Software Engineering Tools For Secure Application Development

Divya Bellamkonda

Software Engineering Tools For Secure Application Development

by

Divya Bellamkonda

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science in

Information Assurance

May 2023

Abstract

Software security has become a crucial part of an organization’s overall security strategy due to increasingly sophisticated attacks at the application layer. One of the major concerns in software engineering is the inadequate use of secure software development methods and tools. Such deficiency is caused by a lack of knowledge and training on available secure tools among software developers. This project conducts a thorough investigation of the tools that can be used by developers throughout the software development life cycle to assist in the development of secure applications, including tools used by individuals and teams, classified by open-source or commercial, tools based on project size, etc. This paper also includes a summary table that provides a quick overview of all the tools listed for developers and individuals to use.

Acknowledgements

I would like to express my sincere gratitude to my chair, Dr. Jim Chen, Ph.D., for his invaluable patience and feedback. I am extremely grateful for his encouragement, prompt response, and sense of humor. I could not have embarked on this journey without the help of my defense committee, who generously shared their knowledge and expertise.

I am also grateful to my classmates and cohort members. Finally, I would be remiss if I did not mention my family, especially my parents and brother. Their faith in me has kept my spirits and motivation high throughout this process.

Table of Contents

List of Tables

List of Figures

Chapter I: Introduction

Introduction

It is important in today's world for software developers to have a thorough understanding of secure tools and systems available in the public domain in order to prevent and protect systems, sensitive data, and networks from attacks. Though many secure development tools are available, there is a lack of systematic study on the tools. Software developers face the challenge of selecting proper tools and learning their complex features. There is a need to build a comprehensive taxonomy to assist software developers in selecting the most appropriate tools for their projects.

Problem Statement

Due to application security misconfigurations, flawed designs, and insecure deployments, lack of information on existing secure tools, and a lack of security awareness training, developers were unable to select the right tool for adding security features to the project as they were mostly involved in delivering the tasks and could not recall the appropriate tools used for secure software development, while attackers attempted to identify software weaknesses and exploit the vulnerability in the system.

Nature and Significance of the Problem

LogRhythm, the company powering today's security operations centers (SOCs), announced the release of its report, *The State of the Security Team: Are Executives the Problem?* The surprising primary findings include 93% of security professionals lack the tools to detect known security threats, and 92% state they are still in need of the appropriate preventative solutions to close current security gaps (*LogRhythm*, 2020).

According to a Ponemon survey (Wike, 2016), 64% of security leaders and directors feel that they lack the tools and resources necessary to monitor external threats. Sixty two percent report a lack of tools and resources to analyze and understand those threats, and 68% report a lack of tools and resources to mitigate external threats.

Objective of the Study

The goal of this study is to create a taxonomy of secure software tools which are useful for developers in developing secure software throughout the software development lifecycle. To achieve this goal the following methods are followed:

- Investigate comprehensive literature on the existence of any current taxonomy in Google Scholar to see what others have done to solve the problem.
- Identify tools based on their security features.
- Classify various tools used during the secure application development based on the number of users that the tool supports the development teams, open source and commercial to better understand the software engineer's expectations and needs.

Study Questions

Is there any similar taxonomy to assist developers in choosing tools to add security features throughout the development process?

Does the developed taxonomy really help developers in identifying the appropriate tool for their project?

Definition of Terms

Cybersecurity attack: Any form of malicious activities that targets IT systems and their users, to gain unauthorized access to the systems and the data or information they contain (*What Is a Cybersecurity Attack?*, 2022).

Design flaw: A design that fails to meet requirements or to serve customer needs resulting in unstable and unusable products, services and environments (Spacey, 2017).

Dynamic code analysis: It is also known as Dynamic Application Security Testing (DAST), and is used to test a running application for potentially exploitable flaws. DAST tools can detect compile-time and run-time vulnerabilities, such as configuration errors that appear only in a realistic execution environment (*What Is Dynamic Code Analysis?*, n.d.).

Open source: It refers to a software programme or platform with easily accessible source code that can be modified or enhanced by anyone.

Requirement elicitation: An activity of the Requirements Engineering process that aims at identifying requirements through intense communication among stakeholders and analysts for the development of projects (*What Is Requirements Elicitation*, 2022).

Security: It is the process of defending critical systems and sensitive data against cyber-attacks.

Security misconfiguration: Inadequate security controls, such as those used for servers or application configurations, network devices, and so on, can lead to security vulnerabilities (*The Impact of Security Misconfiguration and Its Mitigation*, 2020).

*Software design***:** It is a mechanism that converts user requirements into a suitable form, which helps the programmer in software coding and implementation (Javatpoint, 2021).

Software development: Software development, also known as the Software Development Life Cycle (SDLC), is a process that programmers use to create computer programmes. It consists of various phases that give a mechanism for creating products that meet technical specifications and user needs (*What Is Software Development*, 2021).

Software Engineering: It is a branch of engineering that deals with the development of software products (*Whitepaper*, n.d.).

Static code analysis: It is a method of debugging that involves examining source code before running a program. It is accomplished by comparing a set of code to a set of coding rules (Bellairs, 2022).

Threat: Any circumstance or event that has the potential to harm an information system through unauthorized access, data destruction, disclosure, data modification, and/or denial of service. Threats arise as a result of both human actions and natural events (Stoneburner et al., 2004).

Vulnerability: It refers to any weakness in an information system, system processes, or internal controls of an organization ("What Is Vulnerability in Cyber Security?," 2021).

Summary

In this chapter the abstract, problem statement, nature and significance of the problem are described well and the objective for creating a taxonomy for secure software development tools was clearly stated. Study questions are also provided to indicate the outcome of the research done on the secure software development tools. The definitions of the terms were provided at the end to provide a thorough understanding of the topic.

Chapter II: Background and Review of Literature

Introduction

In this chapter, the background and review of literature are clearly stated, with a focus on software vulnerability issues, and the literature linked to the problem is illustrated using statistics and numbers centered on the nature and significance of the problem. Based on prior work in the topic, the literature linked to the methodology is also described.

Background Related to the Problem

According to Stoneburner et al., (2004), a vulnerability is a weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source. This can happen when programmers inadvertently or purposefully leave an exploitable bug in software. End users frequently fail to update their software, leaving it unpatched and vulnerable to exploitation.

Examples of vulnerabilities include:

1. Hardware

   Humidity, dust, soiling, natural disasters, weak encryption, or firmware vulnerability are all factors to consider.

2. Software

   These are some of the software vulnerabilities which include inadequate testing, a lack of audit trails, design flaws, memory safety violations (buffer overflows, over-reads, dangling pointers), input validation errors (code injection, cross-site scripting (XSS), directory traversal, email injection, format string attacks, HTTP header injection, HTTP response splitting, SQL injection).

3. Network

Man-in-the-middle attacks, unsecured network architecture, lack of authentication or default authentication are all examples of vulnerabilities.

4. Personnel

Lack of security awareness and training, poor adherence to security training, bad password management, or downloading malware via email attachments are all examples of inadequate recruiting practices.

5. Physical location

Natural disaster-prone areas, uncertain power sources, or no keycard access are also listed as vulnerabilities which may cause disruption to daily activities.

6. Security strategy

Having no audit, continuity plan, security, or incident response strategy in place within the organization may also lead to vulnerabilities.

Literature Related to the Problem

According to LogRhythm (2020), LogRhythm's Global Study, security teams' stress levels are rising due to a lack of appropriate tools and executive support, impeding their ability to combat threats. Sixty-eight percent of respondents said their company had deployed redundant security tools, with 56% admitting the overlap is unintentional, highlighting the need for better strategic control from management once again. When asked what more help their security programmes require, despite duplicative tools, 58% of respondents answered they still require increased funding for tools. As a result, the report emphasizes the growing need of IT consolidation. Security professionals rate the value of solution consolidation highly, citing top benefits as less maintenance (63%), faster issue detection (54%), identification (53%), and resolution (49%), as well as lower costs (46%) and greater security posture (45%). Despite this, just one out of every three businesses (32%) have a real-time security dashboard that provides a clear, integrated view of all their security solutions (LogRhythm, 2020).

According to a recent Ponemon survey (Wike, 2016), 64% of security leaders believe they lack the necessary tools to monitor external security threats. According to Security Beyond the Traditional Perimeter (*The PhishLabs Blog,* n.d.), a report from Brand Protect and the Ponemon Institute found security leaders have doubts about their organizations' ability to monitor for outside threats.

In a press release, Larry Ponemon (Ponemon Institute, 2016), head of the Ponemon Research Institute, stated, "The majority of security leaders recognise that these external internet dangers jeopardize business continuity. The research reveals a gap in defenses against threats that have shown to be incredibly efficient for cyber criminals while being extremely costly for businesses."

According to *Ponemon Institute*(2016), 79% of respondents believe their defensive architecture to identify and mitigate threats is either non-existent, ad hoc, or inconsistently deployed across the company. Furthermore, 59% of respondents believe that protecting intellectual property from external threats is critical or extremely vital to their company's long-term sustainability. Respondents said their companies had an average of 32 material cyber-attacks every month, or little more than one per month, costing an average of $3.5 million per year. Leaders also believe that monitoring the internet and social media is crucial for gaining intelligence on external dangers. (*Ponemon Institute*, 2016) gave the following as the most important monitoring priorities:

- mobile app monitoring (cited by 62% of respondents)
- social engineering and organizational reconnaissance (61% of respondents)
- branded exploits (59% of respondents)
- spear-phishing infrastructure (58% of respondents)
- executive and high value threats (54% of respondents)

According to Kinzer (2021), Facebook data breach exposed over 533 million individuals' personal information to hackers. The user's name, date of birth, current city, and wall posts were all exposed. A white hat security group discovered the vulnerability in 2021, and it has existed since 2019. This incident occurred when cybercriminals scraped data from Facebook's servers by exploiting a bug in their contact importer. As a result, they could acquire access to millions of people's personal information. Facebook later identified this as an external attack, however the core cause of this breach and others like it comes from a common scenario: misconfiguration problems. What makes these breaches so dangerous is how quickly they may proliferate. Facebook is not the only company experiencing security difficulties as a result of misconfiguration. Many security organizations report an alarming growth in this type of vulnerability, especially with the rise of cloud computing.

According to Chickowski (2017), Fiat Chrysler Automobiles (FCA), which had to recall 1.25 million Dodge Ram pickup trucks owing to a potentially catastrophic fault in its onboard computer software, was one of the most prominent examples of life-or-death software quality issues in 2017. According to Fiat Chrysler Automobiles, the issue has resulted in at least one death and two injuries. The manufacturer also recommended owners to keep an eye out for a warning light on the instrument panel that indicates a possible malfunction. The recall applies to three separate light and medium-duty variants of the full-size Ram pickup sold in North America and other parts of the world during the 2013 and 2016 model years, according to FCA. This figure comprises 1.02 million 2013-16 Ram 1500 and 2500 pickups, and 2014-2016 Ram 3500 trucks sold in the United States, along with 216,007 vehicles sent to Canada, 21,668 to Mexico, and another 21,530 sold outside North America (Eisenstein, 2017). The error leads the truck to disable airbags and seatbelt pretensioners during a rollover crash if the vehicle collides with something in its undercarriage while traveling off-road—exactly the condition in which a driver is probably most likely to roll over (Eisenstein, 2017). If the software problem occurs, owners will receive a warning on the instrument panel. According to the firm, drivers can temporarily resolve the issue by turning off and on their vehicles. If the warning light is turned off, the crash system is back in action. Concerning the Ram recall, FCA was made aware of the issue in December when it was mentioned in a lawsuit involving a 2014 Ram 1500 crash. When the pickup toppled over, the side airbags failed to deploy. According to *Takata Airbag Recall* (2023), Takata, a Japanese supplier, has been at the forefront of the most serious airbag problem. As a result, the greatest automotive recall in history has occurred, affecting up to 70 million airbags used in over 50 million vehicles globally. The Takata problem has now been linked to at least a dozen deaths in the United States, as well as several other fatalities worldwide.

Chickowski (2017) reported on Cloudflare, a content delivery network vendor, that exposed sensitive data from some of the biggest brands on the web, including Uber, Fitbit, and OkCupid, due to a memory leak bug introduced in its codebase. The bug, which had been in production for more than five months, potentially exposed consumers' passwords, cookies, and authentication tokens. The flaw was in Cloudflare's HTML parser, which exposed 1 in every 3.3 million HTTP requests. That amounted to up to 120,000 leaks per piece of exposed data in a single day. Tavis Ormandy, a security engineer with Google's Project Zero, discovered the fault and reported it to Cloudflare. The problem has been affecting Cloudflare's systems since September 2016, although the most significant damage happened between February 13 and February 18, 2017 (Cumming, 2017). According to Cloudflare's incident report, the initial mitigation took 47 minutes, and the problem was repaired in less than seven hours. Cloudflare has collaborated with search engines such as Google, Yahoo, and Bing to remove the data from their caches.

In September 2017, Equifax, one of the three main consumer credit reporting agencies in the United States, announced that its systems had been breached, compromising the sensitive personal information of 143 million Americans (Electronic Privacy Information Center, 2017). The data breach included names, home addresses, phone numbers, dates of birth, social security numbers, and driver’s license numbers. The credit card numbers of approximately 209,000 consumers were also breached. The Department of Homeland Security notified Equifax, Experian, and TransUnion of the vulnerability that occurred.

On March 9, 2017, Equifax administrators received an internal email informing them that the Apache fix had been applied. On March 15, 2017, Equifax's information security staff launched scans to discover systems vulnerable to the Apache Struts issue, however the scans failed to detect the vulnerability. The Equifax breach is unprecedented in scope and severity. The vulnerability that caused the breach was by not updating Apache Struts CVE-2017-5638. Apache Struts is a popular framework for creating Java Web applications maintained by the Apache Software Foundation. The Foundation issued a statement announcing the vulnerability and released an update later.

According to Kinzer (2021) a ransomware attack hit the US-based Colonial Pipeline in May. The corporation owns and maintains a huge pipeline that transports gasoline and other petroleum products from Texas to New Jersey and throughout the Midwest. On April 29, attackers got access to the company's network via a Virtual Private Network (VPN) account with a single compromised password. While operational technology systems were unaffected, the event prompted the company to halt gasoline flow in its mainline as a precaution (and to shut down leaks). This resulted in fuel shortages and rising fuel prices in the Southeast, Midwest, and Northeast parts of the country, with drivers panic-buying at the pump.

The attackers allegedly promised more hacks unless Colonial paid them $5 million in bitcoin — an amount that at the time was more than three times their annual income. The ease with which the hackers gained access to the system is what makes this attack so concerning — it has subsequently been found that the organization did not implement multi-factor authentication (Kinzer, 2021). According to the firm, after a six-day shutdown, pipeline operations restarted on May 12, with all systems and procedures restored by May 15. The FBI launched an investigation three days after the first claims surfaced on social media. It's probable that an insider was responsible for reducing security measures by revealing VPN credentials; however, exactly how attackers acquired access to those credentials remains unclear. This incident exemplifies why it is critical for businesses, particularly those managing sensitive data such as oil pipelines, to have effective cybersecurity protections in place. Multi-factor authentication (MFA) is one such method, and it is becoming more popular among businesses. While Colonial Pipeline chose to pay the ransom demand of around $5 million, close to 50% of the funds had been recovered by June (Perlroth, 2021).

A major vulnerability affecting the popular Java logging package Log4j was revealed in late November (Cimpanu, 2021). It is a remote code execution vulnerability that allows an attacker to take complete control of a system. Shortly after the vulnerability was exposed, malicious actors from all over the world launched a large flood of scanning and exploitation attempts across the internet. Log4Shell is a vulnerability in Log4j (*Log4j – Apache Log4j Security Vulnerabilities*, n.d.), a Java package used to add logging features to Java web and desktop applications. It is controlled by the Apache Applications Foundation, which means it is included in most of their software, and it also has a "seal of high quality code" that makes it a favorite with most enterprise software developers . The vulnerability occurs in programs where user input can result in a log entry, for instance, in apps with input fields or where users can control the text submitted within the log itself. The notion is that an attacker may generate something like this:

```
${jndi:ldap:/attacker.com/script}
```

The JNDI prefix forces the Log4j library to connect to the attacker's domain and run a script stored there when it writes and parses this entry inside a log. Anyone who has used Log4J between 2.10.0 and 2.14.x is vulnerable to attacks. Later, the Apache Software Foundation released a security upgrade for Log4j 2.15.0, which fixed the attack vector. Setting the log4j2.formatMsgNoLookups option to true in the Log4j configuration also avoids exploitation if companies are unable to update. Companies such as Cloudflare stated that none of their products are affected, however Red Hat and N-able (SolarWinds) stated that they have products that are. However, even for an agency like Cybersecurity and Infrastructure Security Agency (CISA), a comprehensive list of what is vulnerable and what is not is not widely available. The good news for some system administrators is that security startup Huntress Labs has produced a free Log4Shell scanner that businesses may use to analyze their own systems.

The Marriott hotel chain was recently fined approximately $23.8 million in penalties as a result of a 2014 data breach (Marriott Data Breach FAQ, 2020). The financial burden is only the beginning of Marriott's problems. More than 300 million guests' credit card information, passport numbers, and birthdates were compromised in the attack on the brand's global guest reservation database. It's one of the biggest data breaches in history. While insurance covered a large portion of Marriott's financial losses, the company's brand reputation will suffer for a long time. The breach occurred in 2014, but it wasn't discovered until 2018, when an internal security tool detected a suspicious attempt to access Marriott's Starwood brands' internal guest reservation database. Marriott acquired Starwood Hotels in 2016, adding 11 new brands to Marriott International's original 19 assets. Marriott discovered that hackers had encrypted data and removed it from the Starwood system during an internal investigation. That data included information from up to 500 million guest records, some of which were duplicates. Crowdstrike cybersecurity expert Ryan Cornateanu told Hotel Tech Report, "The attack on Marriott was hapless, and a popular entry point for adversaries is through email spoofing”. This tactic is used in phishing to get malware onto a target network, where it can then spread laterally across all systems. From there, hackers can obtain account numbers, driver's license numbers, and other sensitive information from loyalty programs and reservation systems. The general data protection regulation has gone a long way toward protecting consumers, but there is only so much that can be done when a hacker is able to secure login credentials or directly access servers.

Literature Related to the Methodology

The paper published by Novak et al. (2010) describes the most widely used static code analysis techniques which are widely used in software development life cycle. Because there are so many different sorts of these tools for so many different purposes and programming languages, they try to categorize them into areas such as technology, rule availability, supported languages, extensibility, and a variety of other categories and subcategories. Before a program is tested or sent into production, static code analysis tools can be used to detect hidden errors in its implementation. Correction of hidden defects during the development cycle can reduce testing effort, reduce the number of operations required, and lower system maintenance costs. Static code analysis tools can be used in a variety of ways, but they all result in higher software quality. They can also help with a variety of security issues. The ability of static code analysis tools to automatically recognize many common programming errors is their most valuable feature. Unfortunately, implementation errors are only part of the problem. Tools are incapable of detecting design and architectural flaws in programs. They cannot find poorly made cryptographic libraries or inappropriately chosen algorithms, and they cannot highlight design problems that can cause major confusion. They also cannot find any passwords or magic numbers hidden in the code. Another weakness of static code analysis tools is that they are prone to "manufacture" so-called "false warnings" or found errors that are not actually errors. These discovered "mistakes" are frequently referred to as "false positives." This article Novak et al. (2010) provides a taxonomy of static code analysis tools, rather than to show which static code analysis tool is better than the others. Therefore, Novak et al. (2010) looked at some static code analyzers and tried to find common characteristics. This paper, on the other hand, does not discuss dynamic application analysis tools, which perform analysis while the application is running.

According to Weber et al. (2005), although recommendations for static analysis tools to aid software security evaluations or to detect security problems were made three decades ago, static analysis and model checking technology has only recently advanced to the point where such tooling is feasible. Tool builders might benefit from having a taxonomy of software security issues to organize the problem space in order to target their technology on a sensible basis. Unfortunately, the only acceptable taxonomies now available are out of date and do not adequately describe security problems present in modern software. They have created a security flaw taxonomy by combining prior efforts to categorize security flaws as well as incident reports in this article. They compared their taxonomy to publicly available data on current high-priority security threats and drew conclusions from the findings. To be more specific, there are a variety of strategies that can be utilized to handle security issues, each with its own set of benefits and drawbacks. Rather than conducting a cross-comparison analysis of each technique against each taxonomy category, which is impractical, they attempted to identify common characteristics that affect difficulty across sets of analyses and relate these aspects to the taxonomy.

A paper by Hein and Saiedian (2009) highlights developments in secure software systems engineering (SSE), on-going issues, and methods for reasoning about threats and vulnerabilities by assessing and categorizing pertinent SSE research. Several difficult risk assessment/mitigation questions (e.g., "what is the possibility of an attack"), as well as practical concerns (e.g., "where do vulnerabilities arise" and "how can vulnerabilities be averted") are addressed. When the topic of security is brought up in the context of development, an almost reflexive reaction is to consider specialized security features such as cryptography, authentication, and copy-protection, and how the development team might add on or integrate software components providing these security features (McGraw, 1999). The tendency to use add-on security components results from the fact that many of these components 1) frequently implement sophisticated and proven encryption and 2) are frequently prepackaged or provided by the underlying operating system (OS) platform. Better vulnerability prevention and risk assessment are dependent on fundamentally agreeing on SSE-specific classifications and models. Efforts like the Common Weakness Enumeration (CWE) lay the groundwork and provide a common language for sharing information and comparing testing coverage of automated testing and static analysis tools. Although the CWE's current focus is on supporting tools for vulnerability prevention, it can also be used to aid in risk assessment efforts. This paper provided a high-level overview of SSE in an attempt to answer practical questions about the field. The challenge that SSE rises to meet is preventing the introduction of vulnerabilities prior to release, rather than patching vulnerabilities after the fact. Outside of the context of formal methods and automated theorem provers (which typically necessitate significant investment and expertise), proving the security of a typical software product necessitates testing for negative consequences (e.g., there are no vulnerabilities). Because testing for a negative can entail indefinite testing time, it is critical to incorporate security from the start and make cost-effective decisions about when to stop testing. Finally, because of the financial benefits, there is a growing interest in secure software engineering. There is a financial case to be made for implementing SSE processes, principles, and best practices throughout the SDLC, particularly early on, rather than applying patches later in maintenance. Software engineering circles have long recognized that software development/maintenance costs rise over time and that the most effective cost savings are realized with early corrective action; "often a hundred times more cost effective" (Boehm & Basili, 2001, p. 3). Hoo et al., (2001) investigated the software development and maintenance costs (e.g., patch development costs) associated with security in particular and concluded that the return on investment (ROI) ranged from 12 to 21%, with the highest rate of return occurring when analysis is performed during application design.

Software requirements serve as the foundation for measuring quality (Bokhari & Siddiqui, 2011). Measurement promotes the improvement of the software process; it aids in the planning, tracking, and management of the software project; and it assesses the quality of the software created. Quality issues such as correctness, security, and performance are frequently critical to a software system's success. Quality should be maintained from the beginning of the software development process. The management of requirements is critical to the maintenance of software quality. With good requirements management, a project may provide the right solution on time and within budget. Checking quality attributes in requirements documents can help to maintain software quality. To measure the requirements engineering phase of the software development lifecycle, metrics such as volatility, traceability, size, and completeness are utilized. Because manual measuring is costly, time consuming, and prone to error, automated tools should be employed. Measurement of requirements metrics is aided by automated requirements technologies.

In this paper, Bokhari & Siddiqui (2011) investigated and analyzed requirements metrics and automated requirements tools in order to assist in selecting the appropriate metrics to monitor software development based on the evaluation of Automated Requirements Tools. Because manual measurement is error-prone and time-consuming, automated measurement tools should be used. The use of automated requirements tools makes metric collection faster and more reliable. The automated requirements tools used for collecting, viewing, and changing requirements are Automated Requirements Tool, Dynamic Object Oriented Requirements Systems, Requirements Use Case Tool, and IBM Rational Rose. These tools manage changes and provide metrics for traceability. The use of automated tools improves requirements management; however, before employing any requirements tool, its function, applications, and limitations must be understood. But, this paper did not focus on the security aspect.

It is critical to have a thorough understanding of existing tools and systems available in the public domain in order to prevent and protect networks from attacks (Hoque et al., 2014). Attacks are classified into a variety of different kinds based on their behavior and the potential impact or severity of damages (Hoque et al., 2014). People use various attack tools to disrupt a network for a variety of reasons. Attackers typically target Web sites or databases, as well as enterprise networks, by gathering information about their vulnerabilities. In general, attackers employ tools that are appropriate for the type of attack they intend to carry out. Various network security research groups and private security professionals have also made a large number of defense tools available. These tools serve different purposes, have different capabilities, and use different interfaces. Existing tools are divided into two categories: tools for attackers and tools for network defenders. Before launching an attack, attackers should first understand the environment in which the attack will be launched. To do so, attackers first collect network information such as machine and service port numbers, operating systems, and so on. After gathering information, attackers use a variety of tools to identify network flaws. Sniffing tools and network mapping/scanning tools are subcategories of information gathering tools. In this survey, a consistent taxonomy of attack tools were provided for the benefit of network security researchers. This paper includes a thorough and organized review of available network tools and systems but was not focused on the software development tools that can help developers in developing an application without any security flaws.

Jagdale et al. (2014) described the Seven Pernicious Kingdoms taxonomy, which simulates the details and consequences of typical mistakes that contribute to security flaws. The taxonomy provides consistent terminology across vulnerability assessment approaches and responsibilities ranging from security practitioners to architects and developers, as well as up the management chain, when used to organize secure coding guidelines. The Seven Pernicious Kingdoms taxonomy's classification technique is based on code-level security concerns that occur in software applications. Seven of the kingdoms are dedicated to source code errors, while the eighth kingdom is concerned with security vulnerabilities generated by the program's configuration or environment.

According to Miele et al. (2018) software development, maintenance, and operation, and software security are part of an ever-evolving field. Software flaws put the software's operation at risk. Through static analysis of source code, software tools have evolved over time to assist in the detection and identification of software vulnerabilities. A software development team can use static analysis techniques to quickly assess their project for vulnerabilities that they are unaware of. Miele et al. (2018) used applications written in C language for testing in this work due to their high vulnerability nature, as C functions have few security mechanisms. As a result, many applications in C inherently have a high risk of vulnerability. They used open source PuTTY 0.68, Wireshark 1.12.1, and Nmap 6.47 to test applications. These applications are well-known in the network software community and are widely used, as evidenced by active user feedback (including those pertaining to vulnerability). They also have a number of stable open source versions available, making them appropriate for the experiment. There are numerous static analysis tools available, each with its own set of features and algorithms. They selected tools for vulnerability testing using Nagy and Mancoridis (2009) criteria. Vulnerabilities affecting software systems are numerous in both cause and effect. Vulnerabilities are influenced by both syntax and semantics. They concentrated on buffer overflow, format string, random number generation, shell, and race condition vulnerabilities in this work. They tested PuTTY, Wireshark, and Nmap using the three tools. They compared the capability of the three tools in detecting and reporting vulnerabilities for analysis. The raw results generated by each tool were then analyzed to determine its strengths and weaknesses. In this paper, Miele et al. (2018) provided a comparative review of three widely used static analysis tools for software vulnerability using open source software in this work, with the goal of assisting software developers in selecting the best tool for their needs. But this did not include a variety of tools used throughout the software development process.

Many computer applications, both proprietary and open source, are impeded by software vulnerabilities (McLean, 2012) . Fortunately, there are a number of static analysis tools available to help spot potential security issues. In this paper, McLean (2012) presented the results of utilizing a variety of static security analysis methods to evaluate multiple subsets of open source code for typical software vulnerabilities. These findings can help other programmers decide which tools to employ when evaluating their own code for static security methods and not for dynamic security methods.

The requirement is the foundation of the entire software development life cycle (Bokhari & Siddiqui, 2009). With good requirement management, a project may provide the right solution on time and within budget. Requirement elicitation, specification, and validation are all critical steps in ensuring the quality of requirement documentation. The software requirement tools can provide a more thorough analysis of all three activities. There are a lot of software requirement tools available, both commercially and freely downloadable, that give a diversity and high quality of software requirement documentation. Furthermore, as the vulnerabilities of software increase, the system requires an additional requirement for security features that protect the software from vulnerabilities and make the software more reliable. A Comparative Study of Software Requirements Tools for Secure Software Development written by Bokhari & Siddiqui (2009) provides a comparative study of requirement tools, demonstrating trends in the usage of methodology for gathering, analyzing, specifying, and validating software requirements, and the results will assist the developer in developing an effective requirement tool.

Almost every software-controlled system faces potential adversaries, ranging from Internet-aware client applications running on PCs to complex telecommunications and power systems accessible via the Internet, to commodity software with copy protection mechanisms (Devanbu & Stubblebine, 2000). Software engineers must be aware of these threats and design systems that have credible defenses while still providing value to customers. In this paper, Devanbu and Stubblebine (2000) presented their perspectives on the research issues that arise in the interactions between software engineering and security. Much of today's software development is based on integrating off-the-shelf components; rarely are new systems built from the ground up.

Middleware technologies (see also the companion paper on middleware [Wolfgang, 2000]) such as COM and CORBA have given rise to a wide range of components, frameworks, libraries, and so on. These are referred to as commercial off-the-shelf software (COTS). A useful summary of research issues in COTS products is very appealing to developers confronted with ever more stringent cost, quality, and time-to-market requirements. However, using these products, especially in safety-critical systems, is fraught with risk. The procurement policies of the customers of safety-critical systems (utilities, government, etc.) have traditionally required software vendors to disclose sufficient details to evaluate their processes and products for safety. These policies, however, are incompatible with current component vendors, who face the risk of intellectual property loss (Wolfgang, 2000).

Summary

The background of the topic is thoroughly presented in this chapter, with an emphasis on software vulnerability issues, and the literature related to the problem is illustrated using statistics and numbers centered on the problem's nature and significance. The literature related to the methodology is also described by critically evaluating the previous work done in the field by giving a clear understanding to the research problem being investigated.

Chapter III: Methodology

Introduction

In this chapter the design of the study is described in detail and the data collection process for the whole project is stated. The work that has previously been completed is listed, along with a concise timeline for what will be completed in the future.

Design of the Study

- Researched for a similar problem using Google Scholar.
- Investigated papers published between 2010 and 2022.
- Found and analyzed papers published between those years because cloud computing came into the picture and many programming languages have evolved since then.
- Classified the tools according to the software development life-cycle and based on project-size, open/commercial to better understand the software engineers' expectations and needs.
- Created a summary table of all the tools to give a brief overview.

Data Collection

Investigation of tools:

Made a comprehensive investigation and analysis of secure application tools that are currently available to get a clear understanding of their usage and effectiveness.

Identification of tools:

- Identified the tools for all six phases of the secure software application development.
- Found various tools for the requirement phase, design phase, implementation phase, testing phase, deployment, and maintenance phase.
- Gathered tools based on project size, open/commercial tools, etc.
- Analyzed whether each tool detects different security risks and threats in each phase of software engineering.

Classification of tools:

- Classified the tools according to the software development lifecycle.
- Created a clear summary table of all the secure tools used during SDLC to get a brief overview for developers and individuals to use.

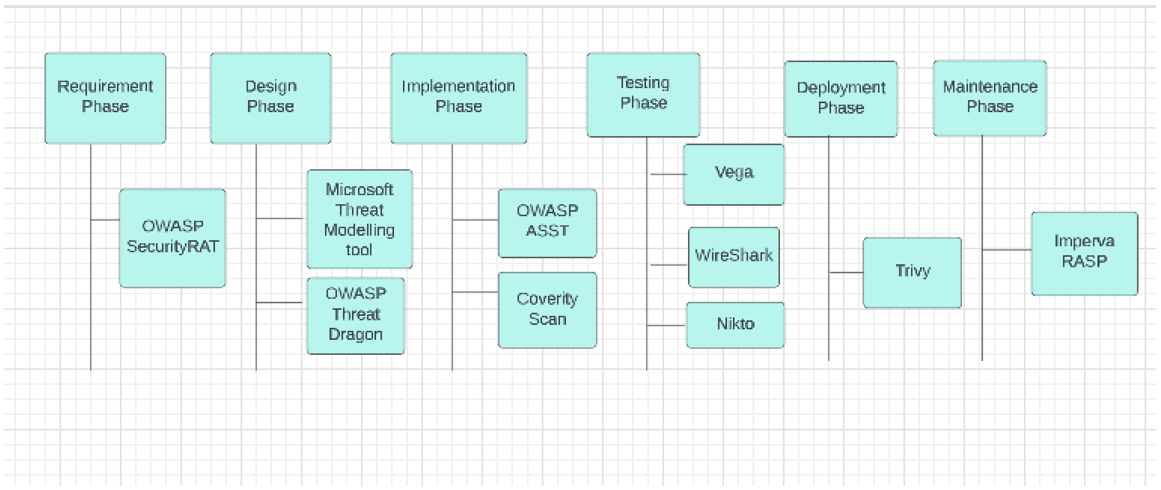
Chapter IV: Data Presentation and Analysis

Secure Open Source tools in SDLC lifecycle

The figure 1 shows a layout of open source tools useful in secure application development for developers and individuals to use throughout the software development life cycle.

Figure 1

Secure Open Source tools in SDLC lifecycle



Requirement Phase

OWASP SecurityRAT

SecurityRAT (Security Requirement Automation Tool) is a tool that assists in managing security requirements in agile development projects (*OWASP SecurityRAT*, 2022). The basic idea is straightforward: you specify the properties of an application (or "artifact") that you're developing. Based on these properties, the tool generates a list of security requirements that you must meet. For each requirement, you can decide whether it should/will be implemented and add your own comment. When you're finished, you can save the specific requirement in a JIRA ticket for future reference as a YAML file (*SecurityRAT*, 2021). Following that, you can create JIRA tickets in bulk for specific requirements and track them with SecurityRAT.

Design Phase

Microsoft Threat Modelling Tool

It enables software architects to identify and mitigate potential security issues early, when they are relatively simple and inexpensive to fix (jegeib, 2022). The tool was created with non-security experts in mind, with the goal of making threat modeling easier for all developers by providing clear guidance on creating and analyzing threat models (jegeib, 2022).

Anyone can use the tool to:

- Communicate about the security architecture of their systems
- Utilize a tested methodology to examine designs for any potential security flaws.
- Manage mitigations for security issues by making suggestions.

OWASP Threat Dragon

OWASP Threat Dragon is a modeling tool used to create threat model diagrams as part of a secure development lifecycle (*OWASP Threat Dragon*, 2023). It can be used to record potential threats and decide on mitigation strategies, as well as provide a visual representation of the threat model components and threat surfaces. Threat Dragon is available as a web application or a desktop application.

Implementation Phase

OWASP ASST

ASST is an open source code scanning tool with a command line interface written in JavaScript. When ASST scans for a project, it examines each file line by line for security flaws. If a vulnerability was discovered, the report will indicate which line in which file the vulnerability was discovered, as well as a "Click Here" link to explain the attack and how to protect against it (*OWASP ASST*, 2020). Currently it focuses on PHP and MySQL programming languages, but because its core functionalities are ready and available to everyone, programmers can contribute and add plugins or extensions to it to add features and make it scan for other programming languages and frameworks such as Java, C#, Python, and so on. As a result, its infrastructure is intended to be shared with other programmers in order to improve and innovate.

Coverity Scan

Coverity Scan finds and fixes defects in Java, C/C++, C#, JavaScript, Ruby, or Python open source project for free (*Coverity Scan - Static Analysis*, n.d.). Tests every line of code and potential execution path and the root cause of each defect is clearly explained, making it easy to fix bugs.

Testing

Vega

Vega is a free and open source web application security scanner and testing platform. It is written in Java, has a graphical user interface, and is compatible with Linux, OS X, and Windows. Vulnerabilities such as reflected cross-site scripting, stored cross-site scripting, blind SQL injection, remote file include, shell injection, and others can be detected using Vega. (*Vega Vulnerability Scanner*, 2014). It also checks for TLS / SSL security settings and identifies ways to improve the security of your TLS servers. Vega includes a quick test scanner and an intercepting proxy for tactical inspection.

- Vega's automated scanner is powered by a website crawler. When given user credentials, Vega can automatically log into websites.
- The Vega proxy can also be set up to run attack modules while the user navigates to the target site through it. To ensure maximum code coverage, semi-automated, user-driven security testing is possible.

Wireshark

Wireshark has a rich and powerful feature set and runs on most computing platforms, including Windows, OS X, Linux, and UNIX (Banerjee et al., 2010). It is used frequently by network professionals, security experts, developers, and educators all over the world. It is open source and distributed under the GNU General Public License version 2. It was created and is maintained by a global team of protocol experts, and it is an example of disruptive technology. Wireshark is a free packet sniffer computer application. It is used for network troubleshooting, analysis, software development, communication protocol development, and education. Wireshark can capture traffic "from the air" and decode it into a format that helps administrators track down issues that are causing poor performance, intermittent connectivity, and other common issues with the appropriate driver support. Wireshark enables users to capture packets traveling across the entire network on a specific interface at a specific time.

Nikto

Nikto is an Open Source (GPL) web server scanner that runs comprehensive tests against web servers for a variety of items, including over 6700 potentially dangerous files/programs, outdated versions of over 1250 servers, and version-specific problems on over 270 servers. It also looks for server configuration items like the presence of multiple index files and HTTP server options, as well as attempting to identify installed web servers and software. Scan items and plugins are frequently updated and can be updated automatically (*Nikto2 | CIRT.Net*, 2023).

Deployment Phase

Trivy

Trivy is an easy-to-use, open source and versatile security scanner. Trivy has scanners that look for security issues and targets where those issues can be found. It is ideal for DevSecOps pipelines because it integrates with CI tools such as Travis, CircleCI, Jenkins, and GitLab. Trivy can scan - Container Images, Filesystems, Git Repository (remote), Virtual Machine Image, Kubernetes, and AWS. It can discover- OS packages and software dependencies in use (SBOM), Known vulnerabilities (CVEs), IaC issues and misconfigurations, Sensitive information and secrets, Software licenses (*Overview - Trivy*, n.d.).

Maintenance Phase

Imperva RASP

RASP, which is built into the application runtime environment, can detect and prevent attacks in real time. Because of today's security challenges, your cloud-native applications require more privacy than network firewalls, which is why Imperva provides protection from within and runs alongside your applications (*RASP Market Leader*, n.d.). It can protect the application using:

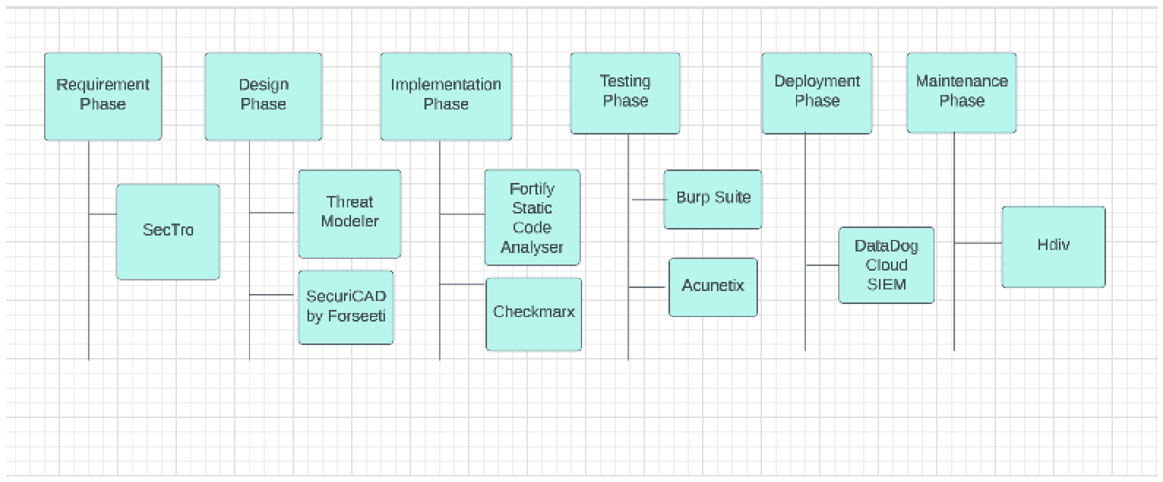
- Application risk mitigation: RASP protects applications from vulnerabilities, allowing teams to focus on business logic rather than leaving your application vulnerable to exploitation.
- Security as business evolves: Cloud native applications require more than just perimeter security due to fading controls and ephemeral workloads. RASP provides internal security and follows your application wherever it goes.

Secure Commercial tools in SDLC lifecycle

The figure 4.2 below shows a layout of commercial tools useful in secure application development for developers and individuals to use throughout the software development life cycle.

Figure 2

Secure Commercial Tools in SDLC Lifecycle



Requirement Phase

Secure Tropos

Secure Tropos (Giorgini et al., 2005) is an agent-oriented software development methodology designed to describe the organization as well as the system in terms of functional and security requirements. Secure Tropos builds on the Tropos methodology (Bresciani et al., 2004) by introducing the concepts of actor, service (i.e., goal, task, resource), and social relationships to define the obligations of actors.

Design Phase

ThreatModeler

ThreatModeler is an automated threat modeling solution that strengthens an enterprise's SDLC by detecting, predicting, and defining threats, enabling security and DevOps teams to make proactive security decisions. ThreatModeler provides a comprehensive view of the entire attack surface, allowing enterprises to reduce overall risk (*ThreatModeler*, 2023).

SecureCAD by Forseeti

It has modeling tools which build models directly in your browser or use existing data to generate parts or an entire model of your architecture (*SecuriCAD Enterprise*, n.d.). It has features such as attack simulations, choke points, finds the most critical paths from a potential attacker to a high-value report generation, and so on.

Implementation Phase

Fortify Static Code Analyzer

Fortify Static Code Analyzer (SCA) is a commercial tool that detects root-cause vulnerabilities using the most comprehensive set of secure coding rules available and supports the most languages, platforms, build environments (Integrated Development Environments, or IDEs), and software component APIs https://www.microfocus.com/media/data-sheet/fortify_static_code_analyzer_static_application_security_testing_ds.pdf. A free trial is available for 15 days (*Fortify on Demand Free Trial,* 2023).

- Conduct static analysis to identify the root causes of security vulnerabilities in source code.
- Detects more than 480 types of software security vulnerabilities across 20 development languages, the most in the industry.
- Receive prioritized results sorted by risk severity, as well as guidance on how to fix vulnerabilities in line-of-code detail.
- Ensure that application security mandates are met.

Checkmarx

Checkmarx is a high-end static code scanner that supports all popular languages. It seamlessly secures your entire codebase, allowing you to deliver and deploy more secure code. It is designed for cloud development generation and delivered from the cloud (*Why Checkmarx*, 2023).

- Scanning of Source Code: Identify and fix additional flaws before releasing your code.
- Open Source Scanning: Identify and eliminate vulnerabilities in your open source code.
- Scanning Interactive Codes: During functional testing, look for vulnerabilities and runtime risks.
- Secure Code Education: Provide devs with engaging, integrated, and targeted AppSec training.
- IaC Open Source Security: Determine and correct insecure IaC configurations that put you at risk.

Testing Phase

Burp Suite

Burp Suite Professional is a web security tester's toolkit that is used to automate repetitive testing tasks before going deeper with its expert-designed manual and semi-automated security testing tools. Burp Suite Professional can assist you in testing for the OWASP Top 10 vulnerabilities. The manual penetration testing features are as follows: Intercept everything your browser sees, effectively break HTTPS, manually test for out-of-band vulnerabilities, expose hidden attack surface, test for clickjacking attacks, rapidly assess your target, and so on (*Features - Burp Suite Professional*, 2023). Automated scanning for vulnerability features are: Remediate bugs effectively, conquer client-side attack surface, configure scan behavior, fuel vulnerability coverage with research, experience browser-driven scanning, fine-tune scan control, and so on.

Acunetix

Acunetix quickly identifies and fixes the vulnerabilities that expose your web applications to attack. It can detect 7,000+ vulnerabilities using a combination of DAST and IAST scanning, including OWASP Top 10, SQL injections, XSS, misconfigurations, exposed databases, and out-of-band vulnerabilities (*Acunetix*, 2023). It resolves vulnerabilities more quickly than remediation. The following are the tool features:

- Remove false positives: Spend less time manually confirming which vulnerabilities are real.
- Identify vulnerable points: You won't have to look for the exact lines of code that need to be fixed.
- Obtain remediation advice: Provide developers with all of the information they need to fix security flaws on their own.

Acunetix Web Vulnerability Scanner performs the following functions:

- Crawls thousands of pages at lightning speed and without interruption.
- In-depth SQL injection and Cross-Site Scripting (XSS) testing, the most comprehensive scanner for these vulnerabilities.
- Acunetix AcuSensor Technology combines black box scanning techniques with feedback from sensors embedded within the source code to provide accurate scanning with low false positives. It does security testing of AJAX and Web 2.0 applications using automatic JavaScript analysis.

- A Login Sequence Recorder for quick and easy testing of password-protected areas and the acunetix deepscan has the capability of interpreting SOAP, XML, AJAX, and JSON.

Deployment Phase

DataDog Cloud SIEM

DataDog Cloud SIEM is a paid tool that includes a 14-day trial period. It functions as both a vulnerability scanner and a SIEM, and it provides quick solutions to security breaches. Continuous scans across cloud accounts, hosts, and containers track security posture and unusual activity in full context, with alerts channeled into your ticketing system. It allows you to analyze everything without having to index and store all of the data. It detects security threats and misconfigurations automatically in real time (Datadog, 2023).

Maintenance Phase

Hdiv

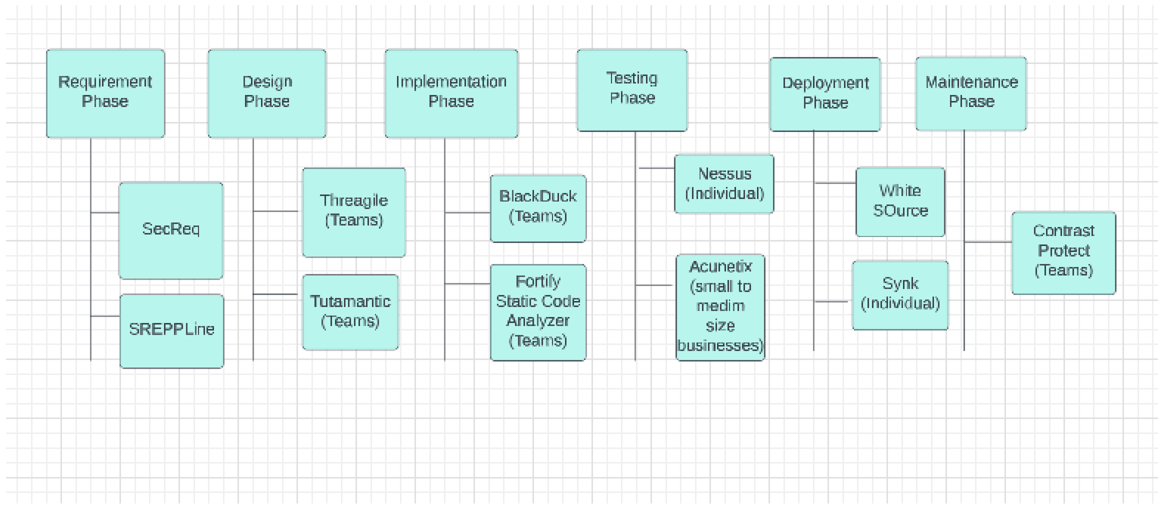
Hdiv Security, a pioneer in application self-protection, is the first product of its kind to provide protection against Security Bugs and Business Logic Flaws throughout the Software Development Lifecycle (SDLC). The unified platform of Hdiv Security makes DevSecOps a reality. Government, banking, aerospace, and Fortune 500 companies are currently using Hdiv's solutions (Documentation | Hdiv Security, 2021). It is more effective than any other solution currently available for combating web application security risks. Maintains strong security without having to be aware of and comprehend all current security threats.

Secure tools used by individuals, teams, and small businesses

The figure 4.3 shows a layout of tools used by individuals, teams, and small businesses useful in secure application development.

Figure 3

Secure tools used by individuals, teams, and small businesses



Requirement Phase

SecReq

SecReq's Common Criteria-based security requirements elicitation section refines high-level security needs through several steps to produce specific security requirements suitable for formulating design solutions and helps to identify potentially secure aspects of service descriptions and functional requirements. It is a combination of three techniques: the CC, the HeRA heuristic requirements editor, and UMLsec. SecReq employs in a systematic manner the security engineering knowledge contained in the CC and UMLsec, as well as security-related heuristics in the HeRA tool. This tool identifies early detection of security-related issues (HeRA), systematic refinement guided by the CC, and has the ability to trace security requirements into UML design models (Houmb et al., 2010).

SREPPLine

It addresses security requirements in the early stages of the product line lifecycle in a systematic and intuitive manner, making it especially suited for product line-based development. It is founded on the application of cutting-edge security requirements techniques, as well as the incorporation of the Common Criteria (ISO/IEC 15408) into the product line lifecycle. It also addresses the reuse of security artifacts by providing us with a Security Resources Repository. Furthermore, it facilitates compliance with the most relevant security standards in terms of security requirement management (Mellado et al., 2023).

Design Phase

Threagile

Threagile enables teams to execute Agile Threat Modeling as seamlessly as possible, even in DevSecOps environments (*Threagile — Agile Threat Modeling Toolkit*, 2020). Threagile is an open-source toolkit that allows you to model an architecture and its assets as a YAML file directly inside the IDE or any YAML editor. When the Threagile toolkit is run, a set of risk-rules perform security checks against the architecture model and generate a report with potential risks and mitigation recommendations. In addition, nice-looking data-flow diagrams and other output formats are generated automatically (Excel and JSON). Risk tracking can also take place within the Threagile YAML model file, so that the current state of risk mitigation is also reported. Threagile can be started or run from the command line or started as a REST-Server.

Tutamantic

Tutamen Threat Model Automator detects and mitigates security threats early during the software design phase. It is intended to enable security at the architectural stage, when the cost of fixing flaws is the lowest (*Home* | *Tutamantic*, n.d.). Reduce human error and inconsistencies with a single variable input. Create a living threat model that changes as the design changes. For its reference frameworks, the Tutamen Threat Model Automator employs the well-known OWASP Top 10, STRIDE, and Common Weakness Enumeration (CWE).

Black Duck

Software composition analysis (SCA) by Black Duck assists teams in managing the security, quality, and license compliance risks associated with the use of open source and third-party code in applications and containers. Over 4,000 organizations worldwide use Black Duck today. It allows developers and DevOps teams to address open source policy concerns while maintaining innovation. This tool costs $525 per team member and can support up to 150 team members (*Black Duck Software Composition Analysis,* 2023).

- Open source is identified within compiled application libraries and executables.
- Tracks vulnerability results and remediation.
- Checks open source code for newly disclosed vulnerabilities.

- Open source vulnerabilities in applications and containers are discovered and fixed.

Testing Phase

Nessus

Nessus can be used by individuals as it scans 16 IP addresses at a time. It runs on a variety of platforms, including the Raspberry Pi. Nessus is used for penetration testing and vulnerability assessments, including malicious attacks. It is a program that searches computers for security flaws that hackers could exploit. Nessus is fully portable regardless of where you are, where you need to go, or how distributed your environment is (*Nessus*, 2023).

WhiteSource(Mend)

WhiteSource provides the best open source security and license compliance management platform available, allowing users to manage and trust open source assets easily and efficiently (*WhiteSource Documentation*, n.d.). Mend has SCA and now offers a static application security testing (SAST) solution to assist your organization in securing proprietary code as well as open source code.

Deployment Phase

Snyk

Snyk detects and fixes vulnerabilities in your code, open source dependencies, containers, and infrastructure as code. This tool simply integrates security knowledge into the existing IDEs, repositories, and workflows. It continuously scans using industry-leading security intelligence to monitor for vulnerabilities while developing and provides actionable fix advice in tools with a single click which can merge and move on with auto PRs (*Snyk,* n.d.).

Maintenance Phase

Contrast Protect

Contrast Protect is a production application and API protection solution that detects and prevents attacks while reducing false positives, allowing developer teams to prioritize vulnerability backlogs (*Contrast Protect*, n.d.). The features of contrast protect are as follows:

- This tool blocks attacks on vulnerabilities that have not been fixed or patched.
- In contrast to perimeter defenses, instrumentation and sensors detect, it also prevents runtime application attacks.
- It determines whether the exploit was successful or not. Protects against a wide range of zero-day attacks without the need for tuning or reconfiguration.
- It also includes game-changing forensics and application protection for all organizations, large and small.

Summary

The tools are categorizing based on open source, commercial are well defined for any developer to quickly identify and use the appropriate one for developing secure applications.

Chapter V: Results and Conclusion

Results and Discussions

The provided taxonomy of tools assists developers to quickly adopt a specific tool based on the features and type of tool required either for teams or individual use, open source or commercial. As a result, data is secure against attackers, businesses, end-users, and endpoint devices. As a result, it contributes to business continuity.

Cybersecurity threats evolve. Intruders adapt by developing new tools and techniques to compromise security as new defenses emerge to stop older threats (https://www.advantio.com/blog/secure-software-development-life-cycle-design-phase). The incentives to compromise the security of deployed IT systems grow as information technology becomes more pervasive in society. As new information technology applications are developed, new venues for criminals, terrorists, and other hostile parties emerge, as do new vulnerabilities that malevolent actors can exploit. The fact that an increasing number of people have access to cyberspace multiplies the number of potential victims as well as the number of potential malicious actors. The summary table below provides all the necessary tools required to make the applications safe and secure from malicious attacks.

Conclusion

The summary table gives a brief overview of tools in a readable format to help find the right tool in the SDLC phases. Cybersecurity is an evolving field in which attackers can develop new tools to attack systems. As a result, tools must be kept up to date in order to combat threats and attacks.

Table 1

Summary Table of Tools for Open Source, Commercial and for Individuals, Teams, and Businesses

Requirement Phase	Design Phase	Implementation Phase	Testing Phase	Deployment Phase	Maintenance Phase
OWASP SecurityRAT: *Generates security requirements. *Tracks specified requirements.	**Microsoft Threat Modelling Tool:** *Enables software architects to identify and mitigate potential security issues early. **OWASP Threat Dragon:** *Creates visual representation of threat components. *Records potential threats.	**OWASP ASST:** *Examines files for security flaws. CLI application. *Scans PHP language. **Coverity Scan:** *SAST tool. Finds and fixes defects in code. Tests every line.	**Vega:** *Does automated, manual and hybrid security tests. *Has an intercepting proxy. **Wireshark:** *Packet sniffer. *Captures packets across the network. **Nikto:** *Web Server scanner. *Updates scan items and plugins automatically.	**Trivy:** *Targets at the security issues. *Ideal for DEVSECOPS pipelines. *Integrates with CI tools.	**ImpervaRASP:** *Detects and prevents attacks in real-time. *Provides internal security for businesses.

Requirement Phase	Design Phase	Implementation Phase	Testing Phase	Deployment Phase	Maintenance Phase
SecTro: *Builds on the tropos methodology. *Introduces the concepts of actor, service and social relationships.	**Threat Modeler:** *Automates threat modeling. *Provides a view of the attack surface. **Security CAD by forseeti:** *Builds models directly in the browser. *Finds critical paths from a potential attacker. *Have attack simulations, choke points etc.	**Fortify Static Code Analyser:** *Supports most languages and build environments. *Conducts static analysis. *Detects 480 types of vulnerabilities. **Checkmarx:** *It’s a static code scanner. *Designed for cloud development generation.	**Burp Suite:** *Automates repetitive testing tasks. *Assists in testing OWASP top 10 vulnerabilities. *Assists in testing OWASP top 10 vulnerabilities. **Acunetix:** *Crawls thousands of pages. *Removes false positives.	**DataDog Cloud SIEM:** *Detects threats and misconfigurations in real-time. *Does continuous scans across cloud accounts. *Sends alerts via ticketing system.	**Hdiv:** *Maintains strong security. *Has a unified platform.

Requirement Phase	Design Phase	Implementation Phase	Testing Phase	Deployment Phase	Maintenance Phase
SecReq: *Facilitates early detection of security-related issues. *Traces security requirements into *UML design models. **SREPPLine:** *Addresses security requirements in early stages. *Facilitates compliance with the most relevant security standards.	**Threagile:** *Enables teams to execute agile threat modeling. *Generates a report with potential risks and mitigation recommendations. **Tutamantic:** *Detects and mitigates security threats early during the design phase. *Reduces human error and inconsistencies with single variable input.	**Blackduck:** *Allows teams to address open source policy concerns. *Checks open source code for newly discovered vulnerabilities. **Fortify Static Code Analyser:** *Supports most languages and build environments. *Conducts static analysis. *Detects 480 types of vulnerabilities.	**Nessus:** *Fully portable. *Individuals can scan 16 IPs at a time. **Acunetix:** *Crawls thousands of pages. *Removes false positives.	**White Source:** *Offers SAST to assist organizations in securing proprietary code as well as open source. **Synk:** *Uses industry-leading security intelligence to monitor vulnerabilities. *Provides actionable fix advice with a single click.	**Contrast Protect:** *Allows developer teams to prioritize vulnerability backlogs. *Includes game changing forensics and application protection for all organizations.

References

93% of Security Professionals Lack the Necessary Tools to Detect Security Threats. (2020, July 28). LogRhythm. https://logrhythm.com/press-releases/93-of-security-professionals-lack-the-necessary-tools-to-detect-security-threats-according-to-logrhythm-report/

Acunetix. *Web Application Security Scanner*. (2023). https://www.acunetix.com/

Banerjee, U., Vashishtha, A., & Mukul, S. (2010). Evaluation of the Capabilities of WireShark as a tool for Intrusion Detection. *International Journal of Computer Applications*, *6*. https://doi.org/10.5120/1092-1427

Bellairs, R. (2022). What Is Static Analysis? Static Code Analysis Overview. *Perforce Software*. https://www.perforce.com/blog/sca/what-static-analysis

Black Duck Software Composition Analysis (SCA). Synopsys. (2023). https://www.synopsys.com/software-integrity/security-testing/software-composition-analysis.html

Boehm, B., & Basili, V. (2001). Software Defect Reduction Top 10 List. *Computer, 34*(1). https://doi.org/10.1109/2.962984

Bokhari, M. U., & Siddiqui, S. T. (2009). A Comparative Study of Software Requirements Tools for Secure Software Development. *Undefined*. https://www.semanticscholar.org/paper/A-Comparative-Study-of-Software-Requirements-Tools-Bokhari-Siddiqui/416630616a9d92ebc0fd4a944481701895743c26

Bokhari, M., & Siddiqui, S. (2011, January 1). *Metrics for Requirements Engineering and Automated Requirements Tools*.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2004). Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, *8*(3), 203–236. https://doi.org/10.1023/B:AGNT.0000018806.20944.ef

Chickowski, E. (2017). *The most epic developer fails of 2017: Lessons learned.* TechBeacon. https://techbeacon.com/app-dev-testing/most-epic-developer-fails-2017-lessons-learned

Cimpanu, C. (2021, December 13). Log4Shell attacks began two weeks ago, Cisco and Cloudflare say. *The Record by Recorded Future*. https://therecord.media/log4shell-attacks-began-two-weeks-ago-cisco-and-cloudflare-say/

Contrast Protect | *Application and API Protection.* Contrast Security*. (n.d.). Retrieved March 13, 2023, from https://www.contrastsecurity.com/contrast-protect*

Coverity Scan—Static Analysis. (n.d.). Retrieved March 14, 2023, from https://scan.coverity.com/

Cumming, J. (2017, February 23). Incident report on memory leak caused by Cloudflare parser bug. *The Cloudflare Blog*. http://blog.cloudflare.com/incident-report-on-memory-leak-caused-by-cloudflare-parser-bug/

Datadog. (2023). Container Security. https://www.datadoghq.com/dg/security/runtime-security/

Devanbu, P. T., & Stubblebine, S. (2000). Software Engineering for Security: A Roadmap. *The Future of Software Engineering*, 227–239.

Documentation. Hdiv Security. (2021). https://hdivsecurity.com/docs/

Download Nessus Vulnerability Assessment - Nessus®. (2023). Tenable®. https://www.tenable.com/products/nessus

Eisenstein, P. (2017, May 15). If you have a Dodge Ram, get this fatal flaw fixed asap. *NBC News*. https://www.nbcnews.com/business/autos/1-25-million-dodge-ram-pickups-recalled-over-fatal-software-n759476

Electronic Privacy Information Center. (2017). *EPIC - Equifax Data Breach*. https://archive.epic.org/privacy/data-breach/equifax/

Features—Burp Suite Professional. (2023). https://portswigger.net/burp/pro/features

Fortify on Demand Free Trial. Micro Focus. (2023). Retrieved March 13, 2023, from https://www.microfocus.com/en-us/products/application-security-testing/free-trial?utm_source=google&utm_medium=cpc&utm_campaign=7018e000000t6gGAAQ&ppc_keyword=fortify%20scan&gclid=Cj0KCQjwk7ugBhDIARIsAGuvgPbiMHMAQveJrbIT_oNwrh8YAnlskjxY4RsJf675OI1GqD8CCMGGlP0aAlA6EALw_wcB

Giorgini, P., Massacci, F., Mylopoulos, J., & Zannone, N. (2005). ST-Tool: A CASE tool for security requirements engineering,*3th IEEE International Conference on Requirements Engineering (RE'05)*, 451–452, https://doi.org/10.1109/RE.2005.67

Hein, D., & Saiedian, H. (2009). Secure Software Engineering: Learning from the Past to Address Future Challenges. *Information Security Journal: A Global Perspective, 18*(1), 8–25. https://doi.org/10.1080/19393550802623206

Home. Tutamantic. (n.d.). Retrieved March 16, 2023, from https://www.tutamantic.com/

Hoque, N., Bhuyan, M. H., Baishya, R. C., Bhattacharyya, D. K., & Kalita, J. K. (2014). Network attacks: Taxonomy, tools and systems. *Journal of Network and Computer Applications, 40*, 307–324. https://doi.org/10.1016/j.jnca.2013.08.001

Houmb, S., Islam, S., Knauss, E., Jürjens, J., & Schneider, K. (2010). Eliciting security requirements and tracing them to design: An integration of Common Criteria, heuristics, and UMLsec. *Requirements Engineering*, *15*, 63–93. https://doi.org/10.1007/s00766-009-0093-9

https://www.advantio.com/blog/secure-software-development-life-cycle-design-phase

https://www.microfocus.com/media/data-sheet/fortify_static_code_analyzer_static_application_security_testing_ds.pdf

Jagdale, P., O'Neil, Y., Sechman, J., & West, J. (2014). The Evolution of a Taxonomy: Ten Years of Software Security. *Semantic Scholar.* https://www.semanticscholar.org/paper/The-Evolution-of-a-Taxonomy-%3A-Ten-Years-of-Software-Jagdale-O%E2%80%99Neil/3f4768930ade8794eb2657354b830caf032132d9

jegeib. (2022, August 25). Microsoft Threat Modeling Tool overview—Azure. https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool

Kinzer, K. (2021, December 31). Top 5 Security Breaches of 2021. *JumpCloud*. https://jumpcloud.com/blog/top-5-security-breaches-of-2021

Log4j – Apache Log4j Security Vulnerabilities. (n.d.). Logging Services. https://logging.apache.org/log4j/2.x/security.html

Marriott Data Breach FAQ: What Really Happened? (2020, December 9). *Hotel Tech Report*. https://hoteltechreport.com/news/marriott-data-breach

McGraw, G. (1999). Software assurance for security. *Computer*, *32*(4), 103–105. https://doi.org/10.1109/2.755011

McLean, R. K. (2012). Comparing Static Security Analysis Tools Using Open Source Software. *2012 IEEE Sixth International Conference on Software Security and Reliability Companion*, 68–74. https://doi.org/10.1109/SERE-C.2012.16

Mellado, D., Fernández-Medina, E., & Piattini, M. (2023). *SREPPLine: Towards a Security Requirements Engineering Process for Software Product Lines*. 220–232. https://www.scitepress.org/Link.aspx?doi=10.5220/0002424702200232

Miele, P., Uwaisem, M. A., & Kim, D.-K. (2018). Comparative Assessment of Static Analysis Tools for Software Vulnerability. *Journal of Computers.* https://doi.org/10.17706/jcp.13.10.1136-1144

Nagy, C., & Mancoridis, S. (2009). Static Security Analysis Based on Input-Related Software Faults. *2009 13th European Conference on Software Maintenance and Reengineering*. https://doi.org/10.1109/CSMR.2009.51

Nikto2. CIRT.net. (2023). Retrieved March 13, 2023, from https://cirt.net/Nikto2

Novak, J., Krajnc, A., & Žontar, R. (2010). Taxonomy of static code analysis tools. *The 33rd International Convention MIPRO*, 418–422.

Overview—Trivy. (n.d.). Trivy. Retrieved March 13, 2023, from https://aquasecurity.github.io/trivy/v0.38/

OWASP ASST. (2020). OWASP. Retrieved March 12, 2023, from https://owasp.org/ASST/

OWASP SecurityRAT. (2022).OWASP Foundation. Retrieved March 13, 2023, from https://owasp.org/www-project-securityrat/

OWASP Threat Dragon. (2023). OWASP Foundation. Retrieved March 12, 2023, from https://owasp.org/www-project-threat-dragon/

Perlroth, N. (2021, May 13). Colonial Pipeline paid 75 Bitcoin, or roughly $5 million, to hackers. *The New York Times*. https://www.nytimes.com/2021/05/13/technology/colonial-pipeline-ransom.html

Ponemon Institute: External Cyber Attacks Cost Enterprises $3.5M/year, 79% of Businesses Lack Comprehensive Strategies to Manage these Risks. (2016, July 18). Business Wire. https://www.businesswire.com/news/home/20160718005304/en/Ponemon-Institute-External-Cyber-Attacks-Cost-Enterprises-3.5Myear-79-of-Businesses-Lack-Comprehensive-Strategies-to-Manage-these-Risks

RASP Market Leader - Secure all Applications by Default. (n.d.). Imperva. Retrieved March 13, 2023, from https://www.imperva.com/products/runtime-application-self-protection-rasp/

SecuriCAD Enterprise. (n.d.). Foreseeti. Retrieved March 14, 2023, from https://foreseeti.com/securicad-enterprise/

SecurityRAT. (2021). Retrieved March 13, 2023, from https://securityrat.github.io/

Snyk - Developer security (n.d.). Snyk. Retrieved March 14, 2023, from https://snyk.io/

Software Engineering - Software Design. (2021). Javatpoint. Www.Javatpoint.Com. https://www.javatpoint.com/software-engineering-software-design

Soo Hoo, K.; Sudbury, A. W.; & Jaquith, A. R. (2001). Tangible ROI through Secure Software Engineering. *Secure Business Quarterly 1*, 2 (Fourth Quarter 2001).

Spacey, J. (2017). *14 Types of Design Flaw*. Simplicable. https://simplicable.com/new/design-flaw

Stoneburner, G., Hayden, C., & Feringa, A. (2004). *Engineering principles for information technology security (a baseline for achieving security), revision a* (NIST SP 800-27ra; 0 ed., p. NIST SP 800-27ra). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-27ra

Takata Airbag Recall: Everything You Need to Know. (2023, February 3). Consumer Reports. https://www.consumerreports.org/cars/car-recalls-defects/takata-airbag-recall-everything-you-need-to-know-a1060713669/

The Impact of Security Misconfiguration and Its Mitigation. (2020). Cypress Data Defense. https://cypressdatadefense.com/

The PhishLabs Blog—PhishLabs. (n.d.). Retrieved March 20, 2022, from https://www.phishlabs.com/blog/

Threagile—Agile Threat Modeling Toolkit. (2020). https://threagile.io/

ThreatModeler—Automated Threat Modeling Solution. (2023). ThreatModeler. https://threatmodeler.com/

Vega Vulnerability Scanner. (2014). Subgraph Retrieved March 13, 2023, from https://subgraph.com/vega/?r=qal-stt

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

What is a Cybersecurity attack? (2022). Rapid 7. https://intsights.com/glossary/what-is-a-cybersecurity-attack

What is Dynamic Code Analysis? (n.d.). Check Point Software. https://www.checkpoint.com/cyber-hub/cloud-security/what-is-dynamic-code-analysis/

What is Requirements Elicitation. (2022). IGI Global. https://www.igi-global.com/dictionary/identifying-requirements-healthcare-information-systems/33121

What Is Software Development: Definition, Processes and Types. Indeed.com. (2021). Indeed Career Guide.

What is Software Engineering—Definition & Introduction. (n.d.). Software Intelligence for Digital Leaders. https://www.castsoftware.com/glossary/what-is-software-engineering-definition-types-of-basics-introduction

What is Vulnerability in Cyber Security? Types and Meaning. (2021, August 25). *Intellipaat Blog*. https://intellipaat.com/blog/vulnerability-in-cyber-security/

WhiteSource Documentation—Confluence. (n.d.). Retrieved March 16, 2023, from https://whitesource.atlassian.net/wiki/spaces/WD/overview

Wike, K. (2016). 64% Of Security Leaders Lack Tools Needed To Understand Security Threats. *Health IT Outcomes.* https://www.healthitoutcomes.com/doc/of-security-leaders-lack-tools-needed-to-u nderst and-security-threats-0001

Why Checkmarx. (2023). Checkmarx Application Security. Retrieved March 13, 2023, from https://checkmarx.com/why-checkmarx/right-choice/

Wolfgang, E. (2000). Software engineering for middleware: A roadmap. In A. Finkelstein (Ed.) *The Future of Software Engineering.* ICSE 2000.