both profiles. This coincidence is not as surprising when considering that only two advisories were adjusted before, due to the rather restricted possible connections in this network.

Probably more revealing results will be achieved when the concept is applied to larger networks. However, the evaluation still showed the possibilities of the extension in *network 1*, and hence served its purpose. Also, even though the extension did not result in any changes of the prioritization for *network 2*, it does not diminish the effect. Rather, it should be considered a confirmation of the initial ranking based on additional information.

5 Summary and Conclusion

5.1 Summary

In this section, the achieved results of the developed information security advisory risk assessment concept are summarized and discussed in respect to the research questions. Afterwards, limitations and possible future research directions are given and a conclusion of this work is drawn.

To understand the concept of security advisories and highlight the current state of advisory assessment, important background information was given and an extensive literature review done. Building on the created knowledge, an advisory assessment concept was proposed, which consists of three consecutive steps: (i) assess the vulnerabilities, (ii) assess the advisories as a whole and (iii) adjust the assessment based on the network configuration of an organization. For each step, the selection of suitable methods was discussed and possible alternatives highlighted. Using two examples, one imaginary and one provided by a national CERT, the applicability of the concept was shown. Furthermore, to show the expandability, an extension was implemented and tested, using available information to improve the results.

Research question one asks the question of how the need of organizations for a security advisory prioritization concept can be fulfilled. By laying the focus on using only data sources that are open and free of cost, even organizations without extensive monetary resources shall be capable to apply it. With the development and implementation of the concept presented in this work, it was proven that the prioritization is definitely possible. The information sources and standards used are open-source, well-known and established concepts in the cybersecurity community. Even though some are maintained by governmental organizations, e.g. NVD, it is not conceivable that those will be closed from the public in the future. Furthermore, an emphasis has been put on highlighting how the different components interact with each other and how they influence the final

calculation. This will allow future implementations to easily replace one concept with another, that serves the same purpose and offers the same outputs. The user is encouraged to do so, if there are methods that yield more precise results in the context of their organization.

Comparing the assessment results to other practised advisory rating concepts, it could be shown that the proposed concept inhibits the significant advantage of providing a definite order of priority. Especially ratings with arbitrary ordinal values, e.g. used by RedHat, have the problem that there will be numerous advisories with the same rating, when several ones are regarded. Of course, they still offer a valuable first indicator for the risk of the advisory, but are not useful for the prioritization process. Also, some related concepts restrict the assessment of an advisory to a single representation of knowledge, e.g. the *DAF-Score* uses a single CVSS vector. This becomes a problem, as the diversity of characteristics regarding multiple vulnerabilities can not be represented with that approach, effectively causing a loss of information.

Even when an advisory has a sophisticated rating by a central publisher, e.g. by a CERT or commercial aggregator, the risk can be completely different in a local environment. The question of how a rating can be appropriately adjusted, is tackled by research question two. With the third step of the assessment concept, a possible solution to that problem has been proposed. Advisories are mapped to the hosts they apply on, which are then regarded in the network configuration of the organization to derive possible attack graphs for an adversary and adjust the ratings of advisories accordingly. The evaluation showed promising results, with advisories being prioritized that could serve as a gateway to the adversary. Furthermore, with the implementation of an extension, the process of upgrading the concept has been demonstrated. The extension reduces possible lateral movement paths using STRIDE impacts derived from CWE scopes of an advisory. To overcome the problem of missing CWE scopes in the open-source data used, a ML model was trained which predicts them based on the advisory's description. For this extension, adjusting the rating algorithm proved to be easy while still notably improving the final results.

Two preliminaries that should not be underestimated are that for the third step, an organization must possess a functional asset management system as well as complete and accessible knowledge of the network. Yet, as could be seen, when the rating is regarded without an adjustment based on the network configuration, it still proved to be useful. The rating experiences the impact of a considerably higher number of vulnerabilities in

an advisory in comparison to others. However, as we have shown, it also regards other aspects that reduce this influence and break the blunt prioritization by number of vulnerabilities as seen in the TTC. These new considered aspects are especially: referencing a known exploited vulnerability, the features of the vulnerability itself and the assumed adversary profile regarding the resources and skill level. As the number of vulnerabilities should have an influence on the rating, especially if it as significant amount, it is a promising observation that the combination with other aspects lead to, in our opinion, a better prioritization order. However, all advisories with a notable amount of referenced vulnerabilities had a high average CVSS base score. Thus, assumptions on advisories with a high vulnerability count, but low average base score, can not yet be made and must be evaluated in future work.

5.2 Limitations and Future Perspective

The proposed advisory assessment concept does not come without limitations, which will be highlighted with possible solutions and general considerations that might be valuable for future work.

General considerations. Numerous methods have been presented and incorporated into the concept described in this work. Ideas to further enhance the assessment can concern all different methods to increase the efficiency of individual components. In general, using more of the available information in the process can be used to improve the calculation results. The selection of used data was made to represent strong indicators, e.g. the existence of an exploit. Further information can however refine assumptions and add clearer insights. For example, is the exploit available in ready to use tools or is a fine-tuning necessary? The evaluation and introduction of more data should be considered in future work to increase the precision of assessment results. Another aspect that could be included in the assessment of advisories is the effort required to implement the solution. A textual description of the necessary steps is provided in each advisory and the idea and possible implementations may be used from related work in the context of the patch prioritization. Furthermore, with the upcoming release of CVSS version 4, the impact of the new metrics have to be evaluated and the calculation has to be adjusted accordingly, e.g. in the *exploit complexity* of the vulnerability assessment. At this point, only two of the various aspects for adversary characterization are used. Assuming other aspects might help to achieve a more fine grained adjustment.

With the currently used shortest path filtering on the advisory network graph, only a single advisory of a host will get the passed on ratings from dependent advisories. However, if there are multiple advisories that present a logical choice to move on to the next host, it might be more precise if both received the ratings. Of course, the advisory currently getting the ratings was chosen due to the fact that it is regarded more important, and when the assessment is run again after it was fixed, the second advisory will be chosen as a replacement and consequently receive the ratings. In future work, it should be evaluated if it is clear for the assessor that the assessment should be run again after addressing an advisory or receiving a new one, because this may change the whole network path selection, or if the order must reflect a suitable selection for the top cases. Either approach requires different computational resources, one to run this assessment concept more often and one for a more comprehensive graph analysis. If applied to larger sized networks, it might be necessary to identify the faster one for performance reasons.

Each advisory has already been assigned to a set of STRIDE threats which indicate the impact of an exploitation. At this point, they are used to filter the network graph in order to better represent possible lateral movement paths. They could be further used to lay a focus on specific threats in the rating process. For example, if for an organization especially information disclosure entails a high risk, a weight factor could be applied to each advisory that impacts this threat type.

An aggregation concept for multiple assessments with variable parameters can help to introduce different perspectives into the assessment. Weighting the different parameter setups would further allow to put an emphasis on the most likely ones. We see this necessity due to the following considerations. At the moment, only a single vantage point for the adversary is used to assess the advisories. However, it is not fallacious that an assessor wants to assume different starting points. For example, employee clients usually have privileges throughout the network which would offer a significant aid to the adversary by allowing previously impossible connections. Since the human factor is a known risk factor, e.g. in terms of phishing, weak passwords or loss of devices, assuming the location of an client as a starting point can help to establish a new perspective. Another possible parameter would be the use of different adversary profiles. As became apparent in the evaluation, the profile can change the outcome of the final result. Therefore, one idea is that each profile would be assumed and accordingly weighted with a different likelihood. This could reflect that a nation state actor is not expected to be interested in one's organization, but an absolute exclusion can not be made.

Limitations. As mentioned before, concerning the ratings without network information, the influence of advisories with a high number of referenced vulnerabilities that have a low average base score must yet be evaluated to better understand the full domain of results of the assessment concept.

This concept was created to add a prioritization order to security advisories under the assumption, that all advisories must be regarded. For that purpose, advisories are assessed in relation to other advisories. This means that if there are only very few or even just one advisory, the concept will not be a good approach to assess the general urgency. In that case other concepts, as for example the DAF-score, can offer a better first indicator. A possible solution to change that behaviour might be to record the scores before the normalization. The validity of those are at this point neither tested nor validated, which should be done in future work.

In some cases, advisories are published to indicate that an asset is not affected by a vulnerability. Those advisories must receive a special treatment, as they are not urgent. Only when an assessor of an organization requires that kind of information, this advisory becomes relevant. Currently, they are not handled specifically, which should be introduced in the future. The question here lies in the identification of such advisories, because they are structured similarly to 'normal' ones. Possible approaches might be to look for advisories that mention not affected products, which might however also be the case when they are incomplete for some reason. Since the textual description usually indicates such advisories, another approach is to implement a NLP concept for the categorization.

For the third part of the assessment concept, the advisories must be matched to the hosts they apply to. Essential preliminaries for this step are a functional asset management system and a matching concept that is also used by the advisories (usually CPE). We account for the fact that those are not given in every organization and the application consequently is limited. However, as shown in the evaluation, the initial assessment values before the adjustment in the network context also provide an improvement in comparison to prior practical concepts.

The ML model was chosen based on an academic literature review and kept after showing adequate results. However, other models or fine-tuning the current approach might lead to even better results. In future work, further evaluation of models is necessary to either improve the current one or confirm the approach. Furthermore, the model was trained solely on German advisories. Even though a multilingual base model was chosen,

the performance was not tested for other languages. First tests showed that for example English input text works as well, but more extensive evaluation in that regard is necessary. Especially English is important, since it is the prominent language for the international cybersecurity community.

5.3 Conclusion

In this thesis, the aim was to develop an assessment concept for security advisories, that can be used by organizations which are struggling with processing the growing amount of information. The concept should provide a prioritization for advisories based on their features. For that purpose, an extensive review of usable methods was performed and a suitable selection made, to create a concept that rates advisories. By first assessing the advisories individually and then in the context of the environment of the organization, a contextualized rating can be determined by the proposed prioritization method. Using solely open information allows for usage regardless of financial resources. The first implementation and evaluation with regard to two example networks showed promising application results of the concept. Various ideas for possible improvements were given, which mainly emphasize the fact that including more information will yield better results. However, the developed concept shows a general construct and different components that are useful for the assessment of security advisories. This work can be used as the foundation to break away from unsuitable advisory ratings and employ a dedicated, automated assessment concept, which is urgently needed to get hold of the rising number of advisories. Future research can build upon the foundational idea and improve individual components.

Bibliography

[1] BSI-Standard 200-3. 2017. Risikoanalyse auf der Basis von IT-Grundschutz. Standard. Version 1. Bundesamt für Sicherheit in der Informationstechnik, Bonn, DE, (Nov. 15, 2017).

[2] ISO 27001:2018. 2018. Information technology — Security techniques — Information security management systems — Overview and vocabulary. Standard. International Organization for Standardization, Geneva, CH, (Feb. 2018).

[3] ISO 31000:2018. 2018. Risk management — Guidelines. Standard. International Organization for Standardization, Geneva, CH, (Feb. 2018).

[4] ISO 55000:2014. 2014. Asset management — Overview, principles and terminology. Standard. International Organization for Standardization, Geneva, CH, (July 2014).

[5] Lillian Ablon and Andy Bogart. 2017. *Zero Days, Thousands of Nights: The Life and Times of Zero-Day Vulnerabilities and Their Exploits*. RAND Corporation, Santa Monica, CA. ISBN: 978-0-8330-9761-3. DOI: 10.7249/RR1751.

[6] 2023. About cwe. The MITRE Corporation. (June 6, 2023). Retrieved June 16, 2023 from https://cwe.mitre.org/about/index.html.

[7] Vida Ahmadi Mehri, Patrik Arlos, and Emiliano Casalicchio. 2022. Automated context-aware vulnerability risk management for patch prioritization. *Electronics*, 11, 21. DOI: 10.3390/electronics11213580.

[8] M. Ugur Aksu, M. Hadi Dilek, E. Islam Tatlı, Kemal Bicakci, H. Ibrahim Dirik, M. Umut Demirezen, and Tayfun Aykır. 2017. A quantitative cvss-based cyber security risk assessment methodology for it systems. In *2017 International Carnahan Conference on Security Technology (ICCST)* (Madrid, Spain). IEEE, New York, NY, USA, 1–8. ISBN: 978-1-5386-1585-0. DOI: 10.1109/CCST.2017.8167819.

[9] Kenneth Alperin, Allan Wollaber, Dennis Ross, Pierre Trepagnier, and Leslie Leonard. 2019. Risk prioritization by leveraging latent vulnerability features in a contested environment. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security* (AISec'19) (London, United Kingdom). Association for Computing Machinery, New York, NY, USA, 49–57. ISBN: 9781450368339. DOI: 10.1145/3338501.3357365.

[10] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. 2002. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security* (CCS '02) (Washington, DC, USA). Association for Computing Machinery, New York, NY, USA, 217–224. ISBN: 1581136129. DOI: 10.1145/586110.586140.

[11] Afsah Anwar, Ahmed Abusnaina, Songqing Chen, Frank Li, and David Mohaisen. 2022. Cleaning the nvd: comprehensive quality assessment, improvements, and analyses. *IEEE Transactions on Dependable and Secure Computing*, 19, 6, 4255–4269. DOI: 10.1109/TDSC.2021.3125270.

[12] 2019. ATT&CK. Lateral movement. The MITRE Corporation. (July 2019). Retrieved Aug. 19, 2023 from https://attack.mitre.org/tactics/TA0008.

[13] Sinchul Back, Jennifer LaPrade, Lana Shehadeh, and Minju Kim. 2019. Youth hackers and adult hackers in south korea: an application of cybercriminal profiling. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (Stockholm, Sweden). IEEE, New York, NY, USA, (June 2019), 410–413. ISBN: 978-1-7281-3027-9. DOI: 10.1109/EuroSPW.2019.00052.

[14] Maria Bada and Jason R.C. Nurse. 2021. Profiling the cybercriminal: a systematic review of research. In *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)* (Dublin, Ireland). IEEE, New York, NY, USA, 1–8. ISBN: 978-1-6654-3092-0. DOI: 10.1109/CyberSA52016.2021.9478246.

[15] Philippe Bourgeois, Oscar Conesa, Bernd Grobauer, and Gareth Price. 2004. EISPP Common Advisory Format Description. Standard EISPP-D3-001-TR. Version 2. EISPP Consortion, (May 20, 2004).

[16] Mehran Bozorgi, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. 2010. Beyond heuristics: learning to classify vulnerabilities and predict exploits. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Dis-*

covery and Data Mining (KDD '10) (Washington, DC, USA). Association for Computing Machinery, New York, NY, USA, 105–114. ISBN: 9781450300551. DOI: 10.1145/1835804.1835821.

[17] BSI. 2023. Cyber-sicherheitswarnungen. Retrieved May 30, 2023 from https://www.bsi.bund.de/dok/14803002.

[18] Timothy Casey. 2007. Threat Agent Library Helps Identify Information Security Risks. White Paper. Intel, (July 2007). DOI: 10.13140/RG.2.2.30094.46406.

[19] Brant A. Cheikes, David Waltermire, and Karen Scarfone. 2011. Common Platform Enumeration: Naming Specification. Interagency Report 7695. Version 2.3. (Aug. 2011). 49 pp. DOI: 10.6028/NIST.IR.7695.

[20] Haipeng Chen, Jing Liu, Rui Liu, Noseong Park, and V.S. Subrahmanian. 2019. Vase: a twitter-based vulnerability analysis and score engine. In *2019 IEEE International Conference on Data Mining (ICDM)* (Beijing, China). IEEE, New York, NY, USA, 976–981. DOI: 10.1109/ICDM.2019.00110.

[21] CISA. [n. d.] Cybersecurity alerts & advisories. Retrieved May 30, 2023 from https://www.cisa.gov/news-events/cybersecurity-advisories.

[22] CISA. 2023. Known Exploited Vulnerabilities Catalog. Cybersecurity & Infrastructure Security Agency. Retrieved Jan. 27, 2022 from https://www.cisa.gov/known-exploited-vulnerabilities-catalog.

[23] CISA. 2022. Traffic light protocol (tlp) definitions and usage. (Aug. 22, 2022). Retrieved May 31, 2023 from https://www.cisa.gov/news-events/news/traffic-light-protocol-tlp-definitions-and-usage.

[24] 2022. CISA Stakeholder-Specific Vulnerability Categorization Guide. Report. CISA, (Aug. 2022). Retrieved June 8, 2023 from https://www.cisa.gov/stakeholder-specific-vulnerability-categorization-ssvc.

[25] Alexander Cobleigh, Martin Hell, Linus Karlsson, Oscar Reimer, Jonathan Sönnerup, and Daniel Wisenhoff. 2018. Identifying, prioritizing and evaluating vulnerabilities in third party code. In *2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW)* (Stockholm, SWE). IEEE, New York, NY, USA, (Oct. 2018). ISBN: 978-1-5386-4142-2. DOI: 10.1109/EDOCW.2018.00038.

[26] FIRST. 2019. *Common Vulnerability Scoring System v3.1*. Rev. 1. FIRST. (June 2019). https://www.first.org/cvss/v3.1/specification-document.

[27] 2023. Computer security resource center. Glossary. National Institute of Standards and Technology. (Mar. 2023). Retrieved July 6, 2023 from https://csrc.nist.gov/glossary.

[28] CrowdStrike. 2023. Lateral movement. (Apr. 2023). Retrieved July 20, 2023 from https://www.crowdstrike.com/cybersecurity-101/lateral-movement.

[29] Trey Darley, Rich Piazza, and Bret Jordan. 2021. STIX. Standard. Version 2.1. OASIS Open, (June 11, 2021). 313 pp. Retrieved June 12, 2023 from https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html.

[30] Stephanie de Smale, Rik van Dijk, Xander Bouwman, Jeroen van der Ham, and Michel van Eeten. 2023. No one drinks from the firehose: how organizations filter and prioritize vulnerability information. In *2023 IEEE Symposium on Security and Privacy (SP)* (San Francisco, CA). IEEE, New York, NY, USA, (May 2023), 1980–1996. DOI: 10.1109/SP46215.2023.00012.

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: pre-training of deep bidirectional transformers for language understanding. (2019). arXiv: 1810.04805 [cs.CL]. DOI: 10.48550/arXiv.1810.04805.

[32] E. W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 1, (Dec. 1959), 269–271. DOI: 10.1007/BF01386390.

[33] Ying Dong, Wenbo Guo, Yueqi Chen, Xinyu Xing, Yuqing Zhang, and Gang Wang. 2019. Towards the detection of inconsistencies in public security vulnerability reports. In *28th USENIX Security Symposium (USENIX Security 19)* (Santa Clara, CA). USENIX Association, Berkeley, CA, USA, (Aug. 2019), 869–885. ISBN: 978-1-939133-06-9.

[34] Ralph Eckmaier, Walter Fumy, Stéfane Mouille, Jean-Pierre Quemard, Nineta Polemi, Rainer Rumpel, and Sławomir Górniak. 2022. RISK MANAGEMENT STANDARDS. Analysis of standardisation requirements in support of cybersecurity policy. Report. ENISA, (Mar. 2022). DOI: 10.2824/001991.

[35] Andreas Ekelhart, Weippl Edgar, and Stefan Fenz. 2008. Semantic potential of existing security advisory standards. In *Inproceedings of the FIRST 2008 Conference-Forum of Incident Response and Security Teams* (Vancouver, CA).

[36] Clément Elbaz, Louis Rilling, and Christine Morin. 2020. Fighting n-day vulnerabilities with automated cvss vector prediction at disclosure. In *Proceedings of the 15th International Conference on Availability, Reliability and Security* (ARES '20) Article 26 (Virtual Event, Ireland). Association for Computing Machinery, New York, NY, USA. ISBN: 9781450388337. DOI: 10.1145/3407023.3407038.

[37] Katheryn A. Farris, Ankit Shah, George Cybenko, Rajesh Ganesan, and Sushil Jajodia. 2018. Vulcon: a system for vulnerability prioritization, mitigation, and management. *ACM Trans. Priv. Secur.*, 21, 4, Article 16, (June 2018). DOI: 10.1145/3196884.

[38] John P. Field, Stephen A. Banghart, and David Waltermire. 2018. Resource-Oriented Lightweight Information Exchange (ROLIE). RFC 8322. RFC Editor, (Feb. 2018). 43 pp. DOI: 10.17487/RFC8322.

[39] Mozilla Foundation. 2023. Mozilla foundation security advisories. Retrieved May 30, 2023 from https://www.mozilla.org/en-US/security/advisories/.

[40] Bundesamt für Sicherheit in der Informationstechnik (BSI). 2018. Lebenszyklus einer Schwachstelle. Tech. rep. Version 2.0. Bonn, DEU, (July 2018). Retrieved May 28, 2023 from https://www.allianz-fuer-cybersicherheit.de/SharedDocs/Downloads/Webs/ACS/DE/BSI-CS/BSI-CS_027.pdf?__blob=publicationFile&v=1.

[41] 2023. GLOBAL THREAT REPORT. Report. CrowdStrike. 42 pp. Retrieved June 13, 2023 from https://www.crowdstrike.com/global-threat-report/.

[42] Oliver Goebel. 2005. Common Announcement Interchange Format. Standard. Version 1.2. CAIF Project, (Nov. 7, 2005). http://www.caif.info/draft-goebel-caif-format.html.

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. Retrieved July 20, 2023 from http://www.deeplearningbook.org.

[44] Geoffrey R. Grimmett and David R. Stirzaker. 2020. *Probability and Random Processes*. (4th ed.). Oxford University Press, New York, NY. ISBN: 978-0-19-884760-1.

[45] Stefan Hagen. 2017. CSAF Common Vulnerability Reporting Framework (CVRF). Standard. Version 1.2. OASIS Committee Specification 01, (Sept. 13, 2017). 101 pp. http://docs.oasis-open.org/csaf/csaf-cvrf/v1.2/csaf-cvrf-v1.2.html.

[46] Anne Honkaranta, Tiina Leppänen, and Andrei Costin. 2021. Towards practical cybersecurity mapping of stride and cwe — a multi-perspective approach. In *2021 29th Conference of Open Innovations Association (FRUCT)* (Tampere, Finland). IEEE, New York, NY, USA, 150–159. ISBN: 978-1-6654-1415-9. DOI: 10.23919/FRUCT52173.2021.9435453.

[47] Allen D. Householder, Garret Wassermann, Art Manion, and Chris King. 2017. The CERT Guide to Coordinated Vulnerability Disclosure. Special Report CMU-/SEI-2017-SR-022. Carnegie Mellon University, (Aug. 2017).

[48] Sotiris Ioannidis, Angelos D. Keromytis, Steve M. Bellovin, and Jonathan M. Smith. 2000. Implementing a distributed firewall. In *Proceedings of the 7th ACM Conference on Computer and Communications Security* (CCS '00) (Athens, Greece). Association for Computing Machinery, New York, NY, USA, 190–199. DOI: 10.1145/352600.353052.

[49] Sarah Isniah, Humiras Hardi Purba, and Fransisca Debora. 2020. Plan do check action (PDCA) method: literature review and research issues. *Jurnal Sistem dan Manajemen Industri*, 4, 1, (July 2020), 72–81. DOI: 10.30656/jsmi.v4i1.2186.

[50] Jay Jacobs, Sasha Romanosky, Benjamin Edwards, Idris Adjerid, and Michael Roytman. 2021. Exploit prediction scoring system (epss). *Digital Threats*, 2, 3, Article 20, (July 2021). DOI: 10.1145/3436242.

[51] Jay Jacobs, Sasha Romanosky, Octavian Suciu, Benjamin Edwards, and Armin Sarabi. 2023. Enhancing vulnerability prioritization: data-driven exploit predictions with community-driven insights. (Mar. 2023). arXiv: 2302.14172 [cs.CR].

[52] Bret Jordan and Drew Varner. 2021. TAXII. Standard. Version 2.1. OASIS Open, (June 10, 2021). 79 pp. Retrieved June 12, 2023 from https://docs.oasis-open.org/cti/taxii/v2.1/os/taxii-v2.1-os.html.

[53] Bill Jung, Yan Li, and Tamir Bechor. 2022. Cavp: a context-aware vulnerability prioritization model. *Computers & Security*, 116, Article 102639, (May 2022). DOI: 10.1016/j.cose.2022.102639.

[54] Daniel Kahneman. 2013. *Thinking, fast and slow.* (1st ed.). Farrar, Straus and Giroux, New York. 499 pp. ISBN: 978-0-374-53355-7.

[55] Linus Karlsson, Pegah Nikbakht Bideh, and Martin Hell. 2020. A recommender system for user-specific vulnerability scoring. In *Risks and Security of Internet and Systems* (Lecture Notes in Computer Science) (Hammamet, Tunisia). Slim Kallel, Frédéric Cuppens, Nora Cuppens-Boulahia, and Ahmed Hadj Kacem, (Eds.) Vol. 12026. Springer International Publishing, Cham, 355–364. ISBN: 978-3-030-41568-6. DOI: 10.1007/978-3-030-41568-6_23.

[56] Rafiullah Khan, Kieran McLaughlin, David Laverty, and Sakir Sezer. 2017. Stride-based threat modeling for cyber-physical systems. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)* (Turin, Italy). IEEE, New York, NY, USA. DOI: 10.1109/ISGTEurope.2017.8260283.

[57] Christiane Kirketerp de Viron, Raluca Stefanuc, Benjamin Boegel, and Maika Fohrenbach. 2022. Cyber resilience act. Presentation. European Commission. (Nov. 2022). Retrieved July 19, 2023 from https://futurium.ec.europa.eu/system/files/2022-11/CRA%20-%20CNECT%20University%20presentation.pdf.

[58] Sebastian Klipper. 2015. *Information Security Risk Management. Risikomanagement mit ISO/IEC 27001, 27005 und 31010.* (2nd ed.). Springer Vieweg, Wiesbaden, (Mar. 3, 2015). 207 pp. ISBN: 978-3-658-08773-9. DOI: 10.1007/978-3-658-08774-6.

[59] Hans-Peter Königs. 2017. *IT-Risikomanagement mit System. Praxisorientiertes Management von Informationssicherheits-, IT- und Cyber-Risiken.* (5th ed.). Springer Vieweg, Wiesbaden. ISBN: 978-3-658-12003-0. DOI: 10.1007/978-3-658-12004-7.

[60] Petros Konstantakos, Panos Chountalas, and Anastasios Magoutas. 2019. The contemporary landscape of asset management systems. *Quality-Access to Success*, 20, 169.

[61] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: a survey. *Information*, 10, 4. DOI: 10.3390/info10040150.

[62] Harjinder Singh Lallie, Kurt Debattista, and Jay Bal. 2020. A review of attack graph and attack tree visual syntax in cyber security. *Computer Science Review*, 35, 46 pages. DOI: 10.1016/j.cosrev.2019.100219.

[63] E. Rutger Leukfeldt, Edward R. Kleemans, and Wouter P. Stol. 2017. Origin, growth and criminal capabilities of cybercriminal networks. an international empirical analysis. *Crime, Law and Social Change*, 67, 1, (Feb. 2017), 39–53. DOI: 10.1007/s10611-016-9663-1.

[64] David John Leversage and Eric James Byres. 2008. Estimating a system's mean time-to-compromise. *IEEE Security & Privacy*, 6, 1, 52–60. DOI: 10.1109/MSP.2008.9.

[65] Qixu Liu, Yuqing Zhang, Ying Kong, and Qianru Wu. 2012. Improving vrss-based vulnerability prioritization using analytic hierarchy process. *Journal of Systems and Software*, 85, 8, (Aug. 2012), 1699–1708. DOI: 10.1016/j.jss.2012.03.057.

[66] Ponemon Institute LLC. 2022. Cost of a Data Breach Report 2022. Report. IBM Corporation, USA, (July 2022). 59 pp.

[67] Amgad Madkour, Walid G. Aref, Faizan Ur Rehman, Mohamed Abdur Rahman, and Saleh Basalamah. 2017. A survey of shortest-path algorithms. (2017). arXiv: 1705.02044 [cs.DS].

[68] Vasileios Mavroeidis, Ryan Hohimer, Tim Casey, and Audun Jesang. 2021. Threat actor type inference and characterization within cyber threat intelligence. In *2021 13th International Conference on Cyber Conflict (CyCon)* (Tallinn, Estonia). IEEE, New York, NY, USA, 327–352. ISBN: 978-1-6654-4709-6. DOI: 10.23919/CyCon51939.2021.9468305.

[69] Miles A. McQueen, Wayne F. Boyer, Mark A. Flynn, and George A. Beitel. 2006. Time-to-compromise model for cyber risk reduction estimation. In *Quality of Protection*. Dieter Gollmann, Fabio Massacci, and Artsiom Yautsiukhin, (Eds.) Vol. 23. Springer US, Boston, MA, 49–64. ISBN: 978-0-387-36584-8. DOI: 10.1007/978-0-387-36584-8_5.

[70] Microsoft. 2022. Microsoft threat modeling tool threats. Stride model. (Aug. 2022). Retrieved July 20, 2023 from https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats.

[71] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: a comprehensive review. *ACM Comput. Surv.*, 54, 3, Article 62, (Apr. 2021), 40 pages. DOI: 10.1145/3439726.

[72] Lucas Miranda et al. 2021. On the flow of software security advisories. *IEEE Transactions on Network and Service Management*, 18, 2, 1305–1320. Hanan Lutfiyya, (Ed.) DOI: 10.1109/TNSM.2021.3078727.

[73] R. A. Miura-Ko and N. Bambos. 2007. Securerank: a risk-based vulnerability management scheme for computing infrastructures. In *2007 IEEE International Conference on Communications* (Glasgow, UK). IEEE, New York, NY, USA, 1455–1460. ISBN: 1-4244-0353-7. DOI: 10.1109/ICC.2007.244.

[74] Pantaleone Nespoli, Dimitrios Papamartzivanos, Félix Gómez Mármol, and Georgios Kambourakis. 2018. Optimal countermeasures selection against cyber attacks: a comprehensive survey on reaction frameworks. *IEEE Communications Surveys & Tutorials*, 20, 2, 1361–1396. DOI: 10.1109/COMST.2017.2781126.

[75] William Nzoukou, Lingyu Wang, Sushil Jajodia, and Anoop Singhal. 2013. A unified framework for measuring a network's mean time-to-compromise. In *2013 IEEE 32nd International Symposium on Reliable Distributed Systems* (Braga, Portugal). IEEE, New York, NY, USA, 215–224. DOI: 10.1109/SRDS.2013.30.

[76] Kris Oosthoek and Christian Doerr. 2021. Cyber threat intelligence: a product without a process? *International Journal of Intelligence and CounterIntelligence*, 34, 2, 300–315. DOI: 10.1080/08850607.2020.1780062.

[77] Ofri Ouzan. 2023. Do You Know KEV? You Should (Because Hackers Do)! Research Report. Rezilion. Retrieved Apr. 3, 2023 from https://info.rezilion.com/rezilion-2023-kev-research.

[78] 2023. Overview. About the cve program. The MITRE Corporation. (July 13, 2023). Retrieved July 14, 2023 from https://www.cve.org/About/Overview.

[79] Sergio Pastrana, Daniel R. Thomas, Alice Hutchings, and Richard Clayton. 2018. Crimebb: enabling cybercrime research on underground forums at scale. In *Proceedings of the 2018 World Wide Web Conference* (WWW '18) (Lyon, France). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1845–1854. ISBN: 9781450356398. DOI: 10.1145/3178876.3186178.

[80] Charles P. Pfleeger, Shari Lawrence Pfleeger, and Jonathan Margulies. 2015. *Security in Computing*. (5th ed.). Prentice Hall. ISBN: 978-0-13-408504-3.

[81] Cynthia Phillips and Laura Painton Swiler. 1998. A graph-based system for network-vulnerability analysis. In *Proceedings of the 1998 Workshop on New Security Paradigms* (NSPW '98) (Charlottesville, Virginia, USA). Association for Computing Machinery, New York, NY, USA, 71–79. ISBN: 1581131682. DOI: 10.1145/310889.310919.

[82] Norbert Pohlmann. [n. d.] Lebenszyklen von Schwachstellen. Retrieved May 28, 2023 from https://norbert-pohlmann.com/glossar-cyber-sicherheit/lebenszyklen-von-schwachstellen.

[83] BSI Pressestelle. 2023. ngermanweltweiter ransomware-angriff. ngermanlaut medienberichten tausende esxi-server verschlüsselt. ngerman. Bundesamt für Sicherheit in der Informationstechnik. (Feb. 2023). Retrieved Feb. 7, 2023 from https://www.bsi.bund.de/DE/Service-Navi/Presse/Pressemitteilungen/Presse2023/230206_ESXi-Schwachstelle-massiv-ausgenutzt.html.

[84] Engla Rencelj Ling and Mathias Ekstedt. 2023. Estimating time-to-compromise for industrial control system attack techniques through vulnerability data. *SN Computer Science*, 4, 3, (Apr. 2023), 318–329. DOI: 10.1007/s42979-023-01750-z.

[85] Jorge Reyes, Walter Fuertes, Paco Arévalo, and Mayra Macas. 2022. An environment-specific prioritization model for information-security vulnerabilities based on risk factor analysis. *Electronics*, 11, 9. DOI: 10.3390/electronics11091334.

[86] Langley Rock, Stefan Hagen, and Thomas Schmidt. 2022. Common Security Advisory Framework. Standard. Version 2.0. OASIS Open, (Nov. 18, 2022). 123 pp. Retrieved June 12, 2023 from https://docs.oasis-open.org/csaf/csaf/v2.0/csaf-v2.0.html.

[87] Frank Romeike. 2018. *Risikomanagement*. (1st ed.). Springer Gabler, Wiesbaden, (Jan. 17, 2018). 260 pp. ISBN: 978-3-658-13951-3. DOI: 10.1007/978-3-658-13952-0.

[88] Dennis M. Ross, Allan B. Wollaber, and Pierre C. Trepagnier. 2017. Latent feature vulnerability ranking of cvss vectors. In *Proceedings of the Summer Simulation Multi-Conference* (SummerSim '17) Article 19 (Bellevue, Washington). Society for Computer Simulation International, San Diego, CA, USA.

[89] Jukka Ruohonen. 2019. A look at the time delays in cvss vulnerability scoring. *Applied Computing and Informatics*, 15, 2, (July 2019), 129–135. DOI: 10.1016/j.aci.2017.12.002.

[90] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. (2020). arXiv: 1910.01108 [cs.CL].

[91] Reginald E. Sawilla and Xinming Ou. 2008. Identifying critical attack assets in dependency attack graphs. In *Computer Security - ESORICS 2008* (LNSC) (Málaga, Spain). Sushil Jajodia and Javier Lopez, (Eds.) Vol. 5283. Springer, Berlin, Heidelberg, 18–34. ISBN: 978-3-540-88313-5. DOI: 10.1007/978-3-540-88313-5_2.

[92] Riccardo Scandariato, Kim Wuyts, and Wouter Joosen. 2015. A descriptive study of microsoft's threat modeling technique. *Requirements Engineering*, 20, 2, (June 2015), 163–180. DOI: 10.1007/s00766-013-0195-2.

[93] Andreas Schaad and Dominik Binder. 2020. Ml-supported identification and prioritization of threats in the ovvl threat modelling tool. In *Data and Applications Security and Privacy XXXIV* (Lecture Notes in Computer Science) (Regensburg, DE). Anoop Singhal and Jaideep Vaidya, (Eds.) Vol. 12122. Springer International Publishing, Cham, 274–285. ISBN: 978-3-030-49669-2. DOI: 10.1007/978-3-030-49669-2_16.

[94] Muhammad Shahzad, Muhammad Zubair Shafiq, and Alex X. Liu. 2012. A large scale exploratory analysis of software vulnerability life cycles. In *2012 34th International Conference on Software Engineering (ICSE)* (Zurich, Switzerland). IEEE, New York, NY, USA, 771–781. DOI: 10.1109/ICSE.2012.6227141.

[95] Ruchi Sharma, Ritu Sibal, and Sangeeta Sabharwal. 2021. Software vulnerability prioritization using vulnerability description. *International Journal of System Assurance Engineering and Management*, 12, 1, (Feb. 2021), 58–64. DOI: 10.1007/s13198-020-01021-7.

[96] Adam Shostack. 2014. *Threat Modeling. Designing for Security.* Wiley, Indiana, USA. ISBN: 978-1-118-80999-0.

[97] Georgios Spanos, Lefteris Angelis, and Dimitrios Toloudis. 2017. Assessment of vulnerability severity using text mining. In *Proceedings of the 21st Pan-Hellenic Conference on Informatics* (PCI '17) Article 49 (Larissa, Greece). Association for

Computing Machinery, New York, NY, USA. ISBN: 9781450353557. DOI: 10.1145/3139367.3139390.

[98] Hartmut Stadtler. 2015. Supply chain management: an overview. In *Supply Chain Management and Advanced Planning: Concepts, Models, Software, and Case Studies*. Hartmut Stadtler, Christoph Kilger, and Herbert Meyr, (Eds.) Springer, Berlin, Heidelberg, 3–28. ISBN: 978-3-642-55309-7. DOI: 10.1007/978-3-642-55309-7_1.

[99] Octavian Suciu, Connor Nelson, Zhuoer Lyu, Tiffany Bao, and Tudor Dumitraş. 2022. Expected exploitability: predicting the development of functional vulnerability exploits. In *31st USENIX Security Symposium (USENIX Security 22)* (Boston, MA, USA). USENIX Association, Berkeley, CA, USA, (Aug. 2022), 377–394. ISBN: 978-1-939133-31-1.

[100] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, (Eds.) Vol. 30. Curran Associates, Inc., 6000–6010. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[101] VMWare. 2021. VMSA-2021-0002. VMware Inc. (Feb. 2021). Retrieved Feb. 7, 2023 from https://www.vmware.com/security/advisories/VMSA-2021-0002.html.

[102] Olivia von Westernhagen. 2021. Jetzt updaten: Kritische Lücke aus VMware ESXi und vCenter Server beseitigt. Heise Medien. (Feb. 2021). Retrieved Feb. 7, 2023 from https://heise.de/-5063860.

[103] Thomas D. Wagner, Khaled Mahbub, Esther Palomar, and Ali E. Abdallah. 2019. Cyber threat intelligence sharing: survey and research directions. *Computers & Security*, 87, (Nov. 2019). DOI: 10.1016/j.cose.2019.101589.

[104] Michał Walkowski, Maciej Krakowiak, Marcin Jaroszewski, Jacek Oko, and Sławomir Sujecki. 2021. Automatic cvss-based vulnerability prioritization and response with context information. In *2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)* (Split, Hvar, Croatia). IEEE, New York, NY, USA, 1–6. ISBN: 978-1-6654-2456-1. DOI: 10.23919/SoftCOM52868.2021.9559094.

[105] Michał Walkowski, Jacek Oko, and Sławomir Sujecki. 2021. Vulnerability management models using a common vulnerability scoring system. *Applied Sciences*, 11, 18, (Sept. 19, 2021). DOI: 10.3390/app11188735.

[106] Yunpeng Wang, Yinghui Wang, Hongmao Qin, Haojie Ji, Yanan Zhang, and Jian Wang. 2021. A systematic risk assessment framework of automotive cybersecurity. *Automotive Innovation*, 4, 3, (Aug. 2021), 253–261. DOI: 10.1007/s42154-021-00140-6.

[107] Arun Warikoo. 2014. Proposed methodology for cyber criminal profiling. *Information Security Journal: A Global Perspective*, 23, 4-6, 172–178. DOI: 10.1080/19393555.2014.931491.

[108] Thomas Wolf et al. 2020. Transformers: state-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (Online). Association for Computational Linguistics, (Aug. 2020), 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6.

[109] Kirsten Wulfmeier and Karl-Heinz Niemann. 2022. Befragung zur Auswertung von Security Advisories im Produktionsumfeld durch KMU. Hochschule Hannover. 24 pp. DOI: 10.25968/OPUS-2303.

[110] Geeta Yadav and Kolin Paul. 2019. Patchrank: ordering updates for scada systems. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* (Zaragoza, ESP). IEEE, New York, NY, USA, 110–117. ISBN: 978-1-7281-0304-4. DOI: 10.1109/ETFA.2019.8869110.

[111] Zhen Zeng, Zhun Yang, Dijiang Huang, and Chun-Jen Chung. 2022. Licality — likelihood and criticality: vulnerability risk prioritization through logical reasoning and deep learning. *IEEE Transactions on Network and Service Management*, 19, 2, 1746–1760. DOI: 10.1109/TNSM.2021.3133811.

[112] Fengli Zhang and Qinghua Li. 2020. Dynamic risk-aware patch scheduling. In *2020 IEEE Conference on Communications and Network Security (CNS)* (Avignon, France). IEEE, New York, NY, USA, 1–9. ISBN: 978-1-7281-4761-1. DOI: 10.1109/CNS48642.2020.9162225.

[113] Yichi Zhang, Lingfeng Wang, Yingmeng Xiang, and Chee-Wooi Ten. 2015. Power system reliability evaluation with scada cybersecurity considerations. *IEEE Transactions on Smart Grid*, 6, 4, 1707–1721. DOI: 10.1109/TSG.2015.2396994.

[114] Andrej Zieger, Felix Freiling, and Klaus-Peter Kossakowski. 2018. The β-time-to-compromise metric for practical cyber security risk estimation. In *2018 11th International Conference on IT Security Incident Management & IT Forensics (IMF)* (Hamburg, Germany). IEEE, New York, NY, USA, (May 2018), 115–133. ISBN: 978-1-5386-6633-3. DOI: 10.1109/IMF.2018.00017.

A Evaluation Result

Here, the full evaluation results are listed. For both test networks, two views of the results are given: (i) the actual computed values and (ii) the derived prioritisation rank. For a comparison to other concepts applied practice, the DAF-score and maximum CVSS base score for each advisory are listed as well.

- *Network 1*
  - (i) Actual values: Table A.1
  - (ii) Derived rank: Table A.2
- *Network 2*
  - (i) Actual values: Table A.3
  - (ii) Derived rank: Table A.4

Table A.1: Evaluation result of network 1. Shows the prioritization order by concept. The profile columns show the calculated values limited to four decimals points. Sorted by the ranking of profile four. Profile names are abbreviated by pX.

			Individual			Network adjusted	
ID	Host	DAF-Score	max. CVSS	p1	p4	p1	p4
Advisory_9081	H1	low	8.1	0.2604	0.4005	0.2604	2.1257
Advisory_20242	H4	low	7.8	0.1565	0.1581	1.1961	1.2630
Advisory_18840	H5	high	9.8	0.8504	0.9187	0.8504	0.9187
Advisory_16585	H3	high	7.5	0.1926	0.3208	0.1926	0.3208
Advisory_16824	H1	high	8.1	0.2616	0.2476	1.7906	0.2476
Advisory_20128	H1	medium	9.8	0.2238	0.2243	0.2238	0.2243
Advisory_493	H2	high	8.8	0.1661	0.1658	0.1661	0.1658
Advisory_17486	H4	low	7.5	0.1403	0.1413	0.1403	0.1413
Advisory_2240	H5	low	7.5	0.1403	0.1413	0.1403	0.1413
Advisory_16358	H2	low	7.8	0.1324	0.1325	0.1324	0.1325
Advisory_1906	H5	low	5.9	0.0489	0.0449	0.0489	0.0449
Advisory_16437	H2	low	5.9	0.0285	0.0259	0.0285	0.0259

Table A.2: Evaluation result of network 1. Shows the prioritization order by concept. The columns display the position when sorted by the according concept. Sorted by the ranking of profile four. Profile names are abbreviated by pX.

			Individual			Network adjusted	
ID	Host	DAF-Score	max. CVSS	p1	p4	p1	p4
Advisory_9081	H1	low	5	3	2	4	1
Advisory_20242	H4	low	7	7	7	2	2
Advisory_18840	H5	high	2	1	1	3	3
Advisory_16585	H3	high	8	5	3	6	4
Advisory_16824	H1	high	4	2	4	1	5
Advisory_20128	H1	medium	1	4	5	5	6
Advisory_493	H2	high	3	6	6	7	7
Advisory_17486	H4	low	9	8	8	8	8
Advisory_2240	H5	low	10	9	9	9	9
Advisory_16358	H2	low	6	10	10	10	10
Advisory_1906	H5	low	12	11	11	11	11
Advisory_16437	H2	low	11	12	12	12	12

Table A.3: Evaluation result of network 2. Shows the prioritization order by concept. The profile columns show the calculated values limited to four decimals points. Sorted by the ranking of profile four. Profile names are abbreviated by pX.

			Individual			Network adjusted	
ID	Host	DAF-Score	max. CVSS	p1	p4	p1	p4
Advisory_2080	H4	high	8.8	0.5850	0.7508	4.9808	5.8655
Advisory_15904	H3	very_high	9.0	0.8306	0.9099	0.8306	0.9099
Advisory_17032	H5	high	7.8	0.7970	0.8915	0.7970	0.8915
Advisory_16830	H7	very_high	9.9	0.7346	0.8549	0.7346	0.8549
Advisory_322	H3	very_high	8.8	0.7307	0.8334	0.7307	0.8334
Advisory_1029	H8	very_high	9.8	0.6935	0.8290	0.6935	0.8290
Advisory_17314	H9	high	8.8	0.6060	0.7559	0.6060	0.7559
Advisory_16809	H11	very_high	9.1	0.6459	0.7393	0.6459	0.7393
Advisory_17868	H6	high	8.8	0.5765	0.7377	0.5765	0.7377
Advisory_15560	H10	very_high	7.8	0.4247	0.6080	0.4247	0.6080
Advisory_462	H6	high	8.1	0.4785	0.5793	0.4785	0.5793
Advisory_416	H8	high	8.1	0.4785	0.5793	0.4785	0.5793
Advisory_2490	H2	high	7.5	0.4208	0.5100	0.4208	0.5100
Advisory_16620	H9	high	9.6	0.5000	0.5000	0.5000	0.5000
Advisory_15534	H1	high	8.8	0.2540	0.2492	0.5080	0.4984
Advisory_16212	H9	very_high	8.1	0.3144	0.4204	0.3144	0.4204
Advisory_1849	H5	high	8.1	0.2499	0.3544	0.2499	0.3544
Advisory_19489	H7	very_high	8.8	0.2854	0.2918	0.2854	0.2918
Advisory_2016	H3	high	9.8	0.2303	0.2344	0.2303	0.2344
Advisory_19765	H8	high	7.5	0.1376	0.1440	0.1376	0.1440

Table A.4: Evaluation result of network 2. Shows the prioritization order by concept. The columns display the position when sorted by the according concept. Sorted by the ranking of profile four. Profile names are abbreviated by pX.

			individual			network adjusted	
ID	Host	DAF-Score	max. CVSS	p1	p4	p1	p4
Advisory_2080	H4	high	11	8	7	1	1
Advisory_15904	H3	very_high	6	1	1	2	2
Advisory_17032	H5	high	18	2	2	3	3
Advisory_16830	H7	very_high	1	3	3	4	4
Advisory_322	H3	very_high	12	4	4	5	5
Advisory_1029	H8	very_high	3	5	5	6	6
Advisory_17314	H9	high	8	7	6	8	7
Advisory_16809	H11	very_high	5	6	8	7	8
Advisory_17868	H6	high	10	9	9	9	9
Advisory_15560	H10	very_high	17	13	10	14	10
Advisory_416	H8	high	15	11	12	12	11
Advisory_462	H6	high	13	12	11	13	12
Advisory_2490	H2	high	20	14	13	15	13
Advisory_16620	H9	high	4	10	14	11	14
Advisory_15534	H1	high	7	17	18	10	15
Advisory_16212	H9	very_high	16	15	15	16	16
Advisory_1849	H5	high	14	18	16	18	17
Advisory_19489	H7	very_high	9	16	17	17	18
Advisory_2016	H3	high	2	19	19	19	19
Advisory_19765	H8	high	19	20	20	20	20

B Extension Evaluation Result

Here, the full evaluation results of the extension are listed. For both test networks, two views of the results are given: (i) the actual computed values and (ii) the derived prioritisation rank. For a comparison to other concepts applied practice, the DAF-score and maximum CVSS base score for each advisory are listed as well. To highlight the effect of the extension, the initial evaluation results were also added.

- *Network 1*
  - (i) Actual values: Table B.1
  - (ii) Derived rank: Table B.2
- *Network 2*
  - (i) Actual values: Table B.3
  - (ii) Derived rank: Table B.4

Table B.1: Evaluation result of network 1. Shows the prioritization order by concept. The profile columns show the calculated values limited to four decimals points. Sorted by the ranking of profile four with extension. Profile names are abbreviated by p*X*.

						Network adjusted			
			Individual			Initial		Extension	
ID	Host	DAF-Score	max. CVSS	p1	p4	p1	p4	p1	p4
Advisory_9081	H1	low	8.1	0.2604	0.4005	0.2604	2.1257	1.7895	2.1257
Advisory_20242	H4	low	7.8	0.1565	0.1581	1.1961	1.2630	1.1961	1.2630
Advisory_18840	H5	high	9.8	0.8504	0.9187	0.8504	0.9187	0.8504	0.9187
Advisory_16585	H3	high	7.5	0.1926	0.3208	0.1926	0.3208	0.1926	0.3208
Advisory_16824	H1	high	8.1	0.2616	0.2476	1.7906	0.2476	0.2616	0.2476
Advisory_20128	H1	medium	9.8	0.2238	0.2243	0.2238	0.2243	0.2238	0.2243
Advisory_493	H2	high	8.8	0.1661	0.1658	0.1661	0.1658	0.1661	0.1658
Advisory_17486	H4	low	7.5	0.1403	0.1413	0.1403	0.1413	0.1403	0.1413
Advisory_2240	H5	low	7.5	0.1403	0.1413	0.1403	0.1413	0.1403	0.1413
Advisory_16358	H2	low	7.8	0.1324	0.1325	0.1324	0.1325	0.1324	0.1325
Advisory_1906	H5	low	5.9	0.0489	0.0449	0.0489	0.0449	0.0489	0.0449
Advisory_16437	H2	low	5.9	0.0285	0.0259	0.0285	0.0259	0.0285	0.0259

Table B.2: Evaluation result of network 1. Shows the prioritization order by concept. The columns display the position when sorted by the according concept. Sorted by the ranking of profile four with extension. Profile names are abbreviated by pX.

						Network adjusted			
			Individual			Initial		Extension	
ID	Host	DAF-Score	max. CVSS	p1	p4	p1	p4	p1	p4
Advisory_9081	H1	low	5	3	2	4	1	1	1
Advisory_20242	H4	low	7	7	7	2	2	2	2
Advisory_18840	H5	high	2	1	1	3	3	3	3
Advisory_16585	H3	high	8	5	3	6	4	6	4
Advisory_16824	H1	high	4	2	4	1	5	4	5
Advisory_20128	H1	medium	1	4	5	5	6	5	6
Advisory_493	H2	high	3	6	6	7	7	7	7
Advisory_17486	H4	low	9	8	8	8	8	8	8
Advisory_2240	H5	low	10	9	9	9	9	9	9
Advisory_16358	H2	low	6	10	10	10	10	10	10
Advisory_1906	H5	low	12	11	11	11	11	11	11
Advisory_16437	H2	low	11	12	12	12	12	12	12

Table B.3: Evaluation result of network 2. Shows the prioritization order by concept. The profile columns show the calculated values limited to four decimals points. Sorted by the ranking of profile four with extension. Profile names are abbreviated by pX.

						Network adjusted			
			Individual			Initial		Extension	
ID	Host	DAF-Score	max. CVSS	p1	p4	p1	p4	p1	p4
Advisory_2080	H4	high	8.8	0.5850	0.7508	4.9808	5.8655	4.9808	5.8655
Advisory_15904	H3	very_high	9.0	0.8306	0.9099	0.8306	0.9099	0.8306	0.9099
Advisory_17032	H5	high	7.8	0.7970	0.8915	0.7970	0.8915	0.7970	0.8915
Advisory_16830	H7	very_high	9.9	0.7346	0.8549	0.7346	0.8549	0.7346	0.8549
Advisory_322	H3	very_high	8.8	0.7307	0.8334	0.7307	0.8334	0.7307	0.8334
Advisory_1029	H8	very_high	9.8	0.6935	0.8290	0.6935	0.8290	0.6935	0.8290
Advisory_17314	H9	high	8.8	0.6060	0.7559	0.6060	0.7559	0.6060	0.7559
Advisory_16809	H11	very_high	9.1	0.6459	0.7393	0.6459	0.7393	0.6459	0.7393
Advisory_17868	H6	high	8.8	0.5765	0.7377	0.5765	0.7377	0.5765	0.7377
Advisory_15560	H10	very_high	7.8	0.4247	0.6080	0.4247	0.6080	0.4247	0.6080
Advisory_462	H6	high	8.1	0.4785	0.5793	0.4785	0.5793	0.4785	0.5793
Advisory_416	H8	high	8.1	0.4785	0.5793	0.4785	0.5793	0.4785	0.5793
Advisory_2490	H2	high	7.5	0.4208	0.5100	0.4208	0.5100	0.4208	0.5100
Advisory_16620	H9	high	9.6	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Advisory_15534	H1	high	8.8	0.2540	0.2492	0.5080	0.4984	0.5080	0.4984
Advisory_16212	H9	very_high	8.1	0.3144	0.4204	0.3144	0.4204	0.3144	0.4204
Advisory_1849	H5	high	8.1	0.2499	0.3544	0.2499	0.3544	0.2499	0.3544
Advisory_19489	H7	very_high	8.8	0.2854	0.2918	0.2854	0.2918	0.2854	0.2918
Advisory_2016	H3	high	9.8	0.2303	0.2344	0.2303	0.2344	0.2303	0.2344
Advisory_19765	H8	high	7.5	0.1376	0.1440	0.1376	0.1440	0.1376	0.1440

Table B.4: Evaluation result of network 2. Shows the prioritization order by concept. The columns display the position when sorted by the according concept. Sorted by the ranking of profile four with extension. Profile names are abbreviated by pX.

						Network adjusted			
			Individual			Initial		Extension	
ID	Host	DAF-Score	max. CVSS	p1	p4	p1	p4	p1	p4
Advisory_2080	H4	high	11	8	7	1	1	1	1
Advisory_15904	H3	very_high	6	1	1	2	2	2	2
Advisory_17032	H5	high	18	2	2	3	3	3	3
Advisory_16830	H7	very_high	1	3	3	4	4	4	4
Advisory_322	H3	very_high	12	4	4	5	5	5	5
Advisory_1029	H8	very_high	3	5	5	6	6	6	6
Advisory_17314	H9	high	8	7	6	8	7	8	7
Advisory_16809	H11	very_high	5	6	8	7	8	7	8
Advisory_17868	H6	high	10	9	9	9	9	9	9
Advisory_15560	H10	very_high	17	13	10	14	10	14	10
Advisory_416	H8	high	15	11	12	12	11	12	11
Advisory_462	H6	high	13	12	11	13	12	13	12
Advisory_2490	H2	high	20	14	13	15	13	15	13
Advisory_16620	H9	high	4	10	14	11	14	11	14
Advisory_15534	H1	high	7	17	18	10	15	10	15
Advisory_16212	H9	very_high	16	15	15	16	16	16	16
Advisory_1849	H5	high	14	18	16	18	17	18	17
Advisory_19489	H7	very_high	9	16	17	17	18	17	18
Advisory_2016	H3	high	2	19	19	19	19	19	19
Advisory_19765	H8	high	19	20	20	20	20	20	20

Glossary

API An Application Programming Interface (API) is a concept that enables the communication with a set of standards and protocols between two software components. Typically, a service, e.g. features or data, is offered by a server which can be accessed by the client by implementing the concepts of the API documentation..

DoS A Denial-of-Service (DoS) attack describes a type of attack which prevents the legitimate use of a system or service. This can for example be achieved by occupying all available resources..

PDCA A quality improvement management methodology, consisting of the steps **P**lan, **D**o, **C**heck and **A**ct.

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original