Journal of Computer Science

Review

Organizing Classification of Application Logic Attacks in Component-based E-Commerce Systems

[1]Faisal Nabi, [2]Jianming Yong, [3]Xiaohui Tao, [4]Muhammad Farhan and [5]Nauman Naseem

[1]CIS, University of Southern Queensland, Australia
[2]CIS, USQ, Australia
[3]CS, USQ, Australia
[4,5]school of IT and Engineering, MIT, Australia

Article history
Received: 06-02-2021
Revised: 12-04-2021
Accepted: 27-05-2021

Corresponding Author:
Faisal Nabi
School of Management and Enterprise, University of Southern Queensland1West St, Darling Heights QLD 4350, Australia
Email: faisal.nabi@yahoo.com

Abstract: This research paper addresses the topic of application logic attack taxonomy that is due to unclear and incorrect implementation in component-based applications. The issue addresses the detection and classification of two separate types of vulnerabilities in component-based applications. The paper completes this aim through organising the classification of each attack and then proposes the classification of logical vulnerabilities and discusses the two distinct forms of weakness and coding faults in the application software found in the mid-level of the framework. The most important argument is to desegregate awareness of attack patterns with boundary profile status relevant to an application logic vulnerability and possible threats. Having review of two different types of attack taxonomies, a logical vulnerability classification based taxonomy is proposed.

Keywords: E-Commerce, Web Software Application, CBS Design Flaws, Logical Attack, Vulnerability and Taxonomy, Software Security Flaw

Introduction

The implementation of advanced mechanisms for managing asynchronous events in web browsers and the advent of many frameworks for rapid prototyping of server-side components have been stimulated by the growth of emerging technologies and the shift from 'conditional' applications to Internet-based platforms (e.g., mail readers). Although new technologies have given significant funding, development, productivity and interoperability advantages, little has been done to fix security concerns. As a consequence, the web applications become more complex, the risk of abuse is increasing (Firesmith, 2005). The risk of violence also increases. An overview of the CVE vulnerability database, for example, reveals that web-based attacks rose from 25% in 2017 to 61% in 2018. The fact that component-based applications are typically accessible through designer firewalls makes it possible for developers with insufficient software protection to build server-side logic more widely under time-to-market pressure. As a result, web applications that are unsafe created and made available over the Internet, making it simple to exploit (Nabi and Nabi, 2017).

The use of best practises in industrial fields such as firewalls, encryption (SSL/TSL), vulnerability scan, security monitoring, etc. (e.g., intrusion, white box and black box) has historically been promoted by security engineering in existing systems to insure proper security. Many security papers and books are unable to provide much detail on the e-commerce framework's security specifications and most of what is written seems to stress the concept of ambiguous security objectives or concentrate on architectural constraints. Usually is either the amount required of a stated particular type of security or the safety implications of non-security Normally, either the amount appropriate to a given security form or the safety effects of non-security specifications are addressed in security processes. Cyber attacks are essential to any component-based security assessment of e-commerce application. In this context, the characteriszation and classification of vulnerabilities is one of the most important fields of study. Several models suggest defining them; such models usually generally describe attacks (Nabi and Nabi, 2017) In addition, experience shows that attack profiles are highly dependent on multiple frontier conditions. This study addresses the problem of the absence of coherent vulnerabilities and

taxonomies to identify and classify two distinct vulnerability classes in the CSB's web-based e-commerce.

This is achieved by organizing the critical classifications that suggest the classification of logical vulnerabilities centred on design faults versus technological faults focused on web application deficiencies and defects at the implementation level from a security evaluation perspective of component-based software applications. Our research methodology relies on grouping that separates or orders the component-based software applications Classifications can be established as either a priority (i.e., non-empirical from an abstract model) or Posteriori Empirical by evaluating the CVE vulnerability database for security breach cases.

Research Background

A taxonomy of recurrent vulnerabilities may contribute to the organisation of today's safety-enhancing knowledge. To detect possible attacks on web application software before it is published to consumers; advanced awareness of vulnerabilities can be useful. We reviewed 25 taxonomies from 1974 to 2017 and analysed different levels of vulnerabilities, property taxonomies, web application vulnerabilities, network vulnerability taxonomy and software vulnerability taxonomy of e-commerce threat classifications before restricting the main scope of this study to address the logical problems of the web software application due to mismatch between design and architecture. However, it depends on web software application during development. Our attack patterns are more detailed to which components could recognise a device design vulnerability.

Most taxonomies have four hierarchical groups within the taxonomy: Structural flaws, environmental deficiencies and codes. We contrasted our taxonomy with the environmental defect class, which is intended to infringe the environmental standards of programmers and their software weakness.

Since most (Nabi and Nabi, 2017) researchers did not find any information on the design vulnerabilities in real-time, they could not provide any information on this vulnerability and its attack classifications.

Research Methodology

Our main objective is to develop the taxonomy of logical weakness in the application layer of distributed multiple-tier e-commerce systems, as stated in the introduction. There are several methodologies to assess the security of information communication technical infrastructure that are developed in various papers and texts, which provide a launchpad into an e-commerce system. We have selected Masera and Nai methodology 2005 as a guide to support our methodology. The authors present in Masera *et al*. (2005) a risk management method for the assessment of complex ICT systems. This approach accepts the fact that a description of the function, components, properties and the relationship between components, assets and the outside world should be first given for the safety evaluation of a system. This can be used to identify defects that influence the system as a whole systematically.

Our research methodology is also focused on the Posteriori Empirical study of CVE vulnerability database data from various levels of e-commerce categories of web-based applications and systems (B2B) and (B2c) from 2002 to 2017. Specific groups of single characteristics are used with a set of taxonomic characters that meet the classification needs of subjective decisions. These classifications are simplest and require a clear selection criterion for individuals to be grouped. For instance, group programmes use encryption or not in their language of programming. The evaluation of potential damage to the components, their propagation to the sys-tem and subsequent attack patterns can be extracted from the evaluation of this information.

As described above, web applications and systems for e-commerce and those elements that form the basis of our methodology are strongly linked to a set of traditional computer security principles, particularly the "five pillars." We also developed a Security Vulnerability Evaluation Model focused on "Five Pillar" Computer Security Elements for component-based e-Commerce software applications and systems. This enables vulnerability to be identified and attacks to patterns that lead to our main goal of classifying logical vulnerabilities (Moore *et al*., 2001).

In the other hand, technological flaws are due to mistake, fault and bug coding at implementation level for a software development framework. During such a process, they can be patched. Furthermore, the use of vulnerability analysis software and web application scanning tools is difficult to repair or identify faults in design. Therefore, no taxonomy provides details on the logical danger of the application layer targeting attacks and patterns related to vulnerabilities and attacks in the mid-level business application logic (the n-tier e-commerce system).

In component Web Applications and Systems, we propose the SVAM for the main computer protection attributes 'Five Columns,' as mentioned, showing the life cycle of the vulnerability and classifying the key point where the vulnerability covers two or more delicate vulnerability classes, such as 'Technical and Logical., as defined in Fig. 1.

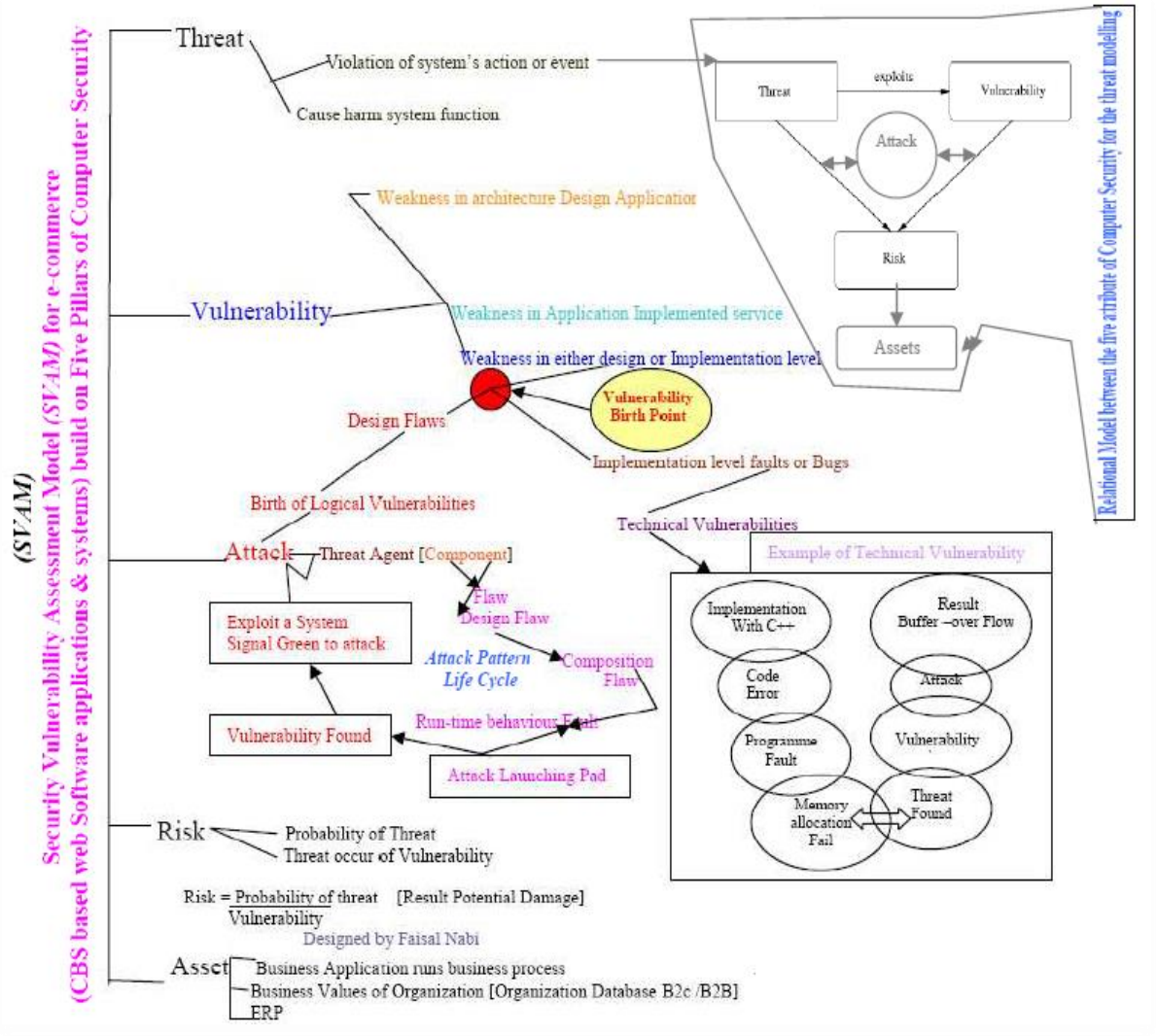

Fig. 1: SVAM model

Related Research Work and Taxonomic Properties

The theoretical analyses are categorised into taxonomy (Simpson, 1945; Moore *et al*., 2001; Masera *et al*., 2005), including their base, principles and procedures and standards. The grouping and/or arrangement of objects (or specimens) into groups is a classification. Non-empirically generated classifications are known as priori classifications. Empirically generated classifications are called subsequent classifications by analysing the data.

Objects, Attributes and Constraints of a System

Object: An object is an "entity" that provides or receives information and possesses a unique name and a collection of operations on it (Longley and Shain, 1990).

Attribute of Object: An object attribute is an object's data component and a derived attribute from another attribute is a later attribute's data component.

Property of Attribute: The attribute property is a property of the attribute, which can be obtained from the attribute by applying a function to the attribute.

Attribute refinement: An attribute refinment is a final refining of attributes wherein larger attributes that contributes to the identification of attributes with assumptions. The refinement attribute can-not contain an attribute element. The refinement attribute can't contain an attribute property.

Attribute Constraint: The Constraint attribute defines the ownership or collection of assumptions regarding this particular attribute.

Table 1 defines attack pattern properties.

Table 1: Attack pattern properties

Pattern name and classification	A unique, descriptive identifier for the pattern
Attack prerequisites	What conditions must exist or what functionality and what characteristics must the target Software has, or what behaviour must it exhibit, for this attack to succeed?
Description	A summary of the assault including the course of action
Related vulnerabilities or weaknesses	What specific vulnerabilities or weaknesses.
Method of attack	Which sort of attack vector utilized (e.g., malicious data entry, maliciously crafted file, Protocol corruption)?

Taxonomic Characters, Object Attributes or Features

The basis for determining a positive classification is the taxonomic character (Simpson, 1961; Glass and Vessey, 1995). These are the characteristics or attributes of the objects. These characters are sometimes referred to as characteristics, attributes or features (Simpson, 1961). Asserts the readiness and objectivity of these properties from the relevant objects.

Concept of Attack Pattern

An assault pattern is the abstraction mechanism to describe how an assault is carried out. It also describes the context in accordance with the pattern model where appropriate and then proposes, proposed ways to mitigate the attack rather than conventional patterns. In other words, a pattern of attack is an inference. In a pattern of attack, the following information is typically given.

With regard to the above-mentioned theory and concepts, discussion and references are based on principles, procedures and rules concerning the taxonomic classification of system objects, attributes, properties and characteristics. We want to first describe clearly the vulnerability of web software applications before moving towards a taxonomic contribution focused on classification and characteriszation of two separate vulnerability categories (Technical vs Logical).

Web Software Application Vulnerability

"The weakness of the Web application software includes misalignment between the application logic and environmental assumptions taken up in development/execution (code written) and the environment within which it is run," we define vulnerabilities in Web Application software (Nabi, 2011).

Taxonomy of Computer Program Security Flaws

A flaw can be defined as malicious or not.

Malicious Flaws

Implemented to cause a breach of the protection deliberately, such as viruses, worms, Trojan-based horses, time bombs and coded trap doors (Landwher *et al*., 1993).

Non-malicious Flaws: Incorporated due to missing specifications or design logic mistake.

During the software life cycle, programmes are graded by the time they are incorporated into the programme. Defaults during development, repair or service are part of the implementation time.

Flaws are concerns that arise in software design. A vulnerability may be a flaw in the software runtime environment. In general, mitigating a defect requires much more work than just a few lines of code. The concern is not just about implementation; the idea behind it is flawed and that is why it is not implemented For example, a design flaw that does not mitigate a simple action such as changes in array boundary (Nabi, 2005) is a sensitive business logic for an untrusted customer application (Nabi, 2005; 2011).

A Taxonomy of Security Faults

Many classification schemes for security faults have been suggested that categorise faults by different criteria as shown in Fig. 2 (Krsul, 1998; Aslam, 1995):

- Coding faults are composed of faults in the software development process that are introduced during software development. These faults are the cause of errors in programming logic and missing or incorrect requirements
- Operational faults Operational faults are called incorrect software deployment. In most situations, failures can be categorized as operational faults (Aslam, 1995)
- Environment faults occur when a programmer does not completely understand the limitations of the usable right modules or the interactions between them (Krsul, 1998)

A Taxonomy of Security Error, Faults and Failures

Error: An error is a developer mistake. It could be a typographical error, misinterpreting a specification, misu nderstanding, etc. (ANSI/IEEE, 1990).

"An error can be the cause of one or more faults"

Fault: Defects can be found in the software code. In particular, the discrepancy between incorrect programming and the correct version (ANSI/IEEE, 1990).

Failures: Faulty code execution can lead to null or more failures when the failure is the [non-empty] difference between the incorrect and correct programme results (ANSI/IEEE, 1990).

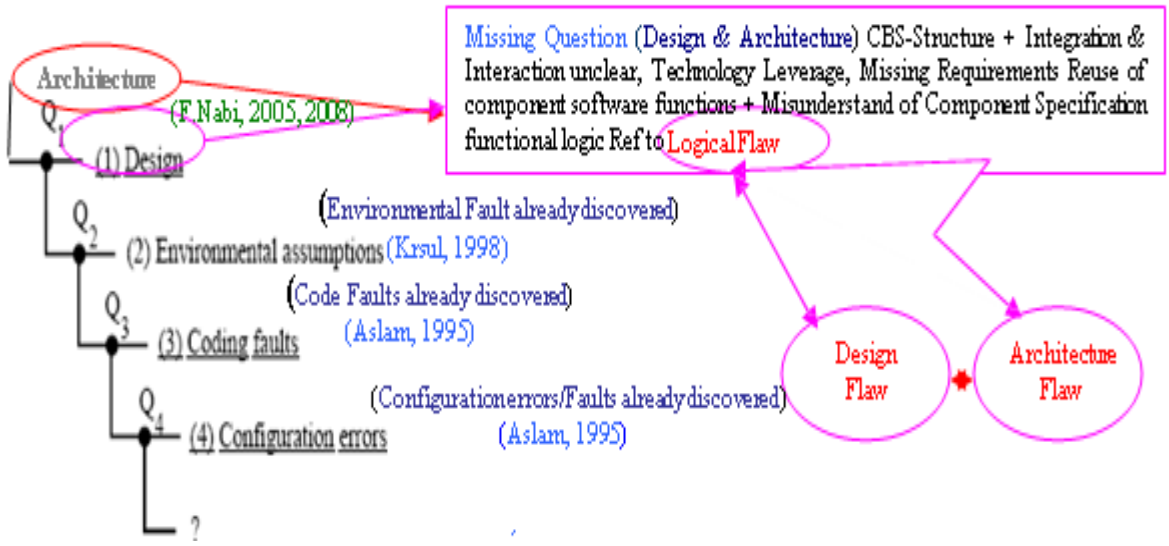

Fig. 2: Taxonomy of Software Vulnerabilities causes

Previous Research Work and Classifications

A detailed understanding of vulnerabilities can help to detect possible attacks on a software programme before they are published to customers. A taxonomy of recurring vulnerabilities can help navigate the details required to increase safety awareness. Between 1974 and 2018, we analysed 21 taxonomies and assessed various levels of vulnerability, classified property taxonomies of e-commerce risks, web application vulnerabilities, network vulnerability taxonomy and software vulnerability taxonomy before restricting the key scope of the analysis to logical attack problems. This is due to a flaw between design and architecture when designing an application with web software.

Taxonomic Classification and Review based Comparison

McPhee (1974) proposed the classification of vulnerability that falls under the category of Design flaw vulnerability, the object of the vulnerability is targeting operating system flaws.

Abbott *et al*. (1976) focus on Layered operation and features that also consider the reason is based on operating system flaws. So this taxonomy is operating system-oriented.

Bisbey and Hollingsworth (1978) Taxonomy is also single dimension targeting operating system based abstract pattern from flaw and automated search flaw. This taxonomy is also operating system-oriented.

Aslam (1995) explained the UNIX security flaw that targets the database vulnerability organization. Overall it is operating system-oriented vulnerability.

Landwher *et al*. (1993) explain the taxonomy of Operating System Flaws categorized vulnerability based on Genesis, Time of introduction and location.

Bishop (1995) explained the UNIX System and Network Vulnerabilities that focus on Effect, Minimum number of components, Source of ID.

Gray (2003) explained the layer-based vulnerability in network operational system.

Jiwnani and Zelkowitz (2004) explained the software flaws in the software development process. This taxonomy is three dimensional.

Pothamsetty and Akyol (2004) explained the Layered based vulnerability targeting the network operational protocol vulnerability.

Tsipenyuk (2005) multi-dimensional coding error-based vulnerability that causes software errors.

Weber *et al*. (2005) focused on also a layer-based software flaw that generates coding analysis and tool-based detection.

Kjaerland (2006) four-dimensional taxonomy explaining the Method of operation and impact of intrusion and its detection.

Bazaz and Arthur (2007) explained the Hierarchical vulnerability taxonomy targeting computer sources and its relation to vulnerability.

Igure and Williams (2008) explained the vulnerability class multi-dimensional attack on computer system resources and process of vulnerability.

Simmons *et al*. (2009) explains five-dimensional network taxonomy focusses on the attack vector, operational process and defense.

Cebula and Young (2010) Hierarchical taxonomy explaining the cyber-attacks and its process to generate vulnerability that cause attacks in the system.

Scott and Angelos (2013) this Hierarchical Network Taxonomy explains the Explore the relationship between events.

Joshi and Singh (2014) five-dimensional taxonomy focusing on attack entity, defence method and target, impact, which explains the nature of the attack.

Joshi *et al*. (2015) review the existing taxonomies related to computer attacks and vulnerability in the system. This mostly, targets the network-based vulnerability detection method overview.

Li *et al*. (2017) represented the software-based vulnerabilities and propose the model to mitigate the software vulnerability issues.

Chen *et al*. (2018) explained the Taxonomy of Internet-of-Things Security and Vulnerabilities that address that internet of things security wholes and related vulnerabilities in the system and applications.

Overall Review and Comparison

There is a number of vulnerabilities and attacks noted previous taxonomies which most do not concentrate on logical software vulnerabilities. This difference clearly identifies the needs for a systematic model and classification of these groups into class vulnerability against technological vulnerability. Therefore, through the vulnerability life cycle in background software process model, we introduced a new taxonomy and its implementation life cycle. This model demonstrates clearly the birth and life cycle of vulnerability.

Classification of Security Threats in e-Commerce

Generally, structural analysis allows a phenomenon to be classified. In particular, a formal e-commerce threat classification would allow managers to develop less fragile system (Álvarez and Petrović, 2003). The following classification properties are recommended for reporting accidents to incident response teams.

- The categories should be mutually exclusive (maximum one for each category) and collectively complete (each specimen should be at least one category). The various categories should be mutually exclusive (one category should be the most suitable for all specimens) and uniformly exhaustive (all specimens should fit in at least one category). In addition, the types should be mutually exclusive
- In each category should be included specific and clear criteria for the specimens to be included in the category
- Not only security experts but also less qualified and seasoned users and administrators can benefit from intuitive and useful taxonomy
- The terminology of taxonomy should comply with existing safety terminology (which can not always be defined easily)

Classification of Web Taxonomy

Chirs and Frank (2005): Addressed a methodology for vulnerability taxonomization and an example of web services, WS architectural model of four components and their connections. It addresses two subclasses. 'Input Format and Input Origin' then contains attack flows based on a category of border state error, which is exceeding an unforeseenly long input that executes arbitrary code from an attacker (programme written in C or C++). (Format and Input Origin). The authorship is the proposed Result Matrix, which is the same that (Aslam, 1995; Krsul, 1998) classifications and almost a copy thereof.

Álvarez and Petrović (2003): Entered the web attack taxonomy. Specific web categories are entry point, aim, HTTP verbs and HTTP headers, which are not covered by general taxonomies and are considered important for the precise classification of Web attacks. However, other types, such as vulnerability to site-specified values (e.g., code injection, HTML handling, etc.), will usually face taxonomies, canonicalization, overload and misspellings. Alvares differentiated & ordered the taxonomy from the point of view of the attacker. The author clarified that because of two vulnerability errors, an attacker might get access to a point that should be a web server or web application entry point looking for an attack.

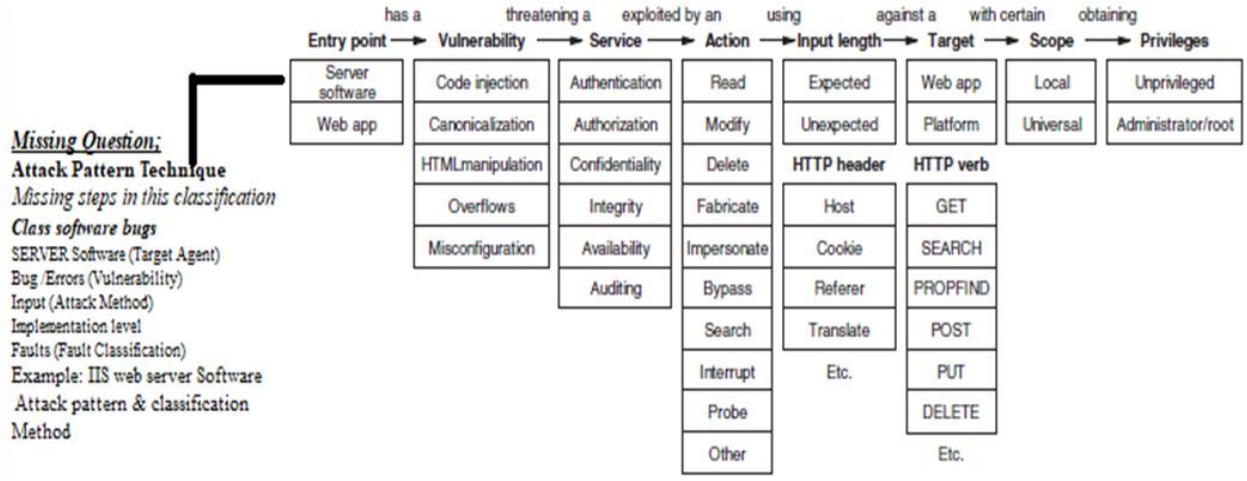

Fig. 3: Web attacks taxonomy (Álvarez and Petrović, 2003)

It reflects the widespread life cycle of a hacker attack based on the HTTP as shown on the Fig. 3. It is also incomplete taxonomy and cannot be called a classification scheme. Since attack patterns cannot be categorised, (Krsul, 1998; Aslam, 1996), classified the classification of vulnerabilities and their characteriszation by their attributes.

Proposed Classification and Types of Logic Attacks

There are different types of logical attacks every time and a particular application function/method must be used for taxonomy. The logical attacks are designed to interrupt the application's logical flow. The logic of implementation is the logical flow that a certain procedure is supposed to be carried out. The software logic contains examples of password recovery, account registration, auction requests and transactions for e-commerce. A website may provide a consumer with a multi-stage process to carry out a certain action properly. An attacker can bypass or use these features to cause website or users damage. As previously stated, the study focuses on the problem of "application logic-based vulnerabilities" as design and architecture differ during development of web applications. In the application logic, we find seven faults/flaws as illustrated in Fig. 4 and then a case that endorse Taxonomy as a source of reference faults for design faults.

In each type of attack, the attack pattern and target agent define the proposed taxonomy contribution. As above, graphical attack pattern methods and vulnerability classes based on application logic are logical presentation as defined in Fig 5. This is further used to categorise each vulnerability because of an attack process, characterised in-group of attacking parameters that determine the essence of the vulnerability.

Case as a Reference: Mars Polar Landing Mission (NASA) Dec 3, 1999

The case for component-based systems and their implementations is discussed here as a reference. The case describes one of the classifications identified above of system composition failures or defects while NASA, USA, takes the component-based approach for mission-critical system development.

Reason of Project Failure

Touchdown Monitor (TDM) component failed to co mply with the requirements contrasted with its functional specification based on the specification integration via c ontract interface, which led to an MPL device design def ault and task failure.

Requirement Modeled of TDM

TDM component is an MPL system software which monitors three landing legs during two downward stages.

Logical Component Information Processing

The Multi-Task Monitoring Calls TDM module receives information from the second module on the leg sensors at 100 times per second. TDM software tracks the three touchdown legs during the first process, which begins at 5 KM above Mars' Surface.

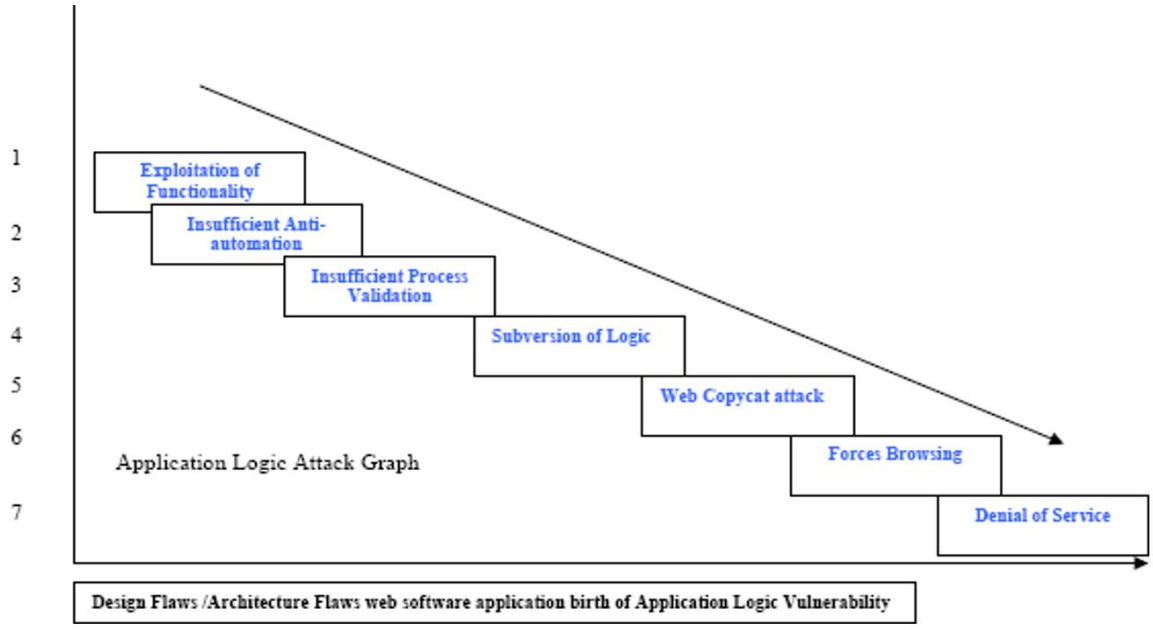

Fig. 4: Application Logic Vulnerbility Graph

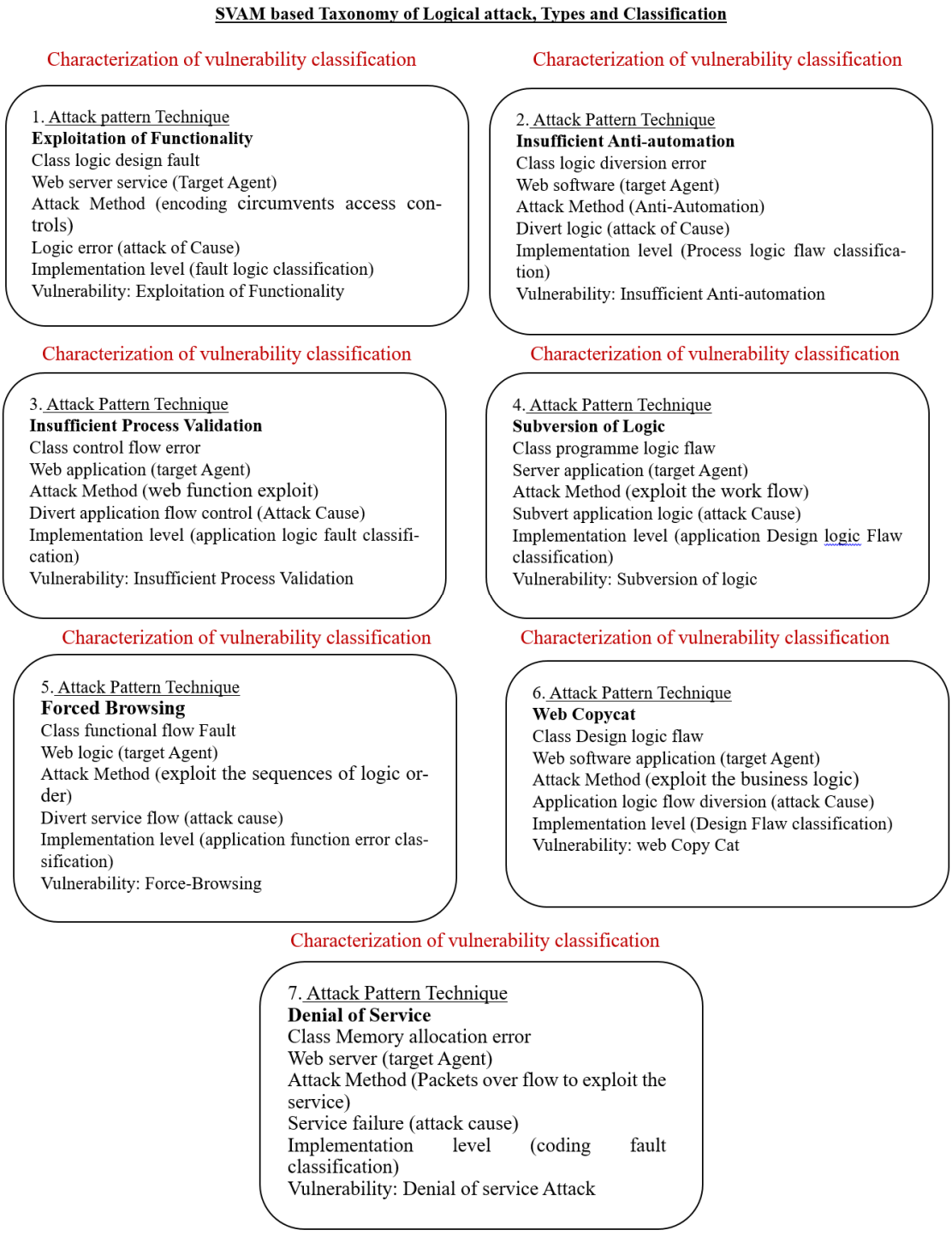

Fig. 5: Characterization of vulnerability

Application Logic of Component

Start reading at First Stage at about 5 km above the surface of Mars, TDM tracks the touchdown legs, one sensor per leg to assess touchdown.

Processing Logic Design

Developer assumed that a known possibility sensor could indicate wrong touchdown signals if-the legs locked in the deployed position. TDM software had to handle this

possible event with a-marking leg that generates a spurious signal with an inappropriate sensor on 2 consecutive sensor readings.

Second Stage

TDM was to track the remainder of the good sensor at around 40 m above the surface. When a sensor had two consecutive Touchdown reading, the TDM programme was instructed to shut down the downwind engine.

What Happened?

One or more of the sensors had 2 consecutive readings in TDM Component Memory before 40 m, leg-sensor information was processed. When MPL crossed the 40 m level, during the first step of descent, TDM changed states and read the storage associated with the leg sensor. Shutdown Engine effect.

Scientific Justification

A developer can design and enforce the requirement in various ways, but the nature of a design failure is that components cause (pre-conditioning, post-condition and invariant) infringements in performing the condition of bad data held by software variables (Chen *et al*., 2018).

Therefore, it has been shown that the problem is not in implementation logic but in design through the application logic technique related to the logical component and its requirement specification rather than a more functional interface specification integration, which resulted in a design defect in the MPL framework and task. This defect's classification is therefore defined as a design defect, which is a logical defect identified by our vulnerability classification through SVAM (Fig. 1).

Logical vs Technical Vulnerability Classification

In view of our study, we would like to suggest a classification and characteriszation of the two categories of vulnerability problems/issues mentioned above (Technical Vs Logical Vulnerabilities). These are categorised as stated above in the classification of each weakness on the basis of their attack process (attack pattern technique). Therefore, by retaining the classification of two separate vulnerability types, we have drawn up a classification tree where all sub-class attacks under each vulnerability class are included. A new taxonomy is shown here with a detailed classification and distinguished by its distinctive signature in the application layer of e-commerce systems. As it is stated in Fig. 6.

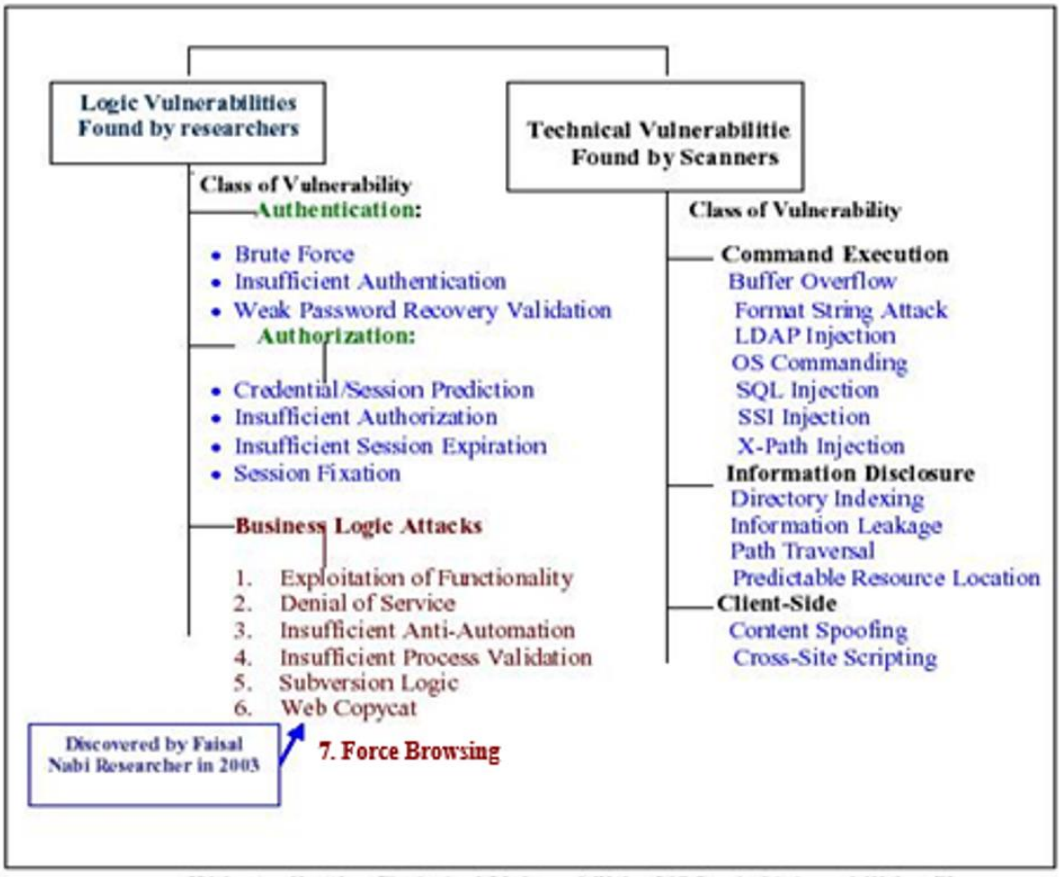

Fig. 6: Logical vulnerabilities Vs technical vulnerabilities

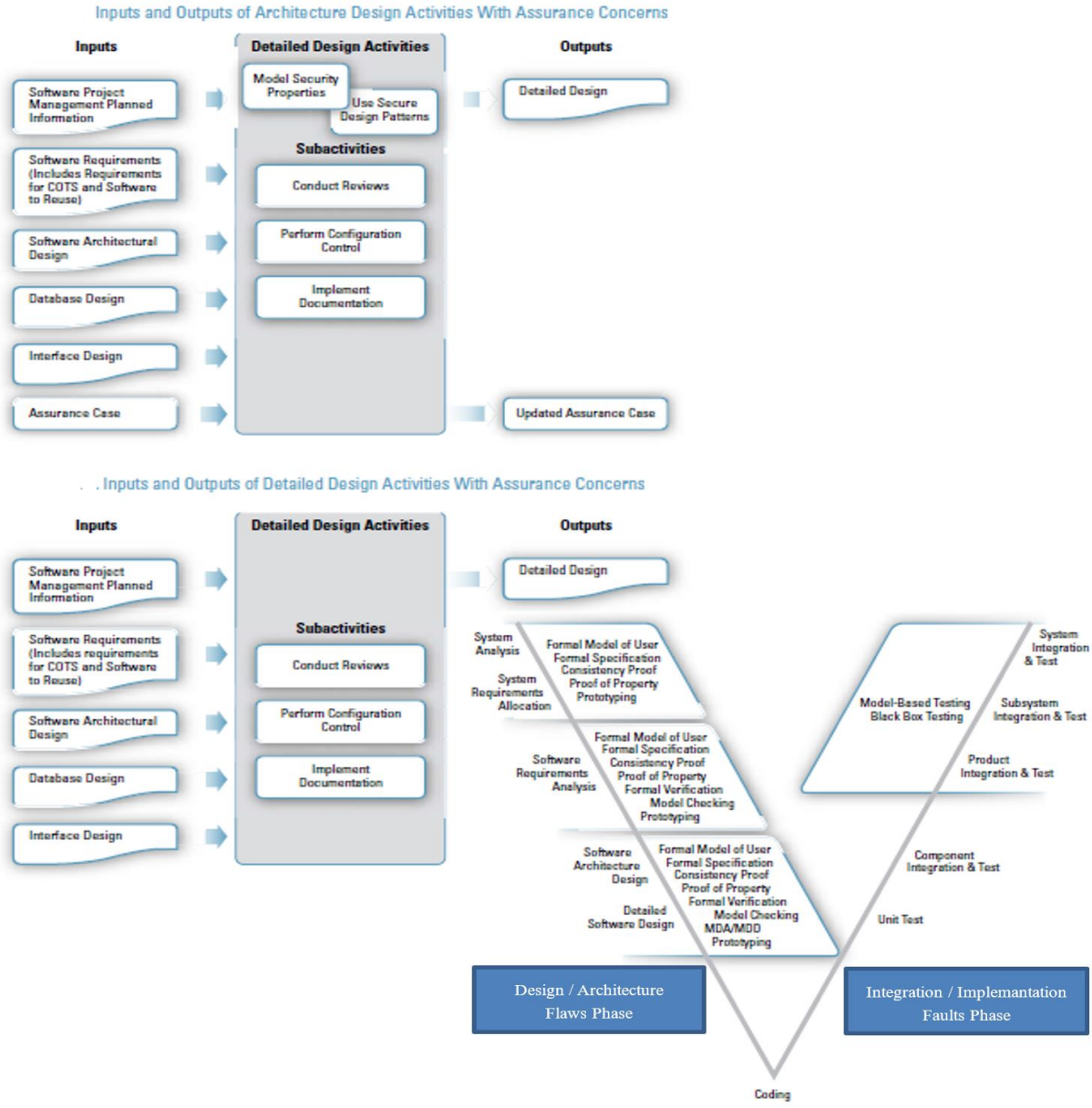

Fig. 7: Vulnerability mitigating in context software design assurance phase process

Mitigation Process in Context SDLC

Attack patterns identify typical methods of software operation. They are derived from a model proposed by the design pattern framework (Li *et al*., 2017) that clearly shows the stages of two distinct life cycles of vulnerability, as shown in Fig. 7. The concept derived from (Joshi *et al*., 2015) that one describes design and architecture, another one shows implementation level, each stage shows two separate causes of vulnerability, such as the design phase refers to a design flaw and architectural flaw and flaws, bugs & errors are seen in the implementation phase. By mismatching a collection of components in a system design that allows the sequence of events occurring in the attack pattern, allows the vulnerability detecting approach is achieved. The proposed model also presents extensive information on all protected system development processes at the design and implementation levels and describes both the two distinct types of vulnerabilities. This helps to understand two distinct life cycles of vulnerability and therefore points out the closeness as stated in the Fig. 7.

Conclusion

For software developers a taxonomy is the footprint for safe system design (Johnson *et al*., 1995). The approach taken in this article focuses in the characteriszation and classification of vulnerabilities of

component-based web-e-commerce applications and of logical vulnerabilities. As a result, safety awareness is increased at the outset of the development process by incorporating the proposed approach and procedure into the design phase. Risk management is required to begin early on so that the protection team can evaluate how the application logic has been strengthened. In the component development software model, we also categorised the two separate vulnerabilities and showed the birth of attack designs because of vulnerability at the various phases of the development cycle, which are helpful for developers in the adoption of protection through design technologies during software design.

Acknowledgment

This work is based on a Research in Australia cyber Banking e-commerce security Busniess logic issues.

Author's Contributions

All authors equally contributed in this work.

Declaration of Interest

- The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this study
- The authors declare the following financial interests/personal relationships, which may be considered as potential competing interests

Ethics Approval

There is no human and animal involved in this research therefore no need of ethical approval for this research

References

Abbott, R. P., Chin, J. S., Donnelley, J. E., Konigsford, W. L., Tokubo, S., & Webb, D. A. (1976). Security analysis and enhancements of computer operating systems. National Bureau of Standards Washingtondc inst for Computer Sciences and Technology. https://apps.dtic.mil/sti/citations/ADA436876

Álvarez, G., & Petrović, S. (2003, July). A taxonomy of web attacks. In International Conference on Web Engineering (pp. 295-298). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45068-8_56

ANSI/IEEE. (1990). ANSI/IEEE Standard Glossary of Software Engineering Terminology. IEEE Press. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=159342

Aslam, T. (1995). A taxonomy of security faults in the Unix operating system. Master's thesis, Purdue University. http://cwe.mitre.org/documents/sources/ATaxonomyofSecurityFaultsintheUNIXOperatingSystem[Aslam95].pdf

Bazaz, A., & Arthur, J. D. (2007, January). Towards a taxonomy of vulnerabilities. In 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07) (pp. 163a-163a). IEEE. doi.org/10.1109/HICSS.2007.566

Bisbey, R., & Hollingsworth, D. (1978). Protection analysis project final report. ISI/RR-78-13, DTIC AD A, 56816. http://nob.cs.ucdavis.edu/bishop/papers/1999-raid/1999-vulclass/1999-vulclass.html

Bishop, M. (1995). A taxonomy of UNIX system and network vulnerabilities. Technical Report CSE-95-10, Purdue University. http://nob.cs.ucdavis.edu/bishop/notes/

Cebula, J. L., & Young, L. R. (2010). A taxonomy of operational cyber security risks. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst. http://www.sei.cmu.edu/library/abstracts/reports/10tn028.cfm

Chen, K., Zhang, S., Li, Z., Zhang, Y., Deng, Q., Ray, S., & Jin, Y. (2018). Internet-of-Things security and vulnerabilities: Taxonomy, challenges and practice. Journal of Hardware and Systems Security, 2(2), 97-110. doi.org/10.1007/s41635-017-0029-7

Chirs, V. B., & Frank, R. J. (2005). A taxonomy methodology applied to web services. Research Report, IBM Zurich Research Laboratory. https://dominoweb.draco.res.ibm.com/f3f9573a5c7b2db4852570750034edf2.html

Firesmith, D. G. (2005, August). A taxonomy of security-related requirements. In International Workshop on High Assurance Systems (RHAS'05) (pp. 29-30). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.6934&rep=rep1&type=pdf

Glass, R. L., & Vessey, I. (1995). Contemporary application-domain taxonomies. IEEE Software, 12(4), 63-76. https://doi.org/10.1109/52.391837

Gray, A. (2003). An historical perspective of software vulnerability management. Information Security Technical Report, 8(4), 34-44. doi.org/10.1016/S1363-4127(03)00005-0

Igure, V. M., & Williams, R. D. (2008). Taxonomies of attacks and vulnerabilities in computer systems. IEEE Communications Surveys & Tutorials, 10(1), 6-19. doi.org/10.1109/COMST.2008.4483667

Jiwnani, K., & Zelkowitz, M. (2004). Susceptibility matrix: A new aid to software auditing. IEEE Security & Privacy, 2(2), 16-21.

doi.org/10.1109/MSECP.2004.1281240

Johnson, R., Gamma, E., Vlissides, J., & Helm, R. (1995). Design pattern: Reusable object-oriented software. Addition Wesley.

Joshi, C., & Singh, U. K. (2014). Admit-A five dimensional approach towards standardization of network and computer attack taxonomies. International Journal of Computer Applications, 100(5), 30-36. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.678.3355&rep=rep1&type=pdf

Joshi, C., Singh, U. K., & Tarey, K. (2015). A review on taxonomies of attacks and vulnerability in computer and network system. International Journal, 5(1), 742-747.

Kjaerland, M. (2006). A taxonomy and comparison of computer security incidents from the commercial and government sectors. Computers & Security, 25(7), 522-538. doi.org/10.1016/j.cose.2006.08.004

Krsul, I. V. (1998). Software vulnerability analysis. West Lafayette, IN: Purdue University. http://coast.cs.purdue.edu/pub/papers/ivan-krsul/krsul-phd-thesis.pdf

Landwher, C., Bull, A. R., McDermott, P. J., & Choi, S. W. (1993). A taxonomy of computer program security flaw. Technical report, Naval Research Laboratory. https://cwe.mitre.org/documents/sources/ATaxonomyofComputerProgramSecurityFlawswithExamples%5BLandwehr93%5D.pdf

Li, X., Chen, J., Lin, Z., Zhang, L., Wang, Z., Zhou, M., & Xie, W. (2017, September). A new method to construct the software vulnerability model. In 2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA) (pp. 225-229). IEEE. doi.org/10.1109/CIAPP.2017.8167212

Longley, D., & Shain, M. (1990). The Data and Computer Security Dictionary of Standards. Concepts and Terms.

Masera, M., Fovino, I. N., & Sgnaolin, R. (2005). A framework for the security assessment of remote control applications of critical infrastructures. In Proceedings of the Twenty-Ninth ESReDA Seminar.

McPhee, W. S. (1974). Operating system integrity in OS/VS2. IBM System Journal, 13, 230-52.

Moore, A. P., Ellison, R. J., & Linger, R. C. (2001). Attack modeling for information security and survivability. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst. https://apps.dtic.mil/sti/citations/ADA388771

Nabi, F. (2005). Secure business application logic for e-commerce systems. Computers & Security, 24(3), 208-217. doi.org/10.1016/j.cose.2004.08.008

Nabi, F. (2011). Designing secure frame work method for e-commerce systems. Journal of Network Security, 12, 29-41.

Nabi, F., & Nabi, M. M. (2017). A process of security assurance properties unification for application logic. International Journal of Electronics and Information Engineering, 6(1), 40-48. http://ijeie.jalaxy.com.tw/contents/ijeie-v6-n1/ijeie-v6-n1.pdf#page=44

Pothamsetty, V., & Akyol, B. A. (2004, November). A vulnerability taxonomy for network protocols: Corresponding engineering best practice countermeasures. In International Conference on Communications, Internet and Information Technology, (pp. 168-175), St. Thomas, US Virgin Islands. https://www.researchgate.net/publication/221425438_A_vulnerability_taxonomy_for_network_protocols_Corresponding_engineering_best_practice_countermeasures

Scott, D., & Angelos, S. (2013). Towards a Cyber Conflict Taxonomy. In: 5th International Conference on Cyber Conflict, (pp. 45-56).

Simmons, C., Ellis, C., Shiva, S., Dasgupta, D., & Wu, Q. (2009). AVOIDIT: A Cyber Attack Taxonomy. University of Memphis, Technical Report CS-09-003.

Simpson, G. G. (1945). The principles of classification and a classification of mammals. Bulletin of the American Museum of Natural History, 85. xvi+350. http://hdl.handle.net/2246/1104

Simpson, G. G. (1961). Principles of animal taxonomy. Columbia University Press, ISBN: 9780231888592.

Tsipenyuk, K., Chess, B., & McGraw, G. (2005). Seven pernicious kingdoms: A taxonomy of software security errors. IEEE Security & Privacy, 3(6), 81-84. doi.org/10.1109/MSP.2005.159

Weber, S., Karger, P. A., & Paradkar, A. (2005). A software flaw taxonomy: Aiming tools at security. ACM SIGSOFT Software Engineering Notes, 30(4), 1-7. https://dl.acm.org/doi/abs/10.1145/1082983.1083209