Contents lists available at ScienceDirect

Scientific African

journal homepage: www.elsevier.com/locate/sciaf

A classification approach for software requirements towards maintainable security

Prudence Kadebu[a,*], Sunil Sikka[a], Rajesh Kumar Tyagi[a], Panashe Chiurunge[b]

[a] *ASET, Amity Education Valley Gurugram, Manesar, Panchgaon, Haryana 122412, India*
[b] *Chinhoyi University of Technology, Private Bag 7724, Chinhoyi, Zimbabwe*

A R T I C L E I N F O

Article history:
Received 17 July 2022
Revised 7 December 2022
Accepted 8 December 2022

Editor: DR B Gyampoh

Keywords:
Security Requirements classification
Machine Learning
Maintainable Security
Software Requirements Datasets

A B S T R A C T

As trends continue to move toward the introduction of intelligent methods to automate software engineering processes, security requirements classification is rapidly turning into a highly potent field for the software engineering community. There are several models for classifying security requirements proposed in the literature. However, their adoption and use is constrained by the absence of substantial datasets to allow for the replication and generalization of studies, using more advanced machine-learning techniques. Furthermore, most researchers in this area, consider Maintainability as purely a non-functional requirement with no relation to security. This has been identified to be a major source of security concerns. The main objective of this study is to propose a software requirements classification approach that considers maintainability as a security requirement. This seeks to ensure that maintenance efforts don't lead to new software vulnerabilities that were previously not present during deployment. A mixed research methodology is adopted as qualitative data is collected from students' project documentation, labelled, and transformed into quantitative form during analysis. As a culmination of this process, a validated original publicly accessible, labelled software requirements dataset of student software project requirements (DOSSPRE) is obtained and presented to support the approach. It contains 1317 software requirements, including security requirements, functional requirements, and other non-functional requirements. Two versions of the dataset are presented: one for binary classification of security requirements vs. non-security requirements and the other for multi-class classification tasks with various more granular security requirements vs. non-security requirements. In both instances, well-known machine learning algorithms are used to verify the dataset. Support Vector Machine (SVM) and Logistic Regression were the top performers in multi-class classification with an average Accuracy of 86% in both cases. Multinomial Nave Bayes topped the other machine learning techniques in binary classification with 91% Precision, 69% Recall, 78% F1-Score, and Accuracy of 86%. The dataset is accessible on this link https://data.mendeley.com/datasets/23xtbvk6yp/1

* Corresponding author.
E-mail addresses: prudence.kadebu@aju.ac.zw (P. Kadebu), ssikka@ggn.amity.edu (S. Sikka), rktyagi@ggn.amity.edu (R.K. Tyagi), panashe.chiu@gmail.com (P. Chiurunge).

https://doi.org/10.1016/j.sciaf.2022.e01496

Introduction

The number of software system breaches has increased dramatically in recent years, with the most affected systems being those used in banking, healthcare [1], e-commerce, and other web applications [2] as well as in safety critical systems [3]. The proliferation of advanced persistent attacks, which are challenging to detect, further exacerbates this problem [4]. Efforts are being made in most enterprises to put security mechanisms in place to protect software systems from the numerous threats that exist today. However, if these safeguards are introduced to the software late in the development process, for instance at the deployment stage, they raise the vulnerabilities and risk of software failure. Furthermore, it cannot be implemented sufficiently to provide a level of defence that equals the sophistication of the methods used by attackers [5]. Protecting the software throughout its development assures that it will be robust and able to withstand attacks, which is then complemented by additional security measures at the deployment [6–8]. This requirement necessitates taking security into account at an early stage of the software development life cycle, beginning with security requirements engineering. In addition to ensuring that security is incorporated into the design and can be further implemented and tested alongside Functional Requirements, which represent the features of the software, this process identifies and documents the Security Requirements necessary for developing secure software systems [9,10]. This way, security is more intentionally implemented rather than being included as an afterthought, which would require refactoring the architecture, which raises maintenance costs.

Security Requirements (SR), which are a type of Non-functional Requirements (NFR), are constraints on the system's behaviour that are typically drawn from industry best practices or standards that specify security features or services that software must provide to protect it from potential compromise [9,11]. These are gathered alongside all other software requirements from documentation from different stakeholders as well as user stories and software reviews.

Security Requirements Elicitation, the method used to gather the SR, aids in Security Requirements Analysis [12–14]. Since these requirements are stated in natural language, it is difficult to apply traditional statistical tools to the analysis process. This is further compounded by most software engineers' lack of security knowledge [15]. Analysis as a process encompasses security requirements classification, which refers to categorising each security requirement into a specific classification model for the security services provided by the software [13]. Most researches focus on functional and non-functional requirements which include security, albeit generally in a coarse-grained manner [15–19]. Those that provide a more fine-grained model, do not provide a corresponding dataset to support it which discourages research in this important area [20]. Additionally, maintainability is regarded as a non-functional requirement unrelated to security to a greater extent. Maintenance activities are also responsible for disrupting the system's security defenses and introducing new vulnerabilities that weren't previously present in the software. All these scenarios make the classification of software requirements a non-trivial task. There are numerous machine learning and natural language processing techniques that make it easier and quicker to classify software requirements [21,22] and carry out other tasks like reliability prediction [23]. But to extract software requirements and classify them, appropriate datasets for training, validation, and testing of machine learning models are necessary [9,24]. The available datasets on software requirements are either too small or skewed toward functional and non-functional requirements, with security coverage being too limited, which hinders replicable research on security requirements classification [24,25,41]. Additionally, the datasets are quite unbalanced, which adversely impacts the classification results [19,26,27].

In this research, a new Dataset of Students' Software Projects Requirements (DOSSPRE) is created with more than double the size of the NFR dataset [25] and slightly smaller than the one from [22] which is created from a combination of datasets. Both datasets comprise Functional Requirements and Non-Functional ones. However, DOSSPRE has more security requirements than the other two, making it appropriate for security requirements research, which has been up to this point constrained by the lack of relevant datasets. Over and above, the goal of this research is to suggest a novel Security Requirements classification methodology that is supported by this new dataset. The specific contributions that this study seeks to make are to:

1. Evaluate existing Software Requirements classification approaches adopted in various research to assess adequacy for Security. As such we take a holistic view of security and elucidate the ambiguities in various approaches.
2. *Propose a classification approach that identifies Maintainability as a Security Requirement for achieving maintainable security.*
3. *Provide a new dataset of Software Requirements with a more fine-grained consideration of Security Requirements validated with 5 Machine Learning Algorithms.*

The research addresses SDG 9 – "Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation" and SDG 16 which seeks to promote "peace. Justice and strong institutions". The premise for this is the fact that digital transformation is impacting all critical sectors and with all the advances and goodwill it has brought, the attack surface of systems has subsequently widened. Cybersecurity becomes critical for maintaining peace and economic stability across the globe. To this end, the research seeks to address the lack of security engineering knowledge in software engineers, yet software quality depends also on its security. Thus, applying intelligent methods to classify security requirements goes a long way to bridging the gap in security domain knowledge in software development. The rest of the paper is organized as guided by the work of [28]: Section 2 reviews the Literature, 3 explores Materials and methods covering Security Requirements Classification, the essence of maintainability in achieving Maintainable Security and a peek into related datasets. This is followed by a look at the Dataset of Students' Software Projects Requirements (DOSSPRE) in section 4. A presentation of the validation results and discussion ensues in section 5 and finally, the conclusion comes in section 6.

Literature review

Security requirements classifications

Research on software requirements classification has grown significantly with the proliferation of approaches based on machine learning that are useful in the field [11,18,29]. But the majority of these studies pay little attention to security and instead concentrate on functional and non-functional requirements. The activities anticipated when inputs are accepted and processed to produce outputs, such as enroling a student in an eLearning system, are described as functional requirements [30]. They are defined using use case scenarios to express the type of interaction between an actor and the system. While functional requirements are more concerned with the functions or services a system offers, non-functional requirements are more concerned with the quality attributes the system possesses [11,13]. Usability, portability, maintainability, performance, reliability, and security are frequently mentioned amongst these. If these are not sufficiently taken care of, the system may be found to be lacking in quality. Kurtanovic and Maalej adopted four NFR categories—usability (US), security (SE), operational (O), and performance (PE)—for the binary and multi-class classification of user comments [16]. In [21] the Non-Functional Requirements are given as "A=Availability, L = Legal, LF = Look and feel, MN = Maintainability, O = Operational, PE = Performance, SC = Scalability, SE = Security, US = Usability, FT = Fault tolerance, and PO = Portability". Security here is defined in a very coarse-grained manner with availability being considered a non-Functional Requirement outside of security. Fault Tolerance can also be taken as a security requirement that may be related to Survivability [20]. This classification has a lot of ambiguity.

In the case of SR, classifications are based upon the security services or principles as suggested in various research [8,11,18,19]. One of the earliest security models in use, albeit a very coarse-grained one, is the CIA Triad, which stands for Confidentiality, Integrity, and Availability [31]. This model, on which organizations can base their security posture, is currently the closest to standardization of a security classification model that we can get. However, the leanness of this model results in a restriction of a narrow focus on security, which may hinder the design and implementation of appropriate security for current systems. Alonge et al. [32] propose a framework for classifying information assets based on the CIA for security risk assessment. Their focus is mainly on information assets and not security requirements. They apply fuzzy logic in the classification task based upon the ISO/IEC 27,001 security standard. Since the introduction of the CIA, several ideas have been put forward to fine-tune the security services or principles. One improvement to it is the Parkerian Hexad [33], which includes Possession, Authenticity, and Utility, each of which is taken into account in conjunction with a different CIA attribute. The goal of this is to change the way information assets are secured so that humans and other factors are taken into account in addition to technology. While the improvement from the CIA is sensible, it is still too lean to cover security requirements adequately.

Identification, Authentication, Authorization, and Audit are listed as the four security principles in [34], while Access Control pertains to Authentication and Authorization in [35]. Authentication, Authorization, Audit, Availability, Integrity, Confidentiality, and Non-Repudiation are presented in ISO7498–2:1989 [36]. In all these works, the security requirements considered are lean. CLASP (Comprehensive, Lightweight Application Security Process [37]), which defines processes for secure software development, identifies Authentication and Integrity, Confidentiality, Availability, and Accountability as core security services together with Authorization (Access Control) indicated as the most fundamental security service. This is a more fine-grained classification of security requirements. In [20], Firesmith classified security requirements into 12 categories which are "Identification, Authentication, Authorization, Immunity, Integrity, Intrusion Detection, Non-Repudiation, Privacy, Security Auditing, Survivability, Physical Protection, and System Maintenance Security Requirements". This classification introduces System Maintenance as a Security Requirement and is fairly fine-grained. However, no dataset is provided to support this classification model. In [38] A Security Requirements taxonomy is presented at 2 levels of abstraction. The first level features basic Security Requirements while the second level contains 3 sub-factors that further refine the Security Requirements. The introduction of sub-factors is key in addressing ambiguities in the classifications which would otherwise, create a challenge of security inadequacy in the software. However, there is no dataset to back up this taxonomy.

In [39], a domain-specific Security Requirements classification is presented for securing the Internet of Things (IoT). They consider the myriad of threats and attacks prevalent in these systems and propose a suitable classification approach. They highlight that vulnerabilities in IoT systems are a result of unrealised Security Requirements. Other domains have also been explored, with the same challenge of the limited availability of adequate and relevant datasets to support security research and development.

Related datasets

This section presents a review of the popular existing datasets in the related works. NFR is a dataset of Software requirements from the PROMISE Repository [25,40]. It has 12 classes, 3 attributes (Project ID, Requirement, and class) and 625 instances of which 255 are Functional Requirements and 370 are Non-functional Requirements constituting 40.80% and 59.20% respectively. Out of the proportion for Non-functional, there are 66 Security Requirements in the dataset constituting

10.56%. This dataset, attributed to 15 student projects has been used in several studies [9,19,22]. It was re-labelled and combined with 7 other industrial projects to give more than 1500 annotated requirements [22]. The requirements are distributed as follows: 674 Functional only, 545 Quality only, 264 both Functional and Quality, and lastly 19 which are categorised as not a requirement, representing information.

SeqReq [41] dataset is made up of Software Requirements Specification (SRS) documents for 3 industry projects ePurse, CPN, and GPS contributing 124, 210, and, 173 requirements respectively. Security Requirements account for 187 i.e., 36.9% of the total of 507 requirements. This dataset features 2 attributes Requirement and requirement class and is suited for binary classification of '*sec* and nonsec'. SeqReq and NFR are combined in [42] to form a Hybrid dataset yielding a total of 803 requirements. Security Requirements in the dataset account for 245 requirements which amount to 30.51%. There are 76% of the requirements taken from SeqReq and 24% from NFR. A document-orientated dataset, PURE (PUblic REquirements dataset), comprising 79 requirements documents from the Web is presented in [24]. It contains 34,268 sentences in natural language suited for Requirements Engineering tasks in general, with little consideration of security. It is clear that most relevant datasets still have fewer Security Requirements as compared to Non-security Requirements. This inhibits the success of applying advanced Machine Learning techniques in this research area.

In summary, while there are similarities amongst many classifications presented in the literature, there are also discrepancies, showing that there isn't a single, globally accepted method for classifying security requirements as shown in Table 1. Challenges of ambiguities are highlighted. Clearly, a one-size-fits-all classification model may not be practical. In addition, the datasets currently available do not sufficiently address security to facilitate research on the multi-class classification of security requirements [22,24,25].

Therefore, we propose a classification method for security requirements that will be supported by a brand-new dataset of Students' Software Projects Requirements (DOSSPRE). This seeks to achieve an increased coverage of Security Requirements in a single original dataset to support research in this area. Furthermore, it considers Maintainability as a critical security requirement. Going forward, we refer to both the dataset and the classification approach, as DOSSPRE.

Material and methods

This section addresses the materials and methods adopted in this research. Based upon the research findings from literature, a classification approach for SR was created. Software Requirements were gathered from students' project documentation. As such, the research can be categorised as primary research since the researchers built a dataset that was previously non-existent to address the problem of scarcity of relevant datasets. Fundamentally, this is empirical research verifiable experimentally. In addition, the research takes on a mixed research approach that considers both qualitative and quantitative data. The data collected was in textual form and was assigned textual labels, making it primarily, qualitative. During analysis, the data was converted from text to vectors making it quantitative. A machine learning model was created to predict requirements classes automatically. The performance of the model on the datasets was evaluated mathematically using percentages. Thus an inductive approach was adopted in data collection, whilst a deductive approach was applied in data analysis.

A new security requirements classification approach (DOSSPRE)

Our approach is based on the idea that classification that only considers SR may not be realistic given how they are gathered from different sources, together with other requirements in natural language, presented for analysis, and then described. The existing literature indicates that requirements classification research has paid little attention to security [15,44]. Nevertheless, it is necessary to consider both functional and non-functional requirements, though with a focus on security. We found that both the SR and NSR techniques now in use have much too much ambiguity and inconsistency. To complement the earlier work of Rjaibi and Rabai [38], we explain each requirement while highlighting sub-factors as identified in the literature.

Non-Security requirements (NSR)

Functional Requirements refer to the system's intended behaviour expressed in terms of system features or the services the system is developed to offer to its users [45]. It can also be defined using Use Case descriptions that detail the actions from accepting inputs, processing them, and generating outputs [30]. Functional Requirements are directly linked to satisfying the user's needs and thus are at the centre of a product's quality. Their scope is also very large depending on the problem domain. Legal in Cleland-Huang et al. can fit very well in Functional Requirements considering that these requirements may be associated with a Government policy such as the incorporation of Value Added Tax (VAT) on sales [25]. This affects the core functionalities of the system as compliance is enforced to avoid possible legal action [46]. Consideration of Functional and Non-functional requirements with respect to web applications is magnified in

Usability considers how easy it is to use the system, its look, and feel, learnability (provision of help options), efficiency as well as human and computer interaction aspects of a system [25]. It pertains to everything about creating a great user experience. This is fused with the operational requirement. Usability can also be related to availability in the sense that a resource that is said to be available should be accessible and usable whenever it is required [33]. It defeats the purpose if it is accessible but not in a readable form.

Table 1
Comparison of Requirements Classification approaches.

Software Requirements Classifications	CIA Triad [31]	Firesmith [13]	ARM [13]	ISO/ IEC 13,335 [43]	ISO 7498–2: 1989 (en [36]	Clelang-Huang [25]	Viega [37]	DOSSPRE	Remarks
Functional						**X**		**X**	Defining intended behaviour or system services offered, Legal
Usability			**X**			**X**		**X**	Ease of use, look and feel, learnability, operational
Portability						**X**		**X**	Transferability, interoperability, platform independence
Performance						**X**		**X**	Accuracy, speed of processing, efficiency, reliability
Scalability						**X**			Maintainability
Look and feel						**X**			Usability
Fault Tolerance						**X**			Survivability
Legal						**X**			Conformance to legal policies
Security						**X**			Defined in a fine-grained manner
Reliability				**X**					Dependability
Manageability			**X**						Ease of managing/ monitoring
Availability	**X**			**X**	**X**	**X**	**X**	**X**	Accessible when required, Limited downtime, reliability
Identification		**X**					**X**		Create a unique profile of an entity, identify an entity for Authentication
Accountability				**X**					Being able to trace the actions of an entity to it, Auditing
Authentication		**X**	**X**	**X**	**X**		**X**	**X**	Confirming an entity's identity includes Identification which attaches a unique profile to an entity
Authorization		**X**			**X**		**X**	**X**	Granting of access based on privileges, Role Management, Access Control
Immunity		**X**						**X**	Internal defence mechanisms include data validation and sanitisation
Integrity	**X**	**X**	**X**	**X**	**X**		**X**	**X**	Preventing unauthorised modification
Intrusion Detection		**X**						**X**	Monitoring and detecting attempts to gain unauthorized access
Confidentiality	**X**	**X**	**X**	**X**	**X**		**X**	**X**	Privacy of information
Access Control			**X**						Prevention of unauthorised access and use of a resource, Authorisation
Auditing		**X**			**X**			**X**	Maintaining tamper-proof records also supports non-repudiation
Survivability		**X**						**X**	Includes fault tolerance, robustness, and recoverability
Maintainability		**X**				**X**		**X**	Scalability, reusability, system configurations, and system updates
Non-Repudiation		**X**		**X**	**X**				A sub-factor of Auditing, Integrity
Physical Protection		**X**							A sub-factor of maintainability, Access control
System Maintenance		**X**							A sub-factor of maintainability, security patching

Portability touches on the extent to which the system is transferable across various platforms for instance Android and IOS mobile Operating Systems as well as its interoperability [30]. This is a very important attribute today as users access systems from numerous devices. Thus, a good system must have platform independence so that users are not restricted to a single operating environment.

Performance looks at how well the system accomplishes its intended tasks. It includes efficiency, accuracy, throughput, responsiveness, speed of execution or delivery of service, and ability to serve multiple users concurrently without crippling the system. The performance of the system may degrade with time for instance due to the increase in users or data that is processed and stored. It can be measured by the number of requests that are processed per unit of time.

Maintainability is a non-functional requirement that is usually associated with characteristics like flexibility, readability of code and documents, and testability. It is also frequently associated with scalability, modifiability, and reusability. Due to its critical significance in software security, it is proposed that it be taken into consideration as a security requirement in this work. As a result, since maintainability is regarded as a security requirement, security is placed at the forefront of all other features. As a result, while it preserves its core principles, it is strongly related to security because software vulnerabilities are most frequently introduced by maintenance activities. This component should be taken into account together with functionality, usability, portability, and performance. We need confidence that modifications to the system won't adversely affect these characteristics. The importance of Maintainability as a Security Requirement can be seen for instance, if the potential risk of keeping dead code without proper updated documentation and effective communication is considered [47]. A deprecated function may be used yet it may be having a security flaw, hence its initial abandonment. The same applies to duplicate code reuse which increases the attack surface of software and the maintenance effort required to fix bugs. The underlying architecture of the software may be the source of major security problems if there is a high cyclomatic complexity which makes bug discovery difficult. Thus Maintainability becomes a key aspect when considering software security.

Security requirements (SR)

The DOSSPRE approach presents SR in a very fine-grained way, with Maintainability included as a crucial Security Requirement. In the binary-classification paradigm, all SR are grouped vs NSR. It is important to keep in mind that some SRs are implemented during development, but some will still need to be applied at deployment and maintained later. The emphasis is on building security into a system through a comprehensive assessment of security needs early in the SDLC. When compared to functional requirements, the SR has a narrower scope.

Availability (AVA) requirement outlines the extent to which system downtime must be kept at a minimum and how authorized users must have access to system resources whenever they need them, notwithstanding disruptions. It is linked to the survivability requirement, which enables maintaining system availability even when it is running in a degraded mode. Understanding the distinction between availability and survivability is crucial. Threats to availability result in downtime, which can be extremely disastrous, especially for banking systems where even a little interruption can cost millions of dollars. Downtime on oxygen supply can also result in instant patient death in healthcare systems. Mechanisms that guarantee availability include redundancy, load balancing, and failovers.

Authentication (THE) requirement specifies the extent to which the system shall be able to verify and confirm correctly the identity of a user wishing to access it. Of importance first is the identification of the user or entity which pertains to creating a unique profile for the user which sets them apart from every other user. This identity must then be confirmed when the user wants to use the system. We linked them together in our classification approach because authentication cannot take place without identification. A digital certificate that is issued by a certification authority is a common method of authentication. Consequently, only trusted entities are allowed to interact with the system.

Authorization (THO) requirement specifies the extent to which the system shall verify that the user wishing to access a service from the system, has the right to access it. It is also indicated as Access Control in [37]. Essentially, this pertains to issues to do with defining user access rights, role assignment, and privileges. Access is granted to a user according to an Access Control List which can be defined in the system. An entity is authenticated first but that does not define what privileges they have. Authorisation then determines their right in the system.

Immunity (IMM) requirement specifies the extent to which the system shall exhibit an internal capacity to defend itself against attacks and intrusions. It includes enabling input validation, configuration management, and any other means for intrusion prevention. Thus, it is an inbuilt defence mechanism against threats.

Integrity (INT) requirement specifies the extent to which the system is protected from unauthorized modification of data and resources it manages. This includes the prevention of data corruption and tampering with the system's audit trail. Integrity can be violated intentionally e.g., by intruders, or unintentionally e.g., by the mistakes of employees. Integrity is guaranteed through hashing, validity checks, and efficient backing up of data.

Intrusion detection (IND) requirement specifies the extent to which the system shall be able to monitor and detect attempts to gain unauthorized access to it. Such attempts should be recorded and alerts generated timeously to prevent damage.

Confidentiality (CON) requirement specifies the extent to which the system shall ensure the privacy of sensitive information and offer personal control over information. This is supported by the authorization requirement as well as internal mechanisms like data encryption and data classification (top secret, secret, confidential) capabilities. Encryption is a popular mechanism to safeguard a system's data from loss of privacy.

Auditing (AUD) requirement specifies the extent to which the system shall monitor and maintain tamper-free records of system activities including system logs, malicious activities, and any system violations. It includes accountability, non-repudiation, problem analysis, and reconstruction of events. It is important to determine what kind of audit records should be kept to support the maintenance of a reliable audit trail. Log files are an important data resource to be well defined and safeguarded [48].

Survivability (SUR) requirement specifies the extent to which the system shall survive an attack on its components [20]. The system shall make a trade-off between integrity and availability as it operates in degraded mode in the face of an attack or network partition. Performance may be greatly affected but services should not be completely disrupted. It is evaluated based on the proportion of data objects that remain unaltered and are available for user access. This includes the Fault Tolerance of a system.

Maintainability (MAI) requirement is defined in terms of ease of locating and repairing faults in software [25]. From a security perspective, maintainability specifies the extent to which the system shall prevent any maintenance activities from disrupting any deployed security mechanism. Firesmith includes System Maintenance as an SR [20].

Maintainability and maintainable security

Maintainability is a non-functional requirement usually associated with such attributes as modifiability, readability, reusability, and scalability. Physical protection of the system may also be included in this category. From a security perspective, maintainability allows for the system to be maintained and updated without introducing vulnerabilities that were previously not present. Exploring the Software Engineering process, the fact that software evolves and that software maintenance is a crucial stage brings about the need to tie it up with security. This is because most controls are affected by system updates and changes that are applied during maintenance activities which include perfective, preventive, and corrective maintenance. The first 2 are usually planned while corrective is a reactive activity to a greater extent as it is normally conducted as a response to system failure or identification of bugs.

Software evolves due to several reasons, including errors that may have been discovered after deployment of the software, the need to incorporate additional functionality, to comply with new regulatory requirements, refactoring to improve the performance of the software, to adapt the software to technological changes in the operating environment as well as to remove obsolete functionalities. All of these may lead to disruption of the Security of the system. Considering maintainability as an NFR separate from Security Requirements increases the risk of the introduction of vulnerabilities in the code every time changes are made to the software. Maintainable security can be achieved through writing maintainable code which makes fixing vulnerabilities faster and easier as well as documenting at every step to ensure that no updates are lost.

According to [47] ‘Having a resilient and scalable security architecture and design is crucial to prevent vulnerabilities.’ Thus, the success in maintaining the security of a system depends upon the extent to which maintainability has been catered for in the development of that system starting with the requirements phase. Consideration of various requirements separately is one of the culprits for most vulnerabilities inherent in software. It is paramount to consider software requirements holistically. Thus, for all system features considered, there is a need to also consider the maintainability, security concerns, and other quality attributes associated with it amongst other things. Fig. 1 shows a proposed classification approach for Software Requirements to support the goal of achieving maintainable security.

Theory

This section details the process undertaken in the creation of the dataset of Students' Software Projects Requirements (DOSSPRE). This is a completely new dataset created from students' projects. The work is inspired by the contributions of [25] which culminated in the NFR dataset as well as Firesmith [20] which presented a fine-grained classification of SR amongst others. Most of the projects were from the area of Information Security and Assurance from Harare Institute of Technology since this is the area that had the potential to yield more Security Requirements. The Methodology in Fig. 2 was applied in the creation of the dataset.

Problem understanding

This research seeks to create a dataset of software requirements with a wide coverage of Security Requirements to aid in the Classification thereof, a task that most Software Engineers find difficult due to a lack of domain knowledge in Software Security [15]. The task is to classify Security Requirements in a more fine-grained manner that set them apart from the other requirements to support the design of specific mechanisms according to the system-specific needs.

Data collection and structuring

Software Requirements were harvested from students' projects in a kind of manual scraping, performed by a group of students. Both functional and non-functional requirements were considered from 105 students' software projects spanning about 5 years. The majority of the requirements were very poorly written. Formatting and restructuring of Requirements were done for better presentation and readability. Generally, each project had an average of 12 requirements, with a minimum of 3 and a maximum of 31 requirements. The total number of elements in the dataset was 1317. Fig. 3 shows the distribution of SR vs NSR. The percentage of SR vs NSR

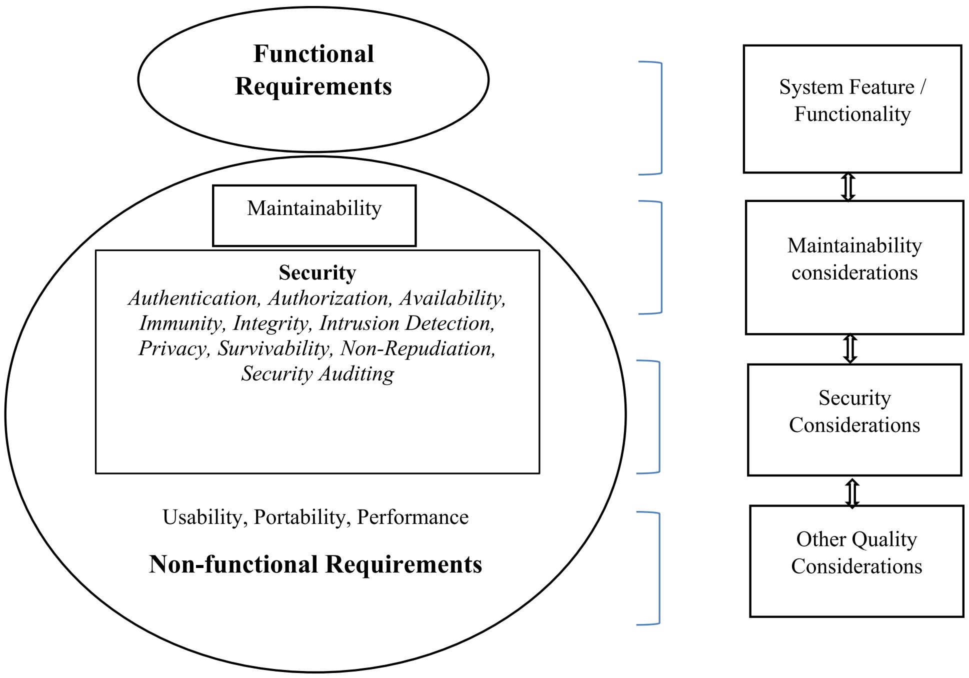
Functional Requirements
Maintainability
Security
Authentication, Authorization, Availability, Immunity, Integrity, Intrusion Detection, Privacy, Survivability, Non-Repudiation, Security Auditing
Usability, Portability, Performance
Non-functional Requirements
System Feature / Functionality
Maintainability considerations
Security Considerations
Other Quality Considerations

Fig. 1. Proposed classification approach for Software Requirements towards Maintainable Security

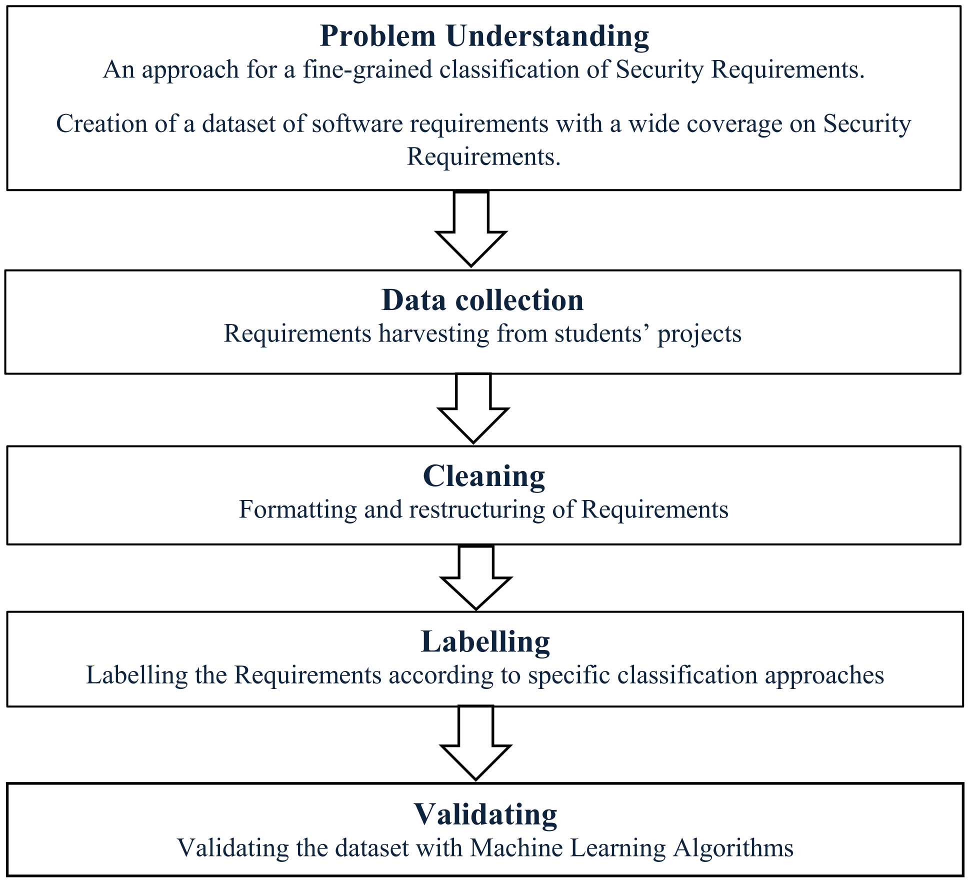
Problem Understanding
An approach for a fine-grained classification of Security Requirements.
Creation of a dataset of software requirements with a wide coverage on Security Requirements.
Data collection
Requirements harvesting from students' projects
Cleaning
Formatting and restructuring of Requirements
Labelling
Labelling the Requirements according to specific classification approaches
Validating
Validating the dataset with Machine Learning Algorithms

Fig. 2. Methodology for dataset creation

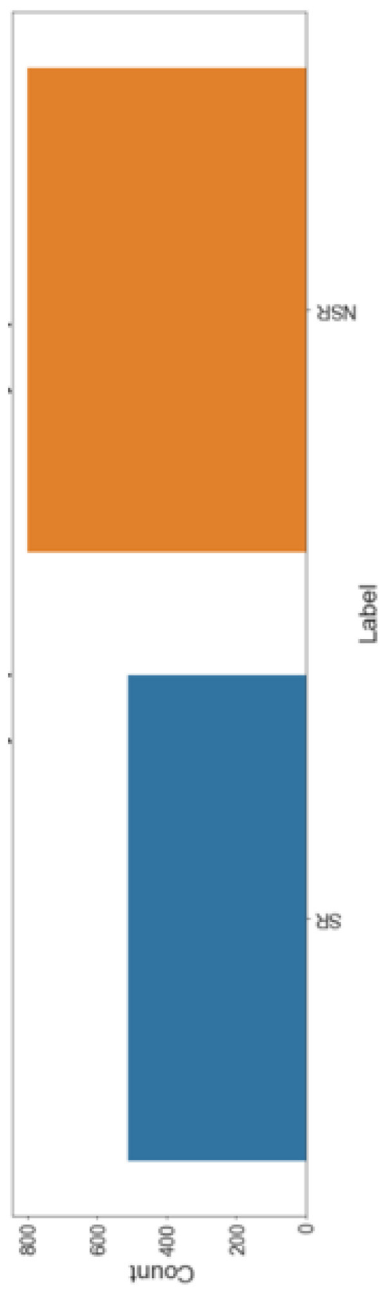

Fig. 3. Number of Non-Security Requirements and Security Requirements

Table 2
Binary Classification Results.

Algorithm	Precision	Recall	F1-Score	Accuracy
SVM				
NSR	0.79	0.88	0.83	0.78
SR	0.75	0.62	0.68	
macro avg	0.77	0.75	0.76	
weighted avg	0.78	0.78	0.78	
Multinomial Naive Bayes				
NSR	0.84	0.96	0.89	0.86
SR	0.91	0.69	0.78	
macro avg	0.87	0.82	0.84	
weighted avg	0.86	0.86	0.85	
Logistic Regression				
NSR	0.78	0.91	0.84	0.78
SR	0.79	0.58	0.67	
macro avg	0.79	0.74	0.75	
weighted avg	0.79	0.78	0.78	
K- Nearest Neighbors Classifier				
NSR	0.68	0.84	0.75	0.66
SR	0.56	0.35	0.43	
macro avg	0.62	0.59	0.59	
weighted avg	0.64	0.66	0.63	
Decision Tree Classifier				
NSR	0.77	0.85	0.81	0.75
SR	0.71	0.59	0.64	
macro avg	0.74	0.72	0.73	
weighted avg	0.75	0.75	0.75	

Data cleaning

This involves the removal of trailing spaces in the dataset. Quoting all the text in the Requirements field using a custom format. The operation added single quotes to all the 1317 Requirements column entries. Data cleaning was done to reduce errors in the data and to eliminate missing values and duplicate data. This has an impact on the quality of data.

Labelling

Labelling the Requirements was a mammoth task as there was a lot of ambiguity emanating from the nature of requirements statements whereby one could fit into more than one category. However, most of the ambiguities were cleared during cleaning and restructuring and the quality assurance procedure which included a team of 3 experts in the field. Binary classification of the data created a distribution of 801 NSR vs 516 SR constituting about 61% and 39% of the dataset respectively.

In multi-class classification, Non-Security Requirements (NSR) included Functional, Usability, Portability, and Performance Requirements. Security Requirements (SR) included Availability (AVA), Authentication (THE), Authorization (THO), Immunity (IMM), Integrity (INT), Intrusion detection (IND), Confidentiality (CON), Auditing (AUD), Survivability (SUR) and Maintainability (MAI). Three annotators collaborated on the labelling task. The first annotator has expertise in Information Security, the second in Machine Learning and Statistical Analysis. The third annotator has competencies in Software Quality. There were several interactive sessions to do the task to increase data quality. The Kappa index is used to ensure the reliability of ratings performed by different parties [49].

Validating the requirements

The dataset was validated with both binary and multi-class classification using Machine Learning techniques i.e., SVM, Multinomial NB, Logistic Regression, K- Nearest Neighbors Classifier, and Decision Tree Classifier. The experiments were run on an Intel Core i7 6500 U 2.50 2.60 GHz processor with 8GB RAM. The basic process for validating the dataset included i) Data Pre-processing ii) Feature extraction iii) Training the Classifier iv) Classification. The training and test split used was 1053 and 264, respectively. The training set is 80% and the test size is 20% of the dataset. Table 2 presents the validation results for binary classification with the DOSSPRE dataset. The best performance was realised with Multinomial NB (86.0%). Table 3 shows the Multi-class Classification model performance from the five classifiers. The classes used are NSR, Availability (AVA), Authentication (THE), Authorization (THO), Immunity (IMM), Integrity (INT), Intrusion detection (IND), Confidentiality (CON), Auditing (AUD), Survivability (SUR) and Maintainability (MAI). Essentially this is NSR vs SR albeit with SR expressed in a fine-grained manner.

Table 3
Multi-class Classification Performance evaluation.

Security Requirement Classes											
Measure	NSR	THE	CON	THO	MAI	IMM	IND	AVA	AUD	SUR	INT
SVM											
Precision	88.00	100.00	78.00	83.00	86.00	73.00	100.00	91.00	91.00	71.00	63.00
Recall	88.00	100.00	93.00	83.00	75.00	73.00	75.00	91.00	91.00	77.00	63.00
F1-score	88.00	100.00	85.00	83.00	80.00	73.00	86.00	91.00	91.00	74.00	63.00
Accuracy	**86.00**										
Multinomial Naive Bayes											
Precision	67.00	100.00	83.00	100.00	83.00	100.00	100.00	64.00	100.00	67.00	75.00
Recall	25.00	45.00	33.00	17.00	31.00	9.00	42.00	97.00	55.00	45.00	16.00
F1-score	36.00	62.00	48.00	29.00	45.00	17.00	59.00	77.00	71.00	54.00	26.00
Accuracy	**67.00**										
Logistic Regression											
Precision	88.00	100.00	82.00	83.00	83.00	86.00	100.00	88.00	100.00	71.00	65.00
Recall	88.00	100.00	93.00	83.00	62.00	55.00	75.00	94.00	82.00	68.00	68.00
F1-score	88.00	100.00	87.00	83.00	71.00	67.00	86.00	91.00	90.00	70.00	67.00
Accuracy	**86.00**										
K-Nearest Neighbors Classifier											
Precision	00.00	71.00	24.00	36.00	56.00	36.00	29.00	63.00	40.00	55.00	57.00
Recall	00.00	45.00	33.00	33.00	31.00	45.00	17.00	80.00	36.00	27.00	21.00
F1-score	00.00	56.00	28.00	35.00	40.00	40.00	21.00	71.00	38.00	36.00	31.00
Accuracy	**56.00**										
Decision Tree Classifier											
Precision	68.00	92.00	67.00	50.00	86.00	83.00	90.00	90.00	71.00	72.00	69.00
Recall	81.00	100.00	80.00	75.00	75.00	91.00	75.00	83.00	91.00	82.00	58.00
F1-score	74.00	96.00	73.00	60.00	80.00	87.00	82.00	86.00	80.00	77.00	63.00
Accuracy	**81.00**										

Results and discussion

A number of Security Requirements classification approaches were considered from the literature, culminating in the DOSSPRE Security Requirements classification approach. In Table 1, a comparison of Software Requirements classifications that include Security was presented. We base our approach on a combination of various existing offerings in the area. A dataset was created from students' project documentation. Each project contributed Software Requirements and there was an average of 12 requirements, both functional and non-functional with a minimum and maximum of 3 and 31 requirements, respectively. The quality of requirements from students' projects was noted to be poor. Most requirements' definitions were scant and much generalised compared to requirements from industry Projects such as in Ferrari et al. [24]. It was noted that students plagiarise each other's requirements, especially on NFR which are often rather generic. Thus, the same requirement would appear multiple times. The requirements were ill-structured making the labelling process very tedious and challenging. Furthermore, most were very ambiguous, for instance:

The system shall offer authentication and authorization to prevent access to the admin console........THO

This requirement was labelled as an Authorisation Requirement (THO) because it speaks to the prevention of access to the admin console which is essentially Access Control and, according to our approach, it translates to an Authorisation requirement. Authentication is still valid in the same case, though authorisation becomes more far-reaching in this case. Our classification approach which we called DOSSPRE from the dataset, includes Functional Requirements, Non-Functional Requirements (Usability, Performance, Portability, and Security) with a greater focus on SR. We streamlined the NFR in [25] as some of them are catered for in other requirements for instance Availability is considered under SR, while Look and Feel is a Usability requirement. The SRs were categorised into 10 distinct classes and this was meant to provide a holistic view of Security in a fine-grained manner with less redundancy. Where certain requirements were left out from the main categories, it was because of their rarity in the dataset which is an indication that they are not commonly included in the requirements elicited. However, because of their importance, they were not completely discarded, rather they were included under the related dominant ones as sub-factors [38] which would still need to be catered for within that requirement. The importance of Maintainability as a security requirement was emphasised as a way to ensure any maintenance that will be applied to software as it evolves would not have the adverse effect of creating new vulnerabilities.

Five Machine Learning techniques (Multinomial Naïve Bayes, K-Nearest Neighbors, SVM, Logistic Regression, and Decision Tree Classifiers) were used to validate the dataset. Precision, Recall, F1-Score as well as Accuracy are the performance measures that were used to evaluate the models. The first three are most suitable for imbalanced data [19] which is the case with DOSSPRE. The best performance for binary classification was noted for Multinomial Naïve Bayes at 84% Precision, 96% Recall, and 89% F1-Score for NSR, shown in Table 2. Similarly, for SR, high performance was noted with 91% Precision, 69% Recall, and 78% F1-Score. Accuracy was 86%. On the other hand, the lowest performance was observed for K-Nearest

Neighbors which had 56% Precision, 35% Recall, and 43% F1-Score for SR, and slightly higher performance of 68% Precision, 84% Recall, and 75% F1-Score for NSR and with overall 66% Accuracy. The performance of a classifier is acceptable if it achieves 70% recall and 60% precision at a minimum, according to Knauss et al. [15]. Thus, the result for K-Nearest Neighbors was acceptable in the prediction of NSR but the same cannot be said for SR. In Table 3, Multiclass classification results are shown with each algorithm having three sets of values i.e. Precision, Recall, and F1-score, as well as prediction accuracy across all the classes. The performance for multi-class classification was better than for binary with SVM and Logistic Regression performing very well for Authentication, Availability, and Auditing Security Requirements. Several perfect predictions were observed for the Authentication Requirement across the board except for K-Nearest Neighbours Classifier. This is a great indication of its importance in the majority of software.

Generally, good model performance was noted on DOSSPRE regardless of the limitation of being unbalanced in the proportion of Security against Non-Security Requirements. NSR has 801 (61%) requirements while SR has 516 (39%) requirements with the least requirements of 33 coming from Integrity. Imbalanced data affects the performance of the algorithms [19]. Sorting and augmentation of the dataset can improve the results. However, there is a great improvement from the previous datasets in terms of the proportion of SR out of the rest of the requirements, NFR (10.56%) [25], SecReq (36.88%) [41], and the Hybrid (30.51%) [42]

DOSSPRE might be affected by a bias toward Security Requirements to fully support other research in Software Requirements Engineering in cases where for instance Maintainability needs to be taken purely as a Non-functional Requirement not related to security. *Re*-labelling may be necessary in that case. However, we identify this as the scenario that leads to vulnerabilities being introduced to software during Maintenance activities.

Conclusion

Consideration of Security Requirements early in the SDLC helps to ensure a quality software product is developed. Knowing the level of security required hinges upon being able to correctly identify and classify the Security Requirements for each software during development. This is critical to the Software Engineering community as new advanced and intelligent methods continue to be explored to make software development smarter and faster as well as result in more secure products. In this research, existing models for requirements classification were reviewed and a classification approach was proposed with Maintainability being considered as a Security Requirement. This is to ensure all system maintenance applied at different stages does not disrupt security mechanisms within the system or introduce new vulnerabilities previously not existing in the system. Research in this area has previously been hampered by challenges such as scarcity of relevant data in existing datasets and in most cases, unavailability of datasets to support some classification models. The existing datasets are too lean in security requirements coverage. To this end, a dataset of students' software requirements projects was created and labelled in 2 versions, for binary classification problems and fine-grained multiclass classification problems, respectively with a major focus on Security Requirements. DOSSPRE has 1317 entries out of which 803 are Non-Security Requirements and 516 Security Requirements which is 39% of all the requirements. This is a great improvement from the Tera-Promise which had 625 requirements of which only 10% were Security Requirements. The DOSSPRE dataset was validated for binary and multi-class classification using five Machine Learning techniques. Higher scores were obtained in binary as compared to multi-class classification. Correctly classifying security requirements will ultimately bridge the gap of a lack of domain knowledge in security in software developers. In the future, we will work on expanding the DOSSPRE dataset to at least 2000 Security Requirements. We will do more extensive experiments on the DOSSPRE dataset. We will explore varied techniques for Feature Extraction to see their effect on the classification as well as to test more algorithms in Ensemble Learning and Deep Learning.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. Authors used their own funds throughout the research.

Declaration of competing interest

All authors have declared no competing interests.

Acknowledgements

We acknowledge the contributions of Eng. Robert Shoniwa, Dr. Kudakwashe Zvarevashe, and Ms. Linnet Svotwa (quality assurance for the dataset) for this work, and the Harare Institute of Technology for assisting with the input of Software Requirements from student projects which contributed immensely to the creation of the dataset.

References

[1] A. Almulihi, F. Alassery, A. Khan, S. Shukla, B. Gupta, R. Kumar, Analyzing the implications of healthcare data breaches through computational technique, Intell. Automat. Soft Comput. 32 (2022) 1763–1779, doi:10.32604/iasc.2022.023460.

[2] P. Salini, S. Kanmani, Model oriented security requirements engineering (MOSRE) framework for web applications, in: Advances in Intelligent Systems and Computing, 177 AISC, 2013, pp. 341–353, doi:10.1007/978-3-642-31552-7_36. no. VOL. 2.
[3] O. Arogundade, O. Abayomi-Alli, S. Misra, L. Fernandez-Sanz, Enhancing misuse cases with risk assessment for safety requirements, IEEE Access 8 (1) (2020), doi:10.1109/ACCESS.2019.2963673.
[4] J.-.W. Jung, S.-.H. Park, S.-.W. Lee, A tool for security requirements recommendation using case-based problem domain ontology, in: 2021 IEEE 29th International Requirements Engineering Conference (RE), 2021, pp. 438–439, doi:10.1109/RE51729.2021.00059.
[5] M. Riaz, J. King, J. Slankas, L. Williams, Hidden in plain sight: automatically identifying security requirements from natural language artifacts, in: 2014 IEEE 22nd International Requirements Engineering Conference, RE 2014 - Proceedings, Sep. 2014, pp. 183–192, doi:10.1109/RE.2014.6912260.
[6] J. Slankas, L. Williams, Automated extraction of non-functional requirements in available documentation, in: 2013 1st International Workshop on Natural Language Analysis in Software Engineering, NaturaLiSE 2013 - Proceedings, 2013, pp. 9–16, doi:10.1109/NAturaLiSE.2013.6611715.
[7] R. Jindal, R. Malhotra, A. Jain, Automated classification of security requirements, in: 2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016, Nov. 2016, pp. 2027–2033, doi:10.1109/ICACCI.2016.7732349.
[8] H. El-Hadary, S. El-Kassas, Capturing security requirements for software systems, J. Adv. Res. 5 (4) (2014) 463–472, doi:10.1016/j.jare.2014.03.001.
[9] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, “Automated classification of non-functional requirements,” in *Requir. Eng.*, Apr. 2007, vol. 12, no. 2, pp. 103–120. doi:10.1007/s00766-007-0045-1.
[10] “C1: define security requirements | OWASP.” https://owasp.org/www-project-proactive-controls/v3/en/c1-security-requirements (accessed Mar. 06, 2021).
[11] M. Lu, P. Liang, in: Automatic Classification of Non-Functional Requirements from Augmented App User Reviews, vol. Part F1286, Association for Computing Machinery, New York, NY, USA, 2017, pp. 344–353, doi:10.1145/3084226.3084241.
[12] G. Sindre, A.L. Opdahl, Eliciting security requirements with misuse cases, Requir Eng.. 10 (1) (Jan. 2005) 34–44, doi:10.1007/s00766-004-0194-4.
[13] N.R. Mead, T. Stehney, Security quality requirements engineering (SQUARE) methodology, in: SESS 2005 - Proceedings of the 2005 Workshop on Software Engineering for Secure Systems - Building Trustworthy Applications, May 2005, pp. 1–7, doi:10.1145/1083200.1083214.
[14] A. Rashwan, Semantic analysis of functional and non-functional requirements in software requirements specifications, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7310 LNAI, 2012, pp. 388–391, doi:10.1007/978-3-642-30353-1_42.
[15] E. Knauss, S. Houmb, K. Schneider, S. Islam, J. Jürjens, Supporting requirements engineers in recognising security issues, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6606 LNCS, 2011, pp. 4–18, doi:10.1007/978-3-642-19858-8_2.
[16] P. Singh, D. Singh, A. Sharma, Classification of non-functional requirements from SRS documents using thematic roles, in: Proceedings - 2016 IEEE International Symposium on Nanoelectronic and Information Systems, iNIS 2016, Jan. 2017, pp. 206–207, doi:10.1109/iNIS.2016.054.
[17] A. Casamayor, D. Godoy, M. Campo, Identification of non-functional requirements in textual specifications: a semi-supervised learning approach, Inf. Softw. Technol. 52 (4) (Apr. 2010) 436–445, doi:10.1016/j.infsof.2009.10.010.
[18] M.A. Haque, M.A. Rahman, and M.S. Siddik, “Non-functional requirements classification with feature extraction and machine learning: an empirical study,” May 2019. doi:10.1109/ICASERT.2019.8934499.
[19] Z. Kurtanovic, W. Maalej, Automatically classifying functional and non-functional requirements using supervised machine learning, in: Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017, Sep. 2017, pp. 490–495, doi:10.1109/RE.2017.82.
[20] D.G. Firesmith, Engineering security requirements, J. Obj. Technol. 2 (1) (2003) 53–68 Journal of Object Technology, doi:10.5381/jot.2003.2.1.c6.
[21] G. Sandhu, A.B. Cse, S. Pal, B. Cse, and P. Pal, “Knowledge extraction in requirement engineering with machine learning perspective,” 2015.
[22] F. Dalpiaz, D. Dell’Anna, F.B. Aydemir, S. Çevikol, Requirements classification with interpretable machine learning and dependency parsing, in: *Proceedings of the IEEE International Conference on Requirements Engineering*, vol. 2019-September, Sep. 2019, pp. 142–152, doi:10.1109/RE.2019.00025.
[23] K. Sahu, F.A. Al-Zahrani, R.K. Srivastava, R. Kumar, Evaluating the impact of prediction techniques: software reliability perspective, Comput. Mater. Continua 67 (2021) 1471–1488, doi:10.32604/cmc.2021.014868.
[24] A. Ferrari, G.O. Spagnolo, S. Gnesi, PURE: a dataset of public requirements documents, in: Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017, Sep. 2017, pp. 502–505, doi:10.1109/RE.2017.29.
[25] J. Cleland-Huang, S. Mazrouee, H. Liguo, and D. Port, “nfr,” Mar. 2007, doi:10.5281/ZENODO.268542.
[26] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (Sep. 2009) 1263–1284, doi:10.1109/TKDE.2008.239.
[27] R. Caruana, “Learning from imbalanced data: rank metrics and extra tasks,” 2000. Accessed: Apr. 03, 2021. [Online]. Available: www.aaai.org
[28] S. Misra, A step by step guide for choosing project topics and writing research papers in ICT related disciplines, Commun. Comput. Inf. Sci. 1350 (2021) 727–744, doi:10.1007/978-3-030-69143-1_55/COVER.
[29] A. Mahmoud, G. Williams, Detecting, classifying, and tracing non-functional software requirements, Requir. Eng. 21 (3) (Sep. 2016) 357–381, doi:10.1007/s00766-016-0252-8.
[30] Y. Singh and R. Maholtra, *Object-Oriented Software Engineering*. New Delhi: Phi Learning, 2012.
[31] J. van der Ham, Toward a better understanding of ‘Cybersecurity, Digital Threats: Res. Practice 2 (3) (Jul. 2021) 1–3, doi:10.1145/3442445.
[32] C. Alonge, O. Arogundade, A. Adesemowo, F. Ibrahalu, J. Adeniran, and A. Mustapha, “Information asset classification and labelling model using fuzzy approach for effective security risk assessment,” 2020, pp. 1–7. doi:10.1109/ICMCECS47690.2020.240911.
[33] G. Pender-Bey, “The Parkerian Hexad: the CIA triad model expanded,” Lewis University. Accessed: Jul. 31, 2021. [Online]. Available: https://cs.lewisu.edu/mathcs/msisprojects/papers/georgiependerbey.pdf
[34] M. Danziger, M.A. da Silva, (PDF) The importance of security requirements elicitation and how to Do It, PMI® Global Congress 2015—EMEA, London, England. Newtown Square, PA, Project Management Institute, 2015 https://www.researchgate.net/publication/304539849_The_Importance_of_Security_Requirements_Elicitation_and_How_to_Do_It accessed Mar. 02, 2021.
[35] M. Stamp, Information Security: Principles and Practice, Wiley, 2011 2nd EditioAccessed: Mar. 02, 2021. [Online]. Available https://www.wiley.com/en-us/Information+Security%3A+Principles+and+Practice%2C+2nd+Edition-p-9780470626399 .
[36] ISO/IEC, “ISO 7498-2:1989(en), information processing systems — open systems interconnection — basic reference model — Part 2: security architecture.” https://www.iso.org/obp/ui/#iso:std:iso:7498:-2:ed-1:v1:en (accessed Jun. 20, 2021).
[37] J. Viega, Building security requirements with CLASP, in: SESS 2005 - Proceedings of the 2005 Workshop on Software Engineering for Secure Systems - Building Trustworthy Applications, May 2005, pp. 1–7, doi:10.1145/1083200.1083207.
[38] N. Rjaibi and L.B.A. Rabai, “Developing a novel holistic taxonomy of security requirements,” 2015. doi:10.1016/j.procs.2015.08.442.
[39] A. Mukalazi, A. Boyaci, The Internet of Things: a domain-specific security requirement classification, in: HORA 2022 - 4th International Congress on Human-Computer Interaction, Optimization and Robotic Applications, Proceedings, 2022, doi:10.1109/HORA55278.2022.9800035.
[40] J. Sayyad Shirabad, T.J. Menzies, The PROMISE Repository of Software Engineering Databases, School of Information Technology and Engineering, University of Ottawa, Canada, 2005 http://promise.site.uottawa.ca/SERepository accessed Feb. 10, 2021.
[41] S.H. Houmb, S. Islam, E. Knauss, J. Jürjens, K. Schneider, Eliciting security requirements and tracing them to design: an integration of Common Criteria, heuristics, and UMLsec, Requir. Eng. 15 (1) (Nov. 2009) 63–93 *2009 15:1*, doi:10.1007/S00766-009-0093-9.
[42] V. Fong, Software Requirements Classification Using Word Embeddings and Convolutional Neural Networks, California Polytechnic State University, 2018.
[43] ISO/IEC, Management of information and communication technology security – Part 1: concepts and models for information and communication technology security managementISO/IEC 13335, 2004.
[44] S. Jaiswal, D. Gupta, Security engineering methods - In-depth analysis, Int. J. Inf. Comput. Secur. 9 (3) (2017) 180–211, doi:10.1504/IJICS.2017.085135.
[45] K. Mahalakshmi, R. Prabhakar, Performance evaluation of non functional requirements, Global J. Comput. Sci. Technol. Softw. Data Eng. (2013).

[46] A. Mustapha, O. Arogundade, S. Misra, R. Damaševičius, R. Maskeliunas, A systematic literature review on compliance requirements management of business processes, Int. J. Syst. Assur. Eng. Manag. 11 (2020), doi:10.1007/s13198-020-00985-w.
[47] B. Vieira, Maintainable Security: 9 best practices to make your software security future proof, Softw. Improv. Group, Medium (2017) https://medium.com/softwareimprovementgroup/maintainable-security-4bd4bf055438. accessed Mar. 06, 2021.
[48] J. Viega, The CLASP Application Security Process, Secure Software Inc, 2005.
[49] O. Ormandjieva, I. Hussain, and L. Kosseim, “Toward a Text classification system for the quality assessment of software requirements written in natural language,” 2007. doi:10.1145/1295074.1295082.