International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-3, February 2020

Detection of Vulnerability Injection Point in Software Development Lifecycle for Effective Countermeasures

Thejasvi N., Shubhamangala B. R.

Abstract— ***This paper takes a deeper look at data breach, its causes and the linked vulnerability aspects in the application development lifecycle. Further, the Vulnerabilities are mapped to the software development life cycle (SDLC) involving requirement elicitation, design, development, testing and deployment phases. Being aware of exact SDLC life cycle where the vulnerabilities are injected, suitable security practices (countermeasures) can be adopted in delivery methodology, which can control the eventual data breaches and safeguard the application from security perspective.***

Our research focuses on Evolution of Vulnerabilities through the application development life cycle, and we have leveraged "Inverted Tree Structure/Attack Tree" and "Affinity Principles" to map the vulnerabilities to right Software Development Life Cycle.

Keywords: Vulnerability, SDLC, Data Breach, DevSecops, Security Requirements.

I. INTRODUCTION

In the new age world of software delivery, Agile methodology is the most popular and widely adopted approach. Agile has several variants such as Scrum, Extreme Programming, Kanban, Scrum Bann, etc. The latest addition for large organizations being SAFe approach. SAFe stands for Scaled Agile framework that caters to Team, Value stream, Program and Portfolio levels.

Inspite of several best practices and proven techniques advocated by these application delivery methodologies, data breach is still a common concern across the globe. Data breach cases in the year 2018 reached all time high with a greater number of well-reputed organizations coming under threat, with their customer information being compromised.

"In computer application security, a vulnerability is a flaw which can be exploited by a threat actor (an attacker/hacker), to execute unauthorized actions within a computer application". This paper takes a deeper look at data breach, its causes and the linked vulnerability aspects in the application development lifecycle. Further, the Vulnerabilities are mapped to the software development life cycle (SDLC) involving requirement elicitation, design, development, testing and deployment phases. Being aware of exact SDLC life cycle where the vulnerabilities are injected, suitable security practices can be adopted in delivery methodology, which can control the eventual data breaches and safeguard the application from security perspective.

To accurately map the vulnerability with SDLC life cycle we use Inverted tree and Affinity approach. The clustering thus derived will serve as key decision support input to adopt suitable stratum of security best practices at early phases of project delivery through appropriate counter measures.

The rest of this paper is structured as follows: We describe the background in section 2 and survey findings in section 3. We describe Industry specific solutions and Related work in section 4 and 5. Problem statement and research methodology is presented in section 6 and 7. We highlight our experiments and results in Section 8. We discuss the findings & conclude in Section 9 & 10.

Revised Manuscript Received on February 05, 2020.

* Correspondence Author

Thejasvi N.*, Computer Science and Engineering, Jain University, Bengaluru, India. E-mail: thejasvin@gmail.com

Dr. Shubhamangala B. R., Professor, Research Head, Bengaluru, India. E-mail: brm1shubha@gmail.com

II. BACKGROUND

The definition of success in today's IT landscape is evolving. The traditional measures of scope, time, and cost are no longer enough in today's competitive environment. Large IT enterprises like Google, Microsoft, Salesforce, and Amazon too are being hit by security vulnerability time-to-time inspite of having best in class security policies and technology advancements in place.

Several IT giants have embraced several security shields such as Deception, Threat Intelligence, Security by design, Security ratings, AI cyber security, BOTS, Threat hunting, including outsourcing to managed security companies inspite of all these counter-measures; data breach incidents are on constant rise year after year.

As per Breach Level Index [1] data breach incidents are getting faster and larger in scope, recent data breach alone in the year 2017 extends around various sectors ranging from Health, Education, Government, Technology, and Financial sector.

Through 2015 onwards, there has been upward trend in both data breach by volume and data breach by sensitivity. With cloud and IOT, data breach means increased sensitive data being more vulnerable.

Few high-profile data breach incidents in recent past:

As per FireEye report, Communications, Media and Entertainment (CME) industry alone has found 91% of breach caused due to failure in conventional defense mechanism.

Uber paid hacker heavy ransom to delete the breached data than disclosing the real breach size, which is now mandated from non-US legislations such as European General Data Protection Regulation (GDPR) law [5].

T-Mobile, Facebook & Reddit were few more high-tech industries segments affected by data breach but were either reluctant to disclose the scale or were late in disclosing.

III. SURVEY FINDINGS

As per the breach data available via https://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/ [3] below were the top reasons of data breach in recent years:

TABLE 1: Data breach – Cause of Breach (Vulnerability reasons)

Year	Category	Organizations affected
2014 till 2018		
	Lost/Stolen Media	Advocate Medical group
	Lost/Stolen Computer	Florida Court, Crescent Health
	Hacked	EBay, Adobe, AOL
2013		
	Accidentally Published	Facebook
	Hacked	Evernote, Living Social, Yahoo
2012		
	Accidentally Published	Apple, LinkedIn
	Hacked	Sony Online Entertainment,
	Poor Security	Accendo Insurance Co.
	Inside Job	Emory Healthcare
	Virus	Massachusetts Govt

TABLE 2: Data breach – by Industry and cause

Major Breach Cause	Industry
Hacked	CME
Inside Job	Health care
Virus	Financial

Common Vulnerabilities & Exposures (CVE) - https://cve.mitre.org/ lists vulnerabilities grouped by year, per type and per product across the industries.

National Vulnerability database (NVD) maintains common weakness enumeration (CWE) coded weakness in a hierarchical structure [16]. NVD database also provides exhaustive list of vulnerabilities in a system, which is further mapped to exact product and version using SWID tags (software identification tags / common platform enumeration tag). Overall Impact of Vulnerability is an outcome of BASE {Exploitability (Attack complexity) and Impact (Integrity and availability)}, Temporal {Reported and Remediation level} & Environmental factors [14]

Each Vulnerability or weakness in the system shall be analyzed quantitatively as well as qualitatively. Based on Criticality/Severity, whether the weakness is exploitable and is publicly available? Alternatively, does the exploitation need custom tailoring to each new setup, and accordingly the patch cycle and urgency can be determined.

Below six are the top vulnerability types in the system across last decade:

- Denial of Service (DoS)
- Remote Code Execution
- Overflow & Memory corruption
- XSS & Injection
- Directory Traversal
- Gain Privilege

Every industry has a different type of threat concentration, for example in Retail industry the key threats are web application attacks leveraging poor validation of inputs, Disk operating systems (DOS) attacks or stolen credentials. In Healthcare industry, data breach is the major threat involving medical records etc. In Financial services industry, DOS, Phishing and social engineering are key issues while new type of threats are emerging like Automatic Teller Machine (ATM) jackpotting

The attack surface and attack modes are continuously changing. AI-powered attacks, sandbox-evading software attacks, Advanced Persistent Threats (APTs) are on the rise and data breach attacks are peaking as well.

Application vulnerabilities are the key risk area for exploits. Vulnerabilities broadly fall into two categories: bugs at the design and code level and flaws at the implementation level (which include configuration, hardware, software, and Perimeter etc.).

Firstly, assets, vulnerabilities and threats are identified. An analysis can then take place to determine the probability of a vulnerability being exploited, the hostile impact of a vulnerability and, finally, the risk levels associated with each threat-vulnerability combination and the SDLC phase in which the vulnerability got introduced. Countermeasures are then be applied to eliminate or prevent the exercise of vulnerabilities.

This paper attempts to map the vulnerability source further to SDLC life cycle using Affinity technique so to actionize against the very introducing of weakness into the system at right phase preferably at early stage as possible.

IV. INDUSTRY SPECIFIC SOLUTION

According to industry reports, cyber-crime damages will cost the world $6 trillion annually by 2021. This represents the greatest transfer of wealth in history through mischievous means, risking the incentives for innovation and investment. There are several industry frameworks around threat modelling and scoring:

STRIDE – Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, or Elevation. STRIDE enables a structural way of categorizing the relevant threats by attacker goals.

DREAD–Damage, Reproducibility, Exploitability, Affected Users, & Discoverability. DREAD enables business view of the Threat.

CVSS – Common Vulnerability scoring system is popular for relative ranking of risks based on how easy a vulnerability is to exploit.

DevOps is already becoming DevSecOps in lieu with application security

OCTAVE stands for Operationally Critical Threat, Asset, and Vulnerability Evaluation.

It is a qualitative risk analysis methodology; Octave is designed and available for large and small organizations.
Standards organizations such as the National Institute of Standards and Technology (NIST), International Organization for Standardization (ISO), the Open Web Application Security Project (OWASP), the Software Engineering Institute (SEI),
Payment Card Industry Data Security Standard (PCI DSS) and few others have been providing guidance and standards for application security. Some of these organizations (ISO, NIST) have the power to enforce the standards they publish with help of rule of law.
From our literature survey, it is found that organizations are still being attacked inspite of several industry best practices and governances put in place.

V. RELATED WORK

Data breach is inherent issue within system security discipline in Information Technology (IT) applications. Extensive research work has taken place to find the reasons of system vulnerabilities, secure development process, security issues in agile methodologies, threat modelling and measuring quality of application.
To author's best understanding and efforts any research undertaken to map the breach incident against the SDLC lifecycle was not found. The summary of the literature survey is as below: Author in [4] discusses challenges and mitigation around integrating security into Agile methodologies. Authors in [7] have undertaken study on threat modelling for enterprise scale and address scalable issues in attack trees. Authors in [8] have report on the "Threat agent and Vulnerability analysis' and introduce the TAME methodology (Vidalis '01). Author in [9] studies the security of specific Myproxy system using attack tree methodology. Authors in [10] studies security centered approaches in Scrum, Extreme Programming and suggests for Risk based security model. Authors in [11] proposes collaboration in secure development process by discussing security centric design patterns and highlights design phase as the most creative part of development process, Authors in [12] contributed a metric to measure quality of application security considering threat resistance, countermeasure effectiveness, vulnerability intensity and breach cost.
The summary of related work indicates the absence of research approach to study data breach incidents and trace them back to vulnerability sources in SDLC life cycle. The research undertaken in this paper is unique in two ways. First, the research was conducted in diverse application domains. projects considered are from low, medium to high complexities and different methodologies of development. Secondly, solution to reduce breach cost is centered on the practical study carried over in IT industries; our process can be used in conjunction with existing development processes thus adapting to the needs of the software delivery methodologies.

VI. PROBLEM STATEMENT

Based on the research gaps identified from the existing literature, our research goal is to study evolution of vulnerabilities and determining injection point in the SDLC life cycle and propose prevention approach at early phase.
This paper addresses below research problems:
Trace back the Data breach incidents to the originating Vulnerability injection source and map to appropriate phase in SDLC life cycle.

VII. RESEARCH METHODOLOGY

Projects chosen for study was selectively spread across four major application domains of software development. The domains considered were communications, healthcare, manufacturing (Mfg.) and banking. Projects chosen were of applications software development type under the Agile methodology with diverse technology platforms. These projects were studied from its inception until post live in a confederation of Capability Maturity Model (CMM)-5 organizations.

TABLE 3: Projects studied

Project	Size (in Person months)	Technology	Complexity level	Domain
P1	20 (S)	Salesforce Communities	Medium	CME
P2	35 (M)	Siebel CRM	Low	Health
P3	40 (L)	Oracle eCommerce	High	Mfg.
P4	42 (L)	Online Banking Application (Java)	High	Banking

Below Techniques have been utilized in this research:

1. **Inverted Tree approach or the Attack tree**
   Attack trees helps in defining and analyzing possible threats expressed in a node hierarchy, allows the decomposition of a nonrepresentational attack into a number of more concrete attack steps.
2. **Affinity**
   An Affinity Diagram gathers large amounts of attributes (ideas, opinions, issues) and organizes them into groupings or clustering based on their natural relationships.

VIII. EXPERIMENT & RESULTS

What path an attacker might take to get into the application? What by-pass mechanisms are required to defeat the existing security measure? What are the possible signs of an attack under a given category of application?
Can existing countermeasures detect or deter the attack?
Answering above interrogations ensures that the organization has considered potential attacks and helps in devising required controls, if existing measures are insufficient.

Experimentation setup:

We worked on four real-time projects as indicated in Table-3 of Research methodology section above out of which 3 projects results are described in section 8A below and fourth project involving banking application results are discussed in detail in Section 8B.

Our first three projects shortlisted for our experimentation was almost getting over, few of the projects had entered technical warranty phase, while few projects were undergoing final user acceptance testing phase. As part of our experimentation, defect dumps were taken from JIRA. The dump had Defect Id, type, subtype, reported date, description of the issue, category of the issue, owner and comments sections. Based on the comments RCA was also conducted and reported in the findings on what exactly went wrong? For focusing on data breach related defects only, security category defects were shortlisted as part of this experiment.

To ensure traceability and proof to establish RCA was indeed correct, Requirements dump, and Design Wiki were also utilized as part of this experiment.

Experiment 1: Defect reports under Security incident – PII data breach in Salesforce communities was picked up.

Experiment 2: Defect reported under security incident – quotation access by unauthorized user was chosen.

Experiment 3: Defect reported under security incident – health data access by unauthorized user was chosen

Experiment 4: Defect reported in CVE-2018-6335 was picked up to depict Denial of service issue.

4th experiment was conducted using a publicly reported data breach incident as details around the breach, the patch applied, root cause of the breach is already available from CVE reference number from the National vulnerability database.

As depicted below an attack path was drawn and based on the defect dump of security category – the RCA was mapped into the attack tree.

Below are few Illustrations of the vulnerability mapping techniques derived from our experiments.

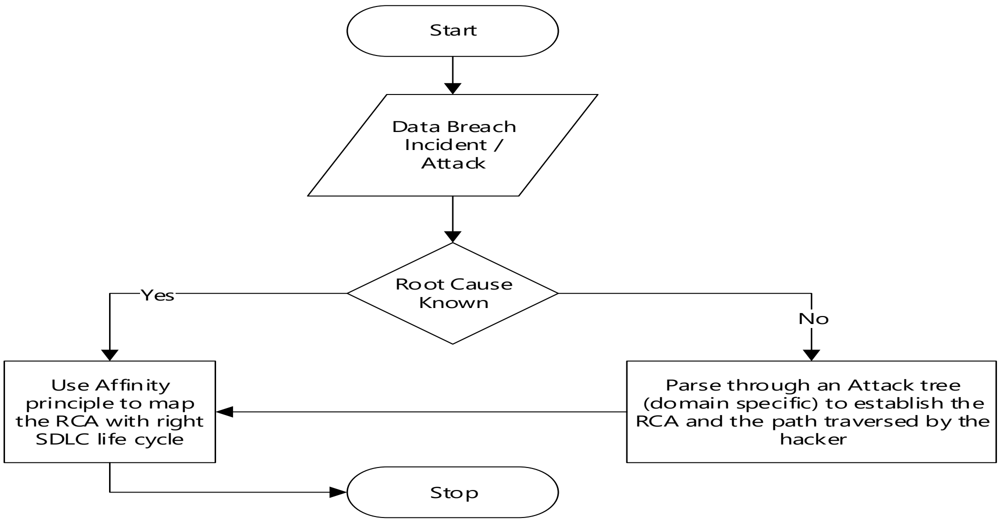
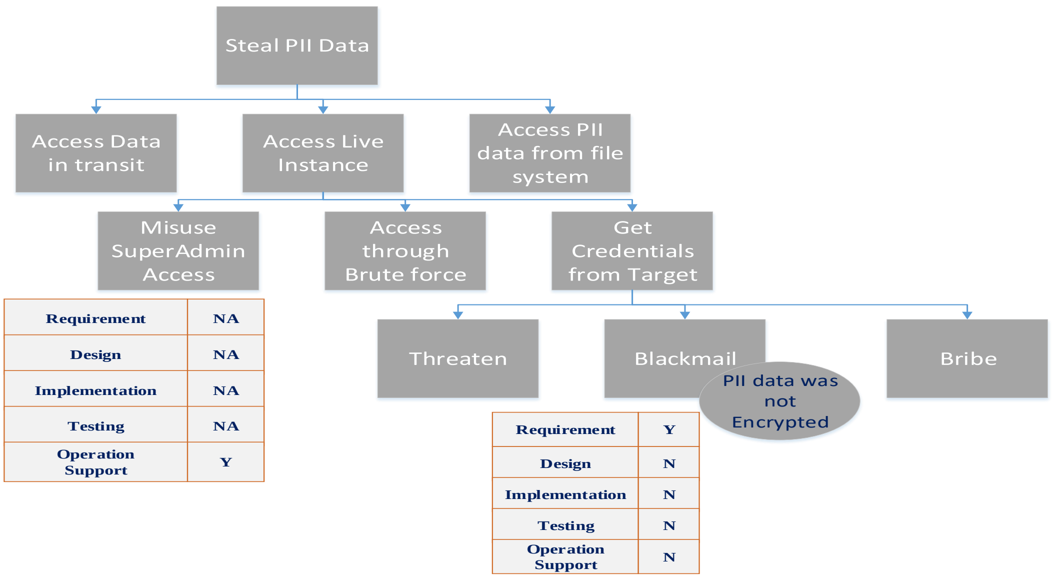

Start
Data Breach Incident / Attack
Root Cause Known
Yes
No
Use Affinity principle to map the RCA with right SDLC life cycle
Parse through an Attack tree (domain specific) to establish the RCA and the path traversed by the hacker
Stop

FIGURE 1: Steps to map a Breach Incident against the Vulnerability source in SDLC stage

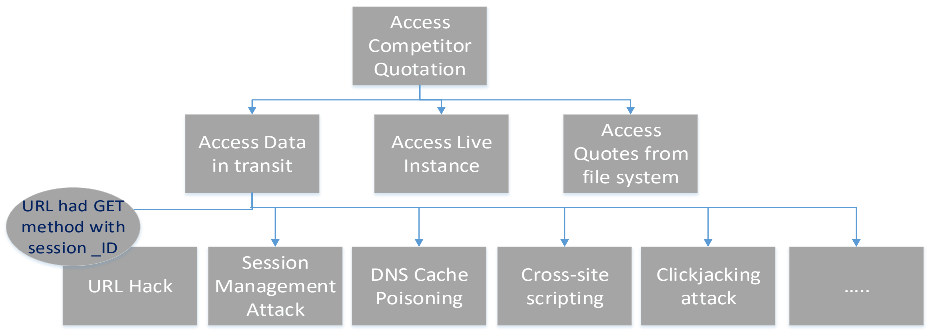

Steal PII Data
Access Data in transit
Access Live Instance
Access PII data from file system
Misuse SuperAdmin Access
Access through Brute force
Get Credentials from Target
Threaten
Blackmail
Bribe
PII data was not Encrypted

Requirement	NA
Design	NA
Implementation	NA
Testing	NA
Operation Support	Y

Requirement	Y
Design	N
Implementation	N
Testing	N
Operation Support	N

FIGURE 2: Attack Tree depicting PII data compromise (Source study: P1 – Salesforce from Table-3 above)

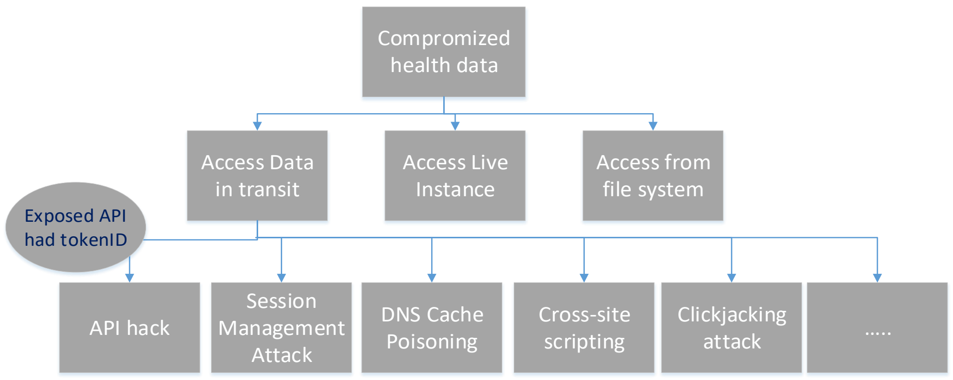
In above attack tree depiction, Personally Identifiable Information (PII) data have been compromised due to two reasons – one misuse of super admin access at operation support level in SDLC life cycle; another due to blackmailing of user to procure the credentials and access the PII data from the live instance of the system; the breach is costlier due to the fact that data at rest was not encrypted. The Affinity principle identifies that the application never had any requirement to encrypt the PII information on live systems, this requirement miss has a costly repercussion as the security incident now can only be fixed by redesign, reimplementation, retesting and redeploying the uplifted encryption capability. The mapping of source SDLC phase is handled by analyzing the breach or defect dump with root cause analysis performed and weighing multiple rationale from the teams involved in various software delivery life cycle. Affinity principle helps in mapping the vulnerability injection point based on the RCA outcome using its natural relationship to most appropriate phase in SDLC.


Access Competitor Quotation
Access Data in transit
Access Live Instance
Access Quotes from file system
URL had GET method with session _ID
URL Hack
Session Management Attack
DNS Cache Poisoning
Cross-site scripting
Clickjacking attack
.....

Requirement	**NA**
Design	**Y**
Implementation	**N**
Testing	**N**
Operation Support	**N**

FIGURE 3: Attack Tree depicting competitor quotes being compromised (Source study: P3 – Oracle E-Commerce from Table-3 above)

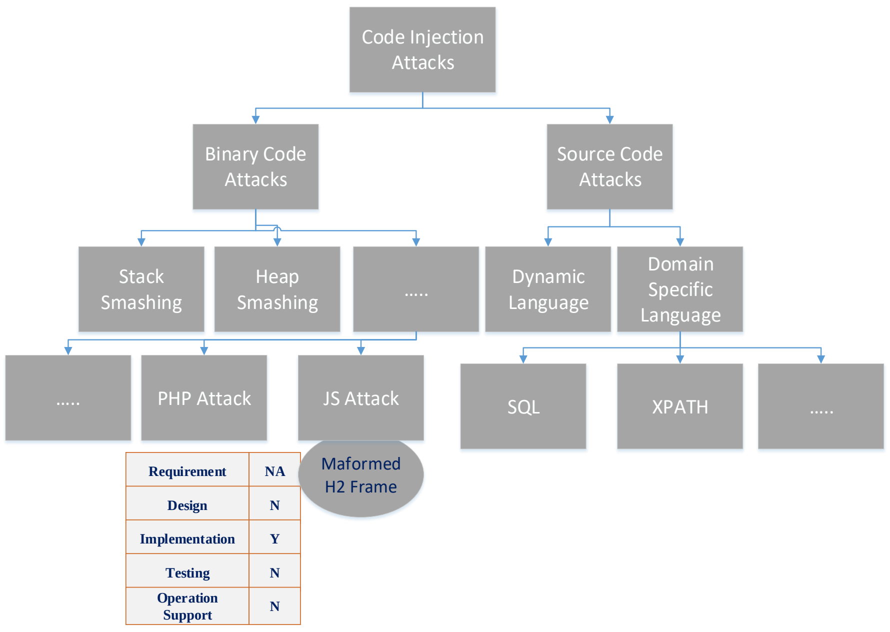
In above attack tree depiction, Quote details of a competitor has been compromised due to URL hack. The detailed Root Cause Analysis (RCA) indicates that though there was no explicit requirement, even then design had adopted privacy by design principle in general, however due to incorrect implementation or choice of protocol (GET method in the Uniform resource locator (URL)) the competitor quotes were easily being downloaded by an URL hacker.
This Vulnerability in the system was not caught in the regular project testing and got deployed onto production and is now caught only in the operation support phase. Source injection of this vulnerability is either design or implementation; during RCA it is also found that design did not mention the GET/POST methods explicitly. Applying affinity principles Design phase carries higher weightage and this design miss was the first source of injection where implementation team also let this vulnerability pass-through, hence this can be treated as design miss. The cost of this breach is to redesign, reimplementation, retest, and redeploy the solution.



Requirement	**NA**
Design	**Y**
Implementation	**N**
Testing	**N**
Operation Support	**N**

FIGURE 4: Attack Tree depicting health data compromise (Source study: P2 – Siebel CRM from Table-3 above)

In above attack tree depiction, health data has been compromised due to insecure Application programming interface (API) design, the breach happened due to multiple federated authentication systems which could use the tokenID to get access to other customer health information without explicitly authorization. Though the requirement need not be explicit, the design has not considered the API signature to be secure enough for external authenticators to exploit! Since implementation and testing has followed the design without explicit vulnerability checks, the breach incident is now costing redesign, reimplementation, retesting and redeployment of revised API. Further as quoted by https://www.cybergrx.com/ around 63% of data breaches are linked to a third party [13]!



Requirement	NA
Design	N
Implementation	Y
Testing	N
Operation Support	N

FIGURE 5: Attack Tree depicting Denial of Service (DoS) attack (Source CVE-2018-6335)

In above attack tree depiction, as reported under CVE-2018-6335 & CWE ID 20, A Malformed h2 frame can cause 'std::out_of_range' exception when parsing priority meta data. This behavior can lead to denial-of-service when using the proxygen server to handle Hyper Text Transfer Protocol (HTTP2) requests. As this was clear case of code injection through header padding, these kind of root causes are mapped under Coding or Implementation stage vulnerabilities.In principle, using attack tree, each data security incident can be analyzed to investigate which path did the hackers took to breach the system, based on the path associated loopholes in the systems which were exploited can be further studied, these vulnerabilities in the system based on root cause analysis and discussions with delivery and testing teams with RCA done to establish in which phase did the vulnerability got introduced into the system. Additionally, audits shall reveal how was data being stolen till now, how did they found out this time, and are there more attack paths yet to be exploited?

Using Affinity diagram, the Vulnerability can be further mapped to suitable SDLC life cycle to compute the cost of breach. RCA and Affinity diagram helps in suitably mapping whether Vulnerability was due to Requirement miss, or Design miss, or Implementation miss, so on.

Most data breach happening on live system reveals that these have been missed through multiple phases of SDLC lifecycle inspite of having Agile, Devops and other latest delivery methodologies.

Result Analysis:

Majority of data security breach are mapped to either not having explicit functional or non-functional requirement from the business or global security stakeholders, followed up incorrect and lack of designs which respect ‘privacy by design’ principles. Implementation misses are next large bucket which often gets missed due to non-adoption of best coding practices and review process with quality assessment tools. Limited functional testing and integration testing with budget constraint and short timelines to launch the application will all contribute to the end product having several vulnerabilities contributed at different SDLC phases waiting to be breached and threatened.

From above experiments, though most of the vulnerabilities injection were at Requirements or design phase, further probing using five-why principle of fishbone approach reveals that majority of the vulnerabilities are due to combination of reasons with different weightage from requirements to design, further ‘Shift left’ approach in software delivery indicates that malformed requirements contributes heavily to vulnerability injection in upcoming phases.

IX. DISCUSSIONS

The security needs to be built into the software from the commencement, and that security undertakings be throughout the delivery functions. Controlling vulnerability injection right at requirements phase ensures time, cost and efforts savings in rest of the phases of software development life cycle.

The contributions of this paper are as follows:

1. Narrow down to the vulnerability injection point to establish effective countermeasures.

2. Attack tree and Affinity combination practically helps in tracing the attach paths taken by the intruder as well as exhibits which other paths are awaiting to be exploited.
3. Best practice adoption at each SDLC development phases shall eliminate the need for costly repudiation remedies at later phases in live applications.

X. CONCLUSIONS

Our experiment provides practical and rigorous approach to detect vulnerabilities injections of technical nature. The prime milestones of the vulnerability injection presented in this paper are Shift left approach, a template of attack tree with affinity model.

The concept of Attack Tree highlights the set of essential attack paths to breach the desired level of security in applications. Viewing security requirement as a countermeasure provides an objective measurement of security in requirements level. Information on the absence of countermeasures facilitates continuous improvement in security requirement elicitation and business analysis.

Detection and opportunity for correction of vulnerabilities in security requirement avoid possible vulnerabilities in the application and enhances the quality of application security. This ensures molding of security quality in the requirement stage. The statement “Security cannot be assured but can be pushed” can be actualized through this process of vulnerability detection.

The nuance will then be in the choices and interpretation of impact and acceptable risk and associated costs / efforts.

This paper limits its scope to detection of vulnerability injection point, the early prevention of technical and logical vulnerability injection at early phase are not catered in this research and could be potential research area for future research.

REFERENCES

1. ‘Data Breach Statistics’, https://breachlevelindex.com/, accessed Jan 2019
2. Brain,Nancy L.Russo: ‘Software development method tailoring at Motorola’ Communications of ACM, 2003, Volume 46, Issue 4, pages 64-70, ISSN: 0001-0782
3. ‘World's Biggest Data Breaches & Hacks’, https://informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-static/, accessed Feb 2019
4. Nguyen .T, Sauter, (2015), ‘Development Methods’, http://www.umsl.edu/~sauterv/analysis/F2015/Integrating%20Security%20into%20Agile%20methodologies.html.htm, accessed Feb 2019
5. Bhandari .G, (2017), https://www.infosys.com/gdpr/Documents/GDPR-industry-geography.pdf, accessed Mar 2019
6. ‘The most infamous data breaches’, https://www.techworld.com/security/uks-most-infamous-data-breaches-3604586/, accessed Jan 2019
7. Bugra Karabey and Nazife Baykal: ‘Attack Tree Based Information Security Risk Assessment Method Integrating Enterprise Objectives with Vulnerabilities’ The International Arab Journal of Information Technology, 2013, Vol. 10, No. 3.
8. S Vidalis and A Jones: ‘Using Vulnerability Trees for Decision Making in Threat Assessment’ School of Computing, University of Glamorgan, Pontypridd, CF37 1DL, Wales, UK available at: www.comp.glam.ac.uk
9. Qiang Duan, ‘Threat Modeling Using Attack Trees’ Consortium for Computing Sciences in Colleges Mid-South Conference, JCSC 2008, 23, 4.
10. Ahmed Alnatheer: ‘The Investigation of Security Issues in Agile Methodologies’ University of Southampton, available at: http://eprints.soton.ac.uk
11. Gunnar Peterson: ‘Collaboration in a secure development process’ Information security bulletin 2004, Vol.9, Page 212.
12. Shubhamangala B. R., Snehanshu saha, and M. Jayalakshmi: ‘The Need for Measuring the Quality of Application Security’ ISACA Journal, 2016, Volume 2
13. ‘Top Third-Party Breaches of 2018’, https://www.cybergrx.com/resources/blog/top-11-third-party-breaches-of-2018-so-far-data-breach-report/ accessed Feb 2019
14. Dugal.D, Rich.D,’Common Vulnerability Scoring System SIG’, https://www.first.org/cvss/ accessed Jan 2019
15. Stroke, ‘Exploit database’, https://www.exploit-db.com/
16. ‘National Vulnerability Database’, https://nvd.nist.gov/vuln/categories accessed Feb 2019
17. Akamai: ‘Evolving threats demand new approaches to security’, Akamai Technologies, Inc. 2015, http://www.akamai.com/dl/brochures/overcoming_security_challenges.pdf.
18. Kitain.L: ‘Root Cause Analysis in the Age of Industry 4.0’, https://medium.com/datadriveninvestor/root-cause-analysis-in-the-age-of-industry-4-0-9516af5fb1d0, 2018
19. ‘Increasing Production Capacity with Automated Root Cause Analysis’, https://iot.seebo.com/hubfs/PDFs%202018/Root%20Cause%20Analysis%20%282%29.pdf, 2017
20. Mee.R, ‘2019 Software Trends’, https://content.pivotal.io/blog/software-trends-for-2019 accessed Apr 2019
21. Dam.R, Sian.T, ‘Stages in the Design Thinking Process’, https://www.interaction-design.org/literature/article/5-stages-in-the-design-thinking-process, 2015
22. ‘CVE Details’, https://www.cvedetails.com accessed Jan 2019
23. Robert Layton and Paul A. Watters, ‘A methodology for estimating the tangible cost of data breaches’ Journal of information security and applications, 2014, 19 vol 321-330
24. David Byers and Nahid Shahmehri Linkoping, Sweden, , ‘Design of a process for software security’ in proceedings of the second internation conference on availability, reliability and security, 2017, pages 196-203, IEEE Computer Society
25. Ludovic-Alexandre Vidal and Franck Marle Malabry, France, ‘A systems thinking approach for project vulnerability management’ Kybernetes 2012, Vol. 41 No. 1/2, pp. 206-228.

AUTHORS PROFILE

Thejasvi N., Department of Computer Science & engineering, Jain University Bengaluru India, has Bachelor’s degree in Information Science and Master’s degree in Telecommunications. He is currently pursuing Ph.D. in Computer Science and Engineering with areas of interest being Application

Shubhamangala B. R., is a researcher in software firm with particular intrests in data science,Data analytics,Application security, Security requirements , Compliance and risk. She has authored paper in the domain of product quality, requirements and application security.She holds a bachelor and master degree. in CS.She is member of IEEE.