Calculating the vulnerability remediation order based on open source intelligence

Author: Richard van Ginkel
        s4599047

External supervisor: Bart Roos
                    bart.roos@northwave.nl

Internal supervisor: Dr. ir. Ileana Buhan
                     ileana.buhan@ru.nl

Second assessor: Dr. ir. Harald Vranken
                Harald.Vranken@ru.nl

October 29, 2021
Abstract

Organizations use vulnerability scans to gain insight into the level of security of their digital infrastructure. Results of such a scan are often scored using Common Vulnerability Scoring System (CVSS). This regularly results in an overwhelming amount of vulnerabilities that are scored high or critical, which makes it difficult to determine which vulnerability should be remedied first.

In this research project, we propose a new score calculation method which helps select the best follow-up actions with the most impact on the security level of the organization. We develop three optional calculation methods that we apply on real vulnerability scanning data. To validate the results, we conduct a survey on experts in the field. Analysis of the results of the survey showed that taking into account the reachability of vulnerabilities and the existence of an exploit improves the scoring of vulnerabilities significantly. Our proposed method that combines those two aspects can help experts in the field of cyber security selecting which vulnerabilities to remedy first.
# Contents

1 Introduction ............................................. 2

2 Preliminaries ........................................... 4
   2.1 Common Vulnerability Scoring System ................. 4
      2.1.1 Base metrics .................................. 5
      2.1.2 Temporal metrics ............................. 6
      2.1.3 Environmental metrics ......................... 6
      2.1.4 Vectors and example .......................... 7
      2.1.5 Version 1 and version 3 ....................... 8

3 Related Work ........................................... 9
   3.1 CVSS .............................................. 9
   3.2 Remediation score calculation ....................... 12
   3.3 Risk assessment .................................. 13
   3.4 Related work and our research question ............ 14

4 Methodology ........................................... 15
   4.1 Developing the methods ............................. 15
      4.1.1 Data collection ................................ 15
      4.1.2 Parameters of our methods .................... 16
      4.1.3 Objectives .................................... 18
      4.1.4 Optional methods ............................. 19
   4.2 Validating our optional methods ..................... 25
      4.2.1 Survey ......................................... 25
      4.2.2 Hypotheses .................................... 25
      4.2.3 Results of the survey .......................... 26
   4.3 Optimal calculation method ......................... 30

5 Conclusions ........................................... 32

Appendices ............................................. 36
Chapter 1

Introduction

In 2020, over eighteen thousand new vulnerabilities have been reported in the National Vulnerability Database (NVD)\(^1\), which is the result of the increasing trend seen over the past years. Vulnerabilities can be exploited by malicious actors to gain access to devices and data, or perform other unwanted actions.

Because of this, organizations want to minimize the number of vulnerabilities in their digital infrastructure. To this end, organizations can scan their infrastructure for vulnerabilities, or let others do so for them, using scanning tools. Such a scan will result in a list of vulnerabilities found, together with the severity score of the vulnerability. These scores are commonly calculated using the Common Vulnerability Scoring System (CVSS)\(^2\), or a calculation method based on CVSS. This way, one can obtain insight into the vulnerabilities that have the highest technical severity.

However, most organizations will face problems in this process. CVSS is limited to technical severity, while the context of the organization and the vulnerability are relevant. Next to that, too many of the found vulnerabilities are scored high or critical. Using version 2 of CVSS, which divides vulnerabilities into the categories low, medium and high, one-third of the vulnerabilities is scored as high, according to the NVD Dashboard\(^3\). CVSS version 3 introduced the category critical. More than half of the vulnerabilities found are scored either high or critical when using this new version of CVSS. This number of high or critical vulnerabilities overwhelms organizations in their mission to keep the digital infrastructure secure.

In the current way of working, someone analyses the results of a vulnerability scan and selects the most important vulnerabilities out of the vast number of high-scored vulnerabilities. These selected vulnerabilities can then be remedied first. The process of analyzing the list of high-scored

---

\(^1\)https://nvd.nist.gov/
\(^2\)https://www.first.org/cvss/
\(^3\)https://nvd.nist.gov/general/nvd-dashboard
vulnerabilities and selecting the most important ones is time-consuming.

Ideally, organizations would periodically have their infrastructure scanned for vulnerabilities. The outcome of these scans would be a short and manageable list of remedies against the vulnerabilities with the biggest potential impact on the organization. Together with managing and solving these vulnerabilities, this forms the concept of vulnerability management.

However, in vulnerability management, most vendors use their own vulnerability and remedy score calculation method. Such a method is based on a version of CVSS. An optimal method is not yet found and vendors are still working to improve their method. How these calculations are designed and work is unclear, since the vendors do not publish their methods. Because of this, open research on this problem is desired. In this research project, we propose a risk and remedy score calculation method that is suitable for vulnerability management, with additional information on the vulnerability taken into account. We formulate this problem as the following research question:

**Can we develop a risk scoring method that helps a human expert select vulnerabilities for remediation?**

The goal of this thesis is to find a suitable method based on actual vulnerability scan data. To this end, we develop several options and conduct a survey among experts to establish the best option.

To answer our research question, we explain different CVSS versions in chapter 2. Research related to our research question is discussed in chapter 3. We found that there has been quite some research on how to build or improve CVSS, however, this thesis takes a different perspective. In chapter 4 we elaborate on the research problem and our optional methods which are applied to vulnerability scan data and the survey that we conducted. Next to that we propose a calculation method based on the results of the survey. In the conclusion, which can be found in chapter 5, we summarize our findings.
Chapter 2

Preliminaries

2.1 Common Vulnerability Scoring System

The Common Vulnerability Scoring System, abbreviated as CVSS, is a framework that is widely used to score vulnerabilities. Using three different metric groups, which will be discussed in the following subsections, one can determine the technical severity of a vulnerability. Calculating using these groups will result in a score in the range of zero to ten. The scores are commonly categorized as low, medium or high, for CVSS version 2. Low covers a score from 0.0 to 3.9, medium from 4.0 to 6.9 and high from 7.0 to 10.0. In version 3, the categories none and critical are added, so the division is done slightly differently. A score of 0.0 is categorized as none and scores 9.0 to 10.0 are categorized as critical. The remaining scores are all still in the same category as was the case with version 2 of CVSS.

The framework is developed by the National Infrastructure Advisory Council (NIAC) and launched in 2005. After that, further development has been done by the Forum of Incident Response and Security Teams (FIRST), an organization that was formed as a response to one of the first big security incidents. As stated on their website, ”FIRST aspires to bring together incident response and security teams from every country across the world to ensure a safe internet for all.” FIRST developed a second and a third version and released those in 2007 and 2015, respectively.

The second version is most common, which is why we discuss this version below. For our research, we will use CVSS version 3 as a building block when it is available, since it contains improvements with respect to version 2. CVSS version 1 and CVSS version 3 are slightly different compared to version 2, as described below.

\[\text{https://www.cisa.gov/niac}\]
\[\text{https://www.first.org/}\]
\[\text{https://www.first.org/cvss/v2/guide}\]
\[\text{https://www.first.org/cvss/v1/guide}\]
\[\text{https://www.first.org/cvss/v3.1/specification-document}\]
2.1.1 Base metrics

As said, CVSS consists of three metrics groups, of which the base metric group is the first one. The base metrics deal with the intrinsic and fundamental characteristics of a vulnerability. These characteristics are not affected by time or environments. This metric group consists of six metrics:

- **Access Vector (AV)**: reflects how the vulnerability is exposed.
- **Access Complexity (AC)**: reflects the complexity of gaining access to the vulnerability.
- **Authentication (Au)**: measures the number of times an attacker must authenticate to exploit the vulnerability.
- **Confidentiality Impact (C)**: measures the impact on confidentiality.
- **Integrity Impact (I)**: measures the impact on integrity.
- **Availability Impact (A)**: measures the impact on availability.

All of these metrics are assigned one of the three possible scores for the metric. For AV, these possible scores are shown below in the example. For AC these scores are *high*, *medium* and *low*. Au has *multiple*, *single* and *none* as possible scores. For the metrics C, I and A, the possible scores are *none*, *partial* and *complete*. Each of these scores corresponds to a value, as can be seen below for the AV, which is shown as an example:

<table>
<thead>
<tr>
<th>Access Vector</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires local access</td>
<td>0.395</td>
</tr>
<tr>
<td>Adjacent network accessible</td>
<td>0.646</td>
</tr>
<tr>
<td>Network accessible</td>
<td>1.0</td>
</tr>
</tbody>
</table>

For a vulnerability that is only reachable when one has local access to the device the vulnerability is on, the value in equation (2.3) is 0.395. This is the lowest possible value, since a vulnerability is more problematic when accessible via the network. With the values for the five other metrics, together with the AV described above, one can calculate the CVSS base score, which is rounded to one decimal, as follows:

$$
BaseScore = ((0.6 \times Impact) + (0.4 \times Exploitability) - 1.5) \times f(Impact) \quad (2.1)
$$

$$
Impact = 10.41 \times (1 - (1 - C) \times (1 - I) \times (1 - A)) \quad (2.2)
$$

$$
Exploitability = 20 \times AV \times AC \times Au \quad (2.3)
$$
\[ f(impact) = 0 \text{ if } Impact = 0, \ 1.176 \text{ otherwise } \quad (2.4) \]

2.1.2 Temporal metrics

In the temporal metrics group, three metrics are presented. These metrics deal with how the threat posed by a vulnerability changes over time. The three metrics are:

- **Exploitability (E)**: measures the current state of exploit techniques or code available.
- **Remediation Level (RL)**: measures the level at which a remedy for the vulnerability exists.
- **Report Confidence (RC)**: measures the degree of confidence in the existence of the vulnerability and the details known.

A score is assigned to these metrics as well. However, for the temporal metrics, four or five possible scores exist. We see that for the RL, there are five possible scores, each with a value assigned to it, as can be seen in this example:

<table>
<thead>
<tr>
<th>RemediationLevel</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Official-fix</td>
<td>0.87</td>
</tr>
<tr>
<td>Temporary-fix</td>
<td>0.90</td>
</tr>
<tr>
<td>Workaround</td>
<td>0.95</td>
</tr>
<tr>
<td>Unavailable</td>
<td>1.00</td>
</tr>
<tr>
<td>Defined</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Using the values of the three metrics, the Temporal score, which is rounded to one decimal, can be calculated as:

\[ TemporalScore = BaseScore \ast E \ast RL \ast RC \quad (2.5) \]

2.1.3 Environmental metrics

Lastly, the environmental score can be calculated using the metrics in the environmental metrics group. This group contains five metrics, which deal with the environment of the vulnerability, i.e. the organization in which the vulnerability exists and its stakeholders. The metrics are:

- **Collateral Damage Potential (CDP)**: measures the potential for loss of life or physical assets through damage or theft.
• **Target Distribution (TD)**: measures the proportion of vulnerable systems.

• **Security Requirements (Confidentiality Requirement (CR), Integrity Requirement (IR), Availability Requirement (AR))**: enables an expert to customize the calculation based on the organization’s requirements for confidentiality, integrity and availability. These three requirements can all be set at *High, Medium, Low* and *Not Defined*.

It is important to note the difference between the impact on and the requirements for confidentiality, integrity and availability. In the base metric group confidentiality, integrity and availability deal with measuring how those criteria are violated within the system where the vulnerability is in. In the Environmental metric group, one can use confidentiality, integrity and availability requirements to ensure, for example, that an infringement of confidentiality will result in a higher score when the organization sees confidentiality as an important requirement.

After all the metrics have been scored and assigned a value, we can calculate the Environmental score using the following formulas:

\[
\text{EnvironmentalScore} = (\text{AdjustedTemporal} + (10 - \text{AdjustedTemporal}) \times CDP) \times TD \tag{2.6}
\]

\[
\text{AdjustedTemporal} = \text{TemporalScore recomputed with the BaseScore's Impact sub-equation replaced with the AdjustedImpact equation} \tag{2.7}
\]

\[
\text{AdjustedImpact} = \min(10, 10.41 \times (1 - C \times CR) \times (1 - I \times IR) \times (1 - A \times AR)) \tag{2.8}
\]

### 2.1.4 Vectors and example

The scoring of a vulnerability can also be expressed as a vector. In the case of CVSS, a vector is a string that consists of all the information with which the score was calculated. We can use base, temporal and environmental vectors. The string consists of the assigned score for each metric, concatenated with a slash between them. For example,

\[
\text{AV:L/AC:M/Au:N/C:N/I:P/A:C}
\]

stands for a base score where the metrics are scored as follows, with the values that belong to the scores:
• **Access Vector (AV):** Local access: 0.395

• **Access Complexity (AC):** Medium: 0.61

• **Authentication (Au):** None: 0.704

• **Confidentiality Impact (C):** None: 0.0

• **Integrity Impact (I):** Partial: 0.275

• **Availability Impact (A):** Complete: 0.660

When we fill in these values in equation (2.1) and its sub-equations we saw at subsection 2.1.1, we get:

\[
BaseScore = ((0.6 \times 7.843935) + (0.4 \times 3.392576) - 1.5) \times 1.176 = 5.4 \quad (2.9)
\]

\[
Impact = 10.41 \times (1 - (1 - 0.0) \times (1 - 0.275) \times (1 - 0.660)) = 7.843935 \quad (2.10)
\]

\[
Exploitability = 20 \times 0.395 \times 0.61 \times 0.704 = 3.392576 \quad (2.11)
\]

\[
f(impact) = 1.176 \quad (2.12)
\]

### 2.1.5 Version 1 and version 3

As discussed above, the second version of CVSS is slightly different than versions 1 and 3.

The difference between version 1 and version 2 is that version 1 does not use the security requirements in the calculation of the environmental score. Instead, the environmental score is calculated based on only the Collateral Damage Potential and the Target Distribution.

Version 3 added two metrics in the base metric group. **User Interaction** and **Scope** are added to the equation. User Interaction describes whether a user, other than the attacker, is needed to perform one or more actions before the vulnerability can be exploited. The Scope captures whether exploiting the vulnerability affects other systems or applications than the one that the vulnerability is in. An attacker may be able to compromise other systems via an entry point in one system.
Chapter 3

Related Work

In this chapter, literature related to our research problem is discussed. As described in the introduction, we propose a method which helps select the vulnerability remediations that have the biggest impact. To this end, research into related topics is reviewed below. In this chapter we review articles on the Common Vulnerability Scoring System (CVSS) first. After that, research on remediation score calculation is discussed, followed by research on the concept of risk assessment. Lastly, our research question is compared to the studies discussed below.

3.1 CVSS

The Common Vulnerability Scoring System, abbreviated as CVSS, is the most commonly used standard for assessing the severity of vulnerabilities in computer systems. Because of this, we discuss relevant research related to CVSS in this section.

Singh et al. [11] researched the impact of the temporal metric group and the environmental metric group of CVSS. As described in chapter 2, these two metric groups are optional. They can be used to calculate a risk score based on additional information on the presence of exploits of the vulnerability and characteristics of the organization in which the vulnerability is present. Singh et al. [11] conclude that the use of the temporal metric group and the environmental metric group results in a more effective way to evaluate the risk level of a vulnerability, and thus, that the use of these metrics is useful in system security. Because of this, we know that the elements used in these metrics must be considered in our method as well. We can not use these metrics as a whole, because our scanning tool, as well as the online sources that we consult, do not offer the information of these metric groups.

Doynikova and Kotenko [2] developed a CVSS-based risk assessment technique. The technique takes variables into account as the attack probabilities, impact of the attack and potential financial loss of the attack. Ex-
periments on their test environment have been done for those three variables separately. The experiments resulted in a tool that combines the three different aspects to get a more accurate assessment of the security situation. This research focuses on known attacks and how to defend against them. In other words, known attacks are used as input, and for each of these is calculated how likely an actual attack on the given environment is. Based on these outcomes, one can then select what countermeasure should be selected. However, in our research, we use vulnerabilities as the basis of our method. Our goal is not to select a countermeasure against an attack, but rather to remedy the vulnerability that made the attack possible. Because of this, the tool proposed in the paper is not suitable for our research.

The paper of Wang et al. [13] proposes an improved version of CVSS. They see the scoring done with the temporal metric group and the environmental metric group as subjective. Because of this, the environmental metrics are not used in the proposed improvement, and the base metrics and the Temporal metrics are slightly altered. For the base metrics, Wang et al. [13] added the type of the server and the operating system on a host in the calculation. For the temporal metric group, two known distributions, the Pareto distribution and the Weibull distribution, are used to calculate the exploitability and the remediation level of a vulnerability. The Report Confidence, which 'measures the degree of confidence in the existence of the vulnerability and the credibility of the known technical details'\(^1\), is not used, since it is seen as subjective and the standard value does not influence the outcome. They illustrate their changed method by a small experiment, showing that their method is more accurate and credible. This research can be useful for our own method. In case that there is no exploit or remediation known, we can use the distributions to calculate an estimate.

Fruhwirth and Mannisto [6] used the same two distributions, the Pareto distribution and the Weibull distribution, to improve the temporal metrics, which results in the same method for these metrics as proposed by Wang et al [13]. However, Fruhwirth and Mannisto [6] were able to improve the environmental metrics as well. This was done by conducting a survey among security managers of different companies. The result was that availability was ranked as most important, while integrity was seen as least important. The results of this survey were translated into weights of the metrics in the environmental metrics of CVSS. With these improvements, they were able to bring down the score by 0.5 on average. This results in fewer critical vulnerabilities as a result of a scan. A decrease of 76 percent was seen, which makes it easier to distinguish the most important vulnerabilities from the others. This research had the goal to improve the temporal metrics of CVSS. Our thesis aims to enrich the base metrics of CVSS with extra information found using open source intelligence.

\(^1\)\url{https://www.first.org/cvss/specification-document}
Houmb et al. [7] use the information given by CVSS and use it as input for a Bayesian Belief Network. This means they do not calculate a risk score, but rather use the variables of a risk score as input. With this input, estimates on the impact and the frequency of a vulnerability being exploited are made. However, both hard and soft evidence is used as additional input. This results in a more subjective method, which is undesirable for our research.

Scarfone and Mell [10] analyzed CVSS version 2. The newer version was compared to the first version of CVSS. They found that the scores that are calculated with the second version are higher on average, but more distributed. Scarfone and Mell mention other possible improvements, but stress that it is important to examine whether increased accuracy of an improvement is worth the added complexity in the calculation method. This research tells us that additions made to CVSS in the second version are interesting for our method, as long as it is possible to automate the calculations on these additional variables.

Elbaz et al. [3] researched the problem that newly disclosed vulnerabilities offer regarding CVSS. After a vulnerability has been disclosed, it will take time before a CVSS score for the vulnerability has been established, since this is done by human analysis of the vulnerability. In this period of time that no CVSS score, together with the information in the CVSS vector, is known, automated tools face problems handling such vulnerabilities. To this end, Elbaz et al. propose a method that estimates the CVSS score and vector, based on the human-readable description of the vulnerability. This research can help when applying our methods to new vulnerabilities. However, the scores that will be calculated with their proposed methods are problematic in the same ways as scores calculated with the standard CVSS.

In the research of Murthy [9], the correlation between CVSS scores in vulnerability disclosures and patching is analyzed. This analysis was based on the health care sector of the United States of America. The paper states that no significant relation between the CVSS score and the frequency of patching could be found. This indicates that the current use of CVSS is not suitable for vulnerability management. One of the aspects that can cause this lack of relationship, is the fact that CVSS does not account for safety, according to Murthy.

Spring et al. [12] discuss whether CVSS needs to be changed. They conclude that this is the case, because CVSS scores severity and not risk, while it is used as a risk score on many occasions. Next to that, the failure to account for context and consequences in CVSS is problematic according to Spring et al. Suggestions on fixing these problems are made. Context and consequences should be added to the equation, which is what we aim to do in our methods. Next to that, any proposed improvement should be accompanied by a study of the consistency of humans scoring using it. This is what we aim to do with the survey conducted on experts in the field of
cyber security. Thus, the research of Spring et al. explains clearly why the problem that we aim to solve is relevant. Next to that, their suggestions validate the approach that we used to solve our research problem.

3.2 Remediation score calculation

The goal of this line of research is to develop a calculation method to find the vulnerability remediations that have the biggest impact on the security level of an organization. Research on this specific topic has barely been done. However, below we will discuss an article that is most closely related to our research problem according to our findings. The contributions to the field will be reviewed, and comparisons to what we contribute in addition will be made.

Farris et al. [4] developed a vulnerability management strategy. To this end, they used two metrics, i.e. time-to-vulnerability remediation (TVR) and total vulnerability exposure (TVE). TVR indicates the time between the detection and remediation of a vulnerability. The TVE is calculated based on multiple variables of the vulnerabilities, such as age, the number of months a vulnerability has persisted and severity, where CVSS is used for the latter. These variables are multiplied by a weight, which is chosen by a system operator. Because of this, these values are rather subjective.

The vulnerability score is then multiplied by a scalar value, which is also determined by the system operator. The results of this multiplication, which are called mitigation utilities, for the vulnerabilities that are not mitigated yet, are then added. The result of this sum is the TVE. The goal of this paper is to plan the workload of an analyst in such a way that the TVE and TVR are reduced. Farris et al. use estimations on how many hours it will take to remediate the vulnerability. Since it is impossible to be sure of this, and the research focused on available time more than on the importance of the vulnerability, there is a need to search for different approaches. Because of this, our thesis can provide relevant insights into the field of cyber security.

TVR is closely related to the research of Chauhan and Pancholi [1]. They describe a theory on availability, which consists of the following concepts:

- **mean time-to-failure (MTTF):** This describes the mean time that a system will run before a failure occurs.

- **mean time-to-repair (MTTR):** This describes the main time that it takes to repair the system after a failure occurs. This is similar to the TVR from the research of Farris et al.

- **mean time-before-failure (MTBF):** This is the sum of MTTF and MTTR, which describes the mean time between two failures.
### 3.3 Risk assessment

An important aspect of our research question is making cyber security measurable. The book *How to measure anything in cybersecurity risk* by Hubbard and Seiersen [8] was studied, in hope to help us answer this part of our research question. This book focuses on making the assessment of cyber risks measurable. The main idea behind the methods proposed is to adjust the standard risk matrix that is commonly used. This matrix has likelihood and impact on the axes. Globally seen, a risk with a high impact and a high likelihood gets rated as a high risk and a risk with a low impact and a low likelihood gets rated as a low risk. In between, there is a category for medium risks for other combinations of impact and likelihood.

The book suggests changing the likelihood from a score from one to five, to an estimate of the chance in a percentage of an event happening in a chosen period of time. For example, a security expert would state that an event has a 10% chance of happening in the next 12 months. Instead of impact scored from one to five, the book focuses on monetized loss. The goal here is to estimate a 90% confidence interval for this loss. For example, an expert would estimate that, if the event occurs, there is a 90% chance that the monetized loss is between one and eight million dollars. Using a *Monte Carlo simulation* one can then calculate a curve that represents the expected loss of an organization based on the likelihood and impact of an event. Based on this expected loss, one can select mitigation for an event.

This book proposes a method to make risk more measurable. However, the underlying problem is still the same. The method completely relies on the estimates of an expert for each vulnerability. The goal of our research project is to propose an automated method based on scan data. Because of this, the proposed methods in the book do not answer our research question.
3.4 Related work and our research question

In our introduction we formulated our research problem as the following question:

**Can we develop a risk scoring method that helps a human expert select vulnerabilities for remediation?**

Above we saw that research has been done that is related to this research question. However, most of the discussed studies were aiming to solve another problem.

The research of Singh et al. [11] tells us which are the important parameters for the calculation method we propose.

The research of Wang et al. [13] is one of the building blocks of our proposed method. The distributions used there are useful for our optional methods when no other information on the vulnerabilities can be found.

Fruhwirth and Mannisto [6] aim to include the context of a vulnerability in the equation, which is similar to our goal. However, they use a different approach to do so.

Elbaz et al. [3] aimed to solve a problem that is outside of our scope, i.e. new vulnerabilities on which little information is known. However, their proposed method could be combined with our optional methods, which would allow us to add such new vulnerabilities to our scope.

Spring et al. [12] discuss the problem that we try to solve in this thesis. They conclude that it is a problem that should be solved, and suggest how the problem can be solved. One of their suggestions is researched in this thesis.

All in all, we see that some attempts to answer our research question have been made. However, the problem is still not solved and thus remains relevant. Because of this, our different approach is of added value in the search for a solution to the research problem.
Chapter 4

Methodology

In this chapter, we will elaborate on our methodology. This chapter is divided into three sections. The first section discusses how the optional methods are developed and describes the requirements an ideal method should meet. The second section describes the survey that was conducted and the results that the survey gave. After that, these results are used to propose our optimal approach.

4.1 Developing the methods

4.1.1 Data collection

Our data is collected using Rapid7’s Nexpose\(^1\), which is professional vulnerability management software. Ideally, one would combine an internal and an external scan for the most complete overview of existing vulnerabilities within an organization. However, we did not have access to such a data set. Because of this, the results of scans at two organizations have been combined, where one of them only had results of internal scans and the other organization only of an external scan. The organization at which the internal scans are done, is split up in 19 different sites. This is done because of the nature of the organization, which means that the 19 sites can be seen as separate organizations.

Anonymity was another reason to use a combined data set. Because the data set consists of data from two organizations, one is not able to link the organizations to the data. The organization that was scanned internally was separated into 19 sites for different scans. Every night, other sites were scanned, which resulted in the full scan after about a week. The external scan was performed at once. The results of the internal scans and the external scan were combined in our complete data set. The combined data set contains a total of 490 vulnerabilities, distributed over 675 endpoints.

\(^1\)https://www.rapid7.com/products/nexpose/
Where possible, we use the information on our parameters given by the tool. Since our method uses the results of a vulnerability scan as input, zero-day vulnerabilities are outside of the scope of this thesis. Kaspersky defines Zero-day vulnerabilities as ‘software vulnerabilities discovered by attackers before the vendor has become aware of it. Because the vendors are unaware, no patch exists for zero-day vulnerabilities, making attacks likely to succeed’. Thus little to no information is known about this kind of vulnerabilities.

However, Elbaz et al. [3] propose a method that aims to solve this problem. When using their method to establish a CVSS score of new vulnerabilities, our methods are applicable to zero-day vulnerabilities as well.

4.1.2 Parameters of our methods

Below the parameters that can help us develop our calculation methods are explained. Here we also elaborate on how we can use the parameters, i.e. how we can express the parameters. The concept ‘parameters’ here means the different blocks of information on a vulnerability that can be used to calculate the risk that the vulnerability offers.

CVSS as parameter

As with most of the risk and severity rating methods, our method uses CVSS as its core. Only the base metrics will be used, since most of the tools used for vulnerability scanning only use the base metrics. Information that is in the temporal metrics is added differently in our methods, as described below. The information that the environmental metrics add, are partially covered by the parameters described also below.

Ideally, we use CVSS version 3. However, this version is not supported for all vulnerabilities yet. When version 3 is not available, we fall back on version 2. The number that we get for this parameter, a number between 0 and 10, indicates the technical severity of the vulnerability that is found. Our other parameters adjust the risk score to the moment in time that the vulnerability is present and how the vulnerability is present in the organization.

Time as parameter

To calculate the two parameters as described below, we need the time since the vulnerability had been disclosed for the first time as input. Our vulnerability scanning tool gives us the date that a vulnerability was disclosed, with which we can calculate the age in days.

---

2https://www.kaspersky.com/resource-center/definitions/zero-day-exploit
Exploit existence as parameter

Vulnerabilities for which an exploit exists, are a greater risk to the organization. Exploits can be used by malicious actors to exploit the vulnerability to their advantage. For one of our methods, additional data on the existence of an exploit was gathered. The following sources are used:

- [https://www.exploit-db.com/](https://www.exploit-db.com/): A search query was run, using a web request, to find out whether an exploit exists.

- [https://vuldb.com/](https://vuldb.com/): Using the API, information on the existence of an exploit and fix for the vulnerability was gathered.

- [https://www.ncsc.nl/actueel/beveiligingsadviezen](https://www.ncsc.nl/actueel/beveiligingsadviezen): Using web requests, a search query was run to see if an advisory for the vulnerability exists. Such an advisory contains information on the existence of exploits and fixes and whether existing exploits are used in the real world.

- [https://www.circl.lu/](https://www.circl.lu/): Using the API, information on the existence of an exploit for the vulnerability was gathered.

However, when no information is offered by any of the sources, there is some uncertainty. One can not know for sure that no exploit exists. To solve this problem of uncertainty, we can estimate the likelihood that an exploit exists. As seen in section 3.1, Wang et al. use the Pareto distribution to do so. This technique was first discovered by Frei et al. [5]. They analyzed 80,000 vulnerabilities and found that we can estimate the exploit level with

$$F(x) = 1 - \left( \frac{0.00161}{x} \right)^{0.260}$$  \hspace{1cm} (4.1)

where $x$ is the time in days since the disclosure of the vulnerability. This will result in a number equal to or smaller than 1, which represents the likelihood that an exploit exists.

Remedy existence as parameter

For the vulnerabilities in our data set, we can do the same for remedy existence as for exploit existence. The web sources listed above, give us information on the availability of a remedy. However, for some vulnerabilities, it is unknown whether a remedy already exists. Because of this, we need to estimate the remediation level as we did above for the exploitation level. Again we can use the research of Frei et al. [5] to do so. For the remediation level, they found that an estimate can be calculated using a Weibull distribution of the form

$$F(x) = 1 - \exp \left( -\frac{x}{0.209} \right)^{4.040}$$  \hspace{1cm} (4.2)
where again $x$ is the time in days since the disclosure of the vulnerability. The result will represent the likelihood that a remedy for a vulnerability exists.

**Number of endpoints as parameter**

The results of a vulnerability scan tell us the number of endpoints a given vulnerability is found on. We want to take this number into account. A vulnerability that is found on multiple devices offers a greater risk than a vulnerability that is only present at one device, assuming the devices are equally critical.

**Reachability as parameter**

We want to distinguish between hosts that are externally reachable and hosts that are only reachable internally. This is because a vulnerability on an externally reachable host is more likely to be exploited, and thus more likely to cause harm in any way. Vulnerabilities on externally reachable hosts put the organization at higher risks.

**Location in network as parameter**

Lastly, we want to distinguish hosts based on their importance to the organization. To keep this as objective as possible, without creating too much overhead, we want to distinguish between network segments. To this end, it is necessary to inventory how the network of a client is split up. Organizations can use a demilitarized zone (DMZ) and a local area network (LAN) for example. A vulnerability on the DMZ might put the organization at greater risk, one might argue, since it can function as an entry point. Depending on the organization, we can use a weight in our calculation for these different network segments. However, this data was not available in our data set and it is hard to gather such data in an automated way. Because of this, this parameter is not used in this thesis, but can be of interest for future work.

### 4.1.3 Objectives

In this section, we list the objectives based on the parameters listed above, that an ideal method would achieve. The objectives are ordered on the expected priority. The results that we get with our different methods should meet these objectives.

- **Ranking based on CVSS:** CVSS is the basis of our method. It indicates the technical severity of a vulnerability. Vulnerabilities with a
high or critical severity should mostly have a higher priority than vulnerabilites with lower severity. When two vulnerabilities have severity scores that are close to each other, other aspects should determine which should be prioritized.

- **Ranking based on exploit existence:** When vulnerabilities seem to be equally severe according to their CVSS score, the existence of an exploit for one of the vulnerabilities can be decisive. However, a vulnerability that is not severe but for which an exploit exists, should not be seen as more important than a vulnerability that is severe for which no exploit is known.

- **Ranking based on reachability:** Whether a vulnerability is reachable externally influences the likelihood of the vulnerability being exploited by an attacker. Next to that, a vulnerability that is accessible via the internet is more critical because it can function as an entry to the network of the organization. Ideally, externally reachable vulnerabilities result in a higher risk score, without shoving down too important internally reachable vulnerabilities.

- **Ranking based on remedy existence:** Another parameter that can be decisive, is the existence of a remedy for a vulnerability. When a vulnerability is critical, it is not relevant whether a remedy exists or one can only turn off the service on which the vulnerability was found. Action should be taken directly in both cases. In case two equally low severity vulnerabilities are found, one could argue that the one with an official fix available has the highest priority. The reason for this is that a vulnerability that is not severe and for which no remedy exists is not worth the effort it will cost to set up a workaround or to turn off the service completely. Then it is more useful to use that effort later to remedy the vulnerability when an official fix exists.

- **Ranking based on the number of endpoints:** If a vulnerability was found on multiple devices in the digital infrastructure, chances might be higher that someone malicious finds and exploits the vulnerability. Because of this, the number of endpoints has a positive relation with the score we want to calculate. However, a vulnerability with a low CVSS score that is present at lots of endpoints, should not overshadow a vulnerability that was found on only one device, but with a critical CVSS score.

4.1.4 Optional methods

Now that we have described all the building blocks that we are going to use, we can develop our methods to calculate the vulnerability that should be
remedied first. We used the data set that was gathered as described above to do so.

The different parameters are isolated in different methods, so that we can determine the effect of the separate parameters. This is done to be able to determine which of the different approaches is most suitable. When using a combined approach before surveying which of the parameters is most suitable, it could become unclear which of the parameters of a combined approach offers the actual improvement. Based on the results of the methods, and the opinion of the experts on the results, we can recommend the most important parameters that should be combined into one method.

In section 4.1.3 we discussed that we wanted to keep the increase of the risk scores by our methods as little as possible, to make sure that vulnerabilities with a high technical severity are seen as a priority by our methods. This is why we decided to maximize the increase at twenty percent. An increase of ten percent was undesirable. According to CVSS version 3, scores in the range from 9.0 to 10.0 are seen as critical. We want our methods to be able to prioritize each of the vulnerabilities in this critical score range. For example, when a vulnerability that is assigned a technical severity score of 9.0, is easily exploitable, externally reachable or was found on a lot of endpoints, we want it to be possibly prioritized over a vulnerability that is assigned a technical severity score of 10.0, and is not (yet) easily exploitable, only internally reachable and was found on only one endpoint. For this to be possible, an increase of ten percent is not enough. Because of this, we chose an increase of twenty percent. We are aware that it is most likely that this percentage is not optimal for the methods. However, to decide on the best parameters for our method, it is suitable. Future research on a bigger data set is needed to establish the optimal percentage. This is out of the scope of this thesis.

A criterium of our methods was that the methods should be functional in practice in an automated manner. The optional methods can all be implemented in any programming language to automate the process.

To compare the different approaches, we used a ranking based on only CVSS as our baseline. A selection of vulnerabilities that was used to compare the different methods, can be found in table 1 of appendix A. This selection was made in such a way that the vulnerabilities that are compared are assigned a high or critical technical severity. This way, we are able to demonstrate the improvements that an optional method results in. The main problem that is faced in the process of vulnerability management, is that the number of high or critical scored vulnerabilities is overwhelming. Because of this, we especially used high- or critical-scored vulnerabilities in our selection.

Since we want to research to what extend a sorted list of vulnerabilities is ordered in the way that experts would remedy, adding lower-scored vulnerabilities to the selection is of no use. Those vulnerabilities will never get
selected as most important to remedy and will always be at the bottom of the list that is the result of one of the optional methods. Doing so would result in all optional methods being equally accurate.

Another criterion for selecting the vulnerability was that the vulnerabilities differ from each other enough to form an opinion of the remediation order. Multiple similar vulnerabilities with the same score are not relevant for our selection, since the new score from one of the optional methods will still result in the similar vulnerabilities being scored equally high.

In the analysis of the method that we suggest at the end of this chapter, the complete data set is used.

Reachability-based

Our first option for a calculation method is based on the reachability of the vulnerability. A vulnerability that is externally reachable offers more risk to the organization. Vulnerabilities that are only reachable internally, are less of a risk to organizations, since an attacker would need to have access already.

Our scanning tool provides the CVSS scores for the found vulnerabilities when available. In our calculations, we aimed to use version 3. However, CVSS version 3 is not available for all vulnerabilities. In such a case, we used CVSS version 2, since the older version was available for all the vulnerabilities in our data set. Both of the versions of CVSS were calculated only based on the base metrics.

Since an internal and external vulnerability scan was done, we can easily distinguish between internally and externally reachable. We used this information to alter the scores of the externally reachable vulnerabilities. The internally reachable vulnerabilities were scored by using the available CVSS version only. The scores of the externally reachable vulnerabilities were increased by twenty percent and calculated with the formula:

\[
\text{Score}_{\text{reachability-based}} = \text{CVSS} \times 1.2
\]  

(4.3)

With this formula, the externally reachable vulnerabilities were assigned a higher score compared to the internally reachable vulnerabilities. This means that the order in which the vulnerabilities are prioritized is altered in such a way that externally reachable vulnerabilities are placed higher on the list. This way, internally reachable vulnerabilities that have a high technical severity, are still also on top of the priority list. The results of this calculation can be found in table 2 of appendix A.

Endpoints-based

Secondly, we calculate the order using the number of endpoints a vulnerability was found on. A vulnerability on multiple endpoints is more likely
to be found and exploited than a vulnerability that is only present on one endpoint. Because of this, vulnerabilities found on more endpoints should be placed higher on the priority list.

The tool we used for the vulnerability scans offers information on the endpoint a vulnerability was found, such as the MAC address and the IP address. Using this information, we can count the number of endpoints a vulnerability was found on.

When we know for each vulnerability on how many endpoints it was found, we can calculate the score based on this. To do so, we again used a maximum of twenty percent increase. The score of the vulnerability on the most endpoints was increased by this percentage. Scores of vulnerabilities that were found on only one endpoint, were not increased. Other vulnerabilities had their score increased with a percentage that was in proportion with the maximum number of endpoints a vulnerability was found on.

For example, if in our data set the vulnerability on the most endpoints was found on hundred of endpoints, the score of this vulnerability would increase by twenty percent. Another vulnerability that was found on fifty endpoints would then have its score increased by ten percent. This is written in the formula:

\[
Score_{\text{endpoints-based}} = CVSS \times (1 + \frac{\text{number of endpoints} \times 0.2}{\text{max number of endpoints}})
\] (4.4)

where \text{number of endpoints} is the number of endpoints the vulnerability, for which we want to calculate the score, was on. \text{max number of endpoints} is the number of endpoints that the vulnerability that was on the most endpoints was found on.

The results of using this calculation method on the selection of vulnerabilities can be found in table 3 of appendix A.

**Exploit-based**

Our third and last method is mainly based on the existence of an exploit for the vulnerability. Next to that, the availability of a remedy is also taken into account. A vulnerability for which an exploit is widely available offers a greater risk to organizations than a vulnerability for which no exploit is known to be available. Because of this, vulnerabilities for which exploits are published should ideally be remedied before vulnerabilities for which no exploits are known. One could argue that vulnerabilities for which an official patch is released are more interesting to pick up than vulnerabilities for which no remedy or fix exists. Something that can not be patched at the moment, might not be worth the time investment to try to do something about it, albeit a workaround or turning off the system.

Our scanning tool provides us with the Common Vulnerabilities and Exposures (CVE) ID number. This is the identifier for the found vulnerability.
Using this number, one can search online sources for additional information on the vulnerability. The vulnerability scanning software also provides us with the date that the vulnerability was first disclosed. For our calculations, this date is used to calculate the age of the vulnerability. The process of searching the databases was automated using a Python script, which can be found via the URL shown below.

This method checks the CVE number first. In some cases, there is no CVE number known for the vulnerability. This can be because the problem found it rather a misconfiguration than an actual vulnerability. In this case, an actual exploit is hard to develop.

However, the age of the vulnerability can still give relevant information about the risk that the situation offers. To estimate this risk, we use the equation (4.1). This will give us a number smaller than or equal to one, which represents the chance that an exploit exists, given the age of the vulnerability. The older the vulnerability, the bigger the chance that an exploit exists. We then multiply the result of this equation with the CVSS score that the scanning tool gave us. Since the result of the equation is equal to or smaller than one, the score will be equal to or lower than the CVSS score. The same was done for the availability of a remedy. Equation (4.2) was used for this, instead of the equation for exploit existence.

However, in most cases, a CVE number is known. Using that number we search the sources mentioned in section 4.1.2. Firstly, we search https://www.exploit-db.com/, which is one of the biggest and most commonly known exploit databases. If we find an exploit here, we multiply the CVSS score by 1.2, to apply the increase of twenty percent to the risk score.

In the case that the search above gives no results, the security advisories database from the Dutch cyber security center (NCSC) is used. This database can be found at https://www.ncsc.nl/actueel/beveiligingsadviezen. In this database, one can search for a CVE number. The result consists of, among other information, parameters and their values that tell relevant information about the vulnerability. Those parameters all have a value assigned. For example, if a fully functional exploit is known, the value 6 is assigned to that parameter. If only a proof of concept is known, a 4 is assigned. The value is 1 if there is no exploit or proof of concept known.

We use the parameters on the existence of an exploit, for which the maximum assigned value is 6, whether the exploit has been seen in the real world, for which is maximum assigned value is 3, and the existence of a fix, for which is the maximum assigned value is 3. The sum of those three maximum values is 12. If this is the case for a vulnerability, we increase the CVSS score with twenty percent to get the risk score. For results lower than 12, the correct multiplier that is in proportion with the maximum value is used. Note that the minimum of the multiplier is 1, so that the score will never get lowered.

In some cases, both of the databases above did not give any results
when searching for the CVE number. In that case, the database of [https://vuldb.com/](https://vuldb.com/) is searched. This database contains information on the level of existence of exploits and remedies as well. Similarly to the calculation for the Dutch cyber security center, a sum of those values is taken. For this database, however, the maximum result of this sum is 9. When the result is 9, we again multiply with 1.2, to get the twenty percent increase. For results lower than 9, the correct multiplier that is in proportion with the maximum value is used.

Lastly, the database of the Computer incident response center Luxembourg, at [https://www.circl.lu/](https://www.circl.lu/) was searched, if none of the other sources gave results. This database was used last, because the least information on exploits and remedies was given by this database. Most of the information that could be found there was about the CVSS scores of the vulnerability, which we already had from our scanning tool. In our vulnerability data set, none of the vulnerabilities had a CVE number with no results in one of the three other databases. Thus, this database is not used in our calculations.

In the case that a CVE-ID is known, but none of these open sources offer any information on the vulnerability, an estimate could be calculated using the equations of Frei et al. [5] In this case, the calculation is similar to when there is no CVE number.

The calculation as described above are summarized in Figure 4.1 below.

![Figure 4.1: Diagram of the exploit-based metric](image)

Our method consults the different sources in the order as described above. When a source results in the information that is desired, our method does not consult the other sources anymore. This way, only one source is used for each vulnerability. Combining those different sources as much as possible could improve this method, and might be a relevant topic for future research.

The results of using this calculation method on the selection of vulnerabilities can be found in table 4 of appendix A.
4.2 Validating our optional methods

Now that we have our three approaches applied on the dataset of our company, we want to know which of the methods is most suitable for selecting the best next action. To this end, we conducted a survey to cyber security experts.

4.2.1 Survey

The survey that was conducted consists of five substantive questions, next to a question on their name and a question to ask for any remarks.

Firstly, we wanted to see whether the answers between offensive and defensive orientated persons did differ significantly. To this end, we asked for the department the respondent was in or which matches their quality and mindset the most. The options here were blue team and red team.

After that, the main question of this survey was presented. The respondents were asked to rate all of the four options that can be found in appendix A. Each of the four options, CVSS-only, reachability-based, endpoints-based and exploit-based, could be rated from one to ten, where one is worst and ten is best. With the answers to this question, we can establish which of the methods is most useful according to the experts and how the methods relate to each other in terms of usability. This question was followed by asking for a short motivation for the choice of the best method.

Lastly, a question about the parameters was presented. We asked to select a maximum of three most relevant parameters out of the seven parameters presented in section 4.1.2. After that, the respondents could enter any important parameters that were missing in the list in their opinion.

The complete survey can be found in appendix B.

4.2.2 Hypotheses

After creating and before sending out the survey, hypotheses were developed. The expectation was that the priority order, together with the vulnerability scores, that was calculated based on the existence of exploits and remedies would be the most useful in the opinion of the experts. This option did result in the highest number of different scores, which we expected to be seen as desirable. The baseline that only took CVSS into account was expected to achieve the lowest score on average, because most of the vulnerabilities are scored the maximum score here. Thus it is unclear which vulnerability should be remedied first when this calculation method is used.

We expected the respondents to answer that the most important parameters are reachability and exploit existence. Those parameters are the parameters that indicate best that the vulnerability might be used and exploited to perform unwanted actions on the systems containing the vulnerability.
Our hypotheses can be summarized as follows:

**H1:** The remedy order calculated based on exploit and remedy existence is deemed most accurate.

**H2:** The remedy order calculated based on CVSS is deemed least accurate.

**H3:** Reachability and exploit existence are seen as the most important parameters.

### 4.2.3 Results of the survey

The survey was filled in by 24 people, all working in the field of cyber security. Some of the open-ended questions resulted in useful insights, in which case the respondent was contacted for further explanation.

**Teams**

As can be seen in figure 4.2, were most of the respondents a member of a blue team. A percentage of 62.5 translates to 15 respondents. 9 respondents, which is 37.5 percent of the population, did affiliate with a red team.

![Figure 4.2: Team division among the respondents](image-url)
Options

In figure 1 of appendix C we see the grades (scores) that were given for each of the priority orders and their scores calculated with the different calculation methods. We divide the scores in four different categories:

- **Insufficient**: scores 1 - 4
- **Acceptable**: scores 5 - 6
- **Good**: scores 7 - 8
- **Very good**: scores 9 - 10

In figure 4.3 below, we see a visualization of how the experts rated the different optional approaches.

![Figure 4.3: Options and their categorized ratings from experts](image)

In this figure, as well as in table 4.2, we can see that the calculation based on the existence of exploits and the calculation based on the reachability were deemed more useful than standard CVSS scores and the calculation based on the number of endpoints. Looking back, we see that hypothesis H1 is correct. The method that is based on the existence of an exploit or remedy was deemed the most accurate, as expected. Next to that, we
see that the list ordered on CVSS was rated as least accurate, which was expected in hypothesis H2.

Next to that, we see that this difference is a little bigger in the results from red team members than in the results from blue team members.

<table>
<thead>
<tr>
<th></th>
<th>Endpoints-based</th>
<th>CVSS-only</th>
<th>Exploit-based</th>
<th>Reachability-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>4.3</td>
<td>4.2</td>
<td>7.7</td>
<td>6.7</td>
</tr>
<tr>
<td>Median</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Average redteam</td>
<td>3.8</td>
<td>3.6</td>
<td>7.9</td>
<td>6.6</td>
</tr>
<tr>
<td>Average blueteam</td>
<td>4.4</td>
<td>4.4</td>
<td>7.1</td>
<td>6.5</td>
</tr>
<tr>
<td>Median redteam</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Median blueteam</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 4.2: Team division in the population

After rating the four options, the respondents were asked to motivate their answers. An important reason to select the reachability-based calculation as a good option was the fact that an externally reachable vulnerability is more likely to be found by undirected attacks, i.e. attacks that are not aimed at a specific target but rather scan the internet for vulnerable targets. Reasons to select the exploit-based calculation as a good option were the differences in the scores, and the fact that it is easier to use a vulnerability when an exploit already exists.

However, out of the 21 respondents that responded to this question, 11, which equals 52.4 percent, indicated that a combination between reachability-based and exploit-based would be their ideal option. Such a method would combine the two best options and their advantages.

Parameters

Lastly, the respondents were asked to select the most important parameters. A maximum of three options out of the list given in section 4.1 could be chosen. After that, extra parameters could be added with an open-ended question.

In figure 4.4 we can see that the parameters CVSS score, reachability and the existence of an exploit are ranked highest by the experts. This corresponds with the answers to the previous question, where the calculations based on these parameters were seen as most useful and accurate. Based on these results, we see that hypothesis H3 was partly correct. The existence of an exploit and reachability are seen as two of the three most important
parameters. However, the fact that CVSS was rated as such an important parameter was not expected in our hypotheses.

The answers to the open-ended question on missing parameters indicate that there is a need for information on the criticality of a vulnerability being exploited in the application that it is on, information on the device that application is on and information on the subnet that the device is in. This is the broader version of the parameter that we describe in section 4.1.2. As discussed in that section, we do not use that parameter in our calculations, because this parameter would require human interaction and subjectivity.

![Figure 4.4: The most important parameters](image)

Legend of figure 4.4:

1. CVSS score
2. Whether the system containing the vulnerability is externally or internally reachable
3. The existence of an exploit
4. In which subnet the system containing the vulnerability is located
5. The existence of a remedy
6. The number of endpoints the vulnerability was detected on
7. The age of the vulnerability
4.3 Optimal calculation method

We saw in the previous section that calculation methods based on the reachability and the existence of exploits are seen as most suitable and accurate. We altered our methods in such a way that a combination of those two methods was used on our data set. 52.4 percent of the experts commented that such a combined approach would result in the best remediation priority list, in their opinion.

Both of the methods rely on a multiplication on the technical severity score, i.e. CVSS. Because of the commutative property of multiplication, the order in which the two methods are applied does not influence the result. We chose to apply the method based on reachability first. In figure 4.5 is shown which steps are performed in our combined approach.

The result of this can be found in table 5 of appendix D. We can see here that the only externally reachable vulnerability in the list is now at the top of the list. The rest of the list remains unchanged compared to the list calculated based on the existence of an exploit for the vulnerability.

The results were calculated by first doing the calculation as described in section on the exploit-based method. The results of the external vulnerability scan were given a flag, by which we could later only alter the externally reachable vulnerabilities. This was done using the method described in section on the reachability-based method.

Using CVSS only, 69 of the 282 vulnerabilities that were found were scored as high with CVSS version 2. This is equal to 24.5 percent of the vulnerabilities. High means a score from 7.0 to 10.0 for this version, i.e. a range of three points. We compare this to the results of our proposed method. Our proposed method results in 12 out of the 282 vulnerabilities that are scored in the top range of three points. This equals 4.3 percent of the total amount of vulnerabilities. Our method was able to bring down this percentage by 82.4 percent.

For our new proposed method, the maximum score is 14.44. The maximum input CVSS score is 10. When the vulnerability is externally reachable, the score of such a vulnerability is increased by twenty percent, resulting in 12. If for this vulnerability then, a fully functional exploit exists, the score 12 is again increased by twenty percent, resulting in the maximum score of 14.44. It is possible to convert all the scores to a scale where 10 is the maximum, as is the case with CVSS. We can do this for our proposed optimal method, as well as for the optional methods that we applied on the data set before. This can be done by multiplying each score by 10 and then dividing the result of that multiplication by the maximum possible score. In our research we chose not to do so, to illustrate the effect that the methods have on the scoring as clearly as possible.

Using CVSS only, 23 out of 282 vulnerabilities were assigned the maximum score of 10, i.e. 8.2 percent. In the results of our proposed method,
none of the vulnerabilities was assigned the maximum score of 14.44. Here we see that our proposed method is able to improve this percentage by 100 percent.

The code of the script that was executed to perform the calculation, that has been visualized in figure 4.5, can be found at the GitHub repository at https://github.com/richardvanginkel/MasterThesisCyberSecurity.

Figure 4.5: Diagram of our proposed optimal method
Chapter 5

Conclusions

This research aims to improve the remediation prioritization of the results of a vulnerability scan. Based on a qualitative analysis of our proposed methods, we found two parameters that can be included in risk score calculation to improve the accuracy significantly. These parameters are reachability and exploit existence. The additional information on a vulnerability that these parameters offer, allows differentiating between the vast amount of vulnerabilities that are scored high or critical. Our proposed method offers a reduction of 82.4 percent of the number of vulnerabilities assigned a high severity score by CVSS. A reduction of one hundred percent of the number of vulnerabilities that are assigned the maximum score was seen.

We developed three risk score calculation methods, that are based on CVSS combined with one or two of the optional parameters. Doing so, rather than combining multiple parameters in one method, we are able to research the added value of each used parameter separately. However, this results in the situation where our ideal method, which uses reachability and exploit existence as parameters, is not included in the survey. This approach, where the two parameters are combined, was widely advised in the survey, which indicated that such an approach is of added value. Furthermore, the selection of the best calculation method by the respondents, as well as the selection of the most relevant parameters, points in this same direction.

The research of Fruhwirth and Mannisto [6] used similar parameters in a different manner, and came to the conclusion that the parameters are useful additions. Their method uses the distributions based on vulnerability data from 2001 to 2005 from the research of Frei et al. [5]. Fruhwirth and Mannisto show that the distributions are of added value. That is why we use it as a fallback in the case that the open sources that we consult offer no additional information on a vulnerability. Using open-source intelligence such as the four online databases consulted in our method, has not been done before. This thesis shows that the addition of such open-source intelligence is of added value to the process of prioritizing on which vulnerability to
remedy first. This makes that our approach is new and relevant to the cyber security field.

Further research is needed to optimize our proposed approach. Testing the method on a bigger data set could help to establish the optimal percentages by which we increase the risk scores in the calculation. Furthermore, additional research can be done to find extra useful parameters. For example, as discussed in this thesis, the subnet in the network in which the vulnerability is present, could offer relevant additional information based on which the calculation can be done. Next to that, combining different online databases for each vulnerability, instead of applying the information of the first result only, could possibly improve our proposed method.

Our project offers a better understanding of the relevant aspects that should be taken into account when selecting the most important vulnerabilities to remedy. Moreover, our proposed method, potentially optimized by others, can help to make the process of selecting the most important vulnerabilities to remedy less of an effort. The proposed method can be used in any type of organization, on which a vulnerability scan is performed. Our method can be applied to the results of a vulnerability scan, which are in the form of a list of vulnerabilities, ordered on their new risk score. Based on this list, the most relevant remedies can be selected.
Bibliography


Appendices
## Appendix A

<table>
<thead>
<tr>
<th>Title</th>
<th>Endpoints</th>
<th>#</th>
<th>Remedy</th>
<th>Reachability</th>
<th>Score</th>
<th>Age (years)</th>
<th>Exploit</th>
<th>Remedy</th>
<th>Remedy</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obsolete Version of Ubuntu</td>
<td>2</td>
<td></td>
<td></td>
<td>Internally</td>
<td>10.0</td>
<td>8.3</td>
<td>None</td>
<td>No</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Inconclusive host with excessive port connection failures</td>
<td>1</td>
<td></td>
<td></td>
<td>Internally</td>
<td>10.0</td>
<td>7.2</td>
<td>None</td>
<td>No</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Undocumented Default Account: diag</td>
<td>1</td>
<td></td>
<td></td>
<td>Internally</td>
<td>10.0</td>
<td>16</td>
<td>None</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VMSA-2020-00023: ESI OpenSSL remote code execution</td>
<td>4</td>
<td></td>
<td></td>
<td>Internally</td>
<td>10.0</td>
<td>1</td>
<td>None</td>
<td>No</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>WMSA-2020-00023: ESI OpenSSP remote code execution</td>
<td>9</td>
<td></td>
<td></td>
<td>Internally</td>
<td>10.0</td>
<td>21.5</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Default Telnet password: admin password &quot;password&quot;</td>
<td>1</td>
<td></td>
<td></td>
<td>Internally</td>
<td>10.0</td>
<td>6.6</td>
<td>Full</td>
<td>Full</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Default or Guessable SNMP community names: private</td>
<td>1</td>
<td></td>
<td></td>
<td>Internally</td>
<td>10.0</td>
<td>5.6</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Allegro Software RomPager. Fortunecookie. Unspeciffed HTTP</td>
<td>1</td>
<td></td>
<td></td>
<td>Internally</td>
<td>10.0</td>
<td>6.6</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Allegro Software RomPager Uns beet buffer overflows in screenOS</td>
<td>1</td>
<td></td>
<td></td>
<td>Internally</td>
<td>10.0</td>
<td>6.6</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Allegro Software RomPager Uns specified buffer overflows in screenOS</td>
<td>1</td>
<td></td>
<td></td>
<td>Internally</td>
<td>10.0</td>
<td>6.6</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Default of Guessable SNMP community names: public</td>
<td>80</td>
<td></td>
<td></td>
<td>Internally</td>
<td>10.0</td>
<td>23</td>
<td>None</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Default of Guessable SNMP community names: private</td>
<td>1</td>
<td></td>
<td></td>
<td>Internally</td>
<td>10.0</td>
<td>1.4</td>
<td>None</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SAP Solution Manager: Missing Authentication Check (User-Experience Monitoring)</td>
<td>5</td>
<td></td>
<td></td>
<td>Internally</td>
<td>9.3</td>
<td>1.2</td>
<td>None</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HP ILO: Code Execution, denial of service</td>
<td>1</td>
<td></td>
<td></td>
<td>Internally</td>
<td>9.3</td>
<td>2.3</td>
<td>None</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Apache Tomcat: Important: Remote Code Execution on Windows</td>
<td>1</td>
<td></td>
<td></td>
<td>Internally</td>
<td>9.3</td>
<td>4.4</td>
<td>None</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Microsoft Windows SMB Remote Code Execution Vulnerability</td>
<td>1</td>
<td></td>
<td></td>
<td>Internally</td>
<td>9.0</td>
<td>3.7</td>
<td>None</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Intel Manageability: Engine: Remote Admin Access Arbitrary Code Execution</td>
<td>1</td>
<td></td>
<td></td>
<td>Internally</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 1:** Order of selection of vulnerabilities based on CVSS
<table>
<thead>
<tr>
<th>Title</th>
<th>Score</th>
<th>Age (years)</th>
<th>Exploit</th>
<th>Remedy</th>
<th>Reachability</th>
<th># Endpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP Solution Manager: Missing Authentication Check (User-Experience Monitoring)</td>
<td>12.0</td>
<td>1.4</td>
<td>None</td>
<td>Yes</td>
<td>Externally</td>
<td>1</td>
</tr>
<tr>
<td>Obsolete Version of Ubuntu</td>
<td>10.0</td>
<td>8.3</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>2</td>
</tr>
<tr>
<td>Inconclusive host with excessive port connection failures</td>
<td>10.0</td>
<td>7.2</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Undocumented Default Account: diag</td>
<td>10.0</td>
<td>16</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>VMSA-2020-0023: ESXi OpenSLP remote code execution vulnerability</td>
<td>10.0</td>
<td>1</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>4</td>
</tr>
<tr>
<td>Default Telnet password: admin password &quot;password&quot;</td>
<td>10.0</td>
<td>10</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Default or Guessable SNMP community names: private</td>
<td>10.0</td>
<td>21.5</td>
<td>PoC</td>
<td>Yes</td>
<td>Internally</td>
<td>9</td>
</tr>
<tr>
<td>Allegro Software RomPager 'Fortune Cookie' Unspecified HTTP Authentication Bypass</td>
<td>10.0</td>
<td>6.6</td>
<td>Full</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Unauthorized remote administrative access to ScreenOS</td>
<td>10.0</td>
<td>5.6</td>
<td>Full</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Allegro Software RomPager Unspecified Buffer Overflows in HTTP Handling</td>
<td>10.0</td>
<td>6.6</td>
<td>Full</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Default or Guessable SNMP community names: public</td>
<td>10.0</td>
<td>23</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>80</td>
</tr>
<tr>
<td>HP iLO: Code Execution, Denial of Service</td>
<td>9.3</td>
<td>1.2</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>5</td>
</tr>
<tr>
<td>Apache Tomcat: Important: Remote Code Execution on Windows</td>
<td>9.3</td>
<td>2.3</td>
<td>Full</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Microsoft Windows SMB Remote Code Execution Vulnerability</td>
<td>9.3</td>
<td>4.4</td>
<td>Full</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Intel Manageability Engine: Remote Admin Access Arbitrary Code Execution</td>
<td>9.0</td>
<td>3.7</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Title</td>
<td>Score</td>
<td>Age (years)</td>
<td>Exploit</td>
<td>Remedy</td>
<td>Reachability</td>
<td># Endpoints</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------</td>
<td>-------------</td>
<td>---------</td>
<td>--------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Default or Guessable SNMP community names: public</td>
<td>10.0</td>
<td>2.3</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>80</td>
</tr>
<tr>
<td>Default or Guessable SNMP community names: private</td>
<td>10.1</td>
<td>21.5</td>
<td>POC</td>
<td>Yes</td>
<td>Internally</td>
<td>9</td>
</tr>
<tr>
<td>Obsolete Version of Ubuntu</td>
<td>10.0</td>
<td>8.3</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>2</td>
</tr>
<tr>
<td>Inconclusive host with excessive port connection failures</td>
<td>10.0</td>
<td>7.2</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Undocumented Default Account: diag</td>
<td>10.0</td>
<td>16</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>4</td>
</tr>
<tr>
<td>WSA-2020-0023: ESXi Openssl remote code execution vulnerability</td>
<td>10.0</td>
<td>1</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Default: Telnet password: &quot;password&quot;</td>
<td>10.0</td>
<td>10</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Allegro Software RomPager: &quot;Fortune Cookie&quot; Unspecified HTTP Authentication Bypass</td>
<td>10.0</td>
<td>6.6</td>
<td>Full</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Unauthorized remote administrative access to ScreenOS</td>
<td>10.0</td>
<td>5.6</td>
<td>Full</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Allegro Software RomPager: Unspecified Buffer Overflows in HTTP Handling</td>
<td>10.0</td>
<td>6.6</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>SAP Solution Manager: Missing Authentication Check (User-Experience Monitoring)</td>
<td>10.0</td>
<td>1.4</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>5</td>
</tr>
<tr>
<td>HP ILC: Code Execution, Denial of Service</td>
<td>9.3</td>
<td>1.2</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Apache Tomcat: Important, Remote Code Execution on Windows</td>
<td>9.3</td>
<td>4.4</td>
<td>Full</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>MS Windows SMB Remote Code Execution Vulnerability</td>
<td>9.0</td>
<td>3.7</td>
<td>Full</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Intel Manageability Engine: Remote Admin Access Arbitrary Code Execution</td>
<td>9.0</td>
<td>3.7</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 3:** Order of selection of vulnerabilities based on the number of endpoints
<table>
<thead>
<tr>
<th>Title</th>
<th>Score</th>
<th>Age (years)</th>
<th>Exploit</th>
<th>Remedy</th>
<th>Reachability</th>
<th># Endpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allegro Software RomPager ‘Fortune Cookie’ Unspecified HTTP Authentication Bypass</td>
<td>12.1</td>
<td>6.6</td>
<td>Full</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Allegro Software RomPager Unspecified Buffer Overflows in HTTP Handling</td>
<td>12.1</td>
<td>6.6</td>
<td>Full</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Unauthorized remote administrative access to ScreenOS</td>
<td>11.9</td>
<td>5.6</td>
<td>Full</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Default or Guessable SNMP community names: private</td>
<td>11.5</td>
<td>21.5</td>
<td>PoC</td>
<td>Yes</td>
<td>Internally</td>
<td>9</td>
</tr>
<tr>
<td>Apache Tomcat: Important: Remote Code Execution on Windows</td>
<td>11.2</td>
<td>2.3</td>
<td>Full</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Microsoft Windows SMB Remote Code Execution Vulnerability</td>
<td>11.2</td>
<td>4.4</td>
<td>Full</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>VMSA-2020-0023: ESXi OpenSLP remote code execution vulnerability</td>
<td>11.0</td>
<td>1</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>4</td>
</tr>
<tr>
<td>Default or Guessable SNMP community names: public</td>
<td>10.8</td>
<td>23</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>80</td>
</tr>
<tr>
<td>SAP Solution Manager: Missing Authentication Check (User-Experience Monitoring)</td>
<td>10.8</td>
<td>1.4</td>
<td>None</td>
<td>Yes</td>
<td>Externally</td>
<td>1</td>
</tr>
<tr>
<td>Default: Telnet password: admin password &quot;password&quot;</td>
<td>9.9</td>
<td>10</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Obsolete Version of Ubuntu</td>
<td>9.8</td>
<td>8.3</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>2</td>
</tr>
<tr>
<td>Inconclusive host with excessive port connection failures</td>
<td>9.8</td>
<td>7.2</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Intel Manageability Engine: Remote Admin Access Arbitrary Code Execution</td>
<td>9.8</td>
<td>3.7</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>HP iLC: Code Execution, Denial of Service</td>
<td>9.8</td>
<td>1.2</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>5</td>
</tr>
<tr>
<td>Undocumented Default Account: diag</td>
<td>9.8</td>
<td>16</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
</tbody>
</table>
Appendix B

Survey Thesis Richard: Scoring vulnerabilities

My thesis is about developing a system to score vulnerabilities. Based on such scoring methods one can select the next best action in improving cyber security. The purpose of this survey is to select the best scoring method based on the opinion of experts in the field.

1. Please enter your name below:
   All data will be completely anonymized. Your name will only be used to reach out to you when answers are unclear.

2. Please select your department. If no answer is applicable, pick which matches your qualities and mindset best.
   - Red team
   - Blue team

The fictitious company Southwave had a vulnerability scan run by one of its employees. The CISO is unsure which vulnerability should be remedied first and Southwave decided to reach out to you as an expert. The tool has given 4 different priority orders of remediating the vulnerabilities, as shown below. Please score under the options how well each option reflects the actual priorities of remediating the vulnerabilities, according to your expertise, i.e. how does it correspond to the order that you would pick up those vulnerabilities.

The results are ordered by their score. Please answer based on how the scores represent the severity of the vulnerability in your opinion. The fields 'Age', 'Exploit', 'Remedy', 'Reachability' and '# Endpoints' give you some context about the vulnerability in the organization. If you are not sure about some vulnerabilities and would like more information, please see the links at the end of the survey.

[Here the four tables that can be found in appendix A were placed in random order.]

3. For each of the severity score distributions, please rate how useful and accurate it is to select the best next action. (1 is worst, 10 is best)
   - Option A:
4. Please motivate your choice for the best option shortly.

5. Which of the following parameters are most important to you when determining the risk of a vulnerability? (Select no more than 3)
   - CVSS score
   - The number of endpoints the vulnerability was detected on
   - Whether the system containing the vulnerability is externally or internally reachable
   - The age of the vulnerability
   - The existence of an exploit
   - The existence of a remedy/fix
   - In which subnet the system containing the vulnerability is located

6. If you are missing (an) important parameter(s), please enter it/them below.

7. If you have any remarks, please write them below.
More information on the vulnerabilities, in random order:
- Default or Guessable SNMP community names: public:
- Obsolete Version of Ubuntu:
  https://www.rapid7.com/db/vulnerabilities/ubuntu-obsolete-version/
- HP iLO: Code Execution, Denial of Service:
  https://nvd.nist.gov/vuln/detail/CVE-2020-11896
- Intel Manageability Engine: Remote Admin Access Arbitrary Code Execution:
- SAP Solution Manager: Missing Authentication Check (User-Experience Monitoring):
  https://nvd.nist.gov/vuln/detail/CVE-2020-6207
- Inconclusive host with excessive port connection failures:
  https://www.rapid7.com/db/vulnerabilities/inconclusive-host/
- Microsoft Windows SMB Remote Code Execution Vulnerability:
- Default Telnet password: admin password "password":
  https://www.rapid7.com/db/vulnerabilities/telnet-default-account-admin-password-password/
- Unauthorized remote administrative access to ScreenOS:
- VMSA-2020-0023: ESXi OpenSLP remote code execution vulnerability:
  https://nvd.nist.gov/vuln/detail/CVE-2020-3992
- Allegro Software RomPager Unspecified Buffer Overflows in HTTP Handling:
  https://nvd.nist.gov/vuln/detail/CVE-2014-9223
- Default or Guessable SNMP community names: private:
  https://nvd.nist.gov/vuln/detail/CVE-2000-0147
- Undocumented Default Account: diag:
  https://www.rapid7.com/db/vulnerabilities/telnet-avaya-default-login-diag/
- Apache Tomcat: Important: Remote Code Execution on Windows:
- Allegro Software RomPager 'Fortune Cookie' Unspecified HTTP Authentication Bypass:
  https://nvd.nist.gov/vuln/detail/CVE-2014-9222
Appendix C

Figure 1: Options and their ratings from experts
Appendix D

<table>
<thead>
<tr>
<th>Title</th>
<th>Score</th>
<th>Age (years)</th>
<th>Exploit</th>
<th>Remedy</th>
<th>Reachability</th>
<th>Endpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP Solution Manager Missing Authentication Check (User Experience Monitoring)</td>
<td>13.0</td>
<td>1.4</td>
<td>None</td>
<td>Yes</td>
<td>Externally</td>
<td>1</td>
</tr>
<tr>
<td>Allegro Software RomPager ‘Fortune Cookie’ Unspecified</td>
<td>12.1</td>
<td>6.6</td>
<td>Full</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>HTTP Authentication Bypass</td>
<td>12.1</td>
<td>6.6</td>
<td>Full</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Allegro Software RomPager Unspecified Buffer Overflows in</td>
<td>11.9</td>
<td>5.6</td>
<td>Full</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>HTTP Handling</td>
<td>11.5</td>
<td>21.5</td>
<td>POC</td>
<td>Yes</td>
<td>Internally</td>
<td>9</td>
</tr>
<tr>
<td>Unauthorized remote administrative access to ScreenOS</td>
<td>11.2</td>
<td>2.3</td>
<td>Full</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Default or Guessable SNMP community names: private</td>
<td>11.2</td>
<td>4.4</td>
<td>Full</td>
<td>Yes</td>
<td>Internally</td>
<td>4</td>
</tr>
<tr>
<td>Apache Tomcat: Important: Remote Code Execution on Windows</td>
<td>11.0</td>
<td>1</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>80</td>
</tr>
<tr>
<td>Microsoft Windows SMB Remote Code Execution Vulnerability</td>
<td>10.8</td>
<td>23</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Default or Guessable SNMP community names: public</td>
<td>9.9</td>
<td>10</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Default Telnet: password: admin password “password”</td>
<td>9.8</td>
<td>83</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>2</td>
</tr>
<tr>
<td>Obsolete Version of Ubuntu</td>
<td>9.8</td>
<td>72</td>
<td>None</td>
<td>No</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Inconclusive host with excessive port connection failures</td>
<td>9.8</td>
<td>37</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
<tr>
<td>Intel Manageability Engine: Remote Admin Access Arbitrary Code Execution</td>
<td>9.8</td>
<td>12</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>5</td>
</tr>
<tr>
<td>HP iLO: Code Execution, Denial of Service</td>
<td>9.8</td>
<td>16</td>
<td>None</td>
<td>Yes</td>
<td>Internally</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5: Order of selection of vulnerabilities based on reachability and the existence of an exploit