University of Essex | Online

UML Flowchart: Summary Post

Course: MSc Computer Science

Module: Secure Software Development (Computer Science)

Assignment: ePortfolio

Date: Saturday 30th October 2021

Student ID: 126853

Post:

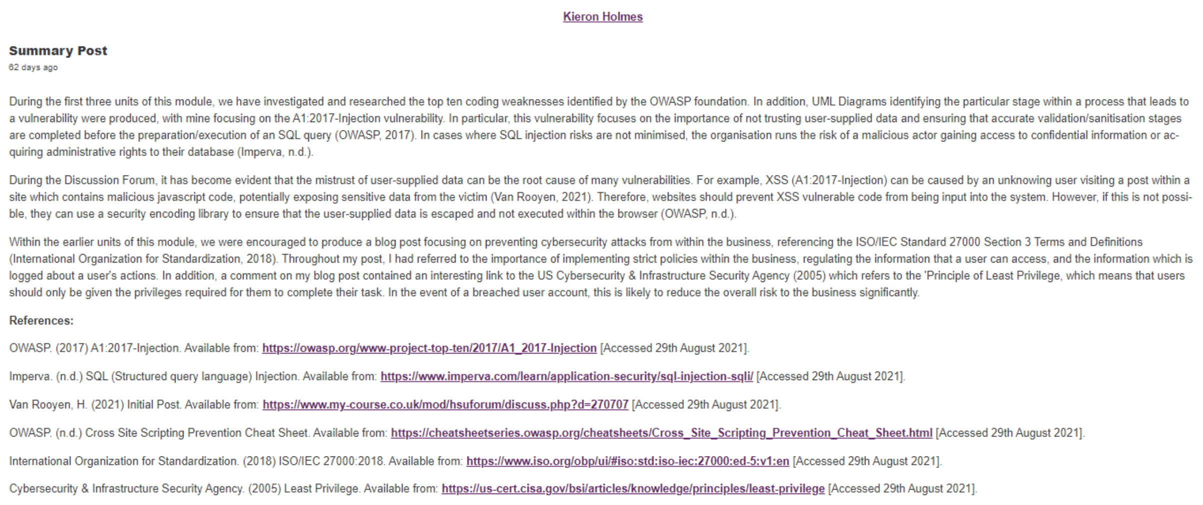
During the first three units of this module, we have investigated and researched the top ten coding weaknesses identified by the OWASP foundation. In addition, UML Diagrams identifying the particular stage within a process that leads to a vulnerability were produced, with mine focusing on the A1:2017-Injection vulnerability. In particular, this vulnerability focuses on the importance of not trusting user-supplied data and ensuring that accurate validation/sanitisation stages are completed before the preparation/execution of an SQL query (OWASP, 2017). In cases where SQL injection risks are not minimised, the organisation runs the risk of a malicious actor gaining access to confidential information or acquiring administrative rights to their database (Imperva, n.d.).

During the Discussion Forum, it has become evident that the mistrust of user-supplied data can be the root cause of many vulnerabilities. For example, XSS (A1:2017-Injection) can be caused by an unknowing user visiting a post within a site which contains malicious javascript code, potentially exposing sensitive data from the victim (Van Rooyen, 2021). Therefore, websites should prevent XSS vulnerable code from being input into the system. However, if this is not possible, they can use a security encoding library to ensure that the user-supplied data is escaped and not executed within the browser (OWASP, n.d.)

Within the earlier units of this module, we were encouraged to produce a blog post focusing on preventing cybersecurity attacks from within the business, referencing the ISO/IEC Standard 27000 Section 3 Terms and Definitions (International

Organization for Standardization, 2018). Throughout my post, I had referred to the importance of implementing strict policies within the business, regulating the information that a user can access, and the information which is logged about a user's actions. In addition, a comment on my blog post contained an interesting link to the US Cybersecurity & Infrastructure Security Agency (2005) which refers to the 'Principle of Least Privilege, which means that users should only be given the privileges required for them to complete their task. In the event of a breached user account, this is likely to reduce the overall risk to the business significantly.

Screenshot:



Kieron Holmes

Summary Post

62 days ago

During the first three units of this module, we have investigated and researched the top ten coding weaknesses identified by the OWASP foundation. In addition, UML Diagrams identifying the particular stage within a process that leads to a vulnerability were produced, with mine focusing on the A1:2017-Injection vulnerability. In particular, this vulnerability focuses on the importance of not trusting user-supplied data and ensuring that accurate validation/sanitisation stages are completed before the preparation/execution of an SQL query (OWASP, 2017). In cases where SQL injection risks are not minimised, the organisation runs the risk of a malicious actor gaining access to confidential information or acquiring administrative rights to their database (Imperva, n.d.).

During the Discussion Forum, it has become evident that the mistrust of user-supplied data can be the root cause of many vulnerabilities. For example, XSS (A1:2017-Injection) can be caused by an unknowing user visiting a post within a site which contains malicious javascript code, potentially exposing sensitive data from the victim (Van Rooyen, 2021). Therefore, websites should prevent XSS vulnerable code from being input into the system. However, if this is not possible, they can use a security encoding library to ensure that the user-supplied data is escaped and not executed within the browser (OWASP, n.d.).

Within the earlier units of this module, we were encouraged to produce a blog post focusing on preventing cybersecurity attacks from within the business, referencing the ISO/IEC Standard 27000 Section 3 Terms and Definitions (International Organization for Standardization, 2018). Throughout my post, I had referred to the importance of implementing strict policies within the business, regulating the information that a user can access, and the information which is logged about a user's actions. In addition, a comment on my blog post contained an interesting link to the US Cybersecurity & Infrastructure Security Agency (2005) which refers to the 'Principle of Least Privilege, which means that users should only be given the privileges required for them to complete their task. In the event of a breached user account, this is likely to reduce the overall risk to the business significantly.

References:

OWASP. (2017) A1:2017-Injection. Available from: **https://owasp.org/www-project-top-ten/2017/A1_2017-Injection** [Accessed 29th August 2021].

Imperva. (n.d.) SQL (Structured query language) Injection. Available from: **https://www.imperva.com/learn/application-security/sql-injection-sqli/** [Accessed 29th August 2021].

Van Rooyen, H. (2021) Initial Post. Available from: **https://www.my-course.co.uk/mod/hsuforum/discuss.php?d=270707** [Accessed 29th August 2021].

OWASP. (n.d.) Cross Site Scripting Prevention Cheat Sheet. Available from: **https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html** [Accessed 29th August 2021].

International Organization for Standardization. (2018) ISO/IEC 27000:2018. Available from: **https://www.iso.org/obp/ui/#iso:std:iso-iec:27000:ed-5:v1:en** [Accessed 29th August 2021].

Cybersecurity & Infrastructure Security Agency. (2005) Least Privilege. Available from: **https://us-cert.cisa.gov/bsi/articles/knowledge/principles/least-privilege** [Accessed 29th August 2021].

References:

Cybersecurity & Infrastructure Security Agency. (2005) Least Privilege. Available from: https://us-cert.cisa.gov/bsi/articles/knowledge/principles/least-privilege [Accessed 29th August 2021].

Imperva. (n.d.) SQL (Structured query language) Injection. Available from: https://www.imperva.com/learn/application-security/sql-injection-sqli/ [Accessed 29th August 2021].

International Organization for Standardization. (2018) ISO/IEC 27000:2018. Available from: https://www.iso.org/obp/ui/#iso:std:iso-iec:27000:ed-5:v1:en [Accessed 29th August 2021].

OWASP. (2017) A1:2017-Injection. Available from: https://owasp.org/www-project-top-ten/2017/A1_2017-Injection [Accessed 29th August 2021].

OWASP. (n.d.) Cross Site Scripting Prevention Cheat Sheet. Available from: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html [Accessed 29th August 2021].

Van Rooyen, H. (2021) Initial Post. Available from: https://www.my-course.co.uk/mod/hsuforum/discuss.php?d=270707 [Accessed 29th August 2021].