4.1. Objects, Attributes and Constraints of a System

Object: An object is an "entity" that contains or receives information, that has a unique name, and that has a set of operations that can be carried out on it [18].

- Attribute of Object of an object is a data component of an object. A derived attribute of another attribute is a data component of the later attribute.
- Property of Attribute is a characteristic of the attribute that can be derived from the attribute through the application of a function to the attribute.
- Attribute refinement is a finite refinement of attributes within larger attributes, and results in the identification of the attributes about which assumptions are made. The at-tribute refinement cannot contain a property of an attribute. Attribute refinement cannot contain a property of an attribute.
- Attribute Constraint identifies the property or set of properties that are being assumed about that attribute.

4.2. Taxonomic Characters, Object Attributes or Features

The bases of the development of successful classifications are taxonomic characters [12][16]. These are the properties or characteristics of the objects that will be classified. Taxonomic characters are also commonly called features, at-tributes or characteristics. [12] argues that such properties should be readily and objectively observable from the objects in question.

5. Concept of Attack Pattern

An attack pattern is an abstraction mechanism for de-scribing how a type of observed attack is executed. Following the pattern paradigm, it also provides a description of the context where it is applicable and then, unlike typical patterns, it gives recommended methods of mitigating the attack. In short, an attack pattern is a blueprint for an exploit. An attack pattern should typically include the following information.

In the light of the above theory and concepts, discussion and references regarding taxonomic classification of system object, attributes, properties, features and constraints based on principles, procedures and rules. We would like to first define clear definition of web software application vulnerability before moving towards the contribution of taxonomy based on classification and characterization of two different categories of vulnerabilities (Technical vs Logical).

5.1. Web Software Application Vulnerability

We define web software application vulnerability as "Web software application vulnerability includes mis-matches between application software Architectural / De-sign logic and the assumptions about the environment made during the development/Implementation (code writing) and operation of the programme and the environment in which the programme executes"[25].

5.2. Taxonomy of Computer Program Security Flaws

A Flaw can be categorized as malicious or non-malicious.

- Malicious Flaws: are intentionally introduced in the system to cause a security violation. These take the form of viruses, worms, Trojan horses, time bombs, and trap doors in the code [17].
- Non-malicious Flaws are introduced because of missing requirements or misunderstanding of Design Logic.

Flaws are software problems that exist in the software's design. A flaw may or may not represent vulnerability in the underlying software. Mitigating a flaw typically involves significantly more effort than simply modifying a few lines of code. The problem does not lie solely in the implementation; the underlying design is flawed, there-fore, any implementation that follows the design would contain the Flaw. For example, performing sensitive business logic in an untrusted client application is a de-sign flaw that cannot mitigated by a simple measure such as modifying array bounds [19][23]. The Flaws are categorized according to the time in the soft-ware life cycle that they were introduced into the system. Time of introduction includes Flaws that were introduced during development, maintenance, or operation.

5.3. A Taxonomy of Security Faults:

Several security fault classification schemes have been proposed that categorized faults according to different criteria [24][26].

- Coding faults are comprised of faults that were introduced during software development. These faults could have been introduced because of errors in programming logic, as well as missing or incorrect requirements.

- Operational faults result from improper installation of software. Most policy errors can be classified as operational faults [26].

- Environment faults result when a programmer fails to completely understand the limitations of the run-time environment or interactions between functionally correct modules [24].

5.4. A Taxonomy of Security Error, Faults, and Failures [15]

- An error is a mistake made by a developer. It might be a typographical error, a misreading of a specifications, a misunderstanding of what a subroutine does, etc.

- Faults are in the text of the program. "An error might lead to one or more faults" a fault is the difference between the incorrect program and the correct version.

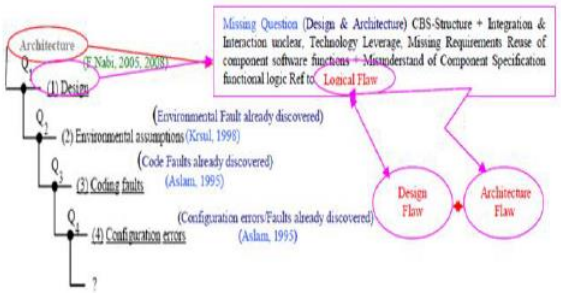

Figure 2. Taxonomy of Software Vulnerability Causes

- The execution of faulty code may lead to zero or more failures, where a failure is the [non-empty] difference between the results of the incorrect and correct pro-gram.

6. Prior Vulnerability Taxonomies Classifications

A taxonomy of recurring vulnerabilities may be helpful in organizing the information needed to increase security awareness. An advanced knowledge of vulnerabilities may be helpful in identifying potential attacks on a web application software before it is released to customers.

We examined 25 taxonomies from 1974 to 2017 and evaluated different level of vulnerabilities, e-commerce threat classifications property taxonomies, web application vulnerabilities, network vulnerability taxonomy and software vulnerability taxonomy before limiting the main scope of this study as focusing on logic attack problems .This is because of mismatch between design / Architecture design flaw, while developing web software application. Our attack patterns are more specific to what method can pinpoint vulnerability in a system design.

6.1. Classification of Security Threats in e-Commerce

In general, categorizing a phenomenon makes systematic studies possible. In particular, an organized classification of threats to e-commerce can help managers build systems that are less vulnerable [14]. An established classification would also be useful when reporting incidents to incident response teams.it is recommended the following properties for the classification for information security:

- The categories should be mutually exclusive (every specimen should fit in at most one category) and collectively exhaustive (every specimen should fit in at least one category).

- Every category should be accompanied by clear and unambiguous criteria defining what specimens are to be placed in that category.

- The taxonomy should be comprehensible and useful not only to experts in security but also to users and administrators with less knowledge and experience.

- The terminology of the taxonomy should comply with established security terminology (something that is not always easy to define).

6.2. Classification of Web Taxonomy

Vanden Berghe and Riordan [13] discussed a methodology for taxonomizing vulnerability and provide an example of web services, WS architectural four components model and their connections. It discusses two subclasses of "In-put Format and Input" Origin then conclude with attack flows based on boundary condition error category as example error caused by a buffer reserved memory being exceeded by unexpectedly long "Input" which execute arbitrary code by an attacker (Programme written in C or C++).The author's Result Matrix which was proposed, gives the same as and almost a copy of Aslam,1995 and Krsul,1998's classifications [13].

Alvares and Slobodan introduced the taxonomy for web attacks. Entry point, target, HTTP Verbs, and HTTP Headers are web-specific categories that they consider important for an accurate classification of web attacks, these are not covered by general taxonomies.

In addition, some categories that can also be met in general taxonomies, such as vulnerability, and web-specific values (e.g. Code injection, HTML manipulation, canonicalization, overflows and misconfiguration). Alvares separated and ordered the taxonomy with the attacker's point of view. An attacker can get access as a result of two vulnerability errors point that may be web server or web application entry points searching for attack

[14]. This is incomplete taxonomy and cannot be called a classification scheme,

It is more likely representing the generalize life cycle HTTP based attacking steps by a hacker that it might take. Since it fails to classify the attack patterns, vulnerabilities classification and their characterization according to their attributes, according to the classification and characterization methods referenced by Krsul [24] and Aslam [26].

7. Taxonomy of Logic Attacks-Types and Classification

The different types of logic Attacks occur each time since they must exploit a function or a feature that is specific to the application. The Logical Attacks focus on the exploitation of a web application's logic flow. Application logic is the expected procedural flow used in order to perform a certain action. Password recovery, account registration, auction bidding, and e-Commerce purchases are all examples of application logic. A web site may require a user to correctly perform a specific multi- step process to complete a particular action. An attacker may be able to circum-vent or misuse these features to harm a web site or its users [25]. As mentioned above the main scope of this study is to focus on "application logic-based vulnerabilities" problem that is because of a mismatch between design and Architecture while developing web software application. We have identified seven vulnerabilities in the application logic and then Taxonomy is supported by a Case as reference cause of classification design flaw.

The proposed contribution of the taxonomy is characterized by attack pattern and target agent in each kind of attack as mentioned above application logic graph-based attack pattern method, vulnerability class is logical. This is then further put into attack pattern technique to classify each vulnerability in the light of attack method, such classifications are characterized in groups of attacking parameters which defines nature of vulnerability.

7.1. Case as a Reference: Mars Polar Landing Mission

The case as a reference is given to study here for component-based systems and its applications. The case dis-cusses the one of the mentioned classifications of fault or defect in system composition while component-based approach is being adopted for mission critical system development by NASA [25].

Reason of Project Failure: Touchdown Monitor (TDM) (Touchdown Monitor) component failed to meet the requirement Specification as compare to its functional specification based on design by contract interface driven specification integration, which gave birth to design flaw in the MPL system and mission Failed.

Requirement Modeled of TDM: TDM is a software component of MPL system that monitor the state of three landing Legs during Two stages of descent.

Logical Component Information Processing: Multi-Tasking executive Calls TDM module at a rate of 100 times per second, receives information on the leg sensors from a second module. These two modules establish interface to TDM. During First stage, starting 5 KM above Mars Surface, TDM software Monitors the three touchdown Legs.

Application Logic of Component: At First Stage start reading at approximately 5 Km above Mars surface, TDM monitors touchdown legs, one sensor each leg to determine touchdown.

Processing Logic design: Developer assumed when Legs lock into deployed position, it was a known possibility sensor might indicate an erroneous touchdown signal. TDM software was to handle this potential event by Marking Leg that generates a spurious signal on two consecutive sensor-reads as having a bad sensor.

Second Stage: Starts about 40 meters above surface, TDM was to monitor the remaining Good sensor. When a sensor had 2 consecutive reads indicating Touchdown, TDM software was to command the descent Engine to Shut down.

What happened? One or more sensor did have 2 Consecutive reads be-fore40 meter Point, Leg-sensor information was stored in TDM Component Memory. When MPL crossed 40-meter point, TDM changed states and read the memory associate with the leg-sensor during first stage of descent. Result shutdown Engine.

Scientific Justification: Developer could have designed and implemented the requirement in many ways, but the Essence of design flaw, is components predicates (pre-condition, Post-condition and Invariants) violated an execution of state of bad sensor information retained by the programme variables.

It has been proved that problem was not in the implementation logic but rather design by contract technique of application logic related to logical component, and its Requirement Specification as compare to Functional, Specification based interface driven integration, which gave birth to design flaw in the MPL system and mission Failed. Therefore, classification of this defect is characterized design flaw, which is a logical defect, as identified by our classification of vulnerability through SVAM (Figure 1).

7.3. Taxonomy of Logical Vulnerabilities vs Technical Vulnerability

In the light of our research, we would like to propose a classification and characterization of above defined two categories of vulnerability issues /problems (Technical vs Logical Vulnerabilities). These are classified on the base of their attack method as mentioned above in the classification of each vulnerability (attack pattern technique). Therefore, keeping view classification of two different categories of vulnerabilities, we have drawn a classification tree all sub-class attack under each class of vulnerability. A new Taxonomy in the application layer of e-commerce systems is depicted here with detailed classification and having characterized by their unique signature of identity.

7.4. Vulnerability Life Cycle in Context Software Process Model

Attack patterns are descriptions of common methods for exploiting software. They are derived from the concept of design patterns [46] the proposed model clearly depicts the stages of two different vulnerabilities life cycle as mentioned in the figure 6.One explains Design and architecture and other one shows Implement level, each phase shows two different vulnerability causes, such as design phase refers to design flaw and architectural flaw and Implementation phase shows faults, bugs and errors. The technique of identifying vulnerabilities is achieved via mismatching a sequence of components in a system de-sign that permits the sequence of events in the attack pattern to occur. If it exists, then the vulnerability may exist in the application being analysed. The proposed model also provide a detailed information of all stages of secure system development process and identifies the two different categories of vulnerabilities, both at design and implementation level .This helps to understand two different vulnerabilities lifecycle and there point of close-ness as mentioned in figure.

8. Conclusion

A taxonomy is a footprint for software designers to per-form secure system engineering. The approach taken in this paper focus on component-based web e-commerce applications and logical vulnerabilities to characterize and then make classification of the vulnerabilities. There-fore, by performing the proposed approach and method in the design phases increases security awareness at the be-ginning of the software process and encourages risk management to begin early so that security team be able determine how to fortify their software application logic. We have also classified the two different categories of vulnerability in component based software development model and depict the birth of attack design by cause of vulnerability into different phases of development life cycle, which is helpful for developers during the software design adoption of security by design technique.

9. References

[1] Firesmith D. G., (2007), A taxonomy of security related requirements, software engineering institute, Carnegie Mellon University.

[2] Schneier, B., Secrets and Lies: Digital Security in a Networked World, John Wiley and Sons, August 2000.

[3] Jones, A., Ashenden, D., (2005), Risk Management for Computer Security: Protecting Your Network and Information Assets, March, Elsevier.

[4] Bishop, M., (2004), Computer Security Art and Science, November, Addison Wesley.

[5] Masera, M., Nai Fovino, I., and Sgnaolin, R., (2005), A Framework for the Security Assessment of Remote Control Ap-plications of Critical Infrastructure. ESReDA 29th Seminar, Ispra.

[6] US National Security Agency (2005), The Mission-Oriented Risk and Design Analysis called (Morda).

[7] Moore, A., Ellison, R., and Linger, R., (2001), "Attack Modelling for Information Security and Survivability," Technical Note, CMU/SEI-2001-TN-001, Software Engineering Institute, Carnegie Mellon University, March.

[8] Grolier Incorporated. (1993), Encyclopaedia Americana, Deluxe Library Edition, Grolier Inc.

[9] EBRIT, (1997), Britannica Online version 97.1.1. http://www.britannica.com (Access Date: 11 December 2018).

[10] WEBOL (1998), Merriam-Webster Online: WWWebster Dictionary. http://www.m-w.com/dictionary.htm (Access Date: 15 November 2018).

[11] Simpson, G.G., (1945), The principles of classification and a classification of mammals. *Bull. Am. Mus. Nat. Hist.* 85: i-xvi, 1-350.

[12] Simpson, G. G., (1961), Principles of Animal Taxonomy. Columbia University Press.

[13] Vanden Berghe, C., and Riordan, J., (2005), A taxonomy methodology applied to web services, IBM Zurich Research Laboratory.

[14] Alvares and Slobodan, (2003), A taxonomy of web attacks, ICWE, 2003 LNCS 2722, Spriger-Verlag Berlin Heidelberg.

[15] IEEE. (1990), ANSI/IEEE Standard Glossary of Software Engineering Terminology, IEEE Press.

[16] Glass, R. L., and Vessey, I., (1995), Contemporary Application-Domain Taxonomies. IEEE Software 12, 4 (July), 63,76

[17] Carl Landwher et al. (1993), A taxonomy of computer program security flaw Technical report, Naval Research Laboratory, November.

[18] Longley, D., and Shain, M., (1990), The Data and Computer Security Dictionary of Standards, Concepts, and Terms. Macmillan Stockton Press.

[19] Faisal Nabi, (2005), "secure business application logic for e-commerce systems" Journal computers and Security.

[20] McPhee. W. S., (1974), "Operating System Integrity in OS/VS2.IBM Sys. J.", vol. 13, no. 3, pp. 230–52.

[21] Abbott, R. P., et al, (1976), "Security Analysis and Enhancements of Computer Operating Systems," Report NBSIR 76- 1041, Institute for Computer Science and Technology, Natl. Bur. of Stnds, Apr.

[22] Bisbey, R., and Hollingsworth, D., (1978), "Protection Analysis Project Final Report", Information Sciences Institute, University of Southern California, Marina Del Rey, CA.

[23] Nabi, F., (2011) "Desgning secure framework method for e-commerce systems" Journal of Network Security.

[24] Krsul, (1998), "Software Vulnerability Analysis", PhD, Purdue University, West Lafayette.

[25] Nabi, F., (2017), A Process of Security Assurance Properties. Unification for Application Logic, International Journal of Electronics and Information Engineering, Vol.6, No.1, pp.40-48, Mar.

[26] Aslam, T., (1995), "A taxonomy of Security Faults in the Unix Operating System," M.S. Thesis, Purdue University.

[27] Bishop, M., (1995), "A Taxonomy of UNIX System and Network Vulnerabilities," Technical Report CSE-95-10, Purdue University, May.

[28] Du W., and Mathur. A. P., (2000), "Testing for software vulnerability using environment perturbation", Proceeding the International Conference on Dependable Systems and Networks (DSN 2000), Workshop on Depend-ability versus Malicious faults, http://www.ceria s.pur-due.edu/homes/duw/research/paper/ftcs30 workshop. (Access Date: 19 Deceember 2018).

[29] Lough, D., (2001), "A Taxonomy of Computer At-tacks with Applications to Wireless Networks," PhD thesis, Virginia Polytechnic Institute and State University.

[30] Piessens, F., (2002), "A taxonomy of causes of soft-ware vulnerabilities in internet software", Supplementary Proceedings of the 13th International Symposium on Software Reliability Engineering.

[31] Gray, (2003), "An Historical Perspective of Software Vulnerability Management," Info. Sec. Tech. Rep., vol. 8, no. 4, April, pp. 34–44.

[32] Jiwnani K., and Zelkowitz, M., (2004), "Susceptibility Matrix: A New Aid to Software Auditing," *IEEE Sec. and Privacy*, vol. 2, no. 2, Mar–Apr, pp.16–21.

[33] Pothamsetty, V., Akyol, B., (2014), "A Vulnerability Taxonomy for Network Protocols: Corresponding Engineering Best Practice Countermeasures," in IASTED Int. Conf. on Commun., Internet, and Inform. Tech. (CIIT), ACTA Press, US Virgin Islands.

[34] Tsipenyuk, K., Chess, B., and McGraw, G., (2005), "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors," IEEE Sec. and Privacy, vol. 3, no. 6, Nov.– Dec. pp. 81–84.

[35] Weber, S., Karger, P. A., and Paradkar. A., (2005), "A Soft-ware Flaw Taxonomy. Aiming Tools", At Security Software Engineering for Secure Systems–Building Trustworthy Applications (SESS'05).

[36] Seacord, R. C., "Secure Coding in C and C++". USA: Addison-Wesley Professional, 2005.

[37] Hansman, S., Hunt R., (2005), "A taxonomy of network and computer attacks". Computer and Security, vol. 24, Issue 1, February, pp. 31-43.

[38] Kjaerland, M., (2006), "A taxonomy and comparison of computer security incidents from the commercial and government sectors". Computers and Security, Volume 25, Issue 7, October, pp 522–538.

[39] FORTIFY, http://www.fortifysoftware.com/ (Access Date: 10 November 2018).

[40] Bazaz, A., Arthur, J. D., (2007), "Towards a Taxonomy of Vulnerabilities", HICSS, 2007, Proceedings of the 40th Annual Hawaii International Conference on System Sciences, pp. 163a, doi:10.1109/HICSS.2007.566.

[41] Igure V. M. and R. D. Williams, (2008), "Taxonomies of Attacks and vulnerabilities in Computer Systems", IEEE Communications Surveys and Tutorials 1st Quarter.

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

[43] Cebula, J. J., and Lisa, R. Y. (2010), "A Taxonomy of Operational Cyber Security Risks", (Carnegie Mellon University / Software Engineering Institute No. CMU/SEI-2010- TN-028). http://www.sei.cmu.edu/library/abstracts /report s/10tn028.cfm (Access Date: 10 November 2018).

[44] Scott, D., and Angelos, S., (2013), "Towards a Cyber Conflict Taxonomy", 5th International Conference on Cyber Conflict K. Podins, J. Stinissen, M. Maybaum (Eds.).

[45] Joshi C., and 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, August.

[46] Johnson, R., Gamma, E., Vlissides, J., Helm, R., (1995), De-sign Pattern: Reusable Object-Oriented Soft-ware, pp 65-75, Addition Wesley.

