Table of Contents

1. Introduction

Queens Medical Centre has proposed the use of appointment and scheduling management information system (ASMIS) software to handle appointment scheduling duties clinic-wide. There have been reports of increased doctor productivity (Forget et al, 2014), patient satisfaction (Ansell et al., 2017), and a reduction in appointment wait times (Horwitz et al., 2011) when such a system is implemented in healthcare settings. While these attributes are advantageous to overall clinic operation, outside user access presents possible network security threats.

Thus this report aims to identify potential security vulnerabilities and prevention in the ASMIS system. Firstly, the software structure will be diagrammed and explained. Secondly, relevant threat scenarios will be outlined and modeled. Finally, the strengths and weaknesses of prevention techniques will be presented and evaluated for an effective combination of prevention measures. The results are meant to provide a security plan to implement alongside ASMIS use.

2. ASMIS Structure

The ASMIS software structure consists of a central database linked to three different user interfaces: the patient interface, the physician interface, and the reception interface. A class diagram *(Figure 1)* has been included below to formulate a complete, static image of the enterprise (Huemer et al., 2015).

2.1 System Database

This system is equipped with a relational database which “arranges data in a tabular fashion” (Batra, 2018 :3) and stores two data classes: physician information and patient

information. Physician information comprises a physician's name, clinic ID, department, and a list of appointment availability times. Clinic identification numbers are automatically stored to

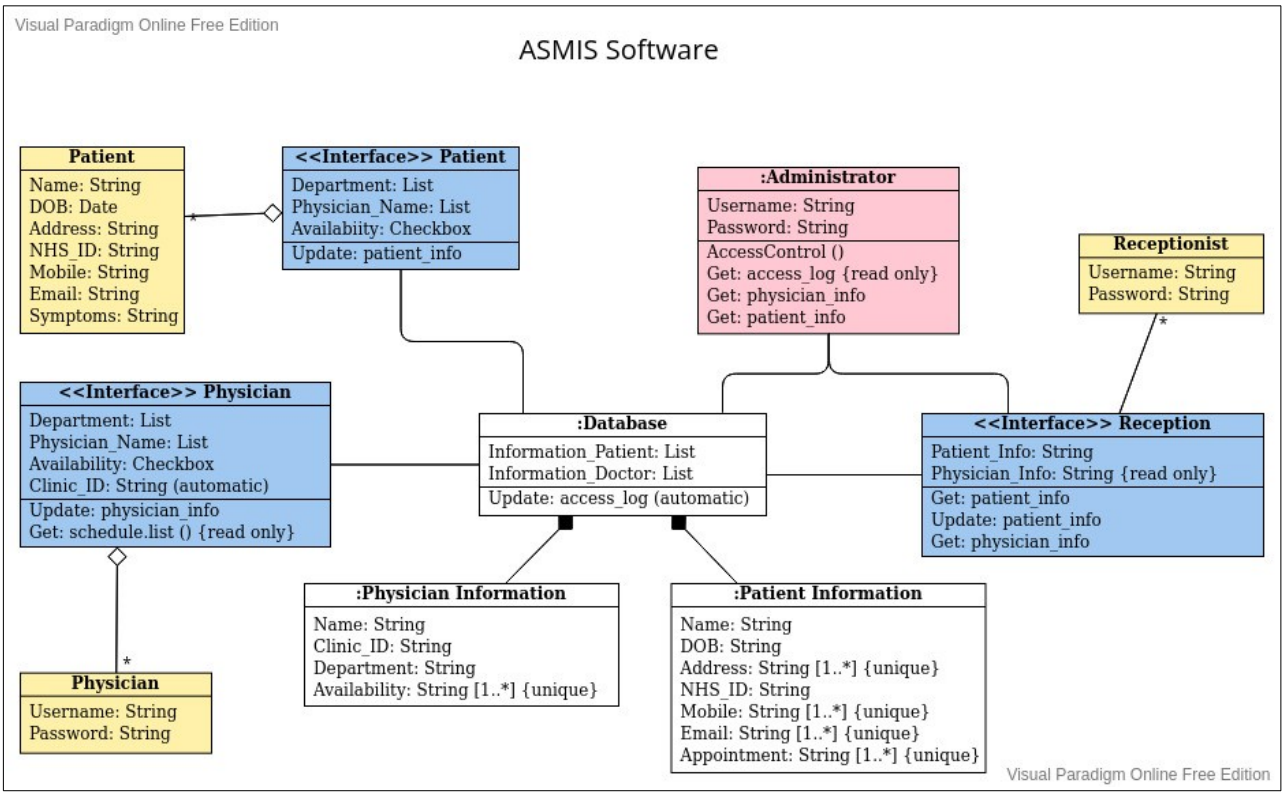

Figure 1: Class Diagram of the ASMIS

prevent redundancy in data (Huemer et al., 2015). Patient information consists of an individual name, date of birth, at least one address, a unique NHS identification number, at least one mobile and email address, and a record of all booked appointments, including sensitive information such as patient symptoms.

2.2 Patient Interface

The patient interface is a one-way interface which allows clients outside the QMC network to provide personal information to and choose appointment details from the database to secure an appointment with a QMC physician.

2.3 Physician Interface

The physician interface is a two-way interface. Physicians are required to login using a clinic username and password and each login is added to the access log, which can be viewed by administrators. Physicians can input their appointment availability and view scheduled appointment times.

2.4 Reception Interface

The reception interface is a two-way interface. Reception staff are required to login using a clinic username and password. They are subject to the same access log requirement as medical staff. Reception staff can view and edit patient information, and can view physician information. As both back-end and front-end personnel, reception staff have the added responsibility not to disclose information from this interface to unauthorized parties (Bygrave et al., 2020).

2.5 Administrator Access

The administrator class has program access through the reception interface only, with no unique interface. Administrators are subject to the same access restrictions as reception staff, which overrides administrative privileges while on the interface. Administrators are also subject to the same login and access log requirement as medical and reception staff. Access logs are not accessible from the reception interface; they are stored through a separate program.

2.6 Access Control

Role-based access control, wherein “access control is granted based on the roles and responsibilities of an individual working in the organization” (Nayak & Rao, 2014: 70) is a key factor

in the ASMIS design, limiting unnecessary usability functions to guard against catastrophic elevation of privilege in the event of a breach (Anderson, 2020). All access control decisions are influenced by GDPR guidelines to ensure the protection of sensitive patient information (Bygrave et al., 2020) by limiting unnecessary access to information in the system.

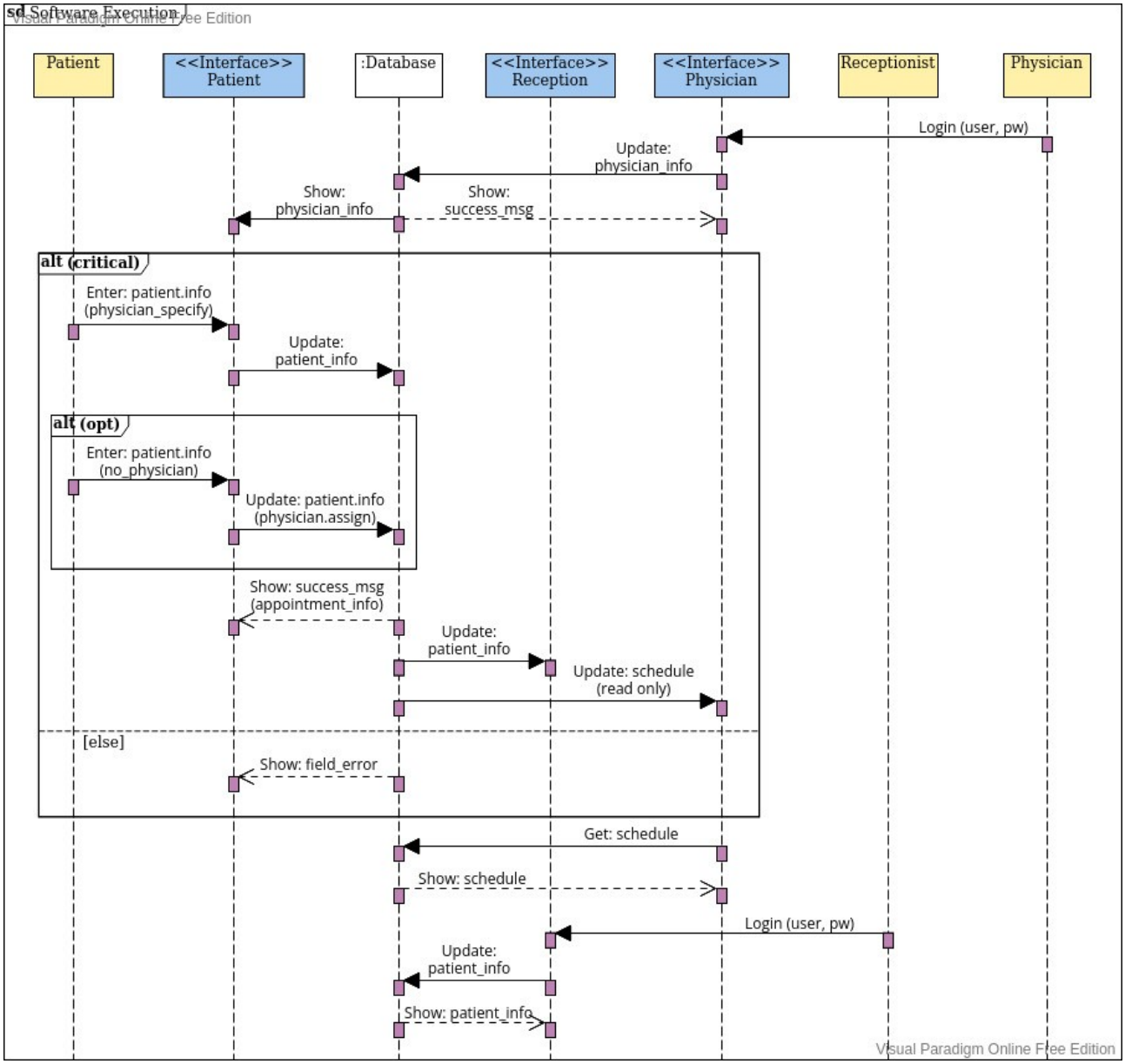

Figure 2: Sequence Diagram of the ASMIS

The sequence diagram above *(Figure 2)* provides "the interactions between [system] objects arranged in time order" (Ahmed & Sulaiman, 2018: 3) to illustrate access controls within

the ASMIS. The patient interface provides no internal access to the QMC network; it may only input patient appointment data and receive a confirmation message. The physician interface does not allow access to patients' personal information or patients' medical records. Similarly the reception interface does not allow writing access to physicians' information.

All interface functions operate as independently as possible within the program to limit unnecessary redundancy (Huemer et al., 2015) and grant certain privileges to only those with the proper credentials. Controls can be implemented by administrators through a port-based network access control list, a full review of which can be found in IEEE's *Port-Based Network Access Control* (2010).

3. Relevant System Security Threats

SQL injection, blind cross-site scripting (BXSS), and phishing attacks are consistently among the most common web application threats (Bach-Nutman, 2022; Costin, 2021; OWASP, 2022). SQL injection and BXSS in particular are highly applicable to the one-way patient interface, as they require public text fields to operate (Stuttard & Pinto, 2011). Each vulnerability is outlined below according to the STRIDE taxonomy (Table 1) (Howard & LeBlanc, 2014), as it is "an essential foundation for defining the security requirements of computer systems" (Bezerra et al., 2020: 2).

Table 1: STRIDE Threat Categories

Threat Types	Spoofing	Tampering With Data	Information Disclosure	Denial of Service	Elevation of Privilege
SQL Injection		X	X		
Blind Cross-site Scripting		X	X		X
Phishing	X	X	X	X	X

3.1 SQL injection

SQL injections target server databases specifically and can lead to information disclosure and data tampering (Askwith et al., 2018). Most SQL injection methods involve 'injecting' code into unprotected text fields to "fool the internal processing mechanism by defaulting the validation of input to be 'true'" (Nayak & Rao, 2014: 130). This activates an underlying SQL Query (Jeong et al., 2012), which then allows access to the database.

3.2 Blind Cross-site Scripting

BXSS is a type of persistent cross-site scripting attack which can lead to information disclosure, data tampering, and/or elevation of privilege (Ballmann, 2021). The attack involves injecting malicious Javascript payloads into unprotected interface text fields which are then "stored and stay in the database of the web application" (Dora & Nemoga, 2021: 321). The saved payload is designed to be activated by back-end users (e.g. reception staff), allowing malicious access to the system (KirstenS, n.d.).

3.3 Phishing Attack

Phishing is a type of spoofing attack where malicious actors “send phishing emails to gain information that can be used for a larger attack” (Bridges et al., 2021: 4). These emails appear to come from a trusted source, such as a bank or employer (Anderson 2020). The goal of phishing is often to gather login credentials (Nayak & Rao, 2014), which can then be used to gain entry onto a network. This access can result in various attack types depending on an actor’s intent (Anderson, 2020).

3.4 Threat Scenarios

There are two ASMIS threat scenarios pertinent to patient confidentiality: unauthorized access to patient personal information and unauthorized access to patient medical records. Both scenarios are diagrammed below using threat models, as such models provide clear illustrations of possible attack routes (Howard & LeBlanc, 2014).

3.4.1 Accessing Patients’ Personal Information

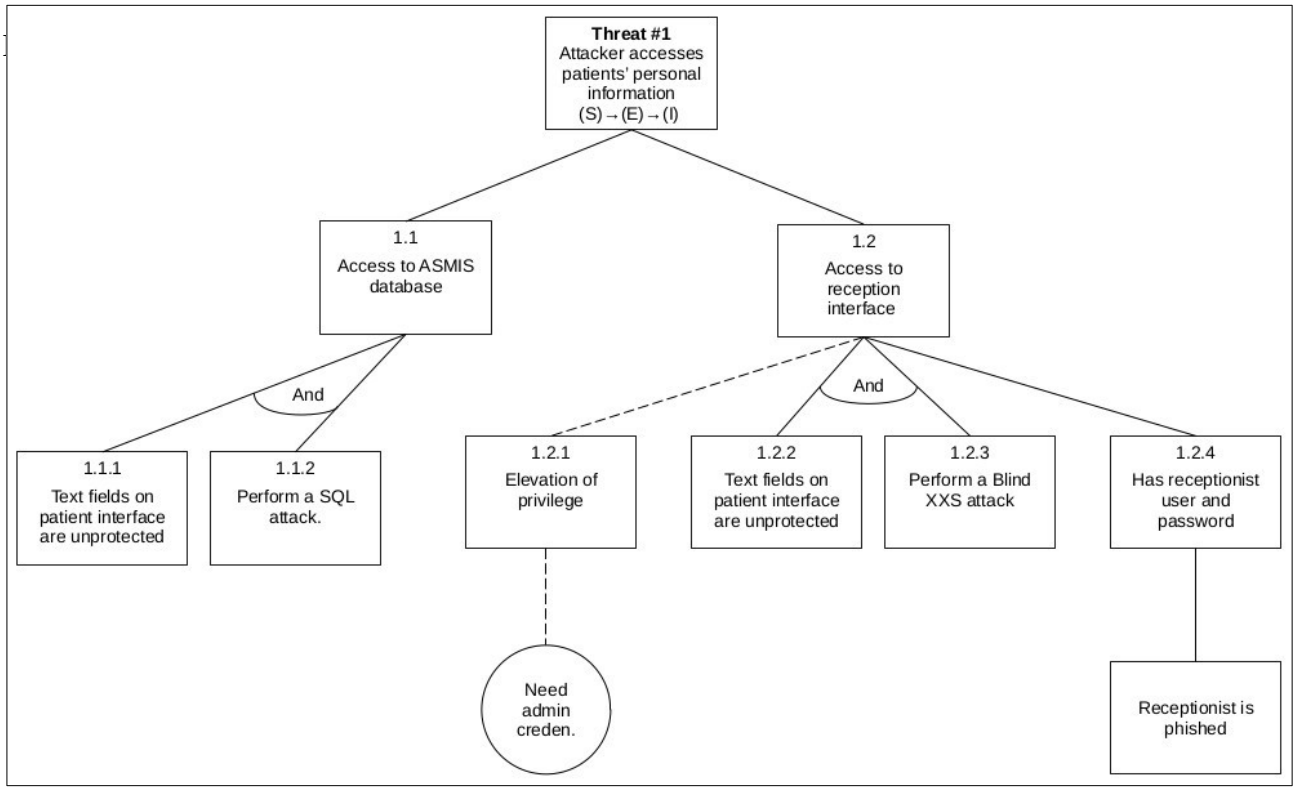

Figure 3: Unauthorized Access to Patients' Personal Information

Patients' appointment information can be accessed through the patient interface and the reception interface *(Figure 3)*. Both SQL injection and BXSS attacks can be performed on the patient interface if text fields are unprotected and thus able to accept malicious code. Phishing attacks require successful spoofing to gain the credentials of a reception staff-member, and then access to the reception interface. Success in any of these scenarios would result in access to patients' appointment information, which could be used for fraudulent activities. Though it is possible a malicious actor could gain an admin elevation of privilege and access information this way, it would require admin credentials at the outset and would thus be less likely.

3.4.2 Accessing Patients' Medical Records

Accessing patient medical records requires entry to the medical records interface *(Figure 4)*. An actor could gain access through the patient or reception interfaces, though not directly. In this scenario, an SQL injection or BXXS attack would be performed under similar circumstances to those

in section 3.4.1. This would provide either physician information from the database or a physician email list from the reception interface. Access to either type of information could lead to a

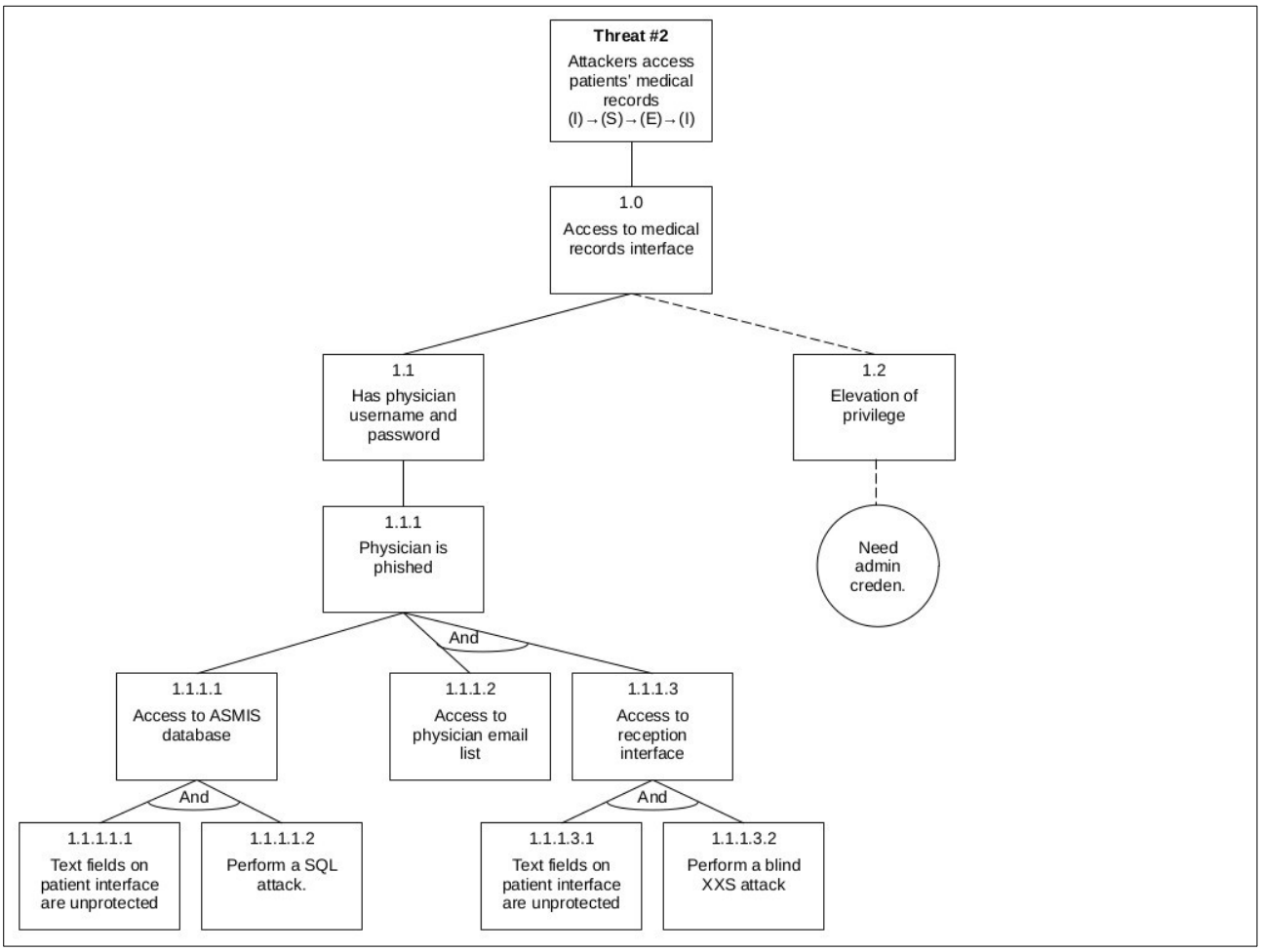

Figure 4: Unauthorized Access to Patients' Medical Records

physician's account being phished and thereafter compromised. Though this attack route involves multiple layers, it would result in a serious elevation of privilege and access to patient medical records. It should therefore be considered a true threat.

Again, though it is possible a malicious actor could gain an admin elevation of privilege and access information this way, it is reasonable to assume that with such privileges an actor would target the medical records interface directly.

4. Vulnerability Prevention Techniques

Security should be designed “into every aspect of [an] application” (Howard & LeBlanc, 2014: 37). For the ASMIS, object-oriented programming (OOP) will be employed to determine the scope of each interface, as this type of programming allows for easy abstraction and encapsulation (Phillips, 2018). Security technologies at the system and attack levels will also be implemented.

4.1 System-Level Approach

Along with the access control measures outlined in section 2.6, authentication and session management controls are crucial aspects of product security. Firstly, user authentication controls provide a means of restricting access to sensitive information. By requiring a clinic-specific username and password, only cleared personnel can access abstracted information unavailable to anonymous users (Hommel & Pöhn, 2021). Session management enforces “effective access control” (Stuttard & Pinto, 2011: 19) by identifying authenticated versus anonymous user requests based on token access. Public-key encryption should be employed in tandem as an additional level of client-server authentication (Nayak & Rao, 2014).

4.2 Attack-Specific Approach

4.2.1 Injection Attacks

User input controls should include input and boundary validation. Input validation can include ‘Reject Known Bad’, ‘Accept Known Good’, sanitation, safe data handling, and semantic checks methods to protect text and hidden-form fields from injection attacks, while boundary validation applies data validation at each abstraction level of the application to protect the server from malicious input (Stuttard & Pinto, 2011).

4.2.1.1 SQL Injection

Prepared statements with parameterized queries, stored procedures, and query whitelisting are three recommended approaches to SQL injection prevention (Bayyapu, 2021). Prepared Statements "are SQL statements that separate statement structure from statement input" (Thomas et al., 2008: 590). These structures can generate static SQL queries through fixed text-field parameters which produce a protected "data typing of user input" (Bayyapu, 2021: 208) before a query is executed, preventing vulnerabilities common to dynamic stored procedures (Kothari et al., 2006).

Parameterized queries "force the developer to first define all the SQL code and then pass in each parameter to the query later" (ibid); this technique allows the database to identify and separate malicious code from query requests. Query whitelisting is input validation which ensures that "the name of tables and columns come from code and not from user parameters" (Bayyapu, 2021: 210). It can involve one or more of the input validation methods listed above and restricts the types of data a user can enter.

4.2.1.2 Blind Cross-site Scripting

BXSS prevention should include input validation, output validation, and the elimination of "dangerous insertion points" (Stuttard & Pinto, 2011; 492). Input validation parameters are similar to the validation measures for SQL injection in that they check "the inputs to the web application against a specification of legitimate value" (Balzarotti et al., 2012: 233). 'Input condition checks' can be carried out to harden input validation; these checks can include null, size, containment match, regex-match, and type functions (Shar et al., 2013), and can limit or eliminate dangerous insertion points on an interface. Output validation can come in the form of output sanitation that

is “automated, context-aware, and robust with respect to real browsers” (Balzarotti et al, 2012: 233), and should be employed before outside output can be applied to the internal program (e.g. updating patient information).

4.2.2 Phishing Attacks

Organizations must organize and disseminate a clearly outlined network “security context in various ways to help staff avoid making mistakes” (Anderson, 2020: 101) that lead to the disclosure of restricted information. Employee education is a flagship method, wherein staff are encouraged to practice safety precautions such as not clicking a hypertext link in any received email, strengthening and changing clinic-only passwords, visiting only approved web URLs, refusing to disclose restricted information over email or telephone communications, and ignoring baiting techniques which attempt to engage victims (Anderson, 2020; Nayak & Rao, 2014).

In addition, anti-phishing software can be used in tandem with employee education to widen the breadth of prevention. Applications such as Off-the-Hook (Armano et al., 2017) or ADVERT (Balcetis et al., 2022) can actively detect user behaviors or possible phishing sites that would otherwise pose a security threat. An application’s ability to scale and its third party credentials would be factors in assessing suitability (Anderson, 2020), but such a tool could be an asset.

5. Continued Surveillance

Though the above prevention methods have been vetted and recommended by experts in the industry, this does not preclude them from vulnerability. Some input validation measures can compromise boundary validation, depending on the allowed parameters (Stuttard & Pinto, 2011).

BXSS attacks can be very difficult to detect using conventional XSS detection methods (Dora & Nemoga, 2021). Actors are also continuously finding ways to override validation measures with new combinations of malicious code (Gupta & Gupta, 2018; Stuttard & Pinto, 2011). Likewise, new and innovative approaches to phishing techniques require ever more careful vigilance and training (Anderson, 2020).

It is important to thus view software development as an iterative endeavor involving periodic “code review (tools) […] penetration testing […] [and] risk-based security testing” (Williams, 2019: 16). Attack-specific breach detection applications, such as the XXSFilt (Gupta & Gupta, 2018) or Black Box Web Application Security Testing (Alazmi et al., 2022) would be useful in tandem, as they provide an additional layer of breach detection.

The security profile of the ASMIS should not be static. Though core security measures will be implemented, constant review and improvement must be implemented in order to provide a truly secure system.

6. Conclusion

In this report the ASMIS has been outlined and diagrammed, as well as three likely vulnerabilities: SQL injection, blind cross-site scripting, and phishing. Two threat scenarios have been explored, and prevention techniques specific to each attack type along with possible weaknesses have been discussed. Finally, measures for continued security have been proposed.

References

Ahmed, M. and Sulaiman, Y. (2018) Communiqué: a planning-based sequence diagrams generator. *The Knowledge Engineering Review*, 33: 1-34.

Alazmi, S. and De Leon, D. (2022) A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners. *IEEE Access*, 10: 33200-33219.

Anderson, R. (2020) *Security Engineering: A Guide to Building Dependable Distributed Systems*. 3rd ed. Indianapolis, USA: Wiley.

Ansell, D., Bjerre, L., Crispo, J., and Simard, B. (2017) Interventions to reduce wait times for primary care appointments: a systematic review. *BMC Health Services Research*, 17(1): 295-304.

Armano, G., Asokan, N., Grondahl, T., Marchal, S., Saari, K., Singh, N. (2017) Off-the-hook: An efficient and usable client-side phishing prevention application. *IEEE Transactions on Computers*, 66(10): 1717–1733.

Askwith, B., Kifayat, McWhirter, P., and K., Shi, Q. (2018) SQL Injection Attack Classification through the Feature Extraction of SQL Query Strings Using a Gap-Weighted String Subsequence Kernel. *Journal of Information Security and Applications*, 40: 199-216.

Bach-Nutman, M. (2022) *Understanding The Top 10 OWASP Vulnerabilities*. [online] Bournemouth: Bournemouth University: 1-4. Available at: https://arxiv.org/pdf/2012.09960.pdf [Accessed 31 July 2022].

Balcetis, E., Huang, L., Jia, S., Zhu, Q. (2022) Advert: An adaptive and data-driven attention enhancement mechanism for phishing prevention. *IEEE Transactions on Information Forensics and Security*, 17: 2585–2597.

Ballmann, B. (2021) *Understand Network Hacks: Attack and Defense with Python 3*. 2nd ed. Berlin: Springer.

Balzarotti, D., Kirda, E., Robertson, W. and Scholte, T. (2012) Preventing Input Validation Vulnerabilities in Web Applications through Automated Type Analysis. In: *IEEE 36th Annual Computer Software and Applications Conference*. IEEE: 233-243.

Batra, R. (2018) *SQL Primer: An Accelerated Introduction to SQL Basics*. Berkeley, USA: Apress.

Bayyapu, N. (2021) SQL Injection Attacks and Mitigation Strategies: The Latest Comprehension. In: K. Daimi and C. Peoples, ed., *Advances in Cybersecurity Mangement*, 1st ed. Cham, CH: Springer: 199-220

Bezerra, J., César, C., de Souza, N., and Hirata, C. (2020) Extending STPA with STRIDE to identify cybersecurity loss scenarios. *Journal of Information Security and Applications*, 55: 1-13.

Bridges, C., Francia III, G. and Francia, X. (2021) Agent-Based Modeling of Entity Behavior in Cybersecurity. In: K. Daimi and C. Peoples, ed., *Advances in Cybersecurity Mangement*, 1st ed. Cham, CH: Springer: 3-18.

Bygrave, L., Dreschler L., Docksey, C., and Kuner C. (2020) *Commentary on the EU general data protection regulation (GDPR). A commentary*. Kettering: Oxford University Press: 103-115, 217-224.

Costin, A., Honkaranta, A. and Leppänen, T. (2021) Towards Practical Cybersecurity Mapping of STRIDE and CWE – a Multi-perspective Approach. In: *29th Conference of Open Innovations Association FRUCT*. [online] Fruct: 150-159. Available at: https://jyx.jyu.fi/bitstream/handle/123456789/76839/FRUCT29_Honkaranta.pdf;jsessionid=09778E527792AA58FAB458996D642470?sequence=1 [Accessed 31 July 2022].

Dora, J. and Nemoga, K. (2021) Ontology for Cross-Site-Scripting (XSS) Attack in Cybersecurity. *Journal of Cybersecurity and Privacy*, 1(2): 319-339.

Forget, P., Paré, and G., Trudel, M. (2014) Adoption, Use, and Impact of E-Booking in Private Medical Practices: Mixed-Methods Evaluation of a Two-Year Showcase Project in Canada. *JMIR Medical Informatics*, 2(2): e24.

Gupta, S. and Gupta, B. (2018) Evaluation and monitoring of XSS defensive solutions: a survey, open research issues and future directions. *Journal of Ambient Intelligence and Humanized Computing*, 10(11): 4377-4405.

Hommel, W., Pöhn, D. (2021) Proven and Modern Approaches to Identity Mangagement. In: K. Daimi and C. Peoples, ed., *Advances in Cybersecurity Mangement*, 1st ed. Cham, CH: Springer, pp.421-444.

Howard, M. and LeBlanc, D. (2014) *Writing secure code*. Redmond, Washington: Microsoft Press: 125-452.

Horwitz, L., Rose, K., and Ross (2011) Advanced Access Scheduling Outcomes. *Archives of Internal Medicine*, 171(13): 1150-1159.

Huemer, C. and Kappel, G., Scholz, M., and Seidl, M. (2015) *UML @ Classroom: An Introduction to Object-Oriented Modeling*. Cham, CH: Springer: 49 - 84.

IEEE (2010) *Port-Based Network Access Control*. IEEE Standard for Local and Metropolitan Area Networks. [online] IEEE. Available at: https://ieeexplore-ieee-org.uniessexlib.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=5409813 [Accessed 12 August 2022].

Jeong, S., Lee, I., Moon, J., and Yeo, S. (2012) A novel method for SQL injection attack detection based on removing SQL query attribute values. *Mathematical and Computer Modelling*, 55(1-2): 58-68.

KirstenS (n.d.) *Cross Site Scripting (XSS) | OWASP Foundation*. [online] Owasp.org. Available at: https://owasp.org/www-community/attacks/xss/ [Accessed 6 August 2022].

Kothari, S., Muthuprasanna, M. & Wei, K. (2006) Preventing SQL injection attacks in stored procedures. In *Australian Software Engineering Conference*. Sydney, NSW: IEEE: 198–206.

Nayak, U. and Rao, U. (2014) *The InfoSec Handbook*. Berkeley, USA: Apress.

Owasp.org (2022) *OWASP Top 10*. [online] Available at: https://owasp.org/Top10/ [Accessed 31 July 2022].

Phillips, D. (2018) *Python 3 object-oriented programming: build robust and maintainable software with object-oriented design patterns in Python 3.8.* 3rd ed. Birmingham: Packt: 7-33.

Shar, L. and Tan, H. (2013) Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns. *Information and Software Technology*, 55(10): 1767-1780.

Stuttard, D. and Pinto, M. (2011) *The Web Application Hacker's Handbook*. 2nd ed. Hoboken, N.J.: Wiley.

Thomas, S., Williams, L. and Xie, T. (2008) On automated prepared statement generation to remove SQL injection vulnerabilities. *Information and Software Technology*, 51(3): 589-598.

Williams, L. (2019) *Secure Software Lifecycle Knowledge Area*. Issue 1.0. The National Cyber Security Centre: 1-39.