Secure Software Development Awareness: A Case Study of Undergraduate Developers

Murimo Bethel Mutanga
Department of ICT, Mangosuthu University of Technology, South Africa
{mutangamb@mut.ac.za}

Abstract
As ubiquitous computing becomes an increasingly inherent component of everyday life due to the rapid growth of communication technologies and globalization, threats against information systems have taken a more latent yet lethal dimension. This emergent digital security challenge has correspondingly motivated a proactive change in the software engineering process in recent decades. This change has inspired more intense research scrutiny on security as a crucial component of any software system. Moreover, in today's virtual world of hyperconnectivity, the most significant vulnerabilities in modern information systems security are software centred. Nevertheless, research shows that software developers often lack the required knowledge and skills in secure software systems development (SSD). Such knowledge ensures that all the resultant software components of each development lifecycle are correctly implemented rather than merely following the SSD lifecycle. Also, the knowledge engenders software security consciousness as a professional attitude amongst developers. Therefore, investigating students' awareness of SSD principles can generate insight into evolving the undergraduate software development curriculum – a path to building future career developers. The study used a voluntary online survey to recruit a sample of 76 undergraduate developers and employed a descriptive approach to data analysis. Among other findings, the study revealed that participants' perception of the threat of software vulnerability impacts their attitude towards security on online and mobile platforms. And that though over 90% of the undergraduate developers took software vulnerability threats either "serious" or "extremely serious", this disposition did not reflect the depth of their knowledge and experience in SSD.

Key Words - **Cyber-security, Framework, Software, threat, ubiquitous-computing, vulnerability**.

1. Introduction

Traditionally, security in software development is often viewed either as a remedy or patch deployed to solve security breaches or as an enhancement to a wholly developed software package [1]. As further emphasized by Alkussayer and Allen [1], developers only pay attention to security considerations as they approach the end of the development lifecycle, which is why such security solutions often come as add-on mechanisms and techniques before software systems deployment. Therefore, security issues were often reactively addressed when prompted by some undetected vulnerability or when such vulnerability may have even been exploited [1], [2].

However, in recent decants, there has been a pragmatic change away from this mundane approach for security in software development to embrace a more proactive approach that advocates the deliberate injection of forethought security ramifications into all stages of the software development lifecycle [1]–[4]. The emergent alternative to secure software development primarily recognizes security requirements as an integral element of the software design and development process; therefore, rather than treating security requirements as an ad-on or a corrective measure, it is implemented as a "designed-in" component.

It has become increasingly imperative to strengthen or reengineer the existing processes for developing secure software. The advancement of the internet and the proliferation of other related sophisticated technologies have escalated the scale of cyber threats against information systems [5]. For instance, as ubiquitous computing becomes an increasingly inherent component of everyday life, it has become increasingly easy to use these technologies in complex ways [6]. However, cyber adversaries who thrive on exploiting information systems vulnerabilities take undue advantage of this ease-of-use and the pervasiveness of the internet [7], [8].

Furthermore, in the broad context, information security research focuses on two fundamental drivers of vulnerability: people-oriented and software-oriented factors [9]. While the former can constitute a potential loophole in information systems security [9], [10], faulty software development in utilizing the appropriate security requirements represents the core weakness in the landscape of information system security. According to Luo et al. [11], such weaknesses are "defects in software's specific implementation or system security policy, which can enable attackers to access or damage the system without authorization".

On the other hand, research on how students undertake software development abound. For example, in the context of this study, the work reported in [12] discussed students' software development knowledge at a more general level.

However, current literature suggests that there has been more emphasis on improving students' programming skills and optimising teaching programming techniques [13]–[15]. But scanty investigations tend to probe students' awareness of emergent security challenges and the state-of-the-art software development principles designed to guarantee secure systems development.

The fact is, as the world becomes more and more interconnected, the landscape and implications of information systems security have drawn more concerns than ever. For instance, amongst other technological evolutions, the emerging trend of the Internet of Things (IoT) has gained momentum in recent years. In conjunction with mobile communication technologies, IoT facilitates the design and supports the deployment of intelligent and ambient devices [16], which essentially makes it possible for things and objects to interact and cooperate between themselves [17] autonomously. With these advances, society will get smarter and smarter with the gradual shift in focus to adopting innovative software-driven systems as the hub of critical resource management to ensure convenient and efficient resource administration and service delivery [8]. At an industrial level, the popularity and reliance on IoT are already rising with critical applications such as smart grids, smart cities, IoT connected factories, smart supply chain management, connected healthcare systems, and smart farming.

The dominant role of software systems is not limited to the industrial sector as governments, research, and other corporate establishments are also heavily reliant on information technology these days. Therefore, software security breaches can have far more reaching consequences.

Because of the essential nature of the global challenge of securing information systems, this study investigates awareness of secure software development principles among South African undergraduate students. This study is motivated first because South Africa has one of Africa's most funded educational systems, the government's strategic interest in advancing local technological content. Second, according to Vadra [18], South Africa's inclusion into the four-member grouping of fast-emerging economies of the world, namely, Brazil, Russia, India, and China, is both a mark of the country's development strides in the African continent and most importantly, a call for a potential shift in focus knowledge-based economy in alignment with the other countries with the block.

As a whole, the findings of this study provide helpful insight into the level of preparedness of upcoming undergraduate programmers to effectively contribute toward secure software development in the industry. Such understanding also contributes to improving the current computer programming teaching curriculum to sufficiently equip undergraduate students with the expertise to address the information system security challenge through secure application development. This study surveyed 2nd and 3rd undergraduate information technology students from a South African University of Technology and qualitatively analyzed the data.

2. Literature Review

There is increasing pressure on software development teams to deliver secure code, as reflected in Forbs' report on cybersecurity [19] as cited in [20]. The report asserts that "Software security and privacy are becoming major issues: almost every week we hear that yet another organization's software systems have been compromised." Yet, this report only reflects a subspace of the emergent cyberwar, which threatens private lives, organizations, and even governments.

While there are many critical factors contributing to achieving an entity's security and privacy, the software used unarguably plays the most central role in whether security breaches occur or not [20]. This argument explains why the emphasis is on secure software development and, therefore, further stresses the crucial responsibility of developers. The authors in [21] explicitly encapsulate this fack when they maintained that "If software developers fail, the cyber-security system fails, which may lead to data breaches." A report by Veracode [22] on the state of software security indicates that over "85 percent of all applications have at least one vulnerability in them and more than 13 percent of applications have at least one very high severity flaw". The authors further cited an IBM-funded survey of 640 participants working for US-based companies developing mobile, revealing that 73% believed that the primary contributor to the challenge of security issues is the developers' lack of understanding of security issues.

The forgoing has birthed a growing and diverse interest in the secure software development research space. For example, in [23], the authors sought an "in-depth understanding of how and why software developers produce security bugs"— their study contributes to the design of interactive tool support for secure software development. Interestingly, the study reveals that developers' conceptual understanding of security is often not aligned with "their attitudes regarding their responsibility and practices for software security". Therefore, they suggest that understanding software security from a developer's perspective is crucial for mitigating security errors. In the same vein, Graff & Van-Wyk R. (2003) [24] postulated "Good People Write Bad Code" for the following three reasons i) Technical factors which have to do with the underlying complexity of the task itself being solved, ii) Psychological factors such as poor mental models and, iii) real-world factors which include, but not limited to production pressures and

absence of financial motivations.

In challenging the status quo in secure software development practice, which relies on using static tools such as checklists, processes and errors to avoid to guide developers, Weir et al. (2020) [20] found the dialectical between the developers and a range of counterparties spans throughout the software development life cycle. Therefore, they proposed "six assurance techniques that are most effective at achieving this dialectic in existing development teams". According to the author, their assurance techniques and corresponding dialectical interactions can potentially enhance the security of development activities.

Another early scholarly study in [25] reports on a framework and methodology for studying the causes of software errors. Among others, in the proposed framework, the authors highlighted the "skill, rule, and knowledge breakdowns" as a crucial human error component in programming activity. Also, Assal & Chiasson (2019) [26] acknowledged the persistence of software vulnerabilities notwithstanding software security initiatives and drive for best practices are thriving in the recent decade. Consequently, Assal & Chiasson (2019) [26] "explore the interplay between developers and software security processes", primarily focusing on human factors of software security, which include the developers' behaviour and motivation. The analysis attributes the problem of security vulnerabilities to "a lack of organizational or process support".

The subject of secure software development has gained research attention widely. However, little has been reported about the impact of developers' attitudes and their lack of understanding of security on secure software development. This study builds on the existing literature to contribute to the above gap.

3. Research Methodology

This section presents the entire research design, which includes the description of the population, sample categorization, data collection instrument, and the approach to data analysis.

3.1 Study Participants

This study targets undergraduate 2nd and 3rd-year Information and Communication Technology students. On the one hand, it was assumed that the 2nd year students have enough exposure to software development training because their three-year study curriculum is highly streamlined to specialize in software development or network engineering. And on the other hand, aside from being in their final year of study, the 3rd year students were undergoing their compulsory work-integrated learning program (WIL), which exposes them to various real-life industry experiences. These two scenarios make the selected population most appropriate for this investigation.

The study used a sample of 76 students drawn up from the study population as described above. And Table 1 presents the characterization of the study's participants.

Table 1: Characterization of the study's population

Gender		Year of Study	
Male	56	2nd	43
Female	20	3rd	33
Total	76		

1.1 Data Collection and Analysis

This study employed an online questionnaire-based survey for gathering data. This data collection method utilized Google Forms – a customizable virtual survey tool that allows researchers to create suitable questionnaires following an existing template. The instrument's suitability was ascertained using a closed-ended questionnaire designed and subjected to evaluation by an independent expert. The validated questionnaire was then used to create a customized questionnaire on Google Forms. This questionnaire elicited demographic and other information related to students' awareness of secure software development principles.

Due to the nature of the information collected, a descriptive approach was used to analyze the data. And a question-by-question analysis was performed to ascertain the level of students' awareness of secure software development principles.

4. Results and Discussion

In this section, a summary of the study's results is presented. First, the data were quantitatively analyzed and presented section-by-section according to the questionnaire design. Second, the results are then interpreted in the discussion subsection section.

In this section, a summary of the study's results is presented. First, the data were quantitatively analyzed and presented section-by-section according to the questionnaire design. Second, the results are then interpreted in the discussion subsection section.

4.1 How students Perceived the threat of software vulnerability

This investigation asked two background questions to explore students' understanding of and, by extension, their attitude towards the critical issue of software security as a global and professional challenge. These questions were as

follows:

Question 1(a): How would you describe the threat posed by software vulnerability to information system security? As stated earlier, this question was asked to enable the researcher to elicit information that can help make the existing software development curriculum more robust and better aligned with current software security realities. Most importantly, such a curriculum can improve the quality of graduate developers in South Africa by ensuring that they are well-grounded in the ethical, theoretical, and professional responsibility of being security conscious when developing commercial systems. Fig. 1 illustrates the outcomes.

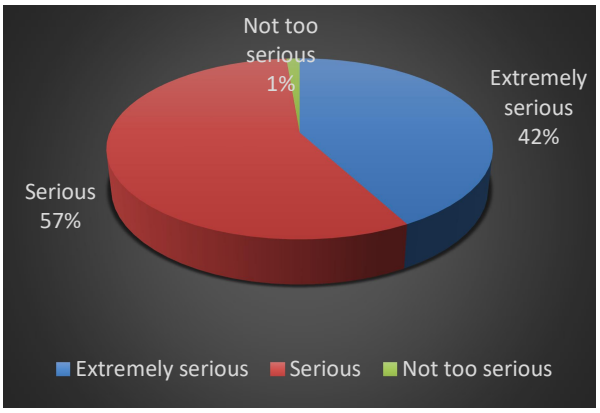

Figure 1: Perception of software vulnerability threat

From the analysis illustrated in Fig. 1, more than 90% of the surveyed population said they took software vulnerability threats either "serious" or "extremely serious". The implication of this outcome is that majority of the undergraduate developers are fully aware of the threat posed by software vulnerability and, therefore, take it as a severe threat to information systems.

Question 1(b): As a developer, which of these types of systems would give you the most security concerns? i) Web/online systems, ii) Mobile systems, iii) Desktop systems.

Understanding the threat posed by software vulnerability to information systems is one thing and knowing the implication of this threat to different information systems is another thing. Therefore, the second question highlighted the students' understanding of the information systems most at risk.

The responses to the second question provided an exciting perspective of the software security awareness of the respondents. This perspective evoked the categorization of the respondents into three groups (G1 – G3), as shown in Fig. 2. G1 represents participants that only picked either Web/online systems or Mobile systems as a security concern. While the responses under the G1 category are not wrong, in the context of the study, such responses reflected a narrow scope of the software vulnerability landscape. On the contrary, participants who believed that both Web/online systems and Mobile systems post the most security concerns were labelled G2. This group was described as "well informed" because their view reflected an accurate understanding of the reality of software systems' vulnerability. Still, the other category of participants, labelled G3, captured responses that included Desktop systems. Such respondents were tagged "unaware" because desktop systems pose the most minimal security risk than the other systems listed in question 2.

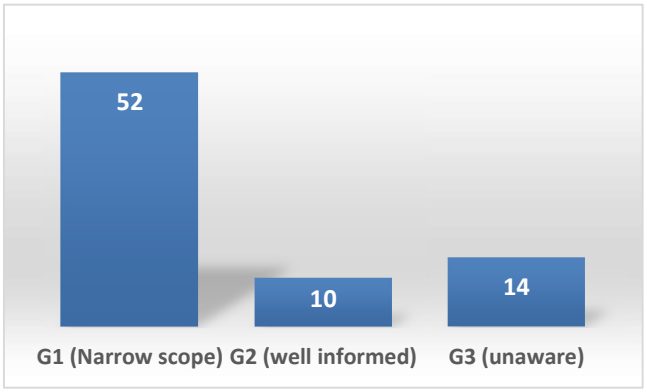

Figure 2: The Scope of information Vulnerability

The consequences of having a narrow scope of software security can equally potentially undermine information systems as being unaware. In this regard, it can be argued that it essentially makes no difference for a developer to have a narrow scope or be unaware of software security vulnerabilities. Therefore, when interpreted in this sense, Fig. 2. above translate to Fig 3 below:

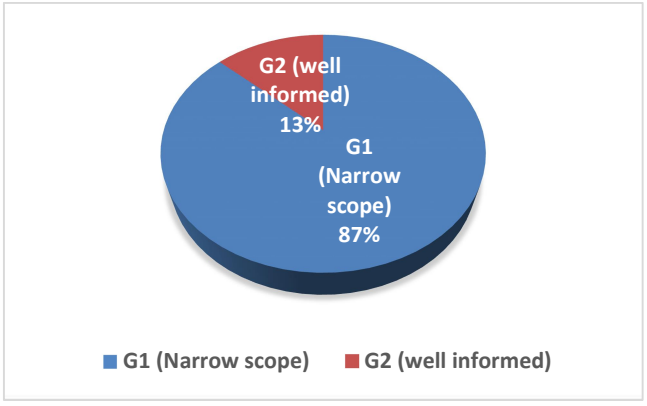

Figure 3: A translation of Figure 2

From the illustration in Fig 3, it becomes striking to note that the result revealed a 74% gap between the developers who are well abreast of the scope of the software system's vulnerability and others who are still unaware. Only 13% of the sample demonstrated an adequate understanding of what software systems they, as future developers, must design with utmost security concerns.

4.2 Examining participants' knowledge of state-of-the-

art industry standards for secure software development.

The need for developers to be informed about the nature and scope of the threat of software vulnerability is hugely critical but being a future career developer requires more knowledge about existing standards for developing secured systems. Therefore, the study poses three sub-questions aimed at helping the study examine how participants are consciously aligning their undergraduate software development experiences and skills with professional standards.

Question 2(a): Do you know about any existing secure software development frameworks (SSDF)? This question tested the extent of the participants' familiarity with existing and most popular professional standards guiding secure software systems development. The outcome is as presented in Fig. 4.

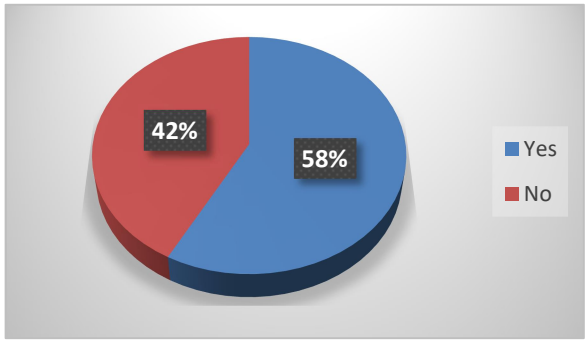

Figure 4: Familiarity with secure software development frameworks

The results depicted in Fig. 4 suggest that a significant number (42% of the sample) of undergraduate developers have either not theoretically or practically interacted with the fundamental frameworks of secure software development. This number is significant because it is only 16% less than the number of participants who reported that they knew about some existing frameworks for developing secure software.

Question 2(b): If you answered "yes" to 3(a), then select all the frameworks that you have known from the list below and continue with 3(c): i) The Fundamental Secure Software Development Guide, ii) The Microsoft SDL, iii) The Integrated Security Development Framework (ISDF), iv) The OWASP's CLASP, v) Software Security TouchPoints.

With this question, the study validates the participants' knowledge depth. The findings enabled the researcher to understand whether the participants can demonstrate classroom knowledge leading to experiential or applicational knowledge (practical experience) or just classroom knowledge. This goal aligns with Kolb's learning framework, which argues that learning is only proven successful when the learners can try out whatever has learned (active experimentation) [19].

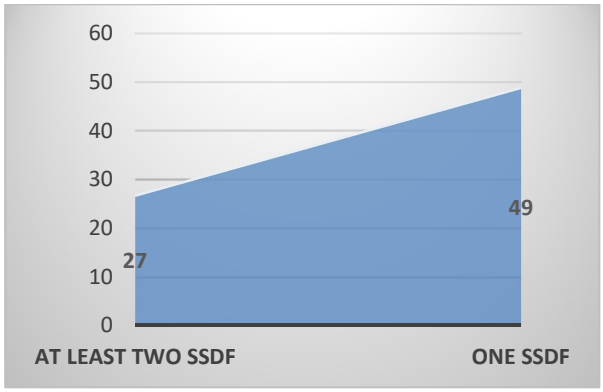

Figure 5 Extent of classroom knowledge about specific SSDFs

As presented in Fig. 5, the results show that fewer participants had a broad understanding of the existing SSDFs. For example, only 36% of the sampled developers know at least two existing SSDFs or have applied them. Whereas 49 participants, representing 64% of the sample, learned only one SSDF or had used it.

Question 2c: Choose the option below that best suits your knowledge of the framework(s) that you selected in 3b above: i) I have only learned about the framework(s), ii) I learned about the framework(s) and applied its outlined best practices.

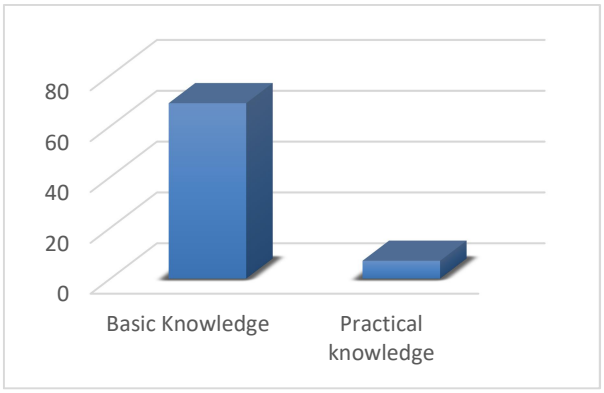

Figure 6: Applicational knowledge of specific SSDFs

The results in Fig. 5 show that 64% of the sample knew at least one SSDF or have applied it. However, as demonstrated in Fig. 6, a further investigation revealed that 69 out of the 76 (91%) admitted that they had only learned about some of the SSDFs. But have not practically applied any of the SSDFs in their software development practice. This finding, therefore, suggests that just 9% of the sampled undergraduate developers have experiential knowledge of software development frameworks in context.

4.3 Attitude towards software systems security

Xie et al. [23] show a disconnect between developers' conceptual understanding of security and their attitudes regarding their responsibility and practices for software

security". Consequently, the next question provided insight into whether the participant's attitude towards online and mobile platforms relates to the way they perceive and may likely handle software security as developers. This aim was achieved in this study by asking the following four questions:

Question 3(a): Choose all the online and mobile platforms (OMPs) that you often use from the list below: i) Facebook, ii) Twitter, iii) Instagram, iv) WhatsApp, v) Linkedln, vi) Email, vii) LMS.

With this question, the researcher sought to understand the participant's level of involvement in the use of various online and social media platforms. This question is motivated by the fact that these platforms constitute software systems' vulnerability tipping point.

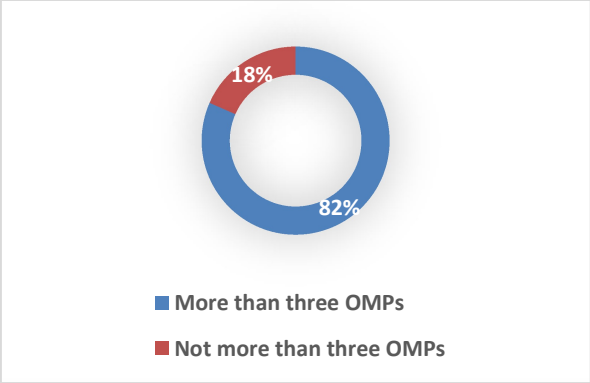

Figure 7: The use of online and mobile platforms

From Fig. 7, all the participants widely used online and mobile platforms. For instance, the results indicate that 62 of the 76 (83%) participants used at least three of the listed OMPs. And the remaining 18% of the sample used at least one but not more than three OMPs.

Question 3(b): Indicate what you usually do when using social media and other online platforms? i) Use personal security settings (UPSS) ii) Use the same password across more than one platform (USPAP) iii) Share password (SP) iv) Use personal security settings, Use the same password across more than one platform (UPSS/SPAP).

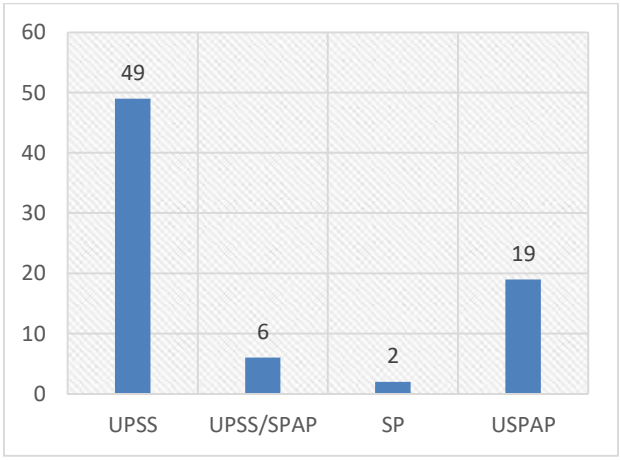

Figure 8: Personal responsibility when using OMPs - 1

Responses to question 3(b) above, as given in Fig. 8, indicate that most participants showed an attitude of security consciousness. For instance, 54 out of the 76 participants used personal security settings, representing 72% of the sampled population. At a personal level, this approach shows that such participants often take personal responsibility to prevent software security breaches. On the contrary, the result also indicates that 21 participants do not use personal security settings. Instead, these participants indicated sharing their passwords or using the same password across different OMPs.

Based on the latter finding, further investigation became apparent. Therefore, as depicted in Fig. 9, the outcome of Fig. 8 was linked to the participants' earlier response to question 1(a).

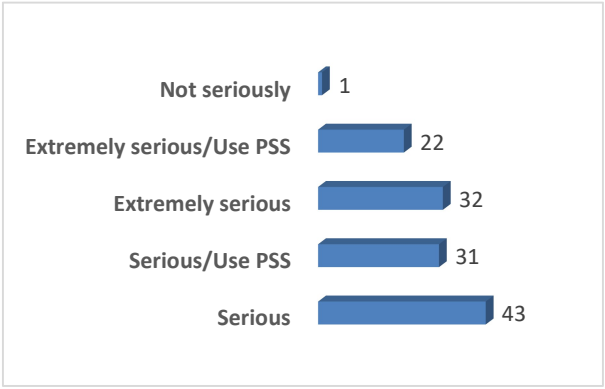

Figure 9: Percept about software security threat vs attitude toward software security 1

As presented in Fig. 9, the results suggest that the participant's attitude towards security on online and mobile platforms is influenced by their perception of the threat of software vulnerability. This claim substantially supports the above findings showing that all the participants who either took the threat of software security "serious" or "extremely serious" were found to use personal security settings. Specifically, only 12 out of the 43 participants who took software security threats "serious" did not use

personal security settings when using various OMPs. And only 10 out of 32 of those who took the threat "extremely serious" did not use personal security settings when using OMPs.

Question 3(c): When using social media platforms, do you change the security and privacy settings or change your passwords regularly? This question provided more information on how the participants explored existing software security features on OMPs. Such information further reveals participants' attitudes towards the threat of software vulnerability.

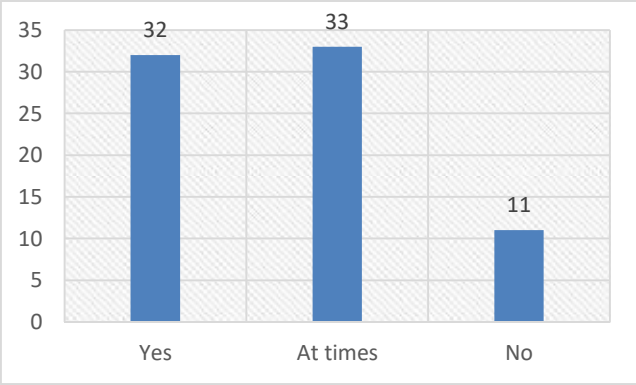

Figure 10: Personal responsibility when using OMPs - 2

As illustrated in Fig. 10, the questions' responses show that most (65 out of 76, that is over 85%) of the respondents either change their passwords regularly or at times. When the results in Fig. 10 were linked to the participants' earlier response to question 1(a), the outcome in Fig. 11 the study further linked participants' attitude towards cyber security to how they perceived the threat of software vulnerability.

For example, as depicted in Fig. 11, the findings suggest that out of 43 participants that claimed to take the threat of software security either "serious", 35 also admitted that they change their password regularly (20) or at times (15). Similarly, of the 32 participants that admitted to taking software security threats "extremely serious", 30 change their password at least sometimes.

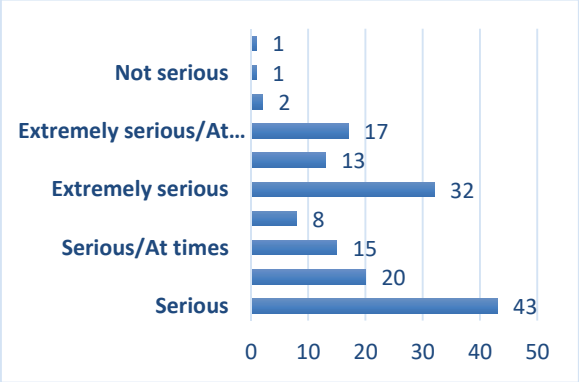

Figure 11: Percept about software security threat vs attitude toward software security 2

4.4 Self-confidence in the knowledge of secure software development (SSD)

Another factor investigated in this study was the participants' confidence in the fundamentals and practices of secure software development. Therefore, in this question, the participants were required to assert their confidence level and provide a personal assessment of their current curriculum with regard to software security.

Question 4: Do you think you have sufficient knowledge in secure software development? If not, what do you think is lacking in your current curriculum?

The above question became imperative because this study sought to enhance the existing software development curriculum. Therefore, the data elicited from the question helped shape the study's contribution.

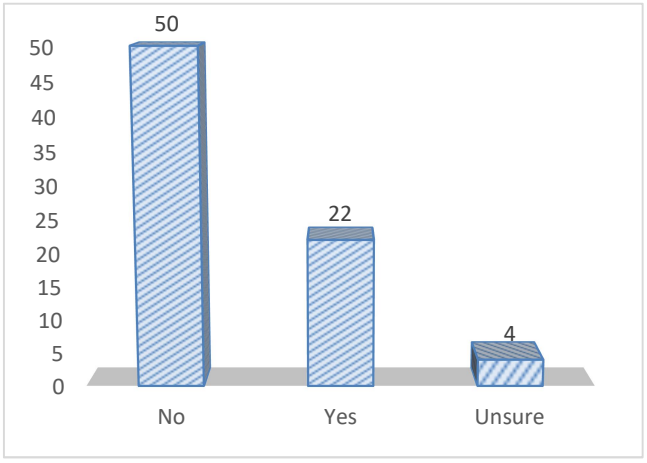

Figure 12: Self-confidence in the knowledge of SSD

Although earlier findings, as shown in Fig. 5, show that 64% of the participants knew at least one of the SSDF, the analysis in Fig. 12 indicates that the majority of the respondent somewhat low confidence in their knowledge of SSD. For instance, concerning question 4 above, only 22 participants responded in the affirmative, while 50 participants admitted they had no confidence in their SSD knowledge.

The responses to the second part of question 4 helped capture the participants' expectations or verdict or their current curriculum, as illustrated in Figs. 13 and 14.

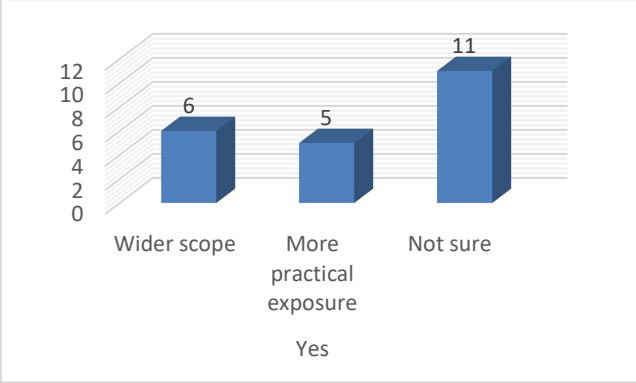

Figure 13: Personal assessment of software development curriculum 1

Fig. 13 presents further analysis of the results in Fig. 12. The first part of the analysis focused on the participants who answered "Yes" to the question: Do you think you have sufficient knowledge in secure software development? As illustrated in Fig. 13, the result indicates that half the number of the participants who acknowledged the confidence in their SSD knowledge were unsure of what may be required to enhance the existing curriculum. But the rest of the participants either believed that extending the scope of the current curriculum on SSD or providing a platform or practical exposure can make a huge difference, as one of the participants stated:

"*Yes, I can say I have the knowledge, but it is not much enough. When it comes to the topic of secure software development, we need to dive deeper and learn everything because they are very important.*"

Similarly, another participant admitted:

"*Yes, I do have knowledge on secure software development even though it is just basic knowledge. I don't really know a lot of detail in the concept, but I have knowledge.*"

Concerning the participant that responded "No" to the question in retrospect, Fig 14 demonstrates the findings. And the results suggest that 44, representing 88% of the participants, saw the need to enhance the existing SD curriculum.

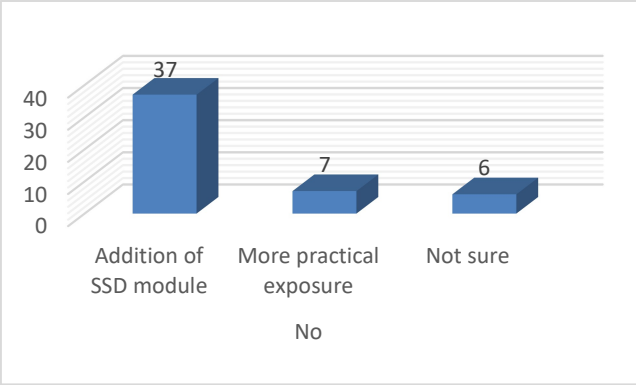

Figure 14: Personal assessment of software development curriculum 2

In responding to what may be lacking in the current curriculum, 37 wants the curriculum to be expanded by explicitly adding a module on SSD. In that respect, the participants quoted below seemed most explicit and representative of the entire feedback.

Participant A: "*No, Security has only been scratched on the surface, but we haven't really dived deep into it and implemented the necessary practices for secure systems outside the obvious sign in and login password.*"

Participant B: "*No, I think we need a specific module or course that teaches practically secure software development*".

Participant C: "*No, because the technology industry is always evolving, so the knowledge that I have on security might be outdated. So, the curriculum must keep up with the times in terms of security updates as a developer.*"

On the contrary, though the remaining 7 participants admitted their SSD inadequacy, they opined that the current curriculum would serve them better if it offered them adequate provision for practical exposure.

5. Discussion and conclusion s

This research attempted to understand undergraduate software developers' perception of software vulnerability threats and the developer's response to information system security. Essentially, the study explored the participants' knowledge of secure software development standards and principles and demonstrated how the developers' sense of personal responsibility in leading a professional attitude of software security consciousness impacts their perception of software security threats.

The study is motivated by the fact while the demand for software is rapidly growing, the risk of software vulnerabilities equally increases proportionately. Therefore, it has become pertinent to ensure that future career developers are adequately armed with the relevant knowledge and skills in secure software development. Despite the study's relatively moderate sample (76), primarily due to employing a voluntary online survey, the key findings still offered valuable insights that informed the recommendations made.

An overview of and reflection on some key findings is as follows:

- An overwhelming majority of over 90% of the surveyed undergraduate developers took the threat of software vulnerability either "serious" or "extremely serious". Nevertheless, subsequent results suggested that such a majority did not necessarily reflect the depth of their knowledge and experience in secure software development.
- Regarding the awareness software system's vulnerability, the gap between the well abreast undergraduate developers and the others who are

still unaware constituted a striking 74% of the sample. The implication is that lack of adequate practical or simulated secure software development experience may undermine undergraduate developers' understanding of the software vulnerability threat.

- The participants' theoretical knowledge of secure software development framework was constrained by their lack of experiential knowledge, as 91% of the participants admitted that they had only learned about some of the SSDFs but had not applied them.
- The participant's attitude towards security on online and mobile platforms was influenced by their perception of the threat of software vulnerability. Therefore, it can be argued that unless professional training or ethics override the developers' perception of software vulnerability threats, their handling of security in software development may be compromised.
- A vast majority of the sampled undergraduate developers feel dissatisfied with the current software development curriculum. Of this majority, 74% advocate the addition of a module that would explicitly deal with secure software development, while 14% expressed the need for more practical exposure.

Centrally, the study's findings, as a contribution, echoed the need to redesign the undergraduate software development curriculum of South African universities of technology in a manner that would guarantee two things. First, to incorporate and facilitate the use of state-of-the-art platforms that can enable undergraduate developers to gain real-life exposure in software security programming. Second, formulate a standard curricula review mechanism to ensure the curriculum evolves in alignment with current trends in the industry.

References

[1] A. Alkussayer and W. H. Allen, "The ISDF Framework: Towards Secure Software Development," *J. Inf. Process. Syst.*, vol. 6, no. 1, pp. 91–106, 2010.

[2] N. Davis, W. Humphrey, S. T. Redwine, G. Zibulski, and G. McGraw, "Processes for producing secure software: Summary of US national Cybersecurity Summit subgroup report," *IEEE Secur. Priv.*, vol. 2, no. 3, pp. 18–25, May 2004.

[3] S. Faily and S. Faily, "Usable and Secure Software Design: The State-of-the-Art," in *Designing Usable and Secure Software with IRIS and CAIRIS*, Springer International Publishing, 2018, pp. 9–53.

[4] B. Bafandeh Mayvan, A. Rasoolzadegan, and Z. Ghavidel Yazdi, "The state of the art on design patterns: A systematic mapping of the literature," *J. Syst. Softw.*, vol. 125, pp. 1339–1351, Mar. 2017.

[5] M. Z. Gunduz and R. Das, "Analysis of cyber-attacks on smart grid applications," in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, Sep. 2018, pp. 1–5.

[6] S. Ghafur, E. Grass, N. R. Jennings, and A. Darzi, "The challenges of cybersecurity in health care: the UK National Health Service as a case study," *Lancet Digit. Heal.*, vol. 1, no. 1, pp. e10–e12, May 2019.

[7] C. Heitzenrater and A. Simpson, "A case for the economics of secure software development," in *ACM International Conference Proceeding Series*, Sep. 2016, vol. 26-29-Sept, pp. 92–105.

[8] Z. A. Baig *et al.*, "Future challenges for smart cities: Cyber-security and digital forensics," *Digit. Investig.*, vol. 22, pp. 3–13, 2017.

[9] S. Omar, T. Frimpong, and J. B. Hayfron-Acquah, "Information System Security Threats and Vulnerabilities: Evaluating the Human Factor in Data Protection," *Int. J. Comput. Appl.*, vol. 143, no. 5, pp. 0975 – 8887, 2016..

[10] M. Sharma and S. Kaur, "Cyber Crimes Becoming Threat to Cyber Security," *Acad. J. Forensic Sci.*, vol. 2, no. 1, pp. 2581–4273, 2019.

[11] C. Luo, W. Bo, H. Kun, and L. Yuesheng, "Study on Software Vulnerability Characteristics and Its Identification Method," *Math. Probl. Eng.*, vol. 2020, pp. 0–6, 2020.

[12] I. Bassey, D. Afuro, and M. Munienge, "An Investigation of Software Engineering Knowledge of Undergraduate Students," *Int. J. Mod. Educ. Comput. Sci.*, vol. 7, no. 12, pp. 42–50, 2015.

[13] B. Isong, O. Ifeoma, and N. Gasela, "On the integration of agile practices into teaching: An approach to overcoming teaching and learning challenges of programming," in *Proceedings - 2015 International Conference on Computational Science and Computational Intelligence, CSCI 2015*, Mar. 2016, pp. 264–270.

[14] S. Biju, "Benefits of Working in Pairs in Problem Solving and Algorithms - Action Research," *Athens J. Educ.*, vol. 6, no. 3, pp. 223–236, Jan. 2019.

[15] B. Isong, "A Methodology for Teaching Computer Programming: first year students' perspective," *Int. J. Mod. Educ. Comput. Sci.*, vol. 6, no. 9, pp. 15–21, 2014.

[16] Y. Changsheng, H. Kaibin, and C. Hyukjin, "Energy Efficient Mobile Cloud Computing Powered by Wireless Energy Transfer - IEEE Journals & Magazine," *Sci. World J.*, vol. 34, no. 5, pp. 1757–1771, 2016.

[17] C. Stergiou, K. E. Psannis, B. G. Kim, and B. Gupta, "Secure integration of IoT and Cloud Computing," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 964–975, Jan. 2018.

[18] R. Vadra, "Knowledge Economy in BRICS: a Case of South Africa," *J. Knowl. Econ.*, vol. 8, no. 4, pp. 1229–1240, Dec. 2017.

[19] Forbs, "Top 2016 Cybersecurity Reports Out From AT&T, Cisco, Dell, Google, IBM, McAfee, Symantec And Verizon," 2016.

[20] C. Weir, A. Rashid, and J. Noble, "Challenging software developers: dialectic as a foundation for security assurance techniques," *J. Cybersecurity*, vol. 6, no. 1, pp. 1–16, 2020.

[21] A. Alhazmi and N. A. G. Arachchilage, "I'm all ears! Listening to software developers on putting GDPR principles into software development practice," *Pers. Ubiquitous Comput.*, vol. 25, no. 5, pp. 879–892, Oct. 2021.

[22] Veracode, "State of Software Security Volume 9," 2018.

[23] J. Xie, H. R. Lipford, and B. Chu, "Why do programmers make security errors?," in *Proceedings - 2011 IEEE Symposium on Visual Languages and Human Centric Computing, VL/HCC 2011*, 2011, pp. 161–164.

[24] M. Graff and K. Van-Wyk R., *Secure coding : principles and practices*. O'Reilly, 2003.

[25] A. J. Ko and B. A. Myers, "A framework and methodology for studying the causes of software errors in programming systems," *J. Vis. Lang. Comput.*, vol. 16, no. 1–2, pp. 41–84, Feb. 2005.

[26] H. Assal and S. Chiasson, "'Think secure from the beginning': A survey with software developers," *Conf. Hum. Factors Comput. Syst. - Proc.*, May 2019.

[27] T. H. Morris, "Experiential learning – a systematic review and revision of Kolb's model," *Interact. Learn. Environ.*, vol. 28, no. 8, pp. 1064–1077, Nov. 2019.