

Research Article

Prioritization of Application Security Vulnerability Remediation Using Metrics, Correlation Analysis, and Threat Model

Santanam Kasturi^{1,*} , Xiaolong Li², John Pickard³, Peng Li³

¹Department of Technology Management, Indiana State University, Terre Haute, USA

²Department of Electronics and Computer Engineering, Indiana State University, Terre Haute, USA

³Department of Technology Systems, East Carolina University, Greenville, USA

Abstract

As part of a continuing research for evaluating threats posed for exposed attack surface, this study will provide a consolidated view of exploitability of vulnerable applications presenting a web attack surface of an organization exposed to an attacker. While testing and scanning technologies like Static Analysis Security Testing (SAST), Dynamic Analysis Security Testing (DAST), Application Ethical Hack (Penetration Testing), a monitoring technology like the Web Application Firewall (WAF) provides web traffic information of the number of transaction requests for every application under study. To ensure validity, reliability, and completeness of observation multiple applications must be observed. Research from a prior study is referenced that shows correlation between incoming WAF requests and existing vulnerabilities. Using correlation analysis, vulnerabilities metrics, and a threat model analysis help identify pathways to an attack. A vulnerability map-based attack tree can be developed using Common Weakness Enumeration (CWE) and Common Vulnerabilities and Exposures (CVE) information. The threat model analysis and vulnerability-based attack tree can help in simulation studies of possible attacks. This attack tree will show the linkages between vulnerabilities and a lineage pointing to how an attack could travel from the incoming WAF requests to deep down into the application code of exposed and existing, open vulnerabilities travelling laterally to create a more expanded attack crossing trust boundaries using application data flow.

Keywords

Application Security, Vulnerability Metrics, Correlation Analysis, Threat Model, Vulnerability Map, Attack Tree, Simulation Study, Remediation Prioritization

1. Introduction

The State of Software Security report provides a comprehensive outlook on what goes on in the application security world from a code analysis standpoint [1].

The report highlights growing security technical debt year-over-year if left unremedied can lead to a dormant risk that cannot be easily quantified. Many factors influence the

*Corresponding author: skasturi@sycamores.indstate.edu (Santanam Kasturi)

Received: 8 February 2024; **Accepted:** 23 February 2024; **Published:** 13 March 2024



Copyright: © The Author(s), 2024. Published by Science Publishing Group. This is an **Open Access** article, distributed under the terms of the Creative Commons Attribution 4.0 License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

growing number of vulnerabilities, and these include developer experience, choice of programming language, import of open-source software (OSS), and integration of third-party software, the last two mentioned contributing to spread of vulnerabilities in view of multiple business domains within an organization sharing the code. So, it is imperative to objectively assess the volume of vulnerabilities and execute a remediation program to reduce risk gradually over a period.

An effective remediation program requires:

1. That application is reducing security debt.
2. An aggressive three-month closure rate is being pursued.
3. That 50% of flaws are closed in a quarter, rather than in two quarters, with all applications written in any language.

Multiple application security testing and monitoring tools are deployed at different layers of an application architecture

and capture activities that occur at that layer, [Table 1](#), that includes Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), Software Composition Analysis (SCA), Penetration Testing also referred as Application Ethical Hack test (AEH), and Web Application Firewall (WAF) / Hybrid WAF, and Runtime Application Self-Protection (RASP). References and details can be found in [\[1-9\]](#). With web applications exposed to attackers due to their widespread use, vulnerability data discovered through tests reveals existing, new, and evolving weaknesses in application code that has the potential for a threat actor to exploit. Consolidating all application security vulnerability information from monitoring, detection, and discovery tools into a physical system allows for convergence of observation and response to an event that is a threat.

Table 1. Test / Scan Data.

Capability	Application Layer	Vulnerability Information	Alerts
Static Analysis Security Testing (SAST)	Code	CWEs, Severity, Exploitability	Point in time scan reports
Software Composition Analysis (SCA)	Component	CVEs, Severity, Exploitability	Point in time scan reports
API Security	API	CVE's, CWE's, Severity, Exploitability	Point in time reports
Dynamic Analysis Security Testing (DAST)	URL/UI	CWEs, Severity, Exploitability	Point in time scan reports
Penetration Testing	URL/UI	CWEs, Severity, Exploitability	Point in time test reports
Web Application Firewall (WAF)	Deployed in front of the web app server	Blocked Attack Requests, Incoming threats	Continuous monitoring
Hybrid WAF	Deployed as an agent in the web app server	Blocked Attack requests, Incoming threats, target IP and host information	Continuous monitoring
Runtime Application Self Protection (RASP)	Deployed as an agent in the web app server	Blocked Attack requests, Incoming threats, target IP and host information	Continuous monitoring
External Threat Hunting	External Agency	Web header Issues, Ranking	Point in time reports

Historical vulnerability data also provides deeper insights into hidden issues, and these include:

1. Development priorities and pressures lead to piling security debt. Security is a culture, a mindset!
2. Vulnerabilities getting embedded in a spaghetti code that is hard to separate from dependencies and code-sharing. Especially, vulnerabilities in open-source software (OSS) that is freely available and eases developer pains to avoid re-inventing the wheel by using features and functionalities readily available can be hard to remedy, and due to sharing of these components, hard to replace easily.
3. Linkages between vulnerabilities lead to an attack tree

that shows multiple pathways for an attacker.

While the first two items require process improvements and evolving a maturity model [\[10-12\]](#); the third requires a step-by-step approach to monitor and detect, measure, analyze and correlate, and develop an attack tree. Historical vulnerability data that is still unremedied is a dormant risk that can be exploited by insiders as well as outsider threat actors.

2. Sequence of Measurements and Data Organization

The following steps need to be followed as outlined in [\[13\]](#).

1. Identify and choose the attack types in the form of requests.
 2. Identify statistical techniques for correlation analysis.
 3. Identify the CWEs and CVEs involved.
- A brief description of each step is given below.

2.1. Identify and Pick the Attack Types in the Form of Requests

The information from all the test, scan, and monitoring results provides insight of vulnerability and attack information with various attributes as seen in Table 1. Given below is a list of vulnerability categories chosen for this study. These are some of the commonly exposed vulnerabilities exploited by attackers, to expose a weakness in an application code [13-16].

1. Cross-site scripting (XSS)
2. Injection / SQL Injection (SQLI)
3. Path Traversal (Directory Traversal)
4. Command Execution
5. Backdoor

2.2. Identify Statistical Techniques for Correlation Analysis

The next step would be to do a Pearson's (and / or Spearman) correlation analysis as discussed in an earlier work in this series [13]. Identifying dependent and independent variables is critical to make an accurate analysis. The statistical measures that are analyzed from correlation analysis are – Pearson Correlation, Spearman Correlation (non-parametric in case the dependent variable is not a normal distribution) and Significance (two-tailed) over large number of data points N [17].

Two types of correlations are studied [13]:

1. Correlations of vulnerabilities between test types.
2. Correlation of vulnerabilities between valid requests and vulnerabilities.

2.3. Identify CWEs and CVEs Involved

A vulnerability profile can be created using the CWEs (SAST, DAST, and AEH tests / scans) and the CVEs (from SCA) discovered from monitoring and detection methods through a parent-peer-child relationship between CWEs and between CWEs and CVEs, and project an application-to-application spread of an attack [14-16]. This can be further developed into a meaningful attack tree based on the application profile, equating to a threat model. Using the results from a prior study XSS, SQLI, and Command Execution vulnerabilities were found as most suitable from the correlation analysis [13]. A representative vulnerability map can be shown by linking CWEs and CVEs for the purposes of highlighting how multi-application testing and monitoring can provide an insightful view of a topology of vulnerabilities

[14-16]. The point here is to show that vulnerabilities are not only correlated, but also linked, and spread laterally across the enterprise.

3. Security Governance Program to Analyze and Report Vulnerabilities

An effective security governance program ensures that there are enough controls to protect all weaknesses, there are monitoring and detection technologies to scan and report issues, a full visibility of all existing issues and potential risks, and effective remediation plans if any in place to plug gaps in controls, assess monitoring and detection coverage, and improving the security posture of the organization by addressing any inadequacies. The key highlights of an effective governance program entail the following:

1. Effectiveness of measures, which track top threats by monitoring the subtle variations from a known pattern exposes gaps in the implemented controls.
2. Inadequacy of current security posture, measured through continuous monitoring of people, process, and technology that requires periodic revisions and enforcement of newer policies, controls, standards, and strategize for future organizational security investment.
3. Assessing and evaluating the impact of the risks, coming from uncertain threats that lack reliable data and may potentially possess false positives. Prioritization and organization of threats based on relevance, urgency, feasibility of attacks, maturity, and measurability of remediation actions is an absolute necessity considering the vast amount of vulnerability information that a misstep could lead the organization in a completely wrong direction looking at irrelevant objects.
4. Nontechnology threats, usually stem from behavior and mindset of human resources are grouped as 'insider threats' are those that organization has no direct control through technology or by authority of people until after the fact. These require process improvement from lessons learnt through qualitative metrics gathered over time.

3.1. Security Technical Debt

Development priorities and pressures lead to piling security debt and that vulnerabilities getting embedded in a spaghetti code are hard to separate from dependencies and code-sharing, causing much pain, and remaining open for a long time without being remediated. Looking at some hard facts, a case can be made for creating a vulnerability map that spans across all business domains, across all applications giving a topology of vulnerability spread. Without this view, it will be hard to emphasize the risks to the organization due to this expansive vulnerability prevalence across the organization. First, build an effective metrics program for analysis and action, to see the lay of the land, and to understand the net risk. Net risk is a

relative term to define what remains as risk after covering for all known risks through technology, process, and personnel awareness. The theme is that security is a mindset and not an option, debunk the myth that measurement is difficult, asking if what is measured makes sense. Then look at the layers of metrics that are shared with every level of the organization – board metrics, management metrics, and operational ground truth metrics. This comprehensive metrics program will throw light on what lies beneath, and this will be quite a revelation, and therefore it is worth looking at how to build a metrics program [18].

3.1.1. Security Is a Culture, a Mindset

Secure by design must be a data driven conversation to create the space for an effective remediation program. The emphasis should be that security should be a mindset, an embedded feature in a development activity, not an after-thought with reactive chaotic fixes to make up for a lost brand name or equity.

Ignoring secure-by-design principles has resulted in new flaws being introduced by developers inadvertently, besides already existing flaws. Code sharing has benefits if good code is passed around to share common features, but if bad code is being shared it further results flaw-multiplication effect. Remediation and fix rates with the newly introduced flaws reveal that applications have accumulated flaws over time exponentially. Many applications have had no flaws introduced at all, but this could be no scans done in the past, and no scans could be due to no new code being written at all or nothing has changed. Observing code size data helps determine whether there has been a change. The fact that there are

no flaws does not mean no new code was written and this is where code size helps to determine if there was a change and a scan was missed. Integrating application code scanning into the Continuous Integration / Continuous Development (CI/CD) pipeline via API scanning reduces the probability of vulnerabilities being introduced by 2% on average. However, an increase in the application size by 10% has 0.6% chances of introducing one or more new flaws. Applications with higher security debt (as measured by flaw density of one flaw per one mb of code) have a higher correlation to introduce more flaws in future. The more you scan, the fewer newer vulnerabilities are found. So, frequency of scans is important to ensure that we constantly monitor introduction of newer vulnerabilities that adds to the security debt [1].

3.1.2. Running an Effective Metrics Program

All threats are not equally formed and do not require the same response. However, understanding the anatomy of a threat and its genesis is important, and this requires elaborate monitoring, detection, and validation. In the recent decade, the same threats have remained as the top trends: malware, phishing, and credential abuse. Any amount of funding may seem insufficient to meet all the needs for catching these in time to prevent an attack and most breaches still find their origins in one of these threats. The question is how much is enough, and what is the extent of measures and metrics that are needed to make a true analysis to prevent a threat? Table 2 gives an exhaustive list of operational metrics that can provide a deeper insight to catch the hidden threat. These operational metrics can then be used to derive the board and management metrics as appropriate,

Table 2. Ground Truth Metrics.

Questions that a Ground Truth data collection team should be asking	Responses	Comments
What's in scope, what's out of scope?	Include only software applications, do not include infrastructure application as part of the scope in this context.	Include internally accessible applications to take cognizance of insider threat.
What's external internet facing and what's internal facing?	Web applications that are externally and internally accessible.	All customer facing application are called external facing, and most applications catering to internal resources are called as internal facing.
Where was this scanned or tested?	Testing can be of applications hosted on physical on-premises hosts and those hosted on a vendor cloud.	Some scan engines of application code exist in a vendor cloud [1].
What are tools used?	SAST, DAST, SCA, Pen test, WAF, API tests, and External Rating Agencies	[1-9, 14-16, 19-23].
What are the applicable standards?	BSIMM and SAMM, [12], discussed in a later section.	One can engage consulting firms to do an evaluation of current state and identify gaps to recommend improvements.
How many critical and high flaws did we	Measure of flaws by severity is an	Discovered through all the tests and scans, this helps in

Questions that a Ground Truth data collection team should be asking	Responses	Comments
find?	important metric to assess risk of exposure.	reporting application weaknesses to the development teams
How many were carried over from last scan?	This gives us a measure of flaw aging, of how long have these flaws been in existence without remediation.	The ageing of flaws is a clear indication of time spent on remediation. Critical and high severity flaws remaining in the code needs to be analysed for dormancy, of whether they were found recently or were carried for a long time from previous code versions.
Overall, how many critical and high are pending resolutions?	Gives an indication of risk and impact	Set a threshold for risk acceptance and request a timeline or a compliance window for remediation beyond which these must be reported and escalated.
What is the remediation timeline?	Gives a timeline or a plan to remediate before the grace period for compliance expires.	If a remediation timeline is not given it means the flaws are not getting the right visibility and this needs to be reported.
Has something gone out of grace and out of compliance?	All out of grace period flaws become non-compliant per the accepted policy.	This needs to be handled as an issue for remediation.
Are there critical issues still unresolved?	Needs escalation for leadership visibility on the risks these issues pose.	No further deployment of the code can be allowed when the issue is past due for remediation.
What is the reporting format and frequency?	Electronic, dashboards, visualization, and analytics.	Can be daily, weekly, monthly, or even quarterly depending upon the value of information.
What are the data sources?	All the data that come from each test, scan, monitoring, and rating tools	A composite data report needs to be presented for an overall assessment of the security posture.
Who is the responsible party to have found, and who is to remediate?	Everyone involved with the application.	Security is everyone's responsibility.

3.2. Layers of Metrics as Applicable

The information required at each level of the organization is different and must be addressed appropriately to gain organizational support. The Board of Directors metrics are driven by the Information technology (IT) Leader, Chief Information Officer (CIO) and Chief Information Security Officer (CISO). The management metrics are driven by the IT Manager, and the ground truth metrics are driven by the IT Leads at the operational level. Each is discussed briefly below, and these become the basis for developing a maturity model based on a standard industry best practice. A comparison of Building Security In Maturity Model (BSIMM) and Software Assurance Maturity Model (SAMM), two commonly used models in Information Security is explained in [12]

3.2.1. Ground Truth "Metrics" – The Operational Metrics

Gather the operational metrics to begin with, which are the most detailed, Table 2. The board metrics and the management metrics derive the key metrics and key performance indicators (KPI) from the operational metrics, layered in such a way that the appropriate level of leadership can get the state of security posture and the associated risks in the organization.

3.2.2. Management Metrics – Presented to Chief Information Security Officer (CISO)

A mature cybersecurity program requires formalizing the organization's ability to measure and report cybersecurity performance. With an ever-expanding attack surface, and a growing digital sphere that encompasses all types of mobile and wearable technology networked into the internet, IT Leaders, especially the CISO, today are under immense pressure to demonstrate their value beyond just reducing risk with an added requirement to validate their security plans with simulation exercise and show that everything worked as per the plan. Thus, the management metrics are a subset of the operational metrics, with key emphasis on the various security programs that are in place and to demonstrate how effective security governance is.

3.2.3. Board Metrics – Presented to the Board of Directors

The Board looks at the risk to an organization from multiple angles, and what that could mean to the organization overcoming the risks, protecting the brand name, and the business eventually. A look at past issues, strategies, and controls will reveal the adequacy or the opposite so newer strategies and

controls could be evolved. They also look at the success and failure rates of the current measures that are in place and make a blunt assessment of organization's vulnerable areas. The security organization's governance practices highlighted through a dashboard, possibly adding external ratings, assessments, and score by which the organization can compare itself with peers. All of this provides red-flags and vulnerability alerts for addressing issues that demand immediate attention. Many organizations conduct simulation exercises of a threat and study the consequence and response to see how effective Business Continuity and Disaster Recovery are addressed. To that effect, the Board will look at certain metrics and Key Performance Indices (KPI) that reflect the organization's preparedness and security posture. The board metrics is an even high-level metrics, a subset of the management metrics, which provides a risk statement. These metrics put in spotlight the real pain points and the efforts underway to overcome the pain points, to assure the board of directors that the organization is headed in the right direction.

4. An Attack Tree as a Threat Model and Its Benefits

Some key questions the board presents to the IT leadership and the CISO is by asking,

1. If they have done a simulation of an attack.
2. How did the simulation run?
3. Did they find a pathway for an attack?
4. What are the control gaps identified?
5. What is the remediation plan to address those gaps?
6. When is the next simulation run?

4.1. Threat Modeling Process

The simulation and its effectiveness are a continuously improving iterative study, which has now become a mandatory exercise for all organizations. From a purely IT perspective, simulations cannot be enacted without understanding the following [24]:

1. Identification of resources to be protected. They cover the three tenets of security – confidentiality, integrity, and availability, commonly termed as CIA.
2. Documenting the technical architecture that defines the entire system and the technologies involved, and application architecture that defines the business function of the application.

3. Defining the security profile of the applications that details trust boundaries, data flow, identify the entry points of the application into the network, identifying privileged code, and the overall security architecture of the application.
4. Identifying the threats associated with architecture. The MITRE ATT@CK Enterprise Matrix lists 266 threat patterns [25], and most organizations look for those threat patterns and attacks that are spoofing, tampering, repudiation, information disclosure, denial of service (DoS), and elevation of privilege. However, application security vulnerabilities that abound in plenty in the systems, are often viewed as an afterthought, most common of those are mentioned in section 2.1 of this paper and listed below for convenience.
 - 1) Cross-site scripting – XSS
 - 2) Injection / SQL Injection – SQLI
 - 3) Path Traversal / Directory Traversal
 - 4) Command Execution
 - 5) Backdoor
5. Documenting the threats and their attributes
6. Assessing threats, scoring the risk and impact, and assigning a cost in case of a breach.

There are two ways of applying a threat model [24]:

1. To design a system based on the security architecture with the intent to protect assets. The emphasis here is to embed security measures to protect all the assets that are perceived as a potential target of a cyber-attack and are at risk of a costly breach.
2. To view all the pathway for attacks and assets being exposed along that every path of those attacks. In essence this forms an attack tree.

Three types of threat models are in use in general:

1. Attacker-centric method, which is focused on the attacker and the source of attack.
2. Software-centric method, where focus is more on the data flow and the boundaries or the edges where an application belonging to a particular domain hand off data to the next application sitting on the other side of the trust-boundary. This is where the vulnerabilities of one application leads to spread of an attack into the next application, at the edges called as common nodes of traffic.
3. Asset-centric methods, the key assets where everything critical to an organization's business operation begins and ends, from servers, databases, and data itself.

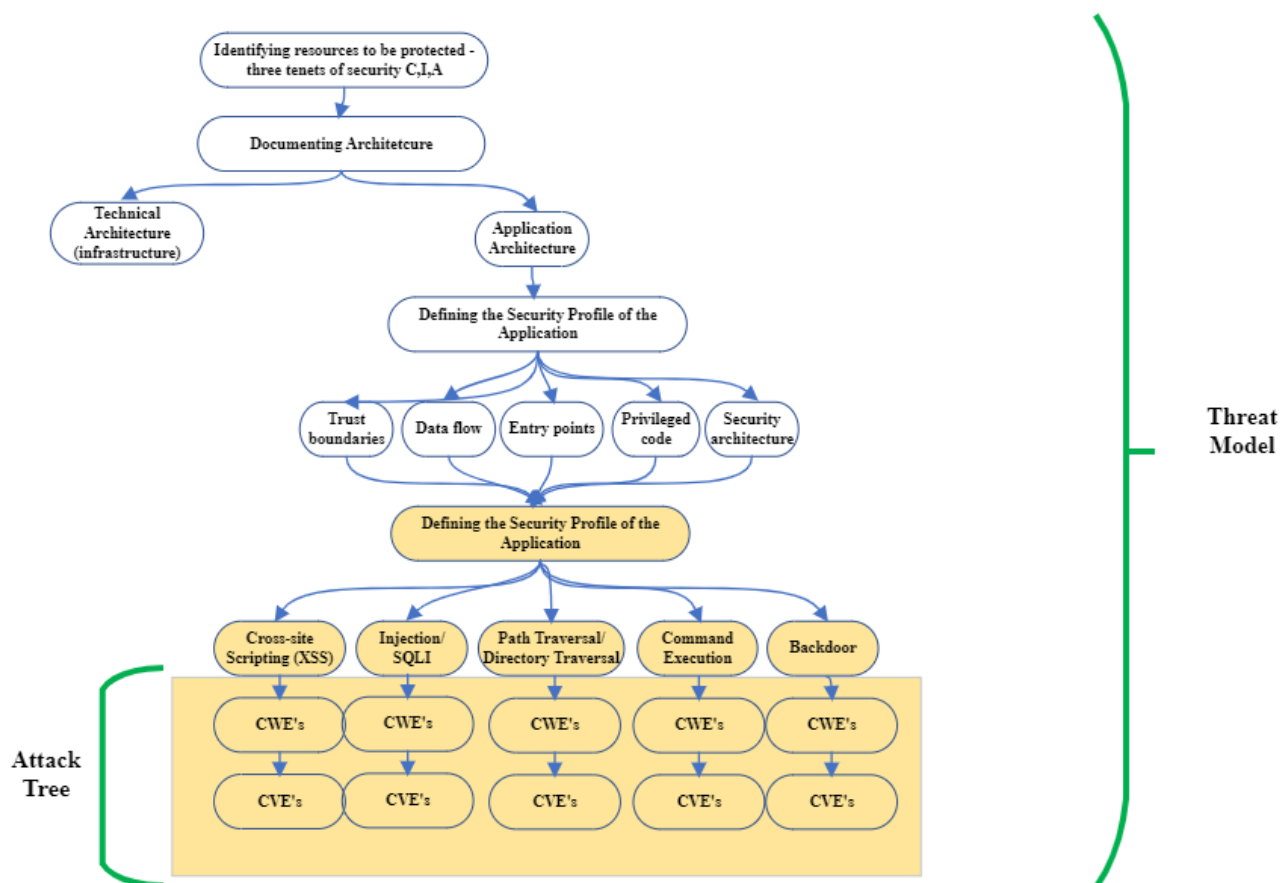


Figure 1. Threat model and attack tree.

4.2. Software-Centric Attack Tree Method

The focus of the present research is the area of application security that does not receive enough visibility. The first step for developing an effective threat model is to look at all vulnerabilities in the systems, as suggested in the software-centric method. A comprehensive threat model should look at infrastructure-based and application-based vulnerabilities. While most threat models are infrastructure-centric looking at exposed ports, servers, end points, and devices, much attention is needed in the application security space especially considering the security technical debt discussed in section 3.1 of this paper. The next step in the process is to develop an attack tree to look at how vulnerabilities are spread across the enterprise. Figure 1 shows the threat model process and the attack tree at the bottom. Web applications have become target of attacks in recent times, and this is a growing issue in cybersecurity. Thus, initial access is very common, and then once in discovery phase when many vulnerabilities are found within an application, the attacker tries to make a lateral move as applications are linked by business functions and shared code. Vulnerabilities existing in one can be linked to those found in other applications and attacker can visualize that a vulnerability spread across application topology can

transmit an attack and spread it laterally which provides the attacker multiple pathways for attack.

When dealing with applications that are stacked with technical debt, and that it has security implications due to unremediated vulnerabilities, the threat actor is looking at the inter-linking of these vulnerabilities as a source-to-target vulnerability lineage, all linked through like a chain, and clearly sees multiple pathways for an attack even across domains and trust boundaries. With hundreds of vulnerabilities, it is not easy to identify specific threat patterns and pathways to a precise attack that the threat actor is planning without developing a vulnerability-based threat model and an attack tree resembling a vulnerability map. The attack tree not only helps in focusing on the possible pathways to attack, but it also helps in creating simulation exercises along the same path as identified by an attack tree. The result of the simulation helps in prioritizing remediation efforts to eliminate the most vulnerable attack paths. The result of simulation also exposes inadequacies in coverage of monitoring and detection, gaps in controls, and a security technical debt that could be the root cause of much of the pain points. While a metrics program reports issues, a simulation exercise following an attack tree shakes up the organization to prioritize remediation efforts.

5. Conclusions

This paper has discussed an approach to use correlation analysis and vulnerability metrics to define prioritization of remediation using a threat model analysis and vulnerability attack tree. The attack tree will show multiple pathways for an attack to shape using vulnerability linkages. By further relating to a parent, peer, or child CWE (including CWEs that follow another CWE and in some cases precede other CWEs) will provide more insight into the attack patterns. These patterns will reveal a multi-vulnerability, multi-application attack pattern which will be hard to visualize without data consolidation and correlation analysis. The correlation analysis tied to the test and scan data supports a vulnerability lineage starting from incoming requests to individual vulnerabilities found in the code that traces a possible attack path. The attack tree will give a better visualization of the possible attack pathways in creating simulations of possible attacks. This simulation can force immediate focus on vulnerabilities along the pathway of the most probable attacks. This prompts a call to action to resolve the vulnerabilities as a priority.

Next step in this research is to create an actual attack tree from existing vulnerability, then create an enterprise-wide vulnerability map with domain specific attack trees, with each tree connecting with an adjoining common shared service or group of services and common application or set of common applications between domains that can enable lateral movement of an attack spreading through vulnerabilities. One can also by correlating the incoming WAF requests, combining with a RASP technology, and integrating with an incident response system that comprises of a Security Information and Event Management (SIEM) / Extended / Security Orchestration and Automated Response X/SOAR / Extended Detection and Response (XDR), build a comprehensive threat intelligence system that covers networks, devices, end points, and applications. This will be future work.

Abbreviations

AEH: Application Ethical Hack.
 BSIM: Building Security in Maturity Model
 CIA: Confidentiality, Integrity, and Availability
 CIO: Chief Information Officer
 CISO: Chief Information Security Officer
 CI/CD: Continuous Integration / Continuous Development
 CVE: Common Vulnerabilities and Exposures
 CWE: Common Weakness Enumeration
 DAST: Dynamic Analysis Security Testing
 OSS: Open-Source Software
 OWASP: Open Worldwide Application Security Project
 RASP: Runtime Application Self-Protect
 SAMM: Soft-ware Assurance Maturity Model
 SAST: Static Analysis Security Testing
 SCA - Software Composition Analysis
 SIEM: Security Information and Event Management

SCA: Software Composition Analysis
 SOAR: Security Orchestration, Automation, and Response
 SQLI: SQL Injection
 XDR: Extended detection and Response
 XSOAR: Extended Security Orchestration, Automation, and Response
 XSS: Cross-Site Scripting
 WAF: Web Application Firewall

Acknowledgments

This research has been supported and guided by Dr. Xiaolong Li of Indiana State University, USA, and Dr. John Pickard and Dr. Peng Li of East Carolina University, USA.

Dr. Xiaolong Li is a professor in the Department of Electronics and Computer Engineering Technology at Indiana State University. He received his PhD in Computer Engineering from the University of Cincinnati in 2006. His primary areas of research include modeling and performance analysis of MAC protocol, Internet of Things, Wireless Ad Hoc networks, and sensor networks.

Dr. John Pickard is a professor of Information and Cybersecurity Technology at East Carolina University, North Carolina, USA. He received his PhD in Technology Management from Indiana State University in 2014. His main research areas are internet protocols, convergence of information and operations technologies, and Internet of Things applications.

Dr. Peng Li received his Ph.D. in Electrical Engineering from the University of Connecticut. His professional certifications include CISSP, RHCE and VCP. Dr. Li is currently an Associate Professor at East Carolina University. He teaches undergraduate and graduate courses in programming, computer networks, information security, web services and virtualization technologies. His research interests include virtualization, cloud computing, cybersecurity, and integration of information technology in education.

Conflicts of Interest

The authors declare no conflicts of interests.

References

- [1] Veracode. State Of Software Security Vol. 11. *Veracode*, [state-of-software-security-volume-11-veracode-report.pdf](https://www.veracode.com/resources/state-of-software-security-volume-11-veracode-report.pdf)
- [2] Checkmarx. Correlation: The Application Security Testing Imperative in Modern Application Development”, *Checkmarx*, https://www.forrester.com/report/the-forrester-wave-software-composition-analysis-q3-2021/RES176091?ref_search=3502061_1674835391293&utm_source=PANTHEON_STRIPPED&utm_medium=email&utm_campaign=summit21na&utm_content=blog&category-id=a89c0000000AKp1AAG%3Futm_source%3DPANTHEON_STRIPPED

- [3] Carielli, S., DeMartine, A., Provost, A.C. and Dostie, P. The Forrester Wave™: Software Composition Analysis, Q3 2021-The 10 Providers That Matter Most And How They Stack Up. *Forrester*, August, https://www.forrester.com/report/the-forrester-wave-software-composition-analysis-q3-2021/RES176091?ref_search=3502061_1674835391293&utm_source=PANTHEON_STRIPPED&utm_medium=email&utm_campaign=summit21na&utm_content=blog&categoryid=a89c000000AKp1AAG%3Futm_source%3DPANTHEON_STRIPPED
 - [4] Primeon. Enterprise Applications: Wide Open to Attack in 2018. *Primeon*, https://www.primeon.com/whitepaper/primeon_wp2_r.pdf?1
 - [5] Akamai. Slipping Through the Security Gaps: The Rise of Application and API Attacks. *Akamai*, <https://www.akamai.com/blog/security/the-rise-of-application-and-api-attacks>
 - [6] Carielli, S., DeMartine, A., Provost, A.C. and Dostie, P. The Forrester Wave™: Web Application Firewalls, Q3 2022, The 12 Providers That Matter Most And How They Stack Up. *Forrester*, September, <https://www.forrester.com/report/the-forrester-wave-tm-web-application-firewalls-q3-2022/RES176396>
 - [7] Signal Sciences. Identifying Web Attack Indicators. Available from: <https://info.signalsciences.com/rs/025-XKO-469/images/signal-sciences-white-paper-identifying-web-attack-indicators.pdf>
 - [8] FASTLY. 10 Key Capabilities of the Fastly Next-Gen WAF. *FASTLY*, 2022, <https://learn.fastly.com/security-10-key-capabilities-of-fastlys-next-gen-waf.html>
 - [9] Na, J. Introducing Secure Application: True Runtime Application Self-Protection (RASP) for the Modern Application. *CISCO App Dynamics*, <https://www.appdynamics.com/blog/product/application-security/>
 - [10] Brumfield, C., & Haugli, B. Cybersecurity Risk Management: Mastering the Fundamentals Using the NIST Cybersecurity Framework. *Wiley*, ISBN: 978-1-119-81628-7.
 - [11] Geer, D. E. Jr., & McClure, S. (2016). How to Measure Anything in Cybersecurity. *John Wiley & Sons*, 2016, ISBN 978-1-119-08529-4.
 - [12] Glas, B. (2020). Comparing BSIMM & SAMM: Building Security In Maturity Model (BSIMM) compared to Software Assurance Maturity Model (SAMM). <https://owaspsamm.org/blog/2020/10/29/comparing-bsimm-and-samm/>
 - [13] Kasturi, S., Li, X., Pickard, J., and Li, P. Understanding Statistical Correlation of Application Security Vulnerability Data from Detection and Monitoring Tools. *2023 33rd International Telecommunication Networks and Applications Conference*, Melbourne, Australia, 2023, pp. 289-296, <https://doi.org/10.1109/ITNAC59571.2023.10368476>
 - [14] MITRE. 2023 CWE Top 25 Most Dangerous Software Weaknesses, *CWE - 2023 CWE Top 25 Most Dangerous Software Weaknesses (mitre.org)*
 - [15] OWASP. OWASP Top 10. OWASP. <https://owasp.org/Top10/>
 - [16] MITRE. Common Vulnerabilities and Exposures (CVE) Numbering Authority (CNA) Rules. *MITRE*, https://cve.mitre.org/cve/cna/CNA_Rules_v2.0.pdf <https://nvd.nist.gov/vuln>
 - [17] Warner, R. M. (2020) Applied Statistics – II Multivariable and Multivariate Techniques. *SAGE Publications*
 - [18] Christopher, J. D. How to Mature ICS Security with Metrics. *Industrial Control Systems Security*, 2022. <https://www.sans.org/blog/mature-ics-security-with-metrics/>
 - [19] OWASP-API. OWASP API Security Top 10, OWASP. <https://owasp.org/API-Security/editions/2023/en/Oxa2-broken-authentication/>
 - [20] Veracode. State Of Software Security Vol. 10. *Veracode*, <https://www.veracode.com/sites/default/files/pdf/resources/ssre-ports/state-of-software-security-volume-10-veracode-report.pdf>
 - [21] Veracode. State Of Software Security Vol. 12. *Veracode*, <https://www.veracode.com/sites/default/files/pdf/resources/ssreports/state-of-software-security-v12-nwm.pdf>
 - [22] SALT. State of API Security Q1 2023. *SALT LABS*, https://content.salt.security/rs/352-UXR-417/images/SaltSecurity-Report-State_of_API_Security.pdf
 - [23] Morgan, S. (2021). 10 Hot Security Ratings Companies To Watch In 2021. *Cybercrime Magazine*, <https://cybersecurityventures.com/security-ratings-companies/>
 - [24] Hajrić, A., Smaka, T., Baraković, S., and Husić, J.B. Methods, Methodologies, and Tools for Threat Modeling with Case Study, *Telfor Journal*, Vol. 12, No. 1, 2020, <https://scindeks.ceon.rs/Article.aspx?artid=1821-32512001056H>
- Xiong, W., Legrand, E., Aberg, O., and Lagerstrom, R. Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix. *Software and Systems Modeling* (2022) 21: 157–177 <https://link.springer.com/article/10.1007/s10270-021-00898-7>