Zurich University
of Applied Sciences

School of
Engineering
InIT Institute of Applied
Information Technology

Master of Science in Engineering – VT 2

A Scalable Security Control Management Framework

Author	Benjamin Gehring
Supervisor	Dr. Ariane Trammell
Date	31.01.2023

Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

DECLARATION OF ORIGINALITY
Project Work at the School of Engineering

By submitting this project work, the undersigned student confirm that this work is his/her own work and was written without the help of a third party. (Group works: the performance of the other group members are not considered as third party).

The student declares that all sources in the text (including Internet pages) and appendices have been correctly disclosed. This means that there has been no plagiarism, i.e. no sections of the project work have been partially or wholly taken from other texts and represented as the student's own work or included without being correctly referenced.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

City, Date:	**Name Student:**
Schaffhausen, 31.01.2023	Benjamin Gehring

Abstract

Taking the correct security measures for a company's IT infrastructure is a complex and time-consuming process. However, correctly selecting appropriate measures is crucial and can significantly reduce the risk of a successful cyberattack. To support those responsible, various standardization organizations have catalogs of measures in the form of security controls from which the appropriate one can be selected. However, even these summary catalogs are long and include many different controls from which the appropriate ones must be found.

This work is about developing an application that automates this process. The project goal is to find the appropriate security controls for the entire system based on definable requirements. For this purpose, a requirements analysis was used to define the necessary functionalities, and a literature search was conducted to find similar approaches. The information gained was then used to develop and implement a concept that is intended to achieve the set goals.

The product of this work is the Security Control Management (SCM) software, which identifies and assigns appropriate security controls to assets (software, clients, servers, etc.) specified by the user. The analysis is done with NLP techniques, like keyword extraction and clustering, to establish relationships between assets and controls. The developed structure of the application allows to identify similarities and categorize them accordingly. The implementation was then tested by employing a validation process, and the results were checked for their usefulness. It was shown that the software is able to automatically find and correctly assign matching controls based on defined assets.

Contents

Chapter 1

Introduction

Security in a company can encompass an incredible number of aspects. A wide variety of things have to be taken into account and covered to actually be able to operate a system that is secure to a certain degree. It can quickly happen that one loses the overview and does not cover essential requirements. A survey[1] by the Enterprise Strategy Group (ESG) has shown that many companies have problems with security hygiene and the management of security assets. Many different tools are used to manage everything, and sometimes even spreadsheets are used. To facilitate this process, various standardization bodies offer assistance in the form of security controls. However, even these catalogs are already very extensive and include many different controls. Selecting the proper controls for the own company can also be challenging.

This work aims to develop a tool that automates this process. Using natural language processing techniques from the field of artificial intelligence, it should be possible to recognize which controls fit a system automatically. The user should have the possibility to record and describe his IT components in the company. With this data, the application tries to assign reasonable controls. In order to implement this, a requirements analysis is performed as a first step in this thesis. Based on this, a concept for such an application will be developed and implemented. Finally, the implemented solution will be tested for its effectiveness using test data.

1.1 Project Goals

The goal of this work is to develop a tool for automated detection of appropriate security controls for an enterprise. The user should be offered the possibility to record the current state of his system and further specify the individual components regarding their composition, criticality, and other properties. This results in the following goals for this work:

1. Development of a tool for acquiring the current system state.
2. Automated derivation of appropriate security controls from the specified system state.

1.2 Project Scope

The project scope defines conditions that must be fulfilled during this project.

- This work is limited to using one security control standard, the NIST SP 800. Other standards are not considered.
- This work is a PoC. It contains a web interface, but not all functionalities are implemented. The focus is clearly on the complete implementation of the algorithms and their evaluation.

1.3 Document Structure

This work begins with *Chapter 2*, which explains the current and most common security control standards, algorithms used for matching, and other explanations necessary for understanding the work.
Chapter 3 describes the concepts with which the defined requirements are to be achieved and how these are finally implemented within the application. In addition, a requirements analysis is carried out, which determines the target group of the application as well as possible usage scenarios.
Chapter 4 presents the results of the application from Chapter 3. Functional tests are carried out, and it is checked whether the usage scenarios from chapter 3 can be fulfilled with the final product.
Chapter 5 summarizes the results of the experiments and offers an outlook on further improvements of the methods.

1.4 Notation

Within this work, the following notation is used for easier understanding:

1. The developed application is called "Security Control Management" and is abbreviated to "SCM" in this report.

1.4.1 Related Work

This section describes how the literature search was conducted and which tools already exist on the market that offer similar functionality as developed in the context of this project work.

1.4.2 Literature Search

To identify relevant literature and works for this project, this process was divided into several steps. In order for literature to be eligible, it must meet all of the sub-steps. These steps are the following:

1. Search using Google Scholar (GS) using the keywords "Security Controls", "Security Controls Selection", "Automated Security Controls Definiton" and "Selection of optimal Security Controls". Only the first 20 results were considered.
2. Remove duplicates, and publications without publisher or source.

3. Determine whether the content is usable by considering publication titles and, if non-conclusive, abstracts, and conclusions.

4. Deeper investigate remaining papers for valuable content.

The *first step* was to search for possible literature for this work. For this purpose, a number of different search engines were available. These include IEEE, ScienceDirect, ACM, and Google Schoolar. Since GS also includes results from the other search engines, this search engine was selected for the literature search. For the search itself, the keywords "Security Controls", "Security Controls Selection", "Automated Security Controls Definiton" and "Selection of optimal Security Controls" were used. Since this outputs a large number of possible papers, the amount was reduced by considering only the first 20 search results.

In the *second step* every paper was shortly analyzed to identify duplicates and to remove papers which don't have a publisher or a source.

Within the *third step* the title was analyzed to determine if the paper is usable. In the case that the title alone doesn't allow a conclusion also the abstract and the if necessary the conclusion part of the paper was considered.

In *step four* the papers were deeper investigated so the Abstract, Introduction, and main content were analyzed for valuable content which can be used for this project.

1.4.3 Related Tools

During the research, no tools were found that could fulfill the requirements defined in this project. However, there are applications that address the fundamental problem of managing IT assets and security standards and for this reason are briefly mentioned below.

Swiss GRC

Swiss GRC[2] is a software solution for governance, risk, and compliance (GRC). It can be used to record risks and manage controls, guidelines, and security requirements. The tool also offers an ISMS (Information Security Management System) where measures, assets, incidents, audits, and analyses regarding security in the company can be recorded and managed.

BigID

BigID[3] focuses primarily on the management of data. As a result, large companies, in particular, have the opportunity to collect and manage different types of data. This helps to directly address concerns regarding specific regulations from GDPR, NIST, BSI, etc. In addition, security concerns regarding particular data can be identified and managed more easily and quickly.

StandardFusion

StandardFusion[4] is also a GRC tool that offers the possibility to capture assets and evaluate risks based on them. With the help of an integrated threat library, the process of identifying risks is greatly simplified for the user. These can then be evaluated and treated with mitigation controls.

Chapter 2

Theoretical Foundations

The theoretical foundations describe the technologies and services used in the project. On the one hand, the standards for security controls are presented, and on the other hand, the algorithms used are explained.

2.1 Security Controls

A security control is a concrete measure that reduces or completely avoids risks. It describes the theoretical steps that can be taken to address a particular risk. No specific technical details are mentioned regarding how a measure will be implemented in concrete terms. Instead, the controls are divided into different categories, where for each category, different controls can be selected to address a specific risk more strongly or weakly. The selection of security controls usually follows a security requirement analysis. Due to the modular structure of the controls, the necessary measures can be well adapted to this analysis.

2.2 Security Guideline Standards

Security standards are a set of guidelines that aim to protect infrastructure against cyber attacks. Such standards are published by different institutions, differing in their definitions and the organization of the guidelines. Concrete standards for security controls such as NIST SP 800-53[5] or ISO/IEC 27002[6] present measures in the form of controls that a system should implement to minimize the risk of cyber attacks.

2.2.1 ISO

ISO publishes the ISO/IEC 27000[6] family, which includes various standards related to information security. This consists of all possible areas concerning security, such as the definition of terms, requirements for an IT system, specific measures in the form of controls, organizational support, risk management, and analysis, etc. The standard was developed by ISO (International Organization for Standardization) in cooperation with IEC (International Electrotechnical Commission) and is regularly updated.

In terms of security controls, ISO/IEC 27002 is offered, the latest version of which is available under ISO/IEC 27002:2022. The standard is structured so that it offers concrete measures in the form of controls that can be applied in an information security management system (ISMS), defined in ISO/IEC 27001. It is also intended to guide organizations to determine and implement generally accepted controls in their systems. Within this standard, the controls are divided into four categories. A distinction is made between organizational controls, people-related controls, physical controls, and technological controls. The first category includes controls that relate to the administrative aspect of an information security management system. This includes, for example, controls such as correct contact with authorities or special interest groups. The threat intelligence process is also defined, as well as the responsibilities of the management. The second category deals with measures that can be applied to employees in order to minimize the security risk in the company. This includes, for example, awareness training and disciplinary processes and responsibilities after a change of position or termination. The third chapter contains controls to guarantee the physical security of assets. This includes physical access control or the securing of offices and other rooms or facilities in the company. Finally, the fourth category deals with the technical aspects of systems and how they can be secured in this way. Controls that fall under this category are, for example, protection against malware or logging and monitoring of activities in the system. Each control in the standard is briefly summarized and categorized by type (Preventive, Detective, Corrective, etc.). In detail, concrete points are then listed which should be implemented in order to fulfill the control. Further information is also provided, pointing out specific difficulties or valuable tips for the correct implementation.

2.2.2 NIST

Similar to ISO, NIST publishes the NIST SP 800 series[5], which provides various standards and guidance in IT security. The specifications range from information for federal governments and a high-level description of common security principles to the implementation of digital identity services and guidance on the security of industrial control systems.

In the case of NIST, the controls are divided into 20 supercategories. Each of these categories contains various controls, which, depending on the area, cover fundamental security aspects for the category. In addition, the controls can be extended by "control enhancements" to obtain even more comprehensive protection in this area. These are, again, own controls that enable a supplement to the upper control. For example, the "Account Management" control of the Access Control category has the control enhancements "Automated System Account Management", "Disable Account", "Inactivity Logout", etc. For each control and control enhancement, the measures to be taken are explained step by step, but no technical details are given. In addition, a Discussion section also explains why this control is necessary and why it was described in this way.

In addition to the Security Controls document, a Control Baseline is provided with the NIST SP 800-53B standard. This contains a minimum set of controls and enables those responsible in a company to gain a more leisurely start and a better overview of the existing security controls. The associated controls are listed for each supercategory with the corresponding control en-

hancements. For each control, the necessity of the control is shown based on the impact of the system. The impact of a system is classified as low, medium, or high. For example, a system in the area of access control only requires account management with inactivity logout if the impact of the system is at least medium. Access enforcement, on the other hand, is needed for every type of system.

2.2.3 Other Standards

In addition to those already mentioned, there are a few other standards for security controls. One is the BSI, the German Federal Office for Information Security. With BS EN ISO/IEC 27002:2022[7], it offers a standard for security controls based on ISO/IEC 27001. It does not differ significantly from the ISO standard but is published by the BSI.

Another standard that provides guidelines for security controls is CIS[8]. The CIS standard consists of 18 essential controls covering the most crucial information security areas. Companies should implement these to establish basic protection against cyber attacks. The areas range from organizational aspects such as inventory and asset management, data protection, and recovery to concrete measures such as penetration testing, malware defense, etc.

2.3 Natural Language Processing

Natural Language Processing (NLP) is an attempt to understand a text's meaning with a computer's help. For this purpose, various techniques are used, such as sentence splitting, tokenization, stemming, lemmatization, etc. The text is thus broken down into its parts and categorized. Then it is tried to extract the most relevant words that describe the text well.

2.3.1 RAKE

RAKE[9] stands for Rapid Automatic Keyword Extraction and is an algorithm for efficiently extracting keywords. RAKE is based on the observation that essential keywords are often surrounded by standard punctuation or stop words such as "and", "if", "or", etc. As a parameter, RAKE takes a list of stop words, which are then removed from the text to be analyzed in order to identify the keywords. Such keywords are called candidate keywords. Once these are identified, the individual words are evaluated further. The degree and the frequency are used for this. The degree of a sentence describes the number of words in the sentence. The degree for a word in turn describes the sum of all degrees of all sentences in which this particular word occurs. The frequency measures how often a word appears in the text. For each term, a score is calculated, which consists of the degree per word divided by the frequency of the word. The final RAKE score is formed from all scores for each word. The top T words are then selected from the final RAKE score. The value T corresponds to one-third of the number of words in the entire graph. The selected terms represent the chosen keywords that best describe the analyzed text.

2.3.2 spaCy

spaCy[10] is an open-source library that can be used with the Python programming language. As with RAKE, with spaCy, it is possible to export keywords from a text. It supports various NLP techniques such as tokenization, segmentation, part-of-speech (POS) tagging, lemmatization, etc. These techniques are applied in the form of a pipeline where the different steps are executed one after the other. Some methods require trained models, such as POS tagging, to classify individual words into verbs, nouns, and adjectives. Such models are available for spaCy and can be selected by the user. They differ in speed and accuracy. Linguistic annotations are also provided, which analyze the grammatical structure of a text. For the meaning of a text, it can be central whether the noun acts as the subject of a sentence or as an object. Also possible is the recognition of named entities for example, a person, a city, a company, etc. For instance, for the word "Apple", it is possible to recognize that it is a company (depending on the context of the sentence) or a fruit. This is made possible by a pre-trained model which is available for spaCy.

2.3.3 Yake

Yake[11] is a lightweight, unsupervised keyword extraction algorithm. It analyzes the statistical properties of a text to extract the most important keywords. According to its data, the methods perform better in some instances than those of already established algorithms on the market. In addition, this offers the advantage that the system does not have to be trained on a set of documents or depend on specific dictionaries, languages, or domains. To use and test the algorithm, a demo is provided on its website, as well as an API and a Python library that can be used freely.

2.3.4 KeyBERT

KeyBERT[12], as its name suggests, uses BERT embeddings and cosine similarity to extract keywords and key phrases. So in the first step, the BERT algorithm is used to get vectors from the documents (so-called document embeddings). Those documents that are similar get very close vectors. The same is repeated at the word level by creating word embeddings. Similar words have identical or comparable vectors. Then, using Cosine Similarity, the similarity of the document and word vectors is determined to obtain the most similar word or words to a document, which is the keyword that best describes the document. The goal of KeyBERT is to provide a simple and efficient working library for the extraction of keywords. For this reason, a corresponding package is available for Python, and no further dependencies are necessary.

2.3.5 TF-IDF

The TF-IDF measure is a statistical measure to assess the relevance of terms in documents. It is divided into two parts, which are then combined. On the one side, the TF value evaluates how often the word T occurs in document D. On the other side is the IDF value representing the inverse document frequency. The idea behind this is that for two documents, the correspondence of rarely

occurring words is more relevant than that of frequently occurring words. If there is a high match of the word "and", this says nothing about how similar the two documents are in terms of content. However, if there is a match for the word "weather", for example, this is much more significant. The TF-IDF is the product of the values TF and IDF and describes the importance of a word. The more often it appears in the documents, the less important it is and the lower its TF-IDF value.

2.3.6 Latent Dirichlet Allocation (LDA)

LDA[13] is a statistical method for topic modeling, i.e., assigning a topic to a document or text. Topic modeling can be used to simplify and optimize a search, for example. Documents are categorized by topics and if someone searches for a specific one, the documents can be displayed with the corresponding, matching topics. LDA is one of the most popular methods for this kind of task. Each document consists of several words and each topic consists of several words that fit it. LDA tries to find out the topic of a document by examining the words in the document and analyzing to which one they belong. For each term, the probability is calculated that it belongs to a particular topic. The algorithm starts with the fact that certain words that belong to a topic are already known. Now the document to be classified is examined, and the words are analyzed. If terms are found that belong to a topic, the document is assigned to this with a probability. If words are found in the same document that belong to a different topic, it is calculated for which there are more matching words, and the document is finally assigned to this. Likewise, the probability is updated that the words from the defeated topic belong to it since these can apparently also occur in another. For example, if you try to assign pictures to the categories "City" and "Nature", an image with many trees on it clearly belongs to the category "Nature". However, a picture of a city can also have trees on it. So the word "Tree" does not belong so clearly to the category "Nature" even if still more likely to "Nature" than to "City".

2.4 KMeans

KMeans is a clustering algorithm that classifies similar data points into the same clusters. The algorithm starts with K randomly positioned centers. Each data point is now assigned to its nearest center. Once all data points are assigned, the centers are repositioned to form the center of the assigned data points. The data points are now reassigned to the closest centers. The process is repeated until the repositioning of the centers is minimal. Thus, the corresponding clusters are now obtained. The problem with the KMeans algorithm is that the number of clusters K is not known. Too few clusters can lead to too much generalization, where data is combined into one cluster, even though it could be further subdivided. On the other hand, if the number of clusters is too high, the data will be separated too much, although they could be taken together. The Elbow method can be used to determine the optimal K, where several runs with different K are performed, and the average distance of the data points to the center is measured. The K, which has neither too high nor too low distances (i.e., approximately in the middle), is considered optimal. Another problem of

KMeans is the random selection of the starting position for the centers. This can lead to the fact that with different runs, the result turns out differently since the algorithm gets stuck in a local maximum. In this case, the KMeans++ method can be used, which comes very close to the optimal solution. With KMeans++, the centers are initially placed as far away from each other as possible, whereby the position of the first center is again random, and the maximum distance is then calculated for the others.

Chapter 3

Methods

This chapter shows and describes the individual steps of the project. In the beginning, a requirement analysis is performed to define the requirements for the application. Based on this, a concept is presented, and the individual process steps are explained. Afterward a detailed explanation of how the concept was implemented and how the developed solution will be tested is given.

3.1 Requirement Analysis

The requirements analysis is about identifying and defining the necessary functions of the software, from a group of fictitious person profiles, mandatory and optional functionalities of the application result.

3.1.1 Profiles

The following person could be part of the target group:

Alex, CISO of a big company

- Is responsible for the security in his company.
- Would like to take the appropriate security measures based on the company's requirements.
- Has a huge budget available.
- The security standard must be very high because the company has some critical data.
- The measures are all to be implemented internally, for which a dedicated security team is also available.

Matthias, CEO of a startup

- Founded a startup in the information technology sector with his colleague.
- The company hosts some critical data of customers and needs to protect them accordingly.

- The company is still very young, and there are few resources nor a large amount of budget available for security.
- Only basic security experience is available.

Jens, IT manager of an SME

- Owns a small team that is responsible for the entire IT infrastructure of an SME.
- Has already contracted an external company that provides certain security services.
- However, Jens would like to take further measures and stay up to date.

3.1.2 Target Group

The application can be used primarily as a supplement and assistance to existing security services. It is best suited for areas of application where there is already a certain amount of experience with security, and specific decisions must be made as to which concrete measures must now be implemented. In a scenario like Matthias, more than the application is required, since knowledge and budgeting must also be available for the correct implementation of the measures. For users like Jens and Alex, it can be helpful to select the appropriate controls that are specifically tailored to the company's requirements.

3.1.3 User Stories

The following user stories for this application result from the person profiles of the target group. These are divided into optional and mandatory requirements, whereby the application should contain at least the mandatory stories:

Mandatory Requirements

- As a user, I want to be able to capture the existing infrastructure of a company as flexibly as possible.
- Based on the requirements entered by the user, the application should automatically select and specify the appropriate controls.
- As a user and developer, I can add more functionality and content to the application at any time because the focus was placed on a scalable environment that does not require excessive effort for expansion.

Optional Requirements

- As a user, I want a simple, understandable, and efficient UI to see at a glance which controls are necessary for my business.
- As a user, I want to be able to choose between different security standards and see the equivalent in another standard for each control.

- As a user, I would like to integrate my existing asset management system with the security control management software to manage assets in one central location.
- As a user, I would like to be able to capture the implemented controls to get an estimate of the resulting risk automatically.

3.2 Concept

The application aims to assign appropriate controls to a given system. There are a few publications that try to achieve a similar goal. In the work of Barnard et al.[14], a formalized approach was developed for selecting and evaluating security controls. Based on business analysis, a suitable subset of controls is selected, and a control selection process is used to ensure the controls are correct. Yevseyeva et al.[15] use a more mathematical approach to this. A model is modeled in which a given budget is defined and distributed across the controls. Each control has a certain loss prevention, which can arise if the controls are not used, and corresponding risks occur. This approach results in a classical portfolio selection problem. The work of Neubauer et al.[16] uses aspects of both methods. Based on security ontologies, appropriate countermeasures are modeled as security controls. The choice is to minimize the costs and maximize the benefits.

For this work, the security controls will not be selected based on requirements but directly from assets in the enterprise. Therefore, this functionality must be mapped accordingly in the tool's structure. The workflow of the application was defined based on the requirements analysis as follows:

1. User defines his system by capturing assets.
2. The user can assign properties and a description to the assets which describe the asset in more detail.
3. Furthermore, the user can assign constraints that provide more general information about an asset or system.
4. The user can import supported security standards (for this PoC, only NIST Controls) or create them himself.
5. When the user starts the analysis, the system categorizes the controls and assets using NLP.
6. Now, the user can display the matching controls, where they are selected based on the categorization of the NLP as well as further heuristics.

In a *first step*, the user must be given the opportunity to define his current system state. An asset-based approach was chosen for the definition of the requirements. This is because the conditions for each system component can be precisely defined, and there is also the possibility of integrating an already existing asset management, thus ensuring scalability. The user, therefore, has the option to enter assets, which can be divided into different categories such as software, hardware, data, etc. However, this information is not yet sufficient for a well-founded selection of controls. It must be possible to specify the assets

further since they can differ significantly within the same category. For example, the software can be a web store as a web application or an internal CRM tool.

Therefore, another component needs to be added with the properties, which can be assigned in a *second step* to the assets to specify them further. Such a property can be named, e.g., Account Management, Web application, Database, etc. With a property-based approach, assigning the same properties to assets with similar or the same functionality is possible. This reduces redundancy and makes it easier to identify similarities for subsequent matching. The properties can be created by the user or imported via File Import. They contain a description that is to be analyzed by the NLP. In addition, the user can add a description for the asset itself. An alternative algorithm then analyzes this without the need for properties or tags.

However, an asset or system is not only described by properties or the description. Some circumstances determine the necessary controls of an asset independently of its properties. For the SCM, these are referred to as "constraints". Such a constraint can be, e.g., the "Impact", which can have several gradations (e.g., "low", "medium", "high"). If an asset has a high impact, it is particularly worth protecting and may require more controls than an asset with a "low" impact. For this reason, the user should be able to assign such constraints to the assets in the *third step*.

Of course, the system must also contain the controls that are ultimately to be matched to the system. For this reason, in a *fourth step*, the user should be able to import these via a file import or create them directly via the user interface. During the import or the creation, a control should also be assigned suitable constraints. In the case of the NIST controls, each control and its impact is specified. For this work, the focus is on using NIST's Security Controls. This is because they are freely available, and various additional information is provided, such as the impact of each control.

When the user finally starts the analysis in the *fifth step*, the system must be able to categorize the controls. The tag component, which is to be created automatically using NLP, serves this purpose. Such a component consists of several keywords derived from the description of a control, as described in the following sections. The tag component is similar to the property component assigned to assets. These two components describe the control and asset, respectively, and it should be possible to easily match them against each other.

In the *last step*, the matching will be performed, which uses the components created so far to find appropriate controls for the system. For this purpose, two approaches are implemented, which require different elements. In the first approach, similar properties will be matched with similar tags, and the respective assets will get their controls. Also, the constraints are included, which add additional controls to already matched controls. The second approach does not require any tags or properties. Instead, an attempt is made to establish a relationship directly via the description text of an asset and control by comparing them with each other. In the subsequent evaluations, it will be evaluated which of the two approaches is better suited in which cases.

3.3 Natural Language Processing

Natural Language Processing is responsible within the application for analyzing the content to perform matching subsequently. For this purpose, two approaches were implemented, then compared with each other.

3.3.1 Tag-based Apporach

In the tag-based approach, tags are created that can be matched with each other. For the controls, the goal is to extract as much important information as possible from the description text to categorize the control and its functionality well. The same procedure is to be used for assets, but the properties are analyzed here with their description. In order to extract information from these texts, essential keywords are first searched for. The idea is that certain words best represent the entire text. When all controls have been analyzed, a massive list of such keywords is obtained, where each keyword comes from at least one control. Of course, there is also the possibility that a keyword was found in several controls. Furthermore, some keywords have a certain similarity (e.g., Account Management and Account Creation). Such similar or identical keywords also indicate a possible similarity of the associated controls. To recognize this similarity, the keywords are to be clustered, whereby similar words come into the same clusters. This also forms the necessary tags for the controls. In order to be able to classify the tags, a suitable topic is to be found for each tag utilizing topic modeling. Thus, e.g., a tag "Account" can be formed with the keywords "Account", "Account Management", "Account Creation", "Account Roles", etc. The resulting tags are assigned to controls based on their containing keywords. For example, if the "Account Management" keyword comes from the "Access Enforcement" control, the "Account" tag will be assigned to the "Access Enforcement" control. Figure 3.1 illustrates this process.

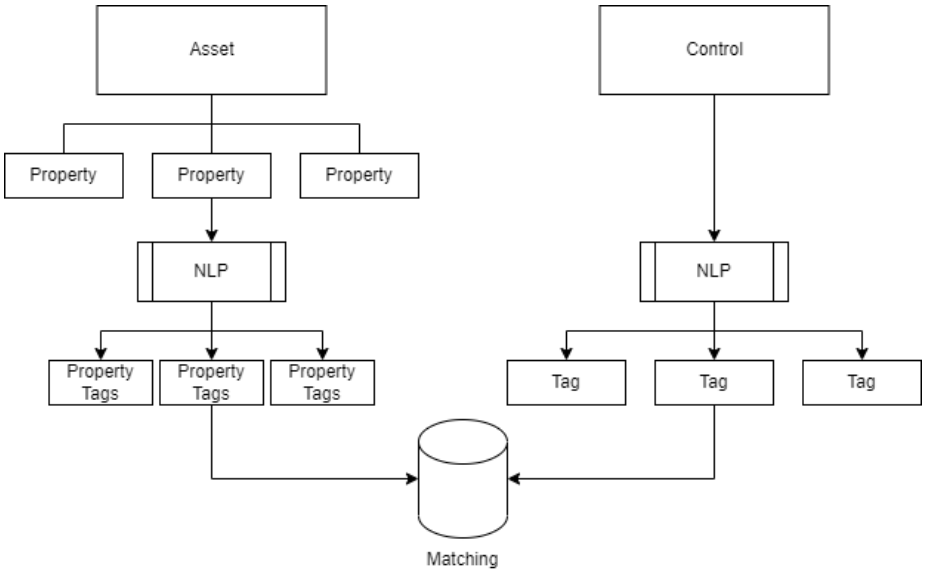


Figure 3.1: Tag-based NLP process overview.

The description texts of the controls and those of the properties of the assets are given as input to the NLP system. Using the explained keyword extraction, clustering and topic modeling, property tags are created on one side and (control) tags on the other. These two tag types are then matched with the matching algorithm.

3.3.2 Description Text comparison Approach

The second approach takes a more straightforward solution based on the concepts in chapter 2 in the book of Peters et al. [17]. In this case, the description text of an asset and a control is directly analyzed and compared with each other. Each asset and each control is considered separately. There are no relationships between assets or controls, and no attempt is made to find a similarity between their types. Only between asset and control is a similarity tried to be identified. The idea is to use NLP to determine how similar the descriptions are in content. If a particular, defined similarity value is reached, it becomes a matching. Figure 3.2 illustrates this approach.

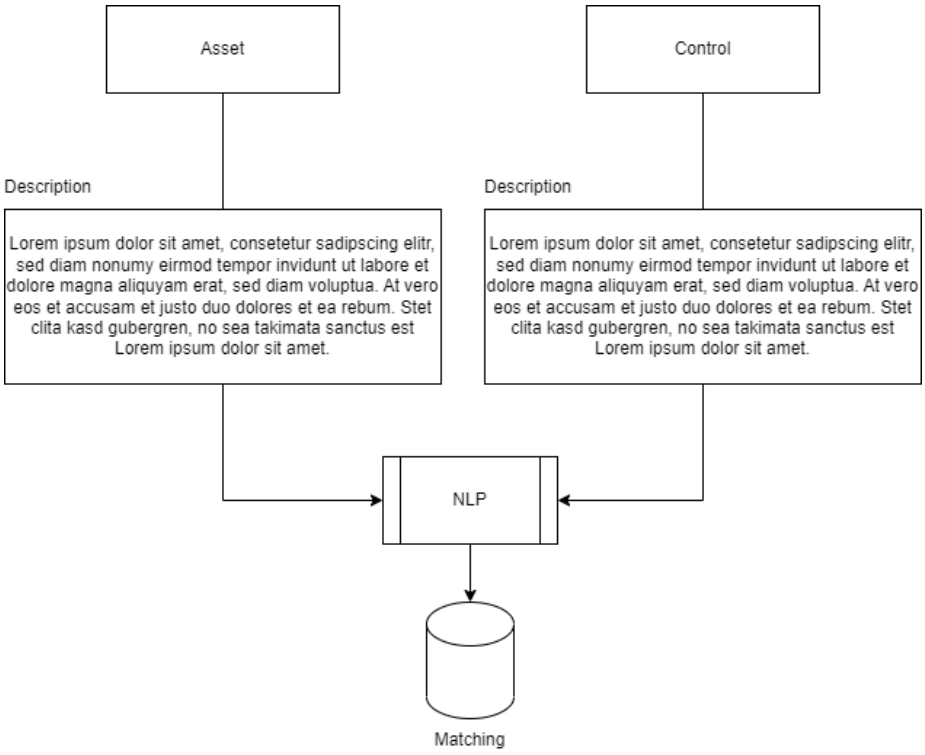


Figure 3.2: Description text comparison NLP process overview.

As shown in the figure, only asset and control pairs are analyzed together. This is repeated for every possible asset/control pair in the system. The NLP process creates a similarity value which is then checked by the matching algorithm and decides whether it is a match. For this approach, a sufficiently detailed description must be available at both the asset and control levels.

3.4 Matching Procedures

Several approaches were implemented and compared to match the controls to the assets. These approaches are listed below:

1. Matching via direct assignment of control tags to assets.
2. Matching tags of controls with property tags of assets.
3. Matching by comparing texts of the control description and the asset description.

In the first case, the user can assign the tags, created from the controls via the NLP, directly to the assets. In this case, those controls are selected which have been tagged with the correspondingly chosen tags. For the assignment of suitable tags to the assets, these are provided with topics that describe the tag.

Thus, a description is selected that represents the keywords within the tag as well as possible.

In the second case, the assignment process is more complex. The properties of the assets are also analyzed using NLP. The comparison is then performed at the keyword level. So each property tag should be compared with each control tag and checked if they contain the same keywords. A match is obtained if there is a certain degree of similarity (this threshold can be defined). The controls to which the tag of the match was assigned are then the controls selected by the system for the corresponding asset. In contrast to the first case, the assignment is automatic without user interaction. Figure 3.3 shows the algorithm.

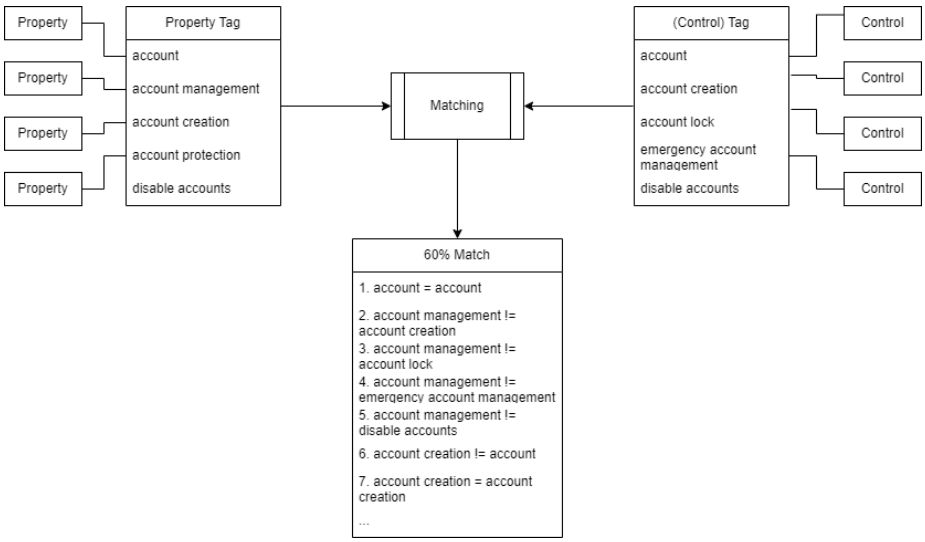


Figure 3.3: Tag-based matching algorithm with property and control tags.

In the figure, a property tag is matched with a control tag containing five keywords each. Also, both tags are assigned to 4 properties and controls, respectively. During matching, each keyword of the property tag is compared with each keyword of the (control) tag. As a result, the algorithm detects that 60% of the property tag keywords match the (control) tag keywords. This results in a match since it was defined (threshold) that at least 5% must match, which was exceeded many times.

The third approach tries to work without tags and is based on the description text comparison approach described in Chapter 3.3.2. The algorithm works very simply. First, the determined comparison values of each asset against all controls are checked value by value with a defined threshold. If a value is above the specified threshold, a match is made. The corresponding control is then assigned to the asset.

3.5 Implementation

This chapter describes the implementation of the Security Control Management Tool, which implements the described concept and methods and thereby tries to reach the defined requirements.

3.5.1 Technology

The main system architecture of the Security Control Management Software is shown in Figure 3.4.

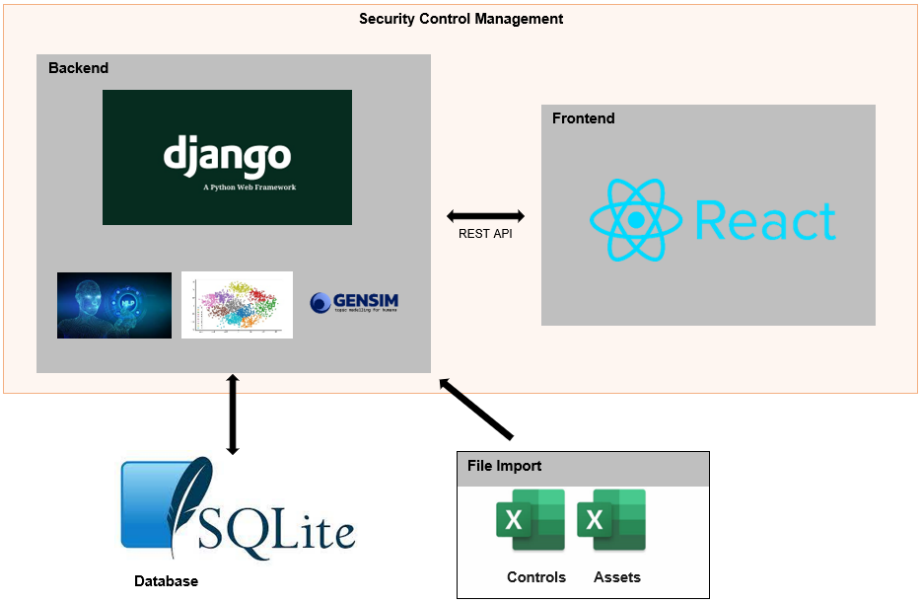


Figure 3.4: Illustration of the system architecture.

The system is a web application, which is divided into a backend and a frontend. For the backend implementation, the Django[18] framework with Python is used. This is because Django is a compelling web framework and therefore meets the application's scalability and maintainability requirements. On the other hand, Django allows the use of the Python programming language, which supports many libraries and is very powerful and most frequently used in the machine learning and natural language processing areas. In addition to Django, various other Python libraries are used for keyword extraction, clustering, and topic modeling. For persistence, an SQL database is connected, allowing controls, assets, tags, etc., and the results of the analyses to be stored and retrieved at any time. As a concrete technology, SQLlite is used for this PoC because it is easy to implement and can be retrieved locally at any time. If necessary, it can be easily converted to a regular, decentralized SQL database without changing the syntax in the code to any great extent. To be able to use the existing controls of the various security standards, an import function was also built, which can read the controls from an Excel file, and the data can be used

in the application. The same applies to the assets, which can be exported from an asset management tool such as Excel or CSV and imported into the security control management tool. The backend provides its functionalities and data via a REST API. To display these accordingly, a frontend was also developed with React[19]. React was used because it gives a simple component-based framework that guarantees good scalability for the further development of the application. Thanks to the REST API, it is possible to exchange the frontend at will and, for example, replace it with an Angluar-based system.

3.5.2 Selection of Algorithms and Libraries

Various external libraries are required for the individual steps of the analysis and evaluation of the controls and assets. Each has different advantages and disadvantages, which were evaluated in various test runs. The results of these are listed in the results chapter. Finally, the individual properties of each library were weighted, and the technologies with the highest weighting were selected.

Keyword Extraction

For the keyword extraction, five different libraries were selected that are suitable for this task. These are KeyBERT, RAKE, spaCy, Textrazor, and Yake. Textrazor is directly omitted since this is an external API and an API key is necessary to be able to make requests. A certain number of requests per day is offered free of charge, but more is needed for the scope of this work. For this reason, Textrazor is not included in the selection. For the keyword extraction algorithms, the keyword quality, the number of keywords, and the distribution of keywords are tested and weighted. The results of this evaluation are shown in Table 3.1 through Table 3.4.

Table 3.1: Performance of the KeyBERT algorithm.

Metric	Weighting (1-5)
Keyword quality	2
Number of keywords	2
Distribution of keywords	3
Total	7

Table 3.2: Performance of the RAKE algorithm.

Metric	Weighting (1-5)
Keyword quality	3
Number of keywords	2
Distribution of keywords	4
Total	9

Table 3.3: Performance of the spaCy algorithm.

Metric	Weighting (1-5)
Keyword quality	4
Number of keywords	5
Distribution of keywords	3
Total	12

Table 3.4: Performance of the Yake algorithm.

Metric	Weighting (1-5)
Keyword quality	5
Number of keywords	5
Distribution of keywords	3
Total	13

The evaluation shows that Yake performs best overall in the extraction of keywords. The quality of the keywords is excellent, and an acceptable amount of keywords is generated, neither too much nor too little. The distribution offers room for improvement but is sufficient.

Clustering

The KMeans algorithm was selected for clustering. This is because it gives good results, performs well, and is one of the most used algorithms for clustering. Furthermore, with KMeans, it is also possible to cluster words or sentences. However, the tests show that different runs can also produce different results. These occur both with KMeans randomized and KMeans++. KMeans++ is used for this application because it is the standard parameter of the corresponding Python library. Based on the tests performed, the cluster size was set to 20% for the (control) tags and also 20% for the property tags. In addition, the tolerance value for cluster cleaning was set relatively high at 600% for the keywords and 800% for the controls in order to eliminate only the outliers and not catch too many of the somewhat larger clusters. What these values mean is explained in more detail in section 3.5.7.

Topic Modeling

In the case of topic modeling, the LDA algorithm can be used, which searches for a suitable topic for a given text. In addition, a custom algorithm was also implemented, which selects the topic based on the most frequently occurring word in a cluster. The most important thing for the topic is that it matches the words and that the user understands what it means. The evaluation of the results, including their weighting, can be seen in Table 3.5 and Table 3.6.

Table 3.5: Performance of the LDA algorithm

Metric	Weighting (1-5)
Topic fits the cluster	3
User understands meaning	3
Total	6

Table 3.6: Performance of the self-developed algorithm

Metric	Weighting (1-5)
Topic fits the cluster	3
User understands meaning	2
Total	5

The LDA algorithm provides slightly better results than the self-developed algorithm. The meaning of a cluster is expressed better, and it is easier to understand for the user because the most used word algorithm sometimes misses too many components of a term. For this reason, the LDA algorithm was chosen for topic modeling.

3.5.3 Structure

Based on the concept defined in chapter 3.2, the structure visible on figure 3.5 results for the Security Control Management application.

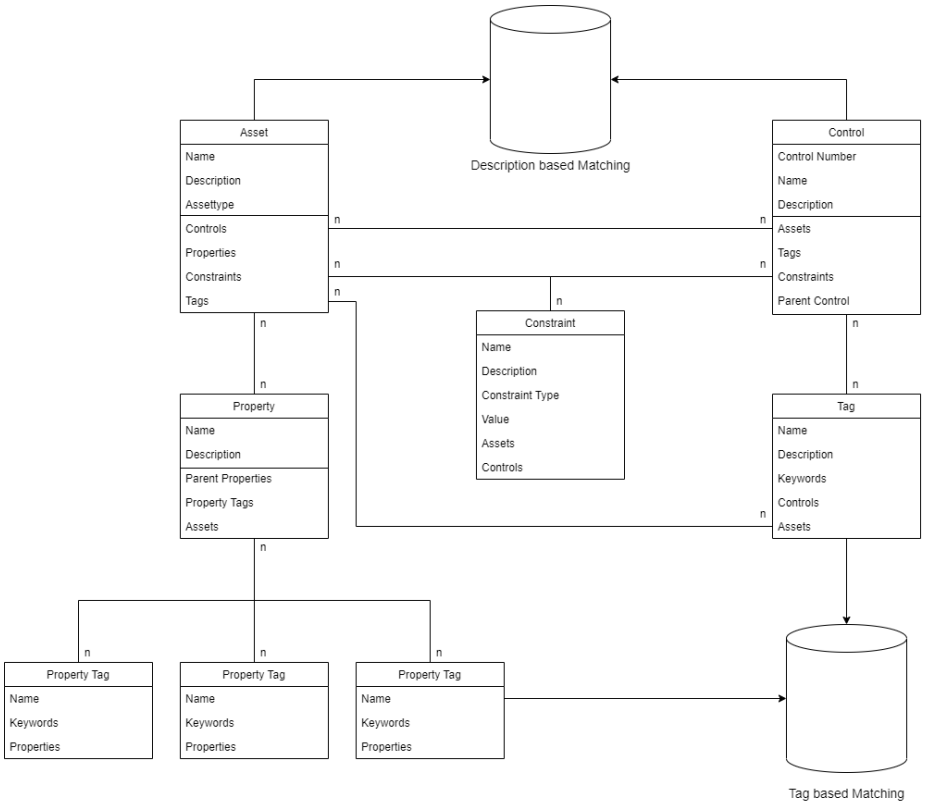


Figure 3.5: Structure of the SCM application.

The left side shows the components of the asset definition, and the right side shows the parts of the controls. In the requirements definition, the asset component is central. An asset consists of a name, description, asset type (category), defined constraints, its properties, and as soon as the matching is done, there are also some controls assigned. The properties, in turn, consist of name, description, and parent properties, thus achieving a hierarchy of properties that can only be selected under certain preconditions. The property tags associated with the properties are created by the NLP. Accordingly, a property tag has keywords determined by the keyword extraction algorithms and a name selected by the topic modeling. These property tags can then be fed into the matching algorithm, thus realizing option 2 of the matching procedures (tag based matching). In order for option 1 to be possible, the tags of the controls can also be assigned directly to the assets. In addition, there is the constraint component which can be assigned to an asset or control. A constraint consists of a name, a description, and possible values. A constraint is only assigned with one of the several possible values if a constraint is set. This circumstance must be mapped accordingly in the structure, whereby an intermediate table is necessary to maintain the assignments. The matching for the constraints should then be done according to these values. For example, they are matched if an asset and control have the same constraint with the same value.

On the right side, the control component is crucial. This consists of Control Number (CN), Name, Description, Parent Control, Constraints, its Tags, and the assigned Assets. Controls can have subordinate controls which extend the actual control. However, such a hierarchy does not go beyond two levels, so a child control cannot have its own children. The correct mapping of this hierarchy is crucial for matching. The affiliation must be clear within the application since it makes no sense logically that a child control is assigned to an asset without its parent control. With this hierarchical structure on the diagram, this is ensured. The tags assigned to the controls consist of Name, Description, and Keywords. As with the property tags, the (control) tags are created by the NLP. For option 2 of the matching procedures (tag-based matching), they form the counterpart to the property tags and, as such, are passed into the matching algorithm where the two tag types are compared. They can also be assigned directly to the asset by the user, which is the first matching option.

To capture the third matching process in the structure of the application, the description of the assets and the controls is required. These are given as input to the description-based matching algorithm. That way, they can be compared directly with each other.

3.5.4 User Interface

The interface should reflect the developed structure as well as possible and make the individual components configurable in a simple way. A lot of emphasis was put on a modular design to be able to extend the interface easily with additional functions. For this purpose, a separate section was developed for each part, which can be accessed via the sidebar. The sidebar can also be used to easily add new sections to accommodate new components and make them editable. A part of the interface can be seen in Figure 3.6.

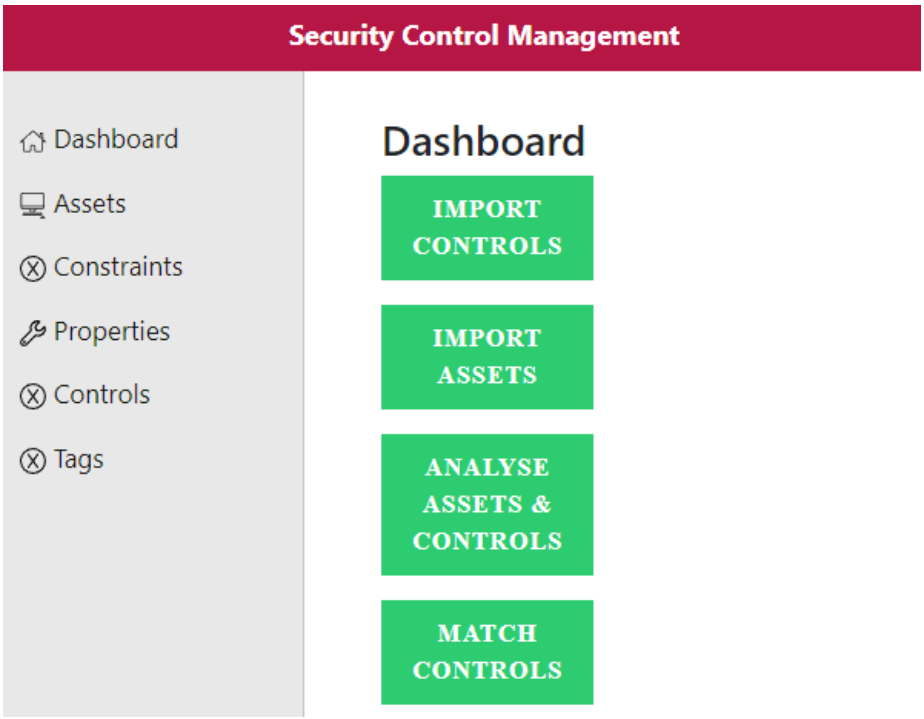


Figure 3.6: Sidebar and dashboard of the Security Control Management software.

Each component contains further submodules that allow editing, creating, and deleting a component. To ensure scalability, the submodules can be extended or removed as needed. In order to control the application as a whole, a dashboard has been developed that provides an overview of the captured entries and allows various actions to be performed, such as importing controls, creating tags via NLP, matching the appropriate controls, deleting specific imported or created data, etc. Once these steps are completed, the results and, thus, the possible controls are also displayed on the dashboard. They are structured according to their hierarchy, and it is directly visible which controls have which subcontrols and which of them are recommended. They are also assigned to each asset and can be viewed there.

3.5.5 File Import

To be able to quickly include external data in the tool, an import interface has been implemented. This allows the controls and the assets to be imported as an Excel file. The interface processes certain relevant columns of the files and directly creates the necessary objects in the system. For example, the columns "Identifier", "Control Name", "Control Text" and "Discussion" are taken into account for the controls. With this information, an object is created in the database for each control, which can then be accessed as desired. The control exists in the same form as if it had been created manually by the user. The relationship of the controls to each other is also taken into account. Certain

controls are subordinate to another control. The importer also extracts this information and relates the corresponding control to its parent.

The same applies to the assets, whereby any number of columns can be imported and also a different number of columns per asset. Only the name and, if applicable, the description remain fixed. All other columns are added as properties and assigned to the asset. Each asset can have any number of properties. These properties are then analyzed by the NLP system and characterize the asset. The user also has the possibility to add his own properties and to complete the information from the Excel file.

3.5.6 Web Crawler

For the description text comparison, a sufficiently detailed description text must be available to calculate reliable TF-IDF values. Since this is not given with the existing data on the asset side, a web crawler was developed for this purpose. For the search, the Bing Search API[20] was used. This is because Microsoft provides startup credit for students, and thus a certain number of queries are free. In addition, Bing Search is a reliable tool that meets this work's requirements. To find suitable descriptions for the assets, the asset's name is searched for, and the prefix "What is" is added to increase the probability of finding a description text. Also, results that have "What is " in the title are preferred. In the case of hardware assets, the name of the operating system used is searched for. A search for the name does not make sense because, especially in the case of servers, the assets are given a name according to an internal naming scheme and therefore it is not possoble to find something about it in the internet. Finally, the Scrapy[21] Python library is used to extract only a website's relevant content. This allows filtering the HTML content of a web page by text. If a website is found that meets all these criteria, its content is extracted and assigned as a description to the asset, where it can then be used for analysis.

3.5.7 Tag-based NLP Process

In the tag-based NLP process, corresponding tags must be generated from the controls' descriptions and properties' descriptions. How this is implemented precisely describes this section.

Structure

The tag based approach is a process that consists of several steps. These are independent and must work well together. The first step is keyword extraction, from which the clusters are generated and then assigned a name via topic modeling. In addition, the relationship of the controls must be taken into account. In the case of NIST controls, there is a particular hierarchy between the controls. Some controls function as parent controls and have control enhancements representing extensions for the control. Thus a structure with parent controls and child controls develops. This fact can be used for NLP to recognize similarities and relationships between them better. Only the parent controls act as independent controls and can be considered in the same context as other parental controls. However, a child control can only be assigned if its parent

control has already been assigned to the asset. This hierarchy must be mapped to the NLP process.

For this reason, only the parent controls are analyzed in the first step according to the procedure described above. In the second step, each parent control is analyzed individually with its child controls. Clusters and, thus, tags are formed only from the keywords of the respective control hierarchy. In addition, each parent control receives its children's tags to reflect that a child control can only be assigned if its parent control has also been assigned. The whole NLP process is illustrated in graphic 3.7:

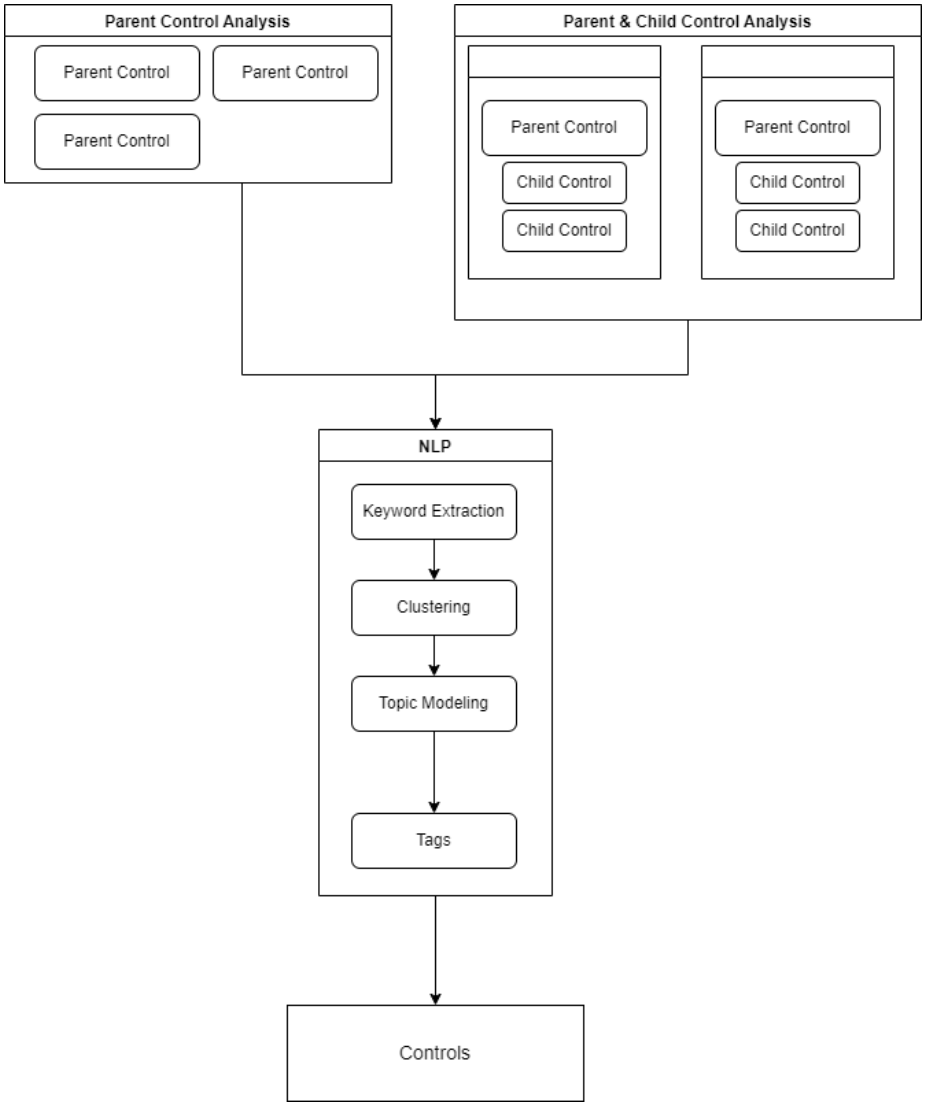


Figure 3.7: Implementation of the NLP process.

First, the parent control analysis is performed, which analyzes all parent

controls, extracts the keywords, clusters them, and creates tags from them. These tags are then assigned to the parent controls. In the second step, each control is passed through the NLP process with its child controls. The tags remain in the hierarchy and are only assigned to the controls in it. Each parent control receives all tags of its children. This is an attempt to make clear that child controls are only similar to their parent control and the other child controls in the same hierarchy and that a child control can only be assigned if its parent has also been assigned since a child control is an "enhancement" of a parent control.

Keyword Extraction

Keyword extraction is the first part of the analysis process. The goal is to find keywords that describe a control and its usability well. Different algorithms are implemented within the security control management software that provide different results. In the case of the NIST controls, the keyword extraction receives the title, the control text, and the discussion for each control as input. The same applies to the assets where the title and the description of each assigned property are passed to the NLP module. These parts are combined into text that is converted into lowercase letters (to avoid the same words with a different spelling) and then analyzed. The algorithms use various techniques of Natural Language Processing. One of these is tokenizing, which splits up a text. Depending on the algorithm, word tokenization is used where the text is split at the spaces, and each word is stored in a list. Some algorithms also match the words with a language model to identify related words. In addition, there is sentence tokenization, where a text's sentences are extracted. This also includes the removal of stop words like "as", "the", "to", "and", "or", etc. Such words are of no use for keyword extraction as they are not relevant keywords and say nothing about the text. Normalizing words, where they are broken down to their basic form, e.g., the word "watched" becomes "watch", is also used in some cases. The words that have been processed in this way are then evaluated with different methods. Each algorithm has a slightly different procedure in this case. Those with the highest score are output as keywords. The words determined, are passed to the clustering algorithm in a data frame. It is also essential that for each keyword, a reference to its descendant control is stored since this reference is needed at the end to assign the tags to the controls.

Clustering

The second step in the Analysis process is clustering. The idea here is to combine similar keywords into a cluster to be able to create tags from them. As input, the cluster function receives the data frame with all keywords and the reference to the associated security controls. For the clustering itself, the KMeans algorithm is used. It is executed with the specified parameter to get the best possible result. The cluster size is defined depending on the number of keywords, and the percentage can be configured. For example, if a rate of 20% is selected, 20 clusters are created for a number of 100 keywords, into which these 100 keywords are divided. Nevertheless, the evaluations have shown that in many cases, KMeans fails to split all keywords optimally. Additional cluster cleaning is performed to avoid large clusters that only serve to accommodate

those keywords that otherwise have no place. On the one hand, those clusters that are much larger than all others are removed. This is done by calculating the average cluster size over all clusters. Furthermore, a certain, definable tolerance value is included within which the clusters may be larger than the average value. All clusters more extensive than the average with tolerance are too large and will be removed. The same applies to the number of controls assigned to a tag/cluster. If there are tags assigned to many more controls than the average, these tags will be removed. The resulting clusters are then passed to the topic modeling function to find a descriptive term for the keywords.

Topic Modeling

With the help of topic modeling, a descriptive term should be found for the clusters to simplify the tags direct assignment to the assets by hand. Two approaches were chosen for topic modeling. The first approach uses the LDA algorithm, which tries to find a matching topic using a probability distribution. As input, the algorithm receives the clusters with their keywords and based on these, it tries to find the term that best describes the remaining terms.

In the second approach, the frequency of a word in the keywords was used to find the appropriate descriptive word as a topic. The keywords of a cluster partly consist of several words themselves. For this reason, they are first separated by the space character and divided into individual words. The obtained words are then counted, and the most frequently occurring word is used as a topic for the cluster. The idea is to find the word that occurs most often in the cluster and therefore fits best to the cluster to describe it. Of course, only actual terms should be considered for this and not stop words such as "the", "a", "also", etc. However, since these words should have already been removed during the keyword extraction, no further measures are taken with this algorithm regarding this topic.

3.5.8 Description Text comparison NLP Process

For this analysis process, the TF-IDF value is used over the descriptions. This allows the TF-IDF value for each word of a description to be calculated and related to the descriptions to determine its similarity. Using the TfidfVectorizer from the Sklearn[22] library, a vector is created which contains the TF-IDF value for each word of a description. A separate vector is created for each description, and all vectors contain all words of each description. If two descriptions have the same words, the vectors have the same value at this point and are therefore similar to a certain degree. In order to receive a unique value for the similarity, the Cosine Similarity between the vectors is computed. Thus one receives a matrix that puts each description text to all others in the relationship and the similarity is evident. In the case of SCM, this is repeated for each asset, and only the similarity between asset and each control text is relevant. This reduces the matrix per asset to a vector of values where each entry represents the similarity of the asset to a corresponding control. This vector is now passed to the matching algorithm which evaluates whether it is a match or not based on the values determined.

3.5.9 Matching

The application includes three matching procedures as presented in chapter 3.4. The first procedure is the simplest and does not require much program logic. As soon as the user assigns a tag to the asset, a relationship between the asset and the tag is established in the system's database via a many-to-many mapping table. The matching algorithm now iterates over this mapping table and reads the controls for each tag to which it is assigned. The controls are then directly related to the asset specified in the mapping table. At the end of the iteration, all controls have been assigned to the assets with the corresponding tags.

For the second method, there are two classes: Asset and tag. An asset consists of a few description properties and a list with all property tags that belong to the asset. In the tag class, a list with the associated keywords is maintained as well as a list with the assigned controls. The matching algorithm now gets two lists, one with all assets and one with all tags. These are then processed in a nested loop. For each asset, all tags are processed. A list with the keywords of all property tags of an asset is created. This list is compared to the keyword list of the tag as described in chapter 3.4. The list is then used as a basis for the matching algorithm. The asset and tag match if the similarity value is above the threshold. The value for the threshold was determined via the tests carried out and, in this case, set at 5%. After a match, the asset is assigned all controls that are in the list with the assigned controls of the tag.

The third matching method analyzes the TF-IDF vector passed by the description text comparison. In the application, a threshold is defined, which must be reached at least so that it is a match. Thus, all values in the vector are iterated through, and each control whose value is above the threshold forms a match with this asset. This process is repeated for all the assets captured, and the matching controls for the assets are obtained. The difficulty is determining the correct thresholds. On the one hand, this must not be set too high, as in this case potentially matching controls will not be recognized as such. On the other hand, it must not be set too low, as in this case, too many controls could be assigned and thus controls that do not fit. It is, therefore, necessary to find the optimal value in between. Based on the tests carried out, the threshold was set at 15%, which is a good compromise.

3.5.10 Metrics

In order to evaluate the matching methodologies, various metrics were collected to assess the results. Table 3.7 shows a list of the implemented metrics.

Table 3.7: Implemented metrics.

Metric	Description
Keyword Counter	Counts the total number of keywords identified by the NLP.
Tag Counter	Counts the total number of tags created by clustering keywords.
Keyword Cluster Distribution	Represents the number of keywords per cluster/tag in a dictionary.
Control Cluster Distribution	Represents the number of controls assigned to each tag in a list. This illustrates the number of controls that would be assigned if this tag was assigned to an asset (manually or automatically).
Average Keywords per Tag	Shows the average number of keywords a tag has.
Average Controls per Tag	Shows the average number of controls assigned to a tag. On average, this number of controls is assigned to an asset per tag.
Matched Control Count	Shows the number of controls matched to the given system.
Matched Tags	Shows the number of tags that have been matched to the given system.
Matching Tag Distribution	Shows within a dictionary which tags have been matched and how many controls have been assigned.
Deleted Tags	Number of tags deleted by cluster cleaning.
Other Metrics	Provides space for additional metrics, such as a detailed view of specific matches and their achieved thresholds.

These metrics are collected during the analysis by the NLP mechanics as well as during the matching of the controls. There are two different types, one for the NLP applied to the controls and one for the NLP of the properties. In both cases, the same measurement values are collected. In a further step, these metrics can now be used to evaluate the quality of a method. But first, evaluating which values could indicate a good matching is necessary. For example, it is possible that a too-low or too-high number of keywords per tag results in a suboptimal outcome. Also, a too high-number of controls only assigned by one tag could cause an unfavorable effect.

3.6 Validation of Implementation

The implementation validation should show whether the application meets the defined requirements. It is also intended to uncover opportunities for improvement and expansion. For this purpose, the tests are divided into two sections. On the one hand, the implementation tests and on the other hand, the acceptance tests, which are both described in the following.

3.6.1 Implementation Tests

A data set from a real asset management system at ZHAW was used for the implementation tests. It contains thousands of software and hardware assets. The implementation tests are divided into several steps:

1. Extract a subset of 200 total entries (100 hardware assets and 100 software assets) from the dataset.
2. Tests of the different keyword extraction algorithms to choose the most suitable one.
3. Performing tests for the clustering algorithm with multiple substeps for cluster size, the appropriate parameter, and various cluster cleaning techniques.
4. Testing of different implementations of the topic modeling algorithm.
5. Final test with all optimal settings determined in the previous tests, focusing on the application's overall performance.

In the *first step*, a subset of 200 entries is extracted from the existing data. The reason for this is to ensure clarity. It is clearer to see which parameters are responsible for which changes and error sources can be isolated more easily. Another reason is the performance, which decreases with increasing data size.

In the *second step*, the keyword extraction algorithms are tested with the existing data. The goal is to determine the best algorithm in terms of keyword quality, number, and distribution of keywords. All components are thereby tested with the same remaining parameters as they were defined in chapter 3.5.2

In the *third step*, tests are performed for the clustering algorithm. Cluster size, parameters, and different cluster cleaning techniques are tested. The cluster size depends on the number of keywords. A proportion of 10 percent, 20 percent, and 30 percent are tested. The KMeans randomized and KMeans++ algorithms are also tested, and how they differ from each other. Finally, cluster cleaning and thus low tolerance values (200% and 300%), as well as high tolerance values (600% and 800%), are tested, and their effects on the tags are analyzed.

The different topic algorithms are tested in the *fourth step*. On the one hand, the LDA algorithm, which automatically extracts the appropriate topic from a given text, and on the other hand, the self-implemented algorithm, which selects the term that occurs most in the cluster. It is tested how well the topic fits to the cluster and if the user understands the meaning and knows which terms are within the cluster.

The *fifth step* is to test the system as a whole. The best settings from the previous steps will be used, which are listed in chapter 3.5.2. It is checked which controls are matched from the existing data set. Relevant here is, above all, which assets receive which controls and how the distribution of these is. For this purpose, corresponding graphics are created to show this. The results of this step are then relevant for the acceptance tests.

3.6.2 Acceptance Tests

The acceptance tests evaluate the application concerning the requirements defined at the beginning. This is based on the results of the implementation tests. It should be checked for each user profile whether and how its needs can be met. This should result in a clear target group for the application, which can be addressed. It can also be checked whether the defined user stories could be implemented and whether they match the corresponding user profiles. To achieve this, each point in the related profile description is compared with the final results. The sum of the fulfilled and unfulfilled points then results in the relevance of the application for the specific user group. Likewise, it can be determined under which circumstances the application can be helpful or less helpful for a user group.

Chapter 4

Results

In this section, the results of the work are explained. These are based on the implementation tests carried out in the validation of the implementation. The individual steps are listed accordingly, and the results are documented. At the end of the chapter, the results of the application, in general, follow with the parameters which were evaluated in the preceding steps. Also at the end are the acceptance tests, which, based on the other results, determine and describe the usability of the application for the defined user profiles.

4.1 Keyword Extraction

For keyword extraction, different algorithms were tried, and their results were compared. These are presented in this section. The results from each algorithm are presented in a separate subsection.

4.1.1 spaCy Algorithm

spaCy generates 7239 keywords over the total 1235 controls. For the 235 properties, 658 keywords are generated. The algorithm is able to recognize individual terms as well as word groups. However, this is not always the case. The algorithm sometimes makes mistakes and, for example, evaluates individual letters as keywords or does not correctly remove special characters and also evaluates these as keywords. An excerpt of the results of spaCy can be seen in Table 4.1.

Table 4.1: Keywords generated from the security controls with spaCy.

Nr.	Keywords
1	advanced threats
2	allocation
3	programming
4	system service
5	b. determine
6	developers
7	vectors
8	trustworthy
9	interagency
10	measurable
11	schematics
12	contractual
13	flaw
14	instrumentation
15	trustworthiness
16	vulnerability
17	sa-11(2
18	live
19	sa-3(2
20	substitution

Regarding cluster size, spaCy results in an average of 6 keywords per tag for the security controls and 3 for the properties. The distribution of keywords among the tags is shown in Figure 4.1, where the top 30 tags with the most keywords are shown.

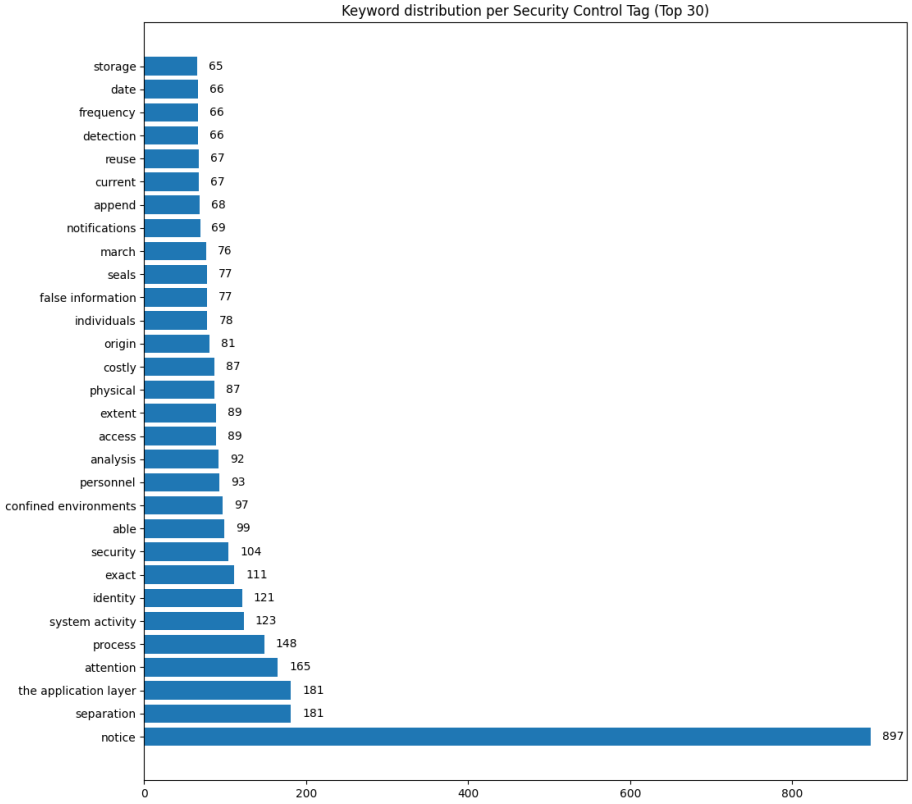


Figure 4.1: Keyword distribution per tag with the spaCy algorithm.

On the graph, it can be seen that one tag stands out very strongly from the others and contains significantly more keywords than all the others. The distribution then drops sharply and approaches the average distribution.

4.1.2 RAKE Algorithm

RAKE extracts a total of 18633 keywords from the 1235 controls. In the case of the 253 properties, 475 keywords are generated. Many actual terms are recognized, including those that consist of several words. However, words are also identified as keywords that do not provide added value. These includes words with special characters, numbers, or stop words that do not allow any statements about the meaning of a text. An excerpt of the results of RAKE can be seen in Table 4.2.

Table 4.2: Keywords generated from the security controls with RAKE.

Nr.	Keywords
1	immediacy
2	h
3	failure
4	f
5	examples
6	disable
7	developer
8	deployed
9	demand
10	defining
11	day
12	confused
13	combination
14	approving
15	anonymous
16	adjust
17	accordance
18	3
19	6 .]
20	au

Regarding cluster sizes, the average cluster size for RAKE is 5 keywords per tag in the case of security controls. The cluster size for the properties of the assets is 3 keywords per tag. Figure 4.2 shows the distribution of keywords among the top 30 security control tags with the most keywords.

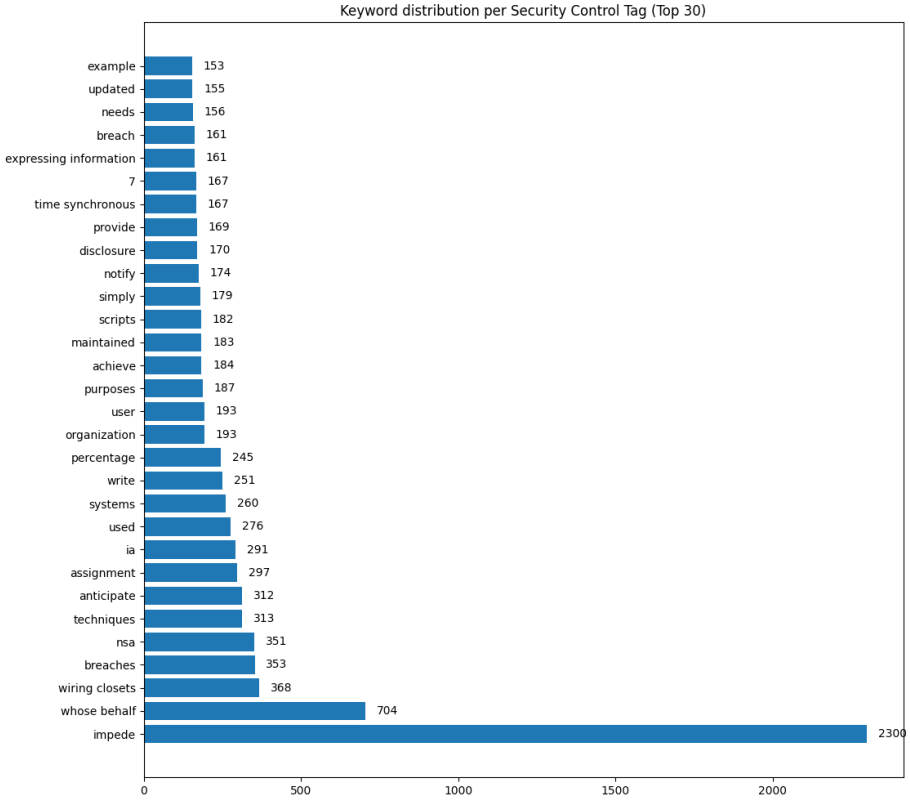


Figure 4.2: Keyword distribution per security control tag with the RAKE algorithm.

RAKE has an even more significant imbalance in distribution compared to spaCy. At least two clusters stand out from the rest and have almost twice and four times as many keywords as the next largest. Generally, the top 30 tags differ way more from the average cluster size, and the subsequent drop in distribution is more substantial.

4.1.3 Yake Algoritm

Yake creates 11274 keywords from the 1235 security controls. In the case of the 235 property tags, a total of 611 keywords were created. The keywords themselves contain practically no invalid words. Both individual terms and word groups that form a term are recognized. Table 4.3 shows an excerpt of Yake's results.

Table 4.3: Keywords generated from the security controls with Yake.

Nr.	Keywords
1	processes
2	principle
3	accomplish
4	allowing only authorized
5	accomplish assigned
6	privilege levels
7	authorized accesses
8	acting on behalf
9	unsuccessful logon attempts
10	logon attempts
11	invalid logon attempts
12	unsuccessful logon
13	logon
14	attempts
15	number
16	attempts is exceeded
17	consecutive invalid logon
18	limit unsuccessful logon
19	invalid logon
20	delay algorithm

In the case of cluster sizes, the use of Yake creates the security controls clusters with an average length of 5 keywords. In terms of clusters for properties, the average size is 3 keywords per cluster. The distribution of the keywords with the Yake algorithm can be seen in Figure 4.3.

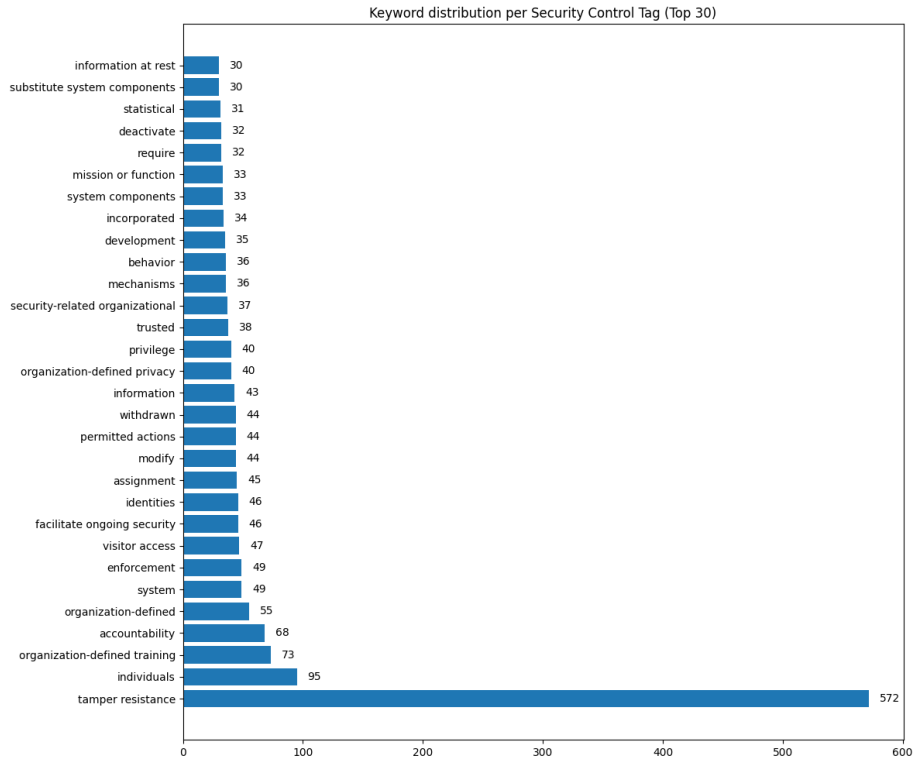


Figure 4.3: Keyword distribution per security control tag with the Yake algorithm.

Also, at Yake, one tag stands out very much from the others and is about 5 times the size of the next largest. After that, the distribution is again close to each other and drops quickly to the average tag size.

4.1.4 KeyBERT Algorithm

Using the KeyBERT library, 1334 keywords are created from the 1235 controls. In the case of property tags, KeyBERT generates 392 keywords over 235 properties. However, the algorithm does not recognize terms that consist of several words. For example, a word like "account management" is not recognized, instead only "account" or "management". Table 4.4 shows an excerpt of KeyBERT's keywords.

Table 4.4: Keywords generated from the security controls with KeyBERT.

Nr.	Keywords
1	authorization
2	privileges
3	account
4	access
5	withdrawn
6	control
7	nan
9	enforcement
10	security
11	enforcing
12	incorporated
13	ac
14	uses
15	marking
16	mp
17	automated
18	restricting
19	privacy
20	unauthorized

From these keywords, clusters with the size of 6 terms on average are created for the security controls. For the properties, the clusters have a size of 4 keywords per cluster. The distribution of the top 30 clusters with the most keywords for the security controls can be seen in Figure 4.4.

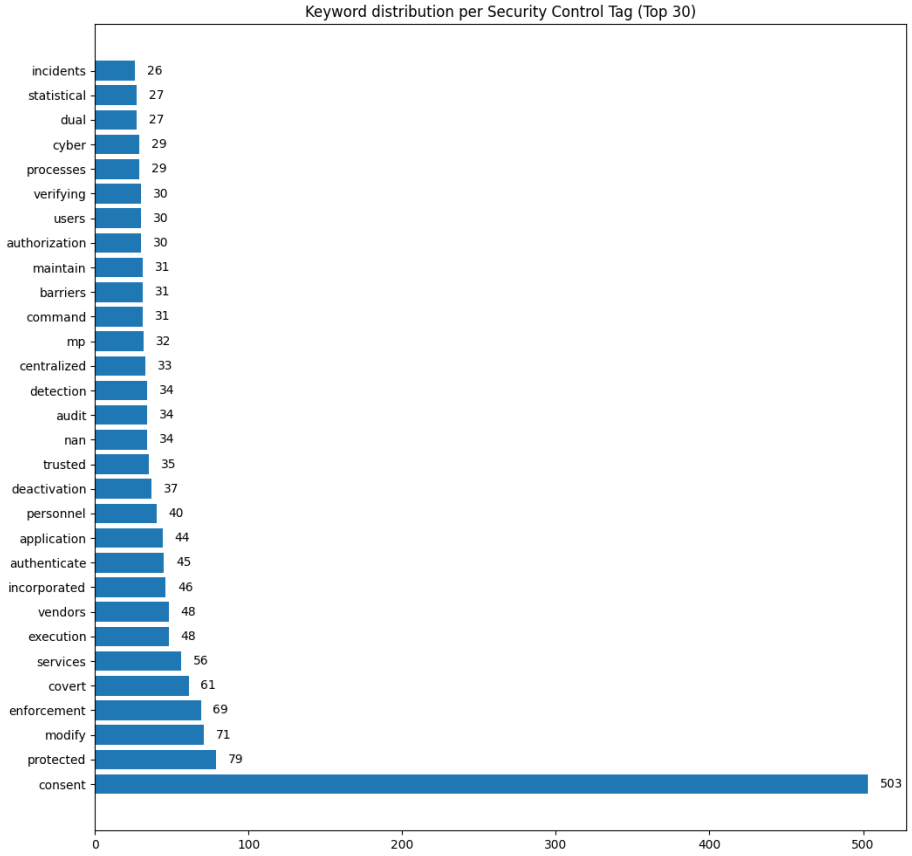


Figure 4.4: Keyword distribution per security control tag with the KeyBERT algorithm.

In KeyBERT, the distribution of keywords is similar to the other algorithms. One tag stands out firmly from the rest, and after that, the distribution drops sharply and approaches the average. The largest tag is about 5 times the size of the following. From this point on, the tags are closer together and no longer have such a large gap.

4.2 Keyword Clustering

The clustering is central for the subsequent matching of the controls. The clusters determine which controls are similar and how many are assigned to the asset during a match. For this reason, an optimal distribution is essential so that a tag does not match everything or has no influence on the assignment. Various methods were implemented to achieve the best possible distribution. The results of these are listed in the following chapter.

4.2.1 Cluster Size

For most cluster algorithms, the number of clusters must be defined in advance. To find a good value, several tests were performed and compared against each other. The number of clusters was set as a fraction of the keywords not to obtain completely different results for the multiple keyword extraction algorithms, each of which generates a different number of keywords. Only tests for clustering the security controls are listed below. The effect of cluster size on properties is similar to that of security controls. For this reason, the properties are not shown.

10 Percent

With a share of 10 percent, a total of 1787 tags are generated. The average share of keywords per tag is 10, with the tag with the most containing 411 terms. The next largest has 136 words. If the tags are matched with the properties, a total of 205 different controls are obtained that were identified as suitable. In the process, 167 of 200 assets were assigned to controls. The various statistics are shown in Table 4.5.

Table 4.5: Statistics for the cluster size of 10 percent.

Metric	Value
Tag Count	1787
Keyword share per tag on average	10
Top 3 of tags with most keywords	411, 136, 119
Top 3 of tags with most associated controls	361, 192, 106
Total amount of matched unique controls	205
Amount of assets with associated controls	167

20 Percent

With a cluster size of 20 percent, a total of 3222 tags were created. The average share of keywords per tag is 5 terms, with the two tags with the most terms having 572 and 95 keywords. In total, 540 different controls could be matched with this cluster size. Of the 200 assets, controls were assigned to 187 of them. The detailed statistics can be seen in Table 4.6.

Table 4.6: Statistics for the cluster size of 20 percent.

Metric	Value
Tag Count	3222
Keyword share per tag on average	5
Top 3 of tags with most keywords	572, 95, 73
Top 3 of tags with most associated controls	376, 188, 98
Total amount of matched unique controls	540
Amount of assets with associated controls	187

30 Percent

With a 30% share of clusters, 4479 tags are created. On average, the tags contain 3 keywords, with the two tags with the most keywords containing 426

and 80 terms, respectively. In total, 563 different controls could be matched with this distribution. Of the total 200 assets, all received at least one control. The complete statistics can be seen in Table 4.7.

Table 4.7: Statistics for the cluster size of 30 percent.

Metric	Value
Tag Count	1911
Keyword share per tag on average	3
Top 3 of tags with most keywords	426, 80, 79
Top 3 of tags with most associated controls	372, 174, 120
Total amount of matched unique controls	563
Amount of assets with associated controls	200

4.2.2 Cluster Algorithms

The KMeans cluster algorithm can be used in 2 variants. On the one hand, with the parameter "randomized", which sets the initial clusters randomly, and on the other hand, with the parameter "KMeans++", which selects the initial parameters with the greatest possible distance to each other, thus minimizing the difference between different runs. The results of the two variants are shown below.

KMeans randomized

KMeans was tested with the "randomized" parameter within five runs. In most cases, the results of the runs differ, with the number of unique matched controls ranging from 381 - 453. However, there are also runs that are identical to each other. The number of keywords is always the same, but the number of tags clustered by KMeans changes. Table 4.8 shows the results of the first run.

Table 4.8: Results of the first run with KMeans randomized.

Metric	Value
Average controls per tag (asset properties)	7
Average keywords per tag (asset properties)	5
Average controls per tag (controls)	4
Average keywords per tag (controls)	5
Number of tags (asset properties)	123
Number of tags (controls)	3218
Matched unique controls	442

Figure 4.5 shows the distribution of keywords from the top 30 clusters with the most terms in the first run.

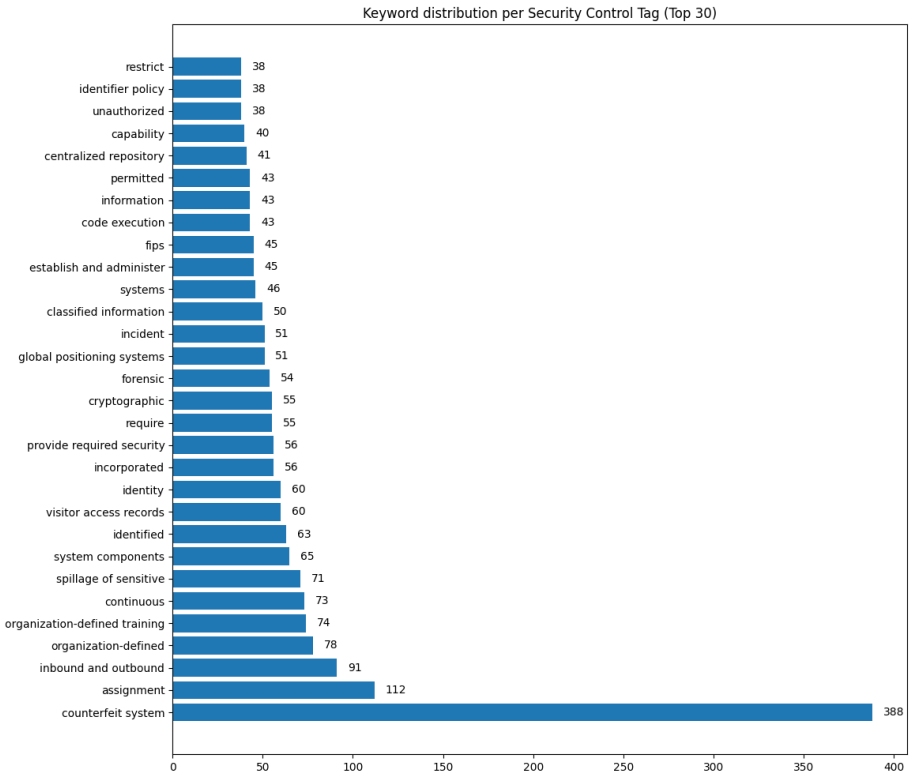


Figure 4.5: Keyword distribution per security control tag with KMeans randomized in the first run.

To illustrate the difference, the last run is also presented as a comparison. In this run, the number of controls increases to 453, whereby more tags are created in the case of the controls. The exact statistics can be seen in Table 4.9.

Table 4.9: Results of the last run with KMeans randomized.

Metric	Value
Average controls per tag (asset properties)	6
Average keywords per tag (asset properties)	4
Average controls per tag (controls)	4
Average Keywords per tag (controls)	5
Number of tags (asset properties)	123
Number of tags (controls)	3246
Matched unique controls	453

The number of tags for the controls has increased to 3246. However, the average distribution has remained the same. The situation is different for the properties, where the number of tags has remained the same, but the average distribution has changed and dropped by one in each case. The distribution of keywords for the controls can be seen in Figure 4.6.

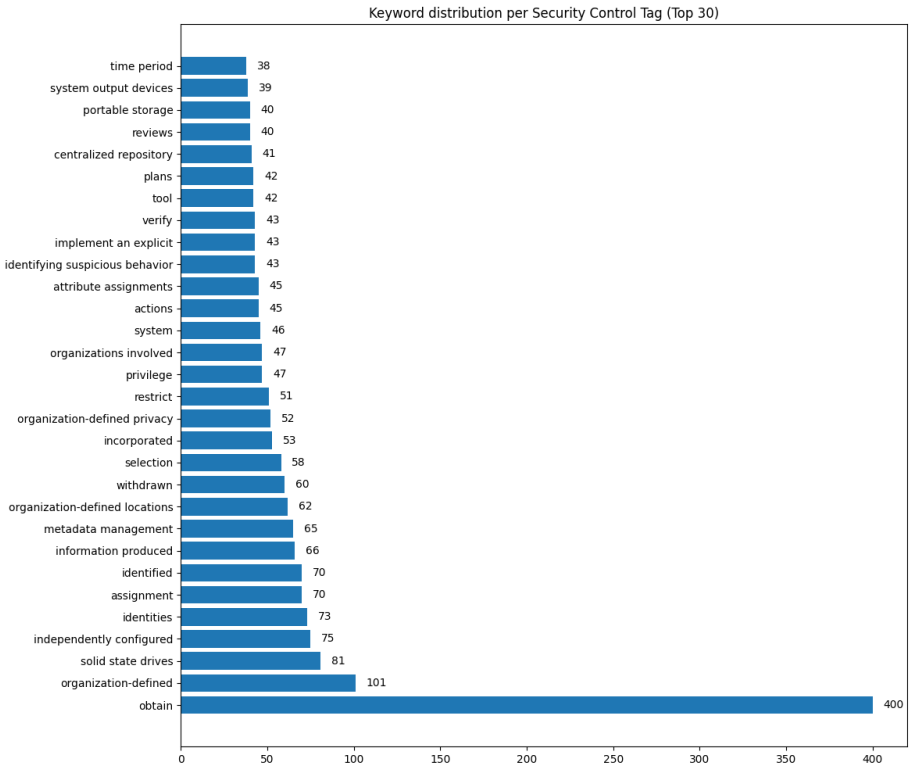


Figure 4.6: Keyword distribution per security control tag with KMeans randomized in the last run.

Figure 4.6 already shows the differences in the distribution of keywords between the first and the last run. Many of the tags have a different number of keywords, which in some cases has also led to them receiving different labels through the topic modelling.

KMeans++

For KMeans++, there are also differences in the individual runs. The number of matched, unique clusters ranges from 412 - 540 and is thus somewhat higher than with the randomized parameter. Table 4.10. shows the statistics of the first run with KMeans++.

Table 4.10: Results of the first run with KMeans++.

Metric	Value
Average controls per tag (asset properties)	7
Average keywords per tag (asset properties)	6
Average controls per tag (controls)	4
Average keywords per tag (controls)	5
Number of tags (asset properties)	123
Number of tags (controls)	3206
Matched controls	412

Figure 4.7 shows the distribution of keywords for the controls in the first run with KMeans++.

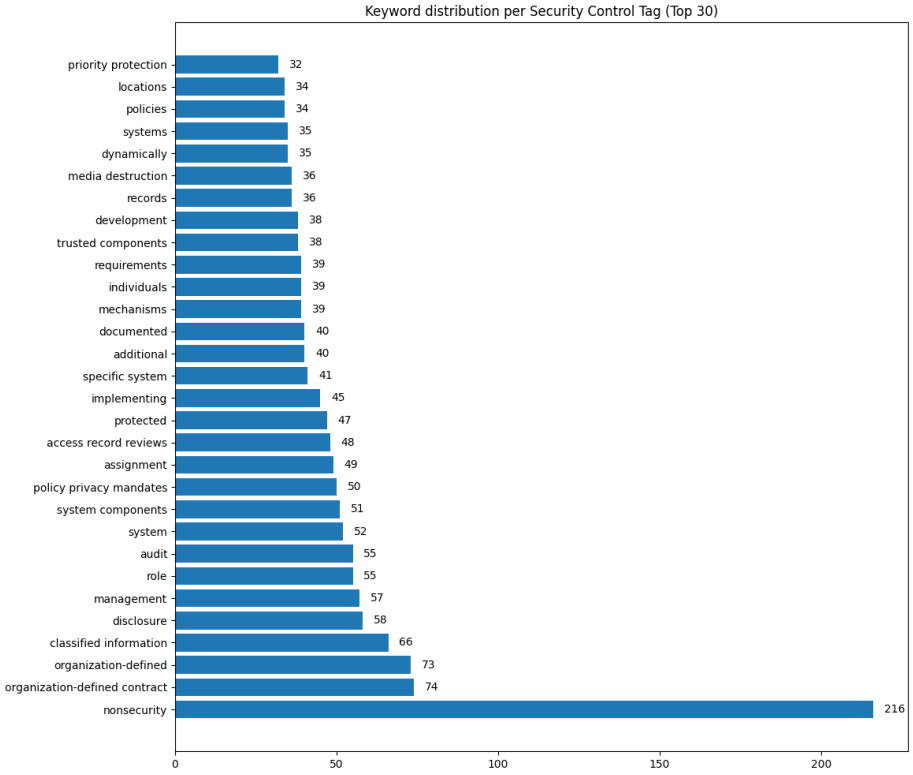


Figure 4.7: Keyword distribution per security control tag with KMeans++ (first run).

In the last run, the number of matched, unique controls increased to 471. However, the number of tags created is almost identical. Table 4.11 shows the exact statistics of the last run with KMeans++.

Table 4.11: Results of the last run with KMeans++.

Metric	Value
Average controls per tag (asset properties)	5
Average keywords per tag (asset properties)	4
Average controls per tag (controls)	4
Average keywords per tag (controls)	5
Number of tags (asset properties)	123
Number of tags (controls)	3207
Matched controls	471

The distribution of the property tags has changed, but that of the controls has remained identical. Only the number of tags created for the controls has increased by 1. Figure 4.8 shows the distribution of keywords for the controls in the last run.

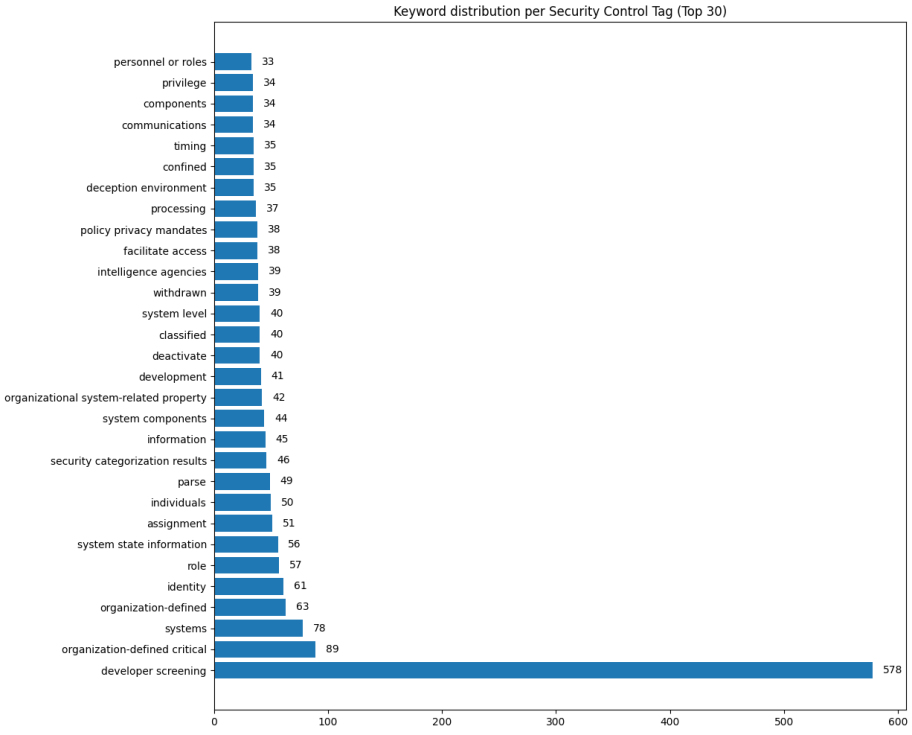


Figure 4.8: Keyword distribution per security control tag with KMeans++ (last run).

The distribution of the top 30 tags in the last run differs greatly from that in the first run. The largest tag has almost twice as many keywords in the last run as in the first. After that, the two are similar again, but there are slight differences almost everywhere and some of the tags have different names.

4.2.3 Cluster Cleaning

Two different variants of the tolerance values were tested during cluster cleaning. On the one hand, a low tolerance of 200% for the keyword size and 300% for the control number. On the other hand, a high tolerance of 600% and 800% was tested. The results are shown in the two sections below.

Low Tolerance

With the low tolerance values, the number of tags drops from 3222 to 2902. So, in total, 320 tags were deleted. The distribution of the top 30 controls with the most keywords can be seen in Figure 4.9.

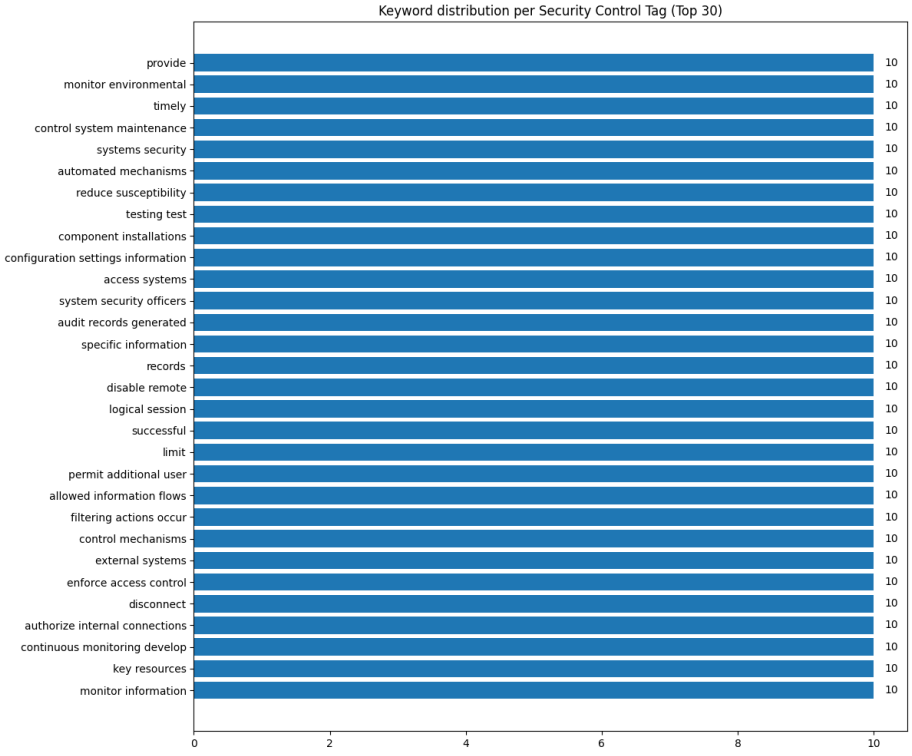


Figure 4.9: Keywords distribution after cluster cleaning with low tolerance.

The cluster sizes have been greatly reduced, and thus all clusters in the top 30 range have the same size. The average number of keywords per cluster is 5, so all clusters in the top 30 are at the maximum possible tolerance, as the graph shows. Similar behavior can be seen in the distribution of the controls among the clusters, visible in Figure 4.10.

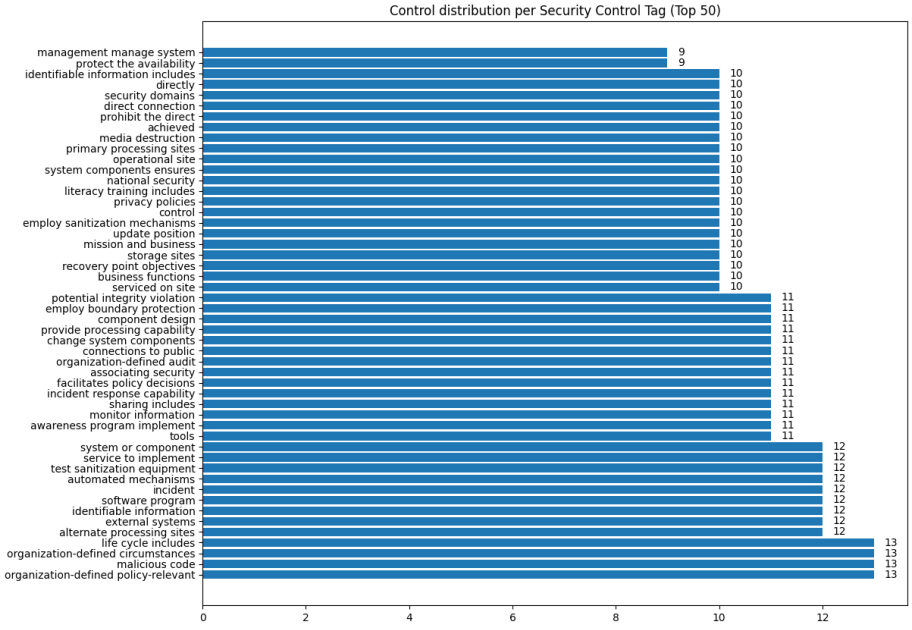


Figure 4.10: Control distribution after cluster cleaning with low tolerance.

Also, in this case, the distribution is a lot closer together. Although it is not quite as close as with the keywords, the differences are not significant. This change is also noticeable in the number of matched unique controls. From the previous 494 controls that were identified, now only 274 remain. The number of total matches has decreased massively from 8667 to 3284. Also, a few assets no longer have controls assigned to them. Of the total 200 assets, 172 have been assigned. This is a reduction of 28 assets with assignments compared to the state without cluster cleaning.

High Tolerance

With the high tolerance values, the reduction of tags is no longer quite as high as with the low tolerance values. The number of tags has reduced from 3222 to 3176. So a total of 46 tags have been deleted. Figure 4.11 shows the distribution of the top 30 clusters.

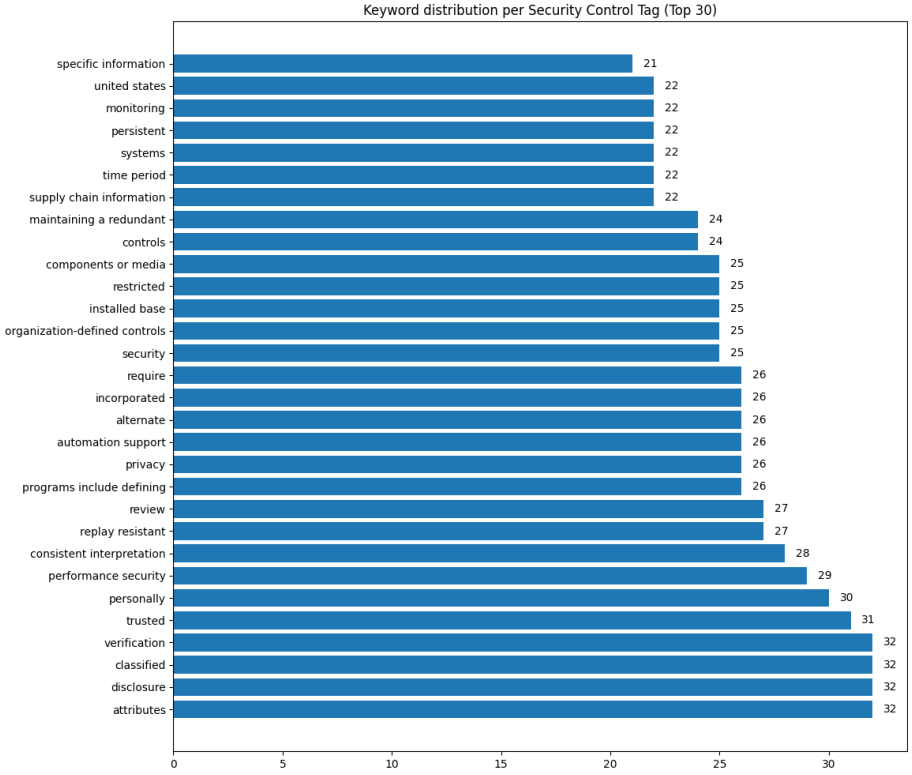


Figure 4.11: Keywords distribution after cluster cleaning with high tolerance.

In this case, the distribution of keywords is no longer quite as constrained as with the low tolerance values. Nevertheless, the individual cluster sizes are close to each other, and there are no large excursions. The largest 4 clusters are at the tolerance limit, and after that, the size decreases steadily. Similar behavior can be seen in the distribution of the controls to which the tags are assigned, as shown in Figure 4.12.

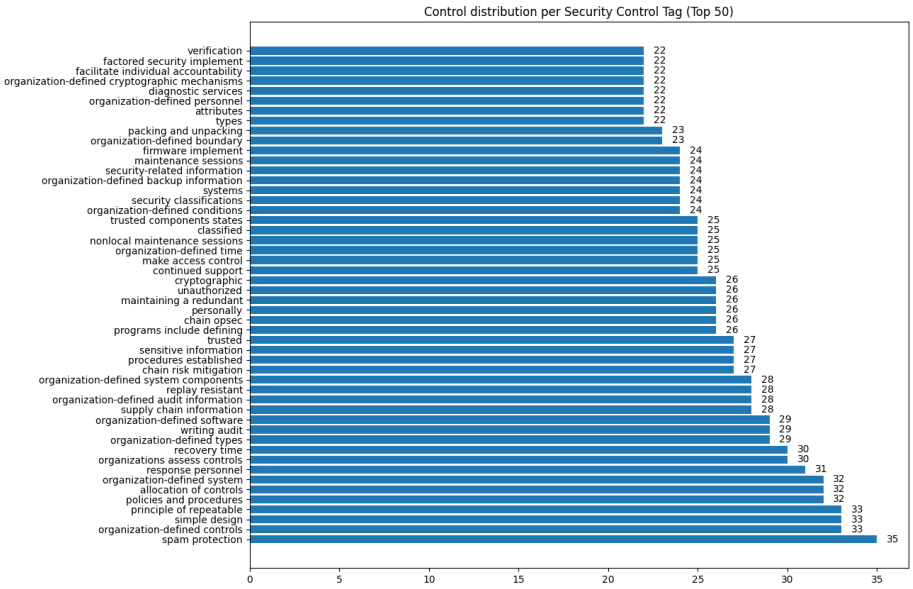


Figure 4.12: Control distribution after cluster cleaning with high tolerance.

The tag "spam protection" is assigned to 35 different controls and thus has the most assignments. After that, the number also decreases steadily, but there are no large gaps, and the distributions remain close.

This cluster adjustment results in the number of matches of unique controls decreasing by 171 to a total of 323. The total number of matches is reduced from 8667 to 5599. This reduction also results in some assets not receiving any controls for the high tolerance values. Of the total 200 assets, 22 no longer have any control assignments after cluster cleaning. This leaves 178 assets that have been assigned controls.

4.3 Topic Detection

For topic modeling, two approaches were tested that search a topic for the cluster/tag. The first approach uses the LDA algorithm, which searches for a suitable topic from a given text. The second approach extracts the word that occurs most in the cluster and uses it as the topic.

4.3.1 Gensim LDA Algorithm

The LDA algorithm has the advantage that it can also recognize related terms, which consist of several words. This allows the user to create more understandable terms in some instances. However, these must still fit the cluster and describe it as well as possible. Table 4.12 shows an excerpt of two clusters whose topics were created with LDA.

Table 4.12: Topics created with the LDA algorithm.

monitoring includes scanning	security and privacy
monitoring	privacy policies
monitoring activities	security and privacy
incident monitoring track	security
monitoring track	privacy plans
access monitoring	privacy
access monitoring includes	privacy requirements
monitoring and tracking	information security
asset monitoring	privacy compliance
vulnerability monitoring	compliance
monitoring tools	privacy representatives require
system monitoring	privacy representatives

The table shows a list of keywords per column, all grouped into a cluster. The heading of the column is the topic determined by LDA for this cluster. The right cluster consists mainly of two terms that should be covered by the Topic. LDA chooses the topic "security and privacy", which covers the terms relatively well. However, LDA tends to select terms that are too long. This can be seen in the cluster on the left. In this case, "monitoring includes scanning" is selected, but for this cluster, only "monitoring" or maybe "monitoring and tracking" would be more appropriate.

4.3.2 Most used Word Algorithm

This algorithm has the advantage that a relevant word is always used for the clusters since it is extracted from the frequency in the cluster itself. However, it does not mean that this word is always the most understandable for the user. Table 4.13 shows an extract of two topics with their keywords.

Table 4.13: Topics created with the most used word algorithm.

life	risk
system development life	risk management programs
development life	risk assessment
life	risk
information life	change in risk
development life cycle	organizational risk
life cycle	privacy risk
information life cycle	organization-wide risk
system life cycle	privacy risk tolerance
life cycle activities	risk monitoring
life cycle process	risk responses
life cycle includes	risk tolerance
cycle	risk tolerance
rigorous system life	risk executive
cycle includes information	position risk
life cycle stage	risk designation

It can be seen that in the case of the risk category, the assignment of the topic by the algorithm may be appropriate, and the word "risk" in this case, describes the cluster relatively well. However, the term "life" is not sufficient for the other cluster. In this case, "life cycle" would have been more appropriate, but only single words are extracted, and related words are not recognized as such. Furthermore, "life" and "cycle" appear as independent words in the cluster.

4.4 Matching Process

Two main approaches are implemented for the actual matching. One is tag matching which requires some preprocessing steps like keyword extraction, clustering, etc., to get the tags. On the other hand, the approach in which the texts are directly compared with each other eliminates the preprocessing steps, provided that qualitatively sufficient description texts are available. The results of both methods are presented in this section.

4.4.1 Tag-based Matching

For matching, tags and their keywords are compared with each other. The number of matching keywords forms the matching ratio. This is the proportion of matching keywords in relation to the total number of keywords. If the ratio is above a certain threshold, a match is made. Figure 4.13 shows the distribution of the ratios as a pie chart. A total of 1094 matches were recorded on tag level, resulting in 323 unique controls and 5599 total matches. It is important to note that the matches on the tag level are lower than the total matches because a match with a tag can lead to an assignment with multiple controls (a tag is usually assigned to several controls).

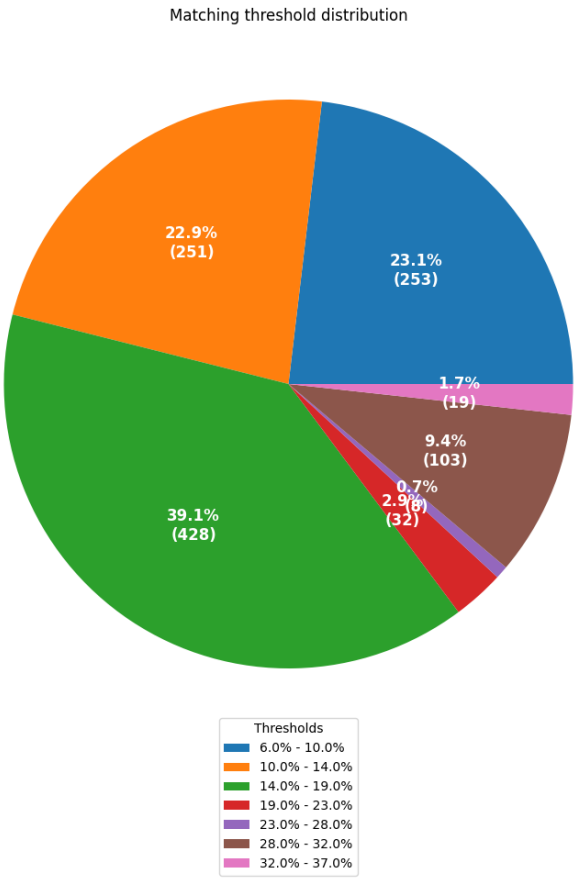


Figure 4.13: Tag-based matching ratio distribution.

The threshold during the tests is set to 5%. Two tags must therefore match by at least 5% in order to match. The graph shows seven areas in which the ratios of the matches are distributed. With 39.1%, most matches happened in the range of 14%-19% (green). This means that out of 1094 matches in total, 39.1% (428 matches) have matched with a ratio of 14% - 19%. So 14% - 19% of the keywords are matched by the compared tags. In the second place, 23.1% (253) of the total 1094 matches have matched with the 6% - 10% ratio. In third place, these most matches were recorded within a 10% - 14% ratio, namely 22.9% (251) of the total 1094. The remaining 14 percent of the 1094 matches at the tag level possess a match ratio of over 19 percent. There were 162 matches between property tag and control tag, with the keywords at least 19% identical. If the

threshold were raised to 10%, we would lose about 253 matches, which would minimize the number of controls found. If too few controls were identified, the threshold could be lowered, and a larger tolerance could be achieved.

4.4.2 Description Text based Matching

For description-based matching, the results depend on the threshold value. With a threshold value of 10%, a total of 15808 matches are recorded, resulting in a total of 833 unique controls. An overview of the distribution with the actual match values (everything above the threshold value is matched) can be seen in Figure 4.14.

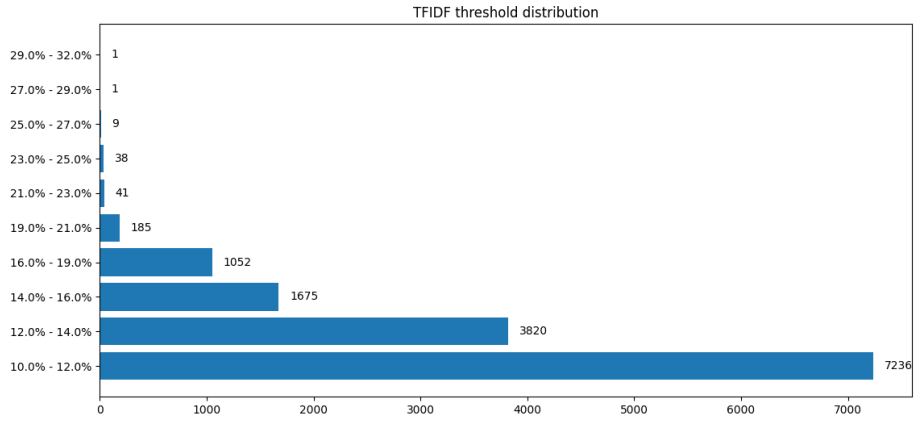


Figure 4.14: Description text based matching threshold distribution.

It is evident that most matches occur within the 10% - 14% range. With a total of 7236 matches, most of the compared texts are 10% - 12% similar. This is followed by the 12% - 14% mark, which has 3820 matches, already significantly less but still a few more than the others. Adjusting the threshold to 15% reduces the number of matches in this case. With a total of 2488 assignments, the number of unique controls is reduced to 370, which is slightly more than a third of the previous value. If the threshold is increased even further to 20%, the number of matches is reduced again by quite a bit. As can be seen in Figure 4.14, above this value, only about 79 matches take place. Accordingly, only 27 unique controls are found which have such a high similarity with the assets. However, if the threshold is set to 5%, a few more matches can be found. There are 108608 matches, of which 1036 unique controls are identified. In this case, most matches occur in the 5% - 8% level, where 72658 matches are found. This is again a significant increase compared to a 10% threshold.

4.5 Validation of Implementation

A distinction is made between the implementation and acceptance tests. In the implementation tests, the application's functionality is examined, and its results

are presented. In the acceptance tests, the software is tested with regard to the defined requirements, and it is checked for which user types it is suitable for.

4.5.1 Implementation Tests

The following section presents the results of the implementation tests. It is divided into two sections, where on the one hand, the results with the tag-based approach are presented and, on the other hand, the results with the description text comparison approach.

Tag-based Matching

The following are the results using the tag-based approach. Table 4.14 shows general statistics obtained with the data and parameters used in the application.

Table 4.14: Summary of the results with the determined parameters.

Description	Value
Total matched unique controls	323
Total matched controls	5599
Number of assets with controls	178
Number of assets without controls	22

A total of 323 out of 1235 unique controls were recognized and assigned to the captured assets. Some of the controls have been assigned to multiple assets. Looking at the total number of matches, the number of controls matched increases to 5599. This means that, on average, each control that was matched was assigned about 17 times, and an asset has an average of 28 controls assigned to it. Of the 200 assets, 178 have received at least one control, which means they remain 22 assets without a control. Figure 4.15 shows the distribution of the top 10 assets with the most control assignments.

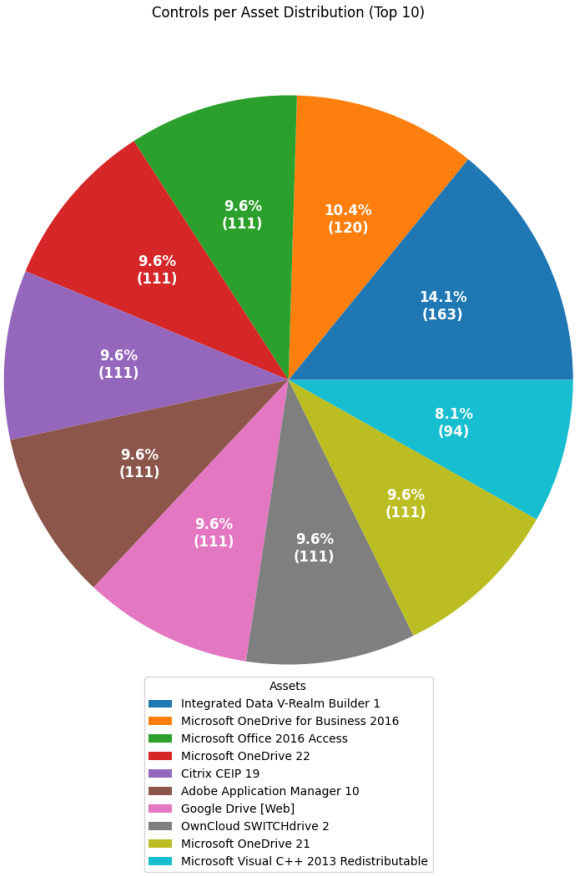


Figure 4.15: Top 10 controls with the most assets assigned.

The asset with the most assignments is "Integrated Data V-Realm Builder 1", to which 163 controls were assigned. This accounts for 14.1% of the top 10. In second place is "Microsoft OneDrive for Business 2016", which has a total of 120 assignments and thus accounts for 10.4% of the top 10. In third place comes "Microsoft Office 2016 Access", with a total of 111 assignments and a 9.6% share of the top 10. The distribution for the following assets remains relatively the same with 6 more showing a 9.6% distribution. This is followed by the last asset, which reaches at least an 8.1% distribution with 94 assignments.

A manual verification was also performed to check if the assigned controls matched the assets. However, since the amount of data is tremendous, only a subset of assets was selected. The selection was made randomly. Table 4.15 shows the results of this verification.

Table 4.15: Evaluation of suitable controls for assets.

Asset	Total assignments	Incorrect assignments
Google Drive [Web]	111	26
Cisco AnyConnect VPN Client	29	3
Microsoft Teams	7	0
Microsoft OneDrive for Business 2016	120	32
SRV-APP-T-202	11	2
Citrix ICA Client	27	0
Microsoft Local Administrator Password Solution 6	35	5
Microsoft Windows 10 Enterprise	10	0
Microsoft SharePoint Online [Web]	29	0
VideoLAN VLC Media Player	34	11
Total	413	79

The table shows that a total of a subset of 10 different, randomly selected assets were manually reviewed. This is a total of 413 assignments that had to be checked. Of these 413 assignments, 79 were identified as incorrect. This means that the assigned control does not match the asset. Thus, 334 assignments were executed correctly in this subset. Thus, the application executed about 80% of the assignments correctly in the analysis with the tag-based approach. However, it should be noted that many of the assets have been assigned the same controls. This is because they are mainly general software-related controls that could be assigned to any software. The same applies to hardware assets. These have primarily hardware-related controls assigned to them (such as virtualization), and there are no major differences between the assets.

Description-based Matching

In the case of the description text comparison approach, a threshold of 15% was chosen, and the algorithm was tested on the given data. Table 4.16 shows an overview of the data from the tests with this methodology.

Table 4.16: Final results with the description-based matching algorithm.

Description	Value
Total matched unique controls	370
Total matched controls	2488
Number of assets with controls	159
Number of assets without controls	41

A total of 370 unique controls were matched. This represents a share of 28% of the total of 1235 available controls. The total number of matches is 1981. On average, each control is assigned to 6 different assets. 159 assets have received at least one control. So 41 assets have not been assigned a single control. The distribution of the matches in detail can be seen in Figure 4.16.

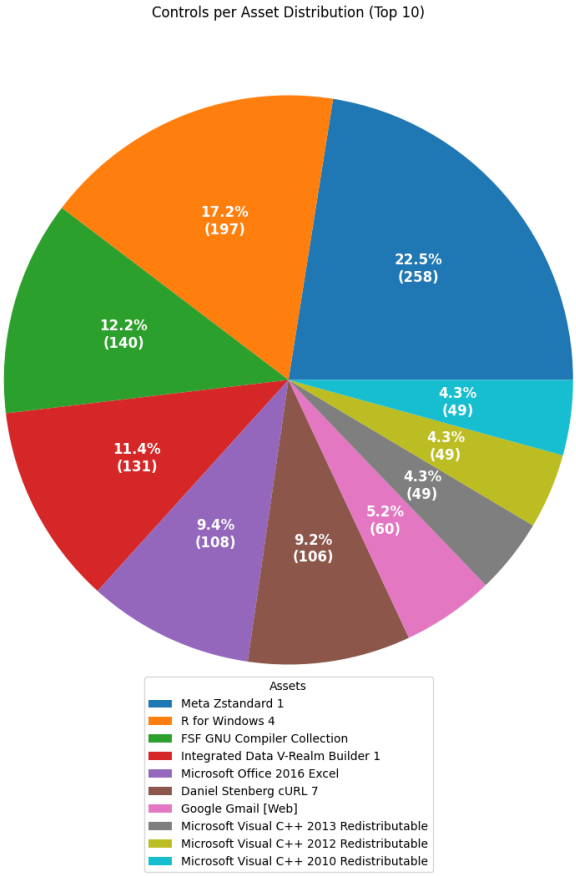


Figure 4.16: Controls per asset distribution with description-based matching.

Most controls were assigned to the "Meta Zstandard 1" asset, with 258 different controls in total. This is a share of 22.5% in the top 10 assets with the most assignments. "R for Windows 4" follows with 197 controls and a share of 17.2%. In third place is "FSF GNU Compiler Collection", with 140 controls and a share of 12.2%. Finally, the last two places in the top 10 are taken by "Microsoft Visual C++ Redistributable" with 49 controls and a share of 4.3%.

When the individual allocations are considered in detail, the results are shown in Table 4.17. In this case, it is again true that the selection of the assets to be tested was random, and thus a subset of the total data was tested.

Table 4.17: Evaluation of suitable controls for assets.

Asset	Total assignments	Incorrect assignments
Google Drive [Web]	3	0
Cisco AnyConnect VPN Client	32	2
Google Gmail [Web]	60	10
Microsoft OneDrive for Business 2016	3	0
SRV-APP-T-202	5	0
Citrix ICA Client	8	0
Microsoft Local Administrator Password Solution 6	3	0
Microsoft Windows 10 Enterprise	2	0
VideoLAN VLC Media Player	19	7
Microsoft Windows Media Player 12	12	3
Total	137	22

For the description-based matching, 10 assets were again selected and manually checked. As far as possible, the same assets were selected as for the tag-based matching in order to obtain a possibility of comparison. If no control was assigned to an asset, it was replaced by another, random one. In total, 137 assignments were recorded. Of these, 22 were identified as incorrect. In total, 115 assignments were classified as correct, which corresponds to a share of 83%. Even with this approach, however, the same assignments were often made to different assets.

Comparison of the Two Approaches

With the two approaches implemented, a direct comparison can now be made, and it can be checked whether the same assets are found. Table 4.14 and Table 4.16 already show the differences. Many of the controls that could be identified by tag-based matching were not found with description-based matching. Especially with assets such as the "Microsoft OneDrive for Business 2016", there are massive differences. With its 2488 matches, description-based matching can generally only show about half as many as tag-based matching. This circumstance is also reflected in the detailed analysis, where many controls that should be matched are not identified. However, if a suitable control is found, this assignment is usually correct, as Table 4.16 shows. On the other hand, tag-based matching was also unable to find some controls that would actually be relevant. For example, for the asset "Cisco AnyConnect VPN Client", the "Remote Access" control was only assigned by description-based matching, but not by tag-based matching. There are generally some similarities between the two methods for some assets, but also many differences. Both description-based matching and tag-based matching can identify some correct controls that are not found by the other algorithm. It is difficult to say how high the number of controls not found is, but based on the comparison of the two algorithms and manual random analysis, the error rate, in this case, is higher than the number of incorrectly assigned controls from Table 4.14 and Table 4.16.

4.5.2 Acceptance Tests

For the acceptance tests, the application's results and functionalities were considered regarding the defined user profiles. Below is a list for each user profile of how they can benefit from the application and what restrictions they should expect regarding their requirements.

Alex, CISO of a big company

For Alex, the Security Control Management Tool can provide a valuable complement to finding suitable security measures for the company. Especially if an existing asset management system is already in place, its data can be used to find proper controls with SCM. The tool is able to make a further recommendation, which can be further evaluated by an already experienced security team and, depending on this, can be included in the security measures catalog.

Matthias, CEO of a startup

Matthias' company is very fresh and lacks know-how in the security area. In this case, the security control management tool provides a first insight into possible security measures for the company. Provided that the requirements and necessary assets can be defined correctly and in detail. For a final decision and a correct implementation of the measures, however, further experience is required. In this case, more than SCM is required as the only decision-making entity.

Jens, IT manager of an SME

Like Alex, Jens has access to an experienced security team. For these requirements, SCM can offer a helpful addition. Especially when further measures need to be taken or every eventuality needs to be considered, the security management tool can provide important inputs. Limitations arise when the assets of the company are not sufficiently documented. If the company's small IT team cannot fully capture them, gaps that cannot be covered may occur. A comprehensive description and characterization of the assets is of great advantage for the best possible recommendation by SCM.

Chapter 5

Discussion

This work shows a possible approach to how security control can be automatically matched to a given IT system. The product of this work is an application that allows capturing its system infrastructure in the form of assets. By specifying the assets using properties or a description, the function of the assets can be described in detail. From this, the software automatically determines the appropriate NIST security controls and assigns them to the assets.

5.1 Project Goals

At the end of this work, the goals defined at the beginning of this project are reviewed.

1. *Development of a tool for the acquisition of the current system state.*
   Capturing the current system state is possible by creating or importing assets. Its functionality can be further specified by assigning properties. It is also possible to connect to an asset management tool by exporting the data to an Excel file and then importing it into the tool using the file import functionality. More complex relationships between the assets are currently only limited mappable and offer an extension possibility for the tool. Likewise, a direct connection to an asset management tool could be implemented without an intermediate stop with an Excel file.

2. *Automated derivation of appropriate security controls from the specified system state*
   The various tests have shown that the application is able to find suitable security controls for the assets captured. Most of the assets from the test data were assigned at least one control. It is also possible to define the importance of an asset via its impact and thus obtain stricter and more detailed controls. The result can be further expanded and optimized by changing various parameters.

5.2 Security Control Managment

The security control management software that resulted as a product of this work is able to find appropriate controls for given assets. It has been shown that

the approach with NLP works, and it is possible to identify the corresponding security controls of the NIST standard for an arbitrary number of assets and asset specifications. Nevertheless, a few points still do not work optimally and can be further improved.

One of the problems is the relatively significant differences that result from changing specific parameters. As the tests have shown, even different runs can give different results because the clusters are created differently. Likewise, the algorithm or methodology choice decides whether certain controls are matched or not. The tests with the NLP algorithms have shown that different libraries deliver very different results and differ enormously in quality. Also, the sizes of the clusters make a difference. It is difficult to determine which value is optimal in this case. Many different runs with different data would be necessary to detect and compare the effects. However, these problems only occur in some cases. Certain controls are always assigned to the same assets relatively reliably. The choice of algorithm or parameter does not play a major role. However, this behavior does not occur often enough at the moment. Further measures to clean up the clusters and keywords could help here.

A further problem that aggravates the above circumstances is the need for more evaluability. The results can be evaluated with specific metrics and manual methods, but more is needed for a clear insight into the performance. The distributions of the clusters, keywords, and controls can give an indication of whether the techniques used perform well, but they are not an unambiguous measure. What is missing is the possibility of making a clear statement about whether a control fits an asset. In the case of security controls, however, this is challenging to achieve since an inevitable subjectivity also plays a role in such an assignment. Which controls exactly fit a given system also depends on the definition of the responsible persons. This can lead to the exact system specifications having different security controls. Furthermore, with the available data, there is no way to automatically assess whether an assignment is correct. In this case, a manual assessment was necessary. Still, with 200 assets and 1235 controls, this is a very time-consuming process, and to do it entirely correctly, it would have to be run several times with different data and parameters. But even then, a subjective perception would flow in again. A solution would be collecting data from existing systems with assigned security controls. The assignments of the system could then be compared with the real assignments, and a precision and recall value could be calculated. The larger the data, the less noise due to subjectivity would occur.

The data is generally a point that can be improved. The security controls are very well defined by NIST, which should be sufficient for the NLP. However, a better quality of the assets is necessary. The dataset available in this work is very extensive, with many different assets divided into software and hardware. However, the specification of these is not sufficient. Different properties are defined for each asset, but these are only described very briefly in a few words. This is a bit too little for an NLP that is supposed to work reliably. Furthermore, certain properties that could improve the evaluation quality are missing. Additional properties such as impact or a detailed description of an asset would be desirable. Furthermore, although there are many data points, they are very similar to each other, which was also shown in the evaluation. The software assets were often assigned the same controls simply because they are software. Individual characteristics that require unique controls are scarce,

and it is difficult to say whether the application would recognize them and what problems still exist. The results can be considered satisfactory based on the given data. Although 323 controls out of 1235 are only about a quarter, a total of over 5500 matches were recorded, and almost every asset received at least one control. Due to the high similarity of the data points, a similarity in the controls is also to be expected, and the values obtained show that many controls were matched, but often the same controls were used. For a reliable evaluation, however, broader data sets are definitely needed, which offer better comparison possibilities and for which more different controls should be matched.

What can also be improved is the performance. Processing the data for matching requires many different steps. With large amounts of data, this process is very time-consuming, and it takes a long time to get the final result. The NLP algorithms and clustering offer little room for improvement. However, the assignment of keywords and clusters to controls and assets can be optimized as well as their evaluation. For the PoC in the context of this work, the current state is sufficient. However, this circumstance should be taken into account in a later extension.

5.3 Future Work and Ideas

Although the automated assignment of security controls to given assets already works with the product of this work, there is room for improvement and enhancement. On the one hand, there are the points that have already been mentioned. A more comprehensive data set would help in many areas. With more detailed asset specifications, a better analysis by the NLP is possible, which makes the allocation more accurate. A data set with already existing allocations would be helpful for better evaluation possibilities. This would allow the tool's results to be compared, making it easier to identify and eliminate problem areas.

An entirely different endeavor would be to try a different approach. Instead of using an NLP methodology for the assignment, Deep Learning could be used for this purpose. This would require a data set containing assets that already have assigned security controls. This data could then be used to create a neural network that establishes a relationship between certain assets and the controls. However, a large amount of data is required for this model. Furthermore, it is questionable how large the influence of subjectivity and the variation of asset specifications is on the quality of the model. The work of Seifeddine et al.[23] goes in this direction. Based on historical data, they try to find the appropriate controls for a new system using machine learning. The researchers achieved a precision value of 93%. However, the assignments are based on security requirements, which must first be determined.

Another approach is to focus more on linguistic rules, as is done in Li's [24] work. Here, an attempt is made to identify security requirements employing linguistic rules and machine learning. The security requirements are divided into different categories such as "threat-based", "asset-based", etc., whereby a different structure of the linguistic rules is defined for each category. This approach could be used to determine security requirements using the methods in Li's work. These are then used to identify potentially suitable controls. Machine learning could also be used for this step, or the model could be extended with the linguistic rules.

Chapter 6

Directories

6.1 Glossary

Term	Description
PoC	Stands for proof of concept and describes an application that proves the feasibility of a project.
ISO	ISO stands for International Organization for Standardization, an organization that develops and publishes a variety of international standards in the fields of technology, management and manufacturing.
NIST	NIST stands for National Institute of Technology, an organization that develops and publishes standards at the national level in the USA.
BSI	BSI the Federal Office for Information Security is the cyber security authority in Germany. It is responsible for the security of federal authorities and develops various standards in this context.
CIS	CIS the Center for Internet Security is a non-profit organization that develops and publishes best practices for securing IT systems and data.
Clustering	Clustering is used to try to identify similarity structures in data sets. Similar data points are organized into groups, called clusters and the assignment process itself is called clustering.
Cosine Similarity	Cosine Similarity is a measure of the similarity between two sequences of numbers. For vectors, the Euclidean dot product is calculated, which returns a value between 0 and 1.
Local Maxima	The local maximum is the point in a function at which the function in its immediate vicinity can no longer assume a larger value.

6.2 Bibliography

[1] 70[accessed 18 January 2023]. [Online]. Available: https://www.securitymagazine.com/articles/96558-70-of-security-pros-find-security-hygiene-and-posture-more-challenging

[2] Swissgrc. [accessed 27 September 2022]. [Online]. Available: https://swissgrc.com/

[3] Bigid. [accessed 27 September 2022]. [Online]. Available: https://bigid.com/

[4] Standardfusion. [accessed 27 September 2022]. [Online]. Available: https://www.standardfusion.com/

[5] Security and privacy controls for information systems and organizations. [accessed 08 September 2022]. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final

[6] Iso/iec 27002:2022. [accessed 08 September 2022]. [Online]. Available: https://www.iso.org/standard/75652.html

[7] Information security, cybersecurity and privacy protection. information security controls. [accessed 08 September 2022]. [Online]. Available: https://knowledge.bsigroup.com/products/information-security-cybersecurity-and-privacy-protection-information-security-controls-1/standard?_ga=2.122679371.1178444906.1674133514-1564155242.1673512293

[8] Cis critical security controls. [accessed 08 September 2022]. [Online]. Available: https://www.cisecurity.org/controls

[9] rake-nltk. [accessed 5 October 2022]. [Online]. Available: https://csurfer.github.io/rake-nltk/_build/html/index.html

[10] spacy. [accessed 5 October 2022]. [Online]. Available: https://spacy.io/

[11] Yake! [accessed 5 October 2022]. [Online]. Available: http://yake.inesctec.pt/

[12] Keybert. [accessed 5 October 2022]. [Online]. Available: https://maartengr.github.io/KeyBERT/

[13] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," in *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14. MIT Press, 2001. [Online]. Available: https://proceedings.neurips.cc/paper/2001/file/296472c9542ad4d4788d543508116cbc-Paper.pdf

[14] L. Barnard and R. von Solms, "A formalized approach to the effective selection and evaluation of information security controls," *Computers Security*, vol. 19, no. 2, pp. 185–194, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404800878293

[15] I. Yevseyeva, V. B. Fernandes, A. van Moorsel, H. Janicke, and M. Emmerich, "Two-stage security controls selection," *Procedia Computer Science*, vol. 100, pp. 971–978, 2016, international Conference on ENTERprise Information Systems/International Conference on Project MANagement/International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN / HCist 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050916324309

[16] T. Neubauer, A. Ekelhart, and S. Fenz, "Interactive selection of iso 27001 controls under multiple objectives," in *Proceedings of The Ifip Tc 11 23rd International Information Security Conference*, S. Jajodia, P. Samarati, and S. Cimato, Eds. Boston, MA: Springer US, 2008, pp. 477–492.

[17] P. C. Carol Peters, Martin Braschler, *Multilingual Information Retrieval: From Research To Practice*. Springer Berlin, Heidelberg, 2012.

[18] django. [accessed 6 October 2022]. [Online]. Available: https://www.djangoproject.com/

[19] React. [accessed 6 October 2022]. [Online]. Available: https://reactjs.org/

[20] Bing web search api. [accessed 03 January 2023]. [Online]. Available: https://www.microsoft.com/en-us/bing/apis/bing-web-search-api

[21] Scrapy. [accessed 03 January 2023]. [Online]. Available: https://scrapy.org/

[22] scikit-learn. [accessed 03 January 2023]. [Online]. Available: https://scikit-learn.org/stable/

[23] S. M. B. L. C. G. M. M. A. Bettaieb Seifeddine, Shin Seung Yeob, "Using machine learning to assist with the selection of security controls during security assessment," *Empirical Software Engineering*, 2020.

[24] T. Li, "Identifying security requirements based on linguistic analysis and machine learning," in *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*, 2017, pp. 388–397.

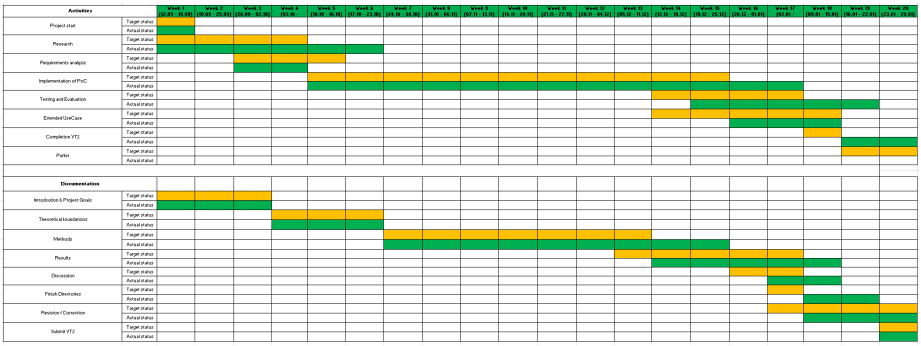
6.3 List of Figures

6.4 List of Tables

Chapter 7

Appendix

7.1 Timetable



7.2 General Conditions

This project is part of the Master of Science in Engineering of the Zurich University of Applied Sciences. It is the first project to fulfill the requirements for the Master of Science in Engineering and focuses on the field of security. Below is a list of the requirements:

General conditions:

1. The project must be written and developed entirely by the student
2. The project must be completed within 450 hours

7.3 API Endpoint Reference

The following section lists and briefly describes the API endpoints of the application.

7.3.1 Assets

- /api/assets/: Allows to query all assets and create an asset.

- /api/asset/pk/: Allows to query all assets and create an asset.
- /api/assettypes/: Allows to query all assets and create an asset.
- /api/asset_control_matches/: Allows to query all assets and create an asset.
- /api/import_assets/: Allows to query all assets and create an asset.

7.3.2 Properties

- /api/properties/: Allows to query all assets and create an asset.
- /property/pk/: Allows to query all assets and create an asset.
- /api/property_tags/: Allows to query all assets and create an asset.

7.3.3 Controls

- /api/controls/: Allows to query all assets and create an asset.
- /api/control/pk/: Allows to query all assets and create an asset.
- /api/parent_controls/: Allows to query all assets and create an asset.
- /api/child_controls/parent_pk/: Allows to query all assets and create an asset.

7.3.4 Tags

- /api/tags/: Allows to query all assets and create an asset.
- /api/tag/pk/: Allows to query all assets and create an asset.
- /api/keywords/: Allows to query all assets and create an asset.
- /api/keyword/pk/: Allows to query all assets and create an asset.

7.3.5 Analysis

- /api/importControls/: Allows to query all assets and create an asset.
- /api/analyseControls/: Allows to query all assets and create an asset.
- /api/matchControls/: Allows to query all assets and create an asset.
- /api/tfidf_matching/: Allows to query all assets and create an asset.
- /api/matching_controls_with_assets/: Allows to query all assets and create an asset.
- /api/matching_controls_with_asset/asset_id/: Allows to query all assets and create an asset.
- /api/tfidf_matching/: Allows to query all assets and create an asset.

7.3.6 Metrics

- /api/metrics/: Allows to query all assets and create an asset.
- /api/create_graphs/: Allows to query all assets and create an asset.

7.3.7 Constraints

- /api/constraints/: Allows to query all assets and create an asset.
- /api/constraint/pk/: Allows to query all assets and create an asset.
- /api/constraint_type/: Allows to query all assets and create an asset.
- /api/constraint_for_asset/pk/: Allows to query all assets and create an asset.
- /api/constraint_association/pk/: Allows to query all assets and create an asset.

7.4 Source Code

The source code for the application can be found on GitHub. The individual modules are listed and briefly described below.

7.4.1 Backend

- **assets**: The asset module which contains the model for the assets and its REST API endpoints. Also included is the file import for importing assets and the web crawler to get the description texts.
- **constraints**: The constraint module contains the models for the constraints as well as the endpoints to access the stored data via the REST API.
- **controls**: The control module contains the models for the controls as well as the endpoints to access the stored data via the REST API.
- **metrics**: The metrics module manages the model for the metrics and the corresponding endpoints for the REST API. Also included is the graph controller which is responsible for creating the graphs. In addition, a class is provided with which metrics can be retrieved and stored throughout the application.
- **nlp**: The NLP module contains the logic for the entire NLP process as well as for the matching algorithms. The algorithms can be started via the REST API and the results are stored directly in the database after completion. There are individual handler classes for the different keyword extraction algorithms, so only the corresponding handler class has to be exchanged to change the algorithm.
- **properties**: The property module contains the models for the properties as well as the endpoints to access the stored data via the REST API.

- **tags**: The tag module contains the models for the tags as well as the endpoints to access the stored data via the REST API.

7.4.2 Frontend

- **general**: Contains general components of the frontend that are used across all other modules. This includes the modal class which provides a generic form of a modal that serves as an input window for certain parameters.
- **assets**: The individual views for the assets are managed in the asset module of the frontend. This includes the overview view and the view for creating and viewing assets.
- **constraints**: The individual views for the constraints are managed in this module. This includes the overview view and the view for creating and viewing constraints.
- **controls**: The individual views for the controls are managed in the controls module. This includes the overview view and the view for creating and viewing controls.
- **dashboard**: The dashboard module contains the view for the dashboard. It contains functionalities to trigger the import of controls and assets in the backend as well as the possibility to start the NLP and the matching algorithms. It is also possible to delete all data via the dashboard.
- **properties**: The individual views for the properties are managed in the properties module. This includes the overview view and the view for creating and viewing properties.
- **tags**: The individual views for the tags are managed in the tags module. This includes the overview view and the view for creating and viewing tags.

7.5 Installation Guide

This section describes the steps required to install and configure the application.

7.5.1 Deployment

There are some basic requirements for successful deployment of the service. The application consists of two parts the backend in Django (scm_backend) and the frontend in React (scm_frontend)

Backend

1. Navigate to the backend folder (scm_backend).
2. Create a virtual environment with python -m venv env.point, regulation parameter, etc.

3. Activate this environment with source env/bin/activate (Linux) or ./env/Scripts/activate.bat (Windows).
4. Install the required Python libraries in requirements.txt with pip install -r ./requirements.txt
5. Install the language model "en_core_web_sm" with python -m spacy download en_core_web_sm.
6. Start the development server with python manage.py runserver
7. The backend is now ready.

Frontend

1. Navigate to the backend folder (scm_backend).
2. Navigate to the frontend folder (scm_frontend).
3. Make sure that Node.js and npm is installed (https://docs.npmjs.com/downloading-and-installing-node-js-and-npm).
4. Install the necessary dependencies with npm install .
5. Start the frontend application with npm start
6. The webpage opens automatically or can be reached via http://localhost:3000.

7.5.2 Environment Variables

Some sensitive config values, such as the api key for the Bing Search API, have to be set using environment variables. For this purpose the .env file is used, which must be created with the following content:

```
BING_SEARCH_V7_SUBSCRIPTION_KEY=<Bing Search API Key>
BING_SEARCH_V7_ENDPOINT=https://api.bing.microsoft.com
```

7.5.3 Config File

The configuration of the application is managed by the config.cfg file. The file contains the following setting options:

```
[Import]
max_controls = <Restrcitons of how much controls should be
    imported (0 = infinite)>
max_assets = <Restrcitons of how many assets should be
    imported (0 = infinite)>

[KeywordExtraction]
algorithm = <Specifies which keyword extraction algorithm to
     use (Rake, Spacy, Yake, Keybert)>

```

```
[Clustering]:
property_cluster_fracture = <Specifies how large the share
    of clusters for the properties should be in relation to
    the keywords (value between 0 and 1)>
control_cluster_fracture = <Specifies how large the share of
     clusters for the controls should be in relation to the
    keywords (value between 0 and 1)>
clustering_algorithm = <Specifies which mode to run for the
    KMeans algorithm (random or k-means++)>
use_cluster_cleaning = <Defines whether cluster cleaning
    should be used or not>
accepted_cluster_size_difference = <defines the tolerance
    value for cluster cleaning for the number of keywords per
     tag>
accepted_cluster_associated_object_difference = <defines the
     cluster cleaning tolerance value for the number of
    assigned objects (controls or properties) per tag>

[TopicModeling]
algorithm = <Specifies the topic modeling algorithm to be
    used (lda or most_used_word)>

[TagBasedMatching]
threshold = <Sets the threshold at which two tags must match
     in order to be matched with each other in tag-based
    matching (between 0 and 1).>

[Tfidf]
threshold = <Sets the threshold at which the description
    texts of the control and asset must match in order to be
    matched during the description-based matching (between 0
    and 1).>
```