ABSTRACT
A well-defined set of business requirements is the foundation for a successful software development. Thus, prior to the development of a software, it is crucial to ensure the clarity of the requirement, especially on what they are building it, why they are building it, and what to expect at the end. Considering the requirements are derived from natural language, requirements engineers (RE) face problems in eliciting and writing security requirements. This is due to their tendency to misunderstand the real needs and the security terms used that lead to vague requirements. Motivated from these problems, this paper proposed a mechanism to check the clarity level of security requirements. Two requirements examples, which are the real business requirements and the certified product requirements were used to demonstrate the clarity mechanism. We also proposed the design of security requirements syntax tree structure (SecReqTS) and evaluated the design using the mechanism. Finally, we compared the results between our proposed SecReqTS with the real business requirements and the certified products requirements.

Keywords: Security Requirements, Good Requirements, Lexical Density, Requirements Clarity.

Introduction
One of the most common requirement engineering (RE) problems in industry is poor requirements quality. These problems relate to ambiguous, incomplete, inconsistent, incorrect, infeasible, unusable, or unverifiable requirements (D. Firesmith 2003). Hence, the quality of software product and the overall subsequent phases are influenced by the requirement phase quality (Alshazly, Elfatatry, and Abougabal 2014; Davis AE Didar Zowghi 2006) According to (Matsugu n.d.), the delivery of late product, poor product quality, degraded design and documentation integrity and invalid features caused by poor requirements is very real, and it has the potential to result in significant impacts. This problem is also proven by research conducted in (Anuar, Ahmad, and Emran 2015) that assert most documented requirement specification were in poor quality. This problem occurs due to the condition that many requirements engineers are inadequately trained as they do not have adequate access to stakeholders and other sources of the requirements or they are not given adequate resources or authority to properly engineer the requirements.

A common cause of IT project overrun is normally due to the impact of low quality requirements, especially in explaining the system behaviour. Common issues that contribute to unclear requirements are vagueness and misinterpretation on the written requirements. The lack of clearly is also due to insufficient detail for creation of required tests, the existence of gaps where system behaviour is unspecified and lack of clarity on specific requirements to be delivered (Roberts 2006). Therefore, it is essential for requirements to be clear and usable so that the stakeholder’s needs are readily found and understood, while mistakes and misunderstandings are avoided (Alsphaugh et al. 2006). According to (Peter Zielczynski n.d.) and (Kar and Bailey 1996), good requirements should be unambiguous, testable (verifiable), clear (concise, terse, simple, precise), correct, understandable, feasible (realistic, possible),
independent, atomic, necessary, implementation-free (abstract). Table 1 shows the example of several low characteristics requirements as per below:-

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incomplete</td>
<td>“The product shall provide status messages at regular intervals not less than every 60 seconds.”</td>
</tr>
<tr>
<td>Not Feasible</td>
<td>“The product shall switch between displaying and hiding non-printing characters instantaneously.”</td>
</tr>
<tr>
<td>Ambiguous</td>
<td>“The HTML Parser shall produce an HTML markup error report which allows quick resolution of errors when used by HTML novices.”</td>
</tr>
</tbody>
</table>

These poor characteristics of requirements are also the major problem to security requirements. This is because low quality security requirements leads to the failure of building secure software. Furthermore, the problems of eliciting and writing security requirements process are tedious and complicated. RE is required to have security experience in the process of eliciting consistent security requirements from the clients-stakeholders. In addition, elicited requirements are usually obtained from natural language, causing most RE to face problems to elicit security requirements from the clients-stakeholders as there are instances of mismatch between the real needs and the security terms used (Banerjee et al. 2015; Houmb et al. 2010). All these problems result in the production of low quality security requirements. This is supported by our survey in (Kamalrudin, Mustafa, and Sidek 2017a), which found that most of the customers did not clearly understand the security needed by their systems. In addition, the captured security requirements did not meet the standards requirements, such as NIST, ISO and Common Criteria.

This paper is organised as follows. After the introduction, we discuss the related works in background and motivation in Section 2. This is followed by Section 3, which proposes a new mechanism to check clarity level for security requirements. We realise the mechanism using real business requirements and certify product requirements. Then, we propose our security requirements tree structure and evaluated the density level. Next, we explain our results and discussions in Section 4. Finally, we end this paper with conclusion and future works.

Background and Motivations
During elicitation process, REs often use written natural language either by themselves or by the clients. These requirements are formed from the discussion and negotiation between both parties; clients and the requirements engineers. However, inconsistencies in requirements often occur due to natural language ambiguities and complexities (Kamsties and Paech 2000)(Kamsties n.d.). Additionally, tedious elicitation process also leads to the development of poor requirements. Besides, REs have the tendency to misunderstand the needs of clients-stakeholders and the security term of security compliance requirements during elicitation process (Houmb et al. 2010)(Banerjee et al. 2015). This problem arises because most them do not have security background. Even though they have undergone the training, they were only given an overview of security architectural mechanisms, such as passwords and encryption rather than the security in the actual security requirements (D. G. Firesmith 2003).

Available guidelines for writing good requirements have been proposed by (Boulila 2015) and these guidelines can be used as guidance for writing requirements. The adherence to this guideline together with the emphasis of quality aspects of the single requirements provide the basic ground to achieve quality requirement specifications. As in (Ramos et al. 2007), they proposed a collection of refactoring to improve the understandability of the requirements and the project organization. While, (Tjong, Hallam,
and Hartley 2006) proposed Natural Language Requirements Patterns for informality, imprecision and ambiguity reduction as well as increasing the quality of Natural Language Requirements Specifications (NLRS). Firesmith (2005) proposed an approach that uses a checklist to increase the quality of the requirement prepared by the requirements team developer. This approach also assists the evaluators to identify requirements specifications defects that exist in the associated requirements specifications during the evaluation process (Tjong, Hallam, and Hartley 2006).

Nigam et al. (2012) also contributed to improve the quality of requirements specification by developing a tool that can detect three types of ambiguities in SRS. This tool helps to detect and resolve ambiguities early as in the requirement analysis phase and it functions to assist requirement analysis personnel to review requirement specifications (Nigam et al. 2012). Besides, Heck and Parviainen (2008) has proposed a method that focuses on the requirement part to certify the quality of software product. It also helps to check the quality of requirements description by finding inconsistencies in the requirement specifications (Heck and Parviainen 2008). Other researcher (Sabriye and Wan Zainon 2018) used NLP technique called parts of speech tagging to automatically detect ambiguity and syntactic ambiguity. This research aims to enhance the understandability and quality of the requirement specifications. In summary, there are several numbers of works done in writing good requirements. However, none was found in security requirements contexts, specifically on the clarity on the syntax level.

**Approach**

Motivated from the gaps in the previous section, a mechanism to check the clarity level of the security requirements syntax is proposed. Here, we adopted the formulation of the Lexical Density (LD) to calculate the clarity level (Daller 2003). LD is used to show the proportion of lexical or content words to the total number of words. LD could possibly consist of nouns, verbs, adjectives and adverbs. We express the LD formula in equation below:

\[
LD = \frac{\sum \text{Lexical Words}}{\sum \text{Number of Words}}
\]

The research conducted by (Johansson 2009) found that content words with high proportion text carry more information than function words with high proportion. Here, functions words are the prepositions, interjections, pronouns, conjunctions and count words.

A high LD indicates that the written requirements are rich of information and are expected to be good and clear requirements, while low LD shows that the requirements lack information. Therefore, our main concern is to check the clarity in percentage for the selected real business requirements and the certified products. Further, we aim to propose our new approach, named security requirements tree structure (SecReqTS) that will assist REs to write good and clear requirements. We evaluate the level of LD provided by real business requirements and certified product to ensure the required percentage that need be fulfilled by certain security requirements using free online tool from (Analyze My Writing n.d.). We then compared the results provided by both requirements and then identify the minimum % requires as a clear security requirements.

**Calculation of LD**

This section demonstrates the evaluation process of LD using two types of requirements based, the first is (1) real business requirements of “Electronic Health Record (EHR)” ((SCP) 2009), and the second set of requirements is from certified product requirements ((CCRA) n.d.). These requirements were selected as they are considered as applications that have critical effect on the business performance of the respective domains. Any failure or interruptions in the systems may result in a great loss to the business entity. Therefore, we expected that security requirements have significant effect on the development of application.

**Real Business Requirements**

The first requirements is from Electronic Health Record (EHR) determined by the Statewide Collaboration Process (SCP) ((SCP) 2009). The requirements are designed for the technical implementation of the projects for the Statewide Health Information Network, New York (SHIN-NY) and interoperable electronic health records. Firstly, we elicit the EHR functional requirements. Next, we key-
in the requirements in the text editor provided by the tool, select the “Lexical Density” tab and click “Analyze Text” as shown in Figure 1(a).

![Analyze My Writing](image1.png)

**Figure 1(a):** Analyze My Writing (Analyze My Writing n.d.)

The tool then checks and analyses the parts of speech in every sentence(s) and displays the results in percentage. Each sentence(s) LD is calculated and the LD for the entire text is displayed as shown in Figure 1(b).

![Analyze My Writing Result](image2.png)

**Figure 2(b):** Analyze My Writing Result (Analyze My Writing n.d.)

We analyse more than 30 requirements and the results and average of the results are shown in Table 2.
Table 2: EHR LD Results

<table>
<thead>
<tr>
<th>Reg</th>
<th>Mean</th>
<th>Adj Mean</th>
<th>N</th>
<th>LD</th>
<th>N of Sents</th>
<th>LD</th>
<th>LD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.2</td>
<td>1.05</td>
<td>12</td>
<td>1.5</td>
<td>10</td>
<td>0.67</td>
<td>1.23</td>
</tr>
<tr>
<td>2</td>
<td>2.3</td>
<td>2.15</td>
<td>12</td>
<td>2.4</td>
<td>10</td>
<td>0.67</td>
<td>1.23</td>
</tr>
<tr>
<td>3</td>
<td>3.4</td>
<td>3.25</td>
<td>12</td>
<td>3.5</td>
<td>10</td>
<td>0.67</td>
<td>1.23</td>
</tr>
<tr>
<td>4</td>
<td>4.5</td>
<td>4.35</td>
<td>12</td>
<td>4.6</td>
<td>10</td>
<td>0.67</td>
<td>1.23</td>
</tr>
<tr>
<td>5</td>
<td>5.6</td>
<td>5.45</td>
<td>12</td>
<td>5.7</td>
<td>10</td>
<td>0.67</td>
<td>1.23</td>
</tr>
<tr>
<td>6</td>
<td>6.7</td>
<td>6.55</td>
<td>12</td>
<td>6.8</td>
<td>10</td>
<td>0.67</td>
<td>1.23</td>
</tr>
<tr>
<td>7</td>
<td>7.8</td>
<td>7.65</td>
<td>12</td>
<td>7.9</td>
<td>10</td>
<td>0.67</td>
<td>1.23</td>
</tr>
<tr>
<td>8</td>
<td>8.9</td>
<td>8.75</td>
<td>12</td>
<td>9.0</td>
<td>10</td>
<td>0.67</td>
<td>1.23</td>
</tr>
<tr>
<td>9</td>
<td>9.0</td>
<td>8.85</td>
<td>12</td>
<td>9.1</td>
<td>10</td>
<td>0.67</td>
<td>1.23</td>
</tr>
<tr>
<td>10</td>
<td>10.1</td>
<td>9.95</td>
<td>12</td>
<td>10.2</td>
<td>10</td>
<td>0.67</td>
<td>1.23</td>
</tr>
</tbody>
</table>

Standard Certified Products
The second requirements is from certified product requirements by Common Criteria ((CCRA) n.d.) which consists of five categories. The chosen requirements is from the most certified product category, which has 1134 certified products. Firstly, we elicit the EHR functional requirements. Next, we key-in the requirements in the text editor provided by the tool, select the “Lexical Density” tab and click “Analyze Text” as shown in Figure 2(a).

![Figure 2(a): Analyze My Writing](Analyze My Writing n.d.)

The tool then checks and analyses the parts of speech in every sentence(s) and displays the results in percentage. Each sentence(s) LD is calculated and the LD for the entire text is displayed as shown in Figure 2(b).
We analyse four (4) certified product requirements and the results with average density level are shown in Table 3(a), Table 3(b), Table 3(c) and Table 3(d) respectively.

Table 3(a): Product 1 LD Results

Table 3(b): Product 2 LD Results

Table 3(c): Product 3 LD Results

Table 3(d): Product 4 LD Results
Design of Security Requirements Tree Structure

We has proposed our security requirements tree structure (SecReqTS) in (Kamalrudin, Mustafa, and Sidek 2017b) as a guidance to the REs for writing security requirements. To do this, we used syntax analysis and keyword matching to parse the textual natural language security requirements to the check the structure of the security requirements. Basically, we derived SecReqTS by analysing and comparing all security requirements from the business requirements to produce the common pattern. In addition, we compared the common pattern that we derived with the sentence structure in the standards. Finally, SecReqTS is composed as in Figure 2.

![Figure 3: SecReqTS (Kamalrudin, Mustafa, and Sidek 2017c)](image)

As shown in Figure 2, SecReqTS consists of six elements. Each of the elements is explained in Table 6 as per below:-

<table>
<thead>
<tr>
<th>Table 4: SecReqTS Terms Definition (Kamalrudin, Mustafa, and Sidek 2017c)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Term</strong></td>
</tr>
<tr>
<td>Subject</td>
</tr>
<tr>
<td>Verb</td>
</tr>
<tr>
<td>Security Keywords</td>
</tr>
<tr>
<td>Security Mechanism</td>
</tr>
<tr>
<td>Object</td>
</tr>
</tbody>
</table>

We then evaluated the LD for our SecReqTS as shown in Table 5 below:-

Submit Date: 05.07.2018, Acceptance Date: 22.08.2018, DOI NO: 10.7456/1080SSE/320

Research Article - This article was checked by Turnitin
Copyright © The Turkish Online Journal of Design, Art and Communication
Results and Discussions

LD of Real Business Requirements and Certified Product
Based on the density evaluation, LD for business requirements presented in the previous section has an average of 58% of clarity level. More than half of the requirements have below than average density level. While, for the four certified products selected, we found that the average is in between 55-65%, which are 59.74%, 64.96%, 61.09%, 54.88% respectively. It is clearly shown that the certified product requirements have higher percentage in comparison to the real business requirements, which shows that the existing standards provides better requirements guidelines compared to normal business requirements.

LD of Security Requirements Syntax
Based on the findings, LD for security requirements tree structure presented in the previous section has an average of 67.93% of clarity level. In comparison with the real business requirements, our security requirements structures are proven better because the density level are higher. Subsequently, our structure also contributed higher percentage than the certified product. It has been shown that our security requirements tree structure contributes higher clarity percentage and aligns with the standards. Thus, we believe our structure has the ability to assist REs to write good security requirements due to the proposition that higher density contributes to a clearer meaning of requirements.

Conclusion and Future Works
In summary, we found that the clarity percentage is more than 65% using our SeqReqTS approach. It can be seen that our approach is good as comparable to certified product results. Significantly, this finding contributes towards writing clear security requirements leading to produce good security requirements. As for future work, we are motivated to improve our security requirements approach by incorporating our template with stakeholders concerns to ensure that our approach considers the concern’s of the stakeholders in developing secure software. We strongly believe that our approach will simplify the way of writing good security requirements to support novice requirement engineers.

Acknowledgement
We would like to express our gratitute to the university, UTeM and MoHE for the research funding: FRGS/1/2015/ICT01/FTMK/02/ F00325.

REFERENCES
“Analyze My Writing.”
——. 2017b. “A Template For Writing Security Requirements.” In Submitted to APRES 2017,.
Matsugu, Brad. “Poor Requirements - What Impact Do They Have?”
Peter Zielczynski. Requirements Management Using IBM® Rational® RequisitePro®.
Ramos, Ricardo et al. 2007. “Improving the Quality of Requirements with Refactoring.” In VI Simpósio Brasileiro de Qualidade de Software, (SBQS2007),.
Roberts, Clare. 2006. “Do Your IT Projects Suffer from Requirements Clarity Issues?”