International Journal of Humanities and Information Technology (IJHIT)
e-ISSN: 2456 –1142, Volume 04, Issue 1-3 (January-September 2022), ijhit.info

Security Assessment Framework for Microservices-Based Applications Using Automated Penetration Testing

Sreenivasa Rao Basavala

Independent Researcher, Brambleton, Virginia, USA

Email: basavala@gmail.com

ABSTRACT

Applications that employ microservices have become very popular following their scalability, modularity, flexibility and continuous delivery of software. Nonetheless, the complexity of their distributed nature brings about complex security issues such as the relatively vulnerable service interfaces, insecure inter-service communications, and vulnerabilities in authentication, weak authorization policies, and augmented attack surfaces. Conventionally used security assessment methods tend to rely on manual penetration testing which is time-consuming, inconsistent and is hard to scale in a dynamically changing microservices environment. In an endeavor to identify, analyze and prioritize the security weaknesses in a systematic way across the distributed service architectures, this study has introduced a Security Assessment Framework of Microservices-Based Applications using Automated Penetration Testing. The proposed framework is a combination of automated reconnaissance, service discovery, the identification of vulnerabilities, the analysis of the attack-path and controlled penetration testing as a cohesive workflow evaluation. The interfaces, APIs, authentication systems, the communication system, as well as the publicly accessible services are security tested with the aim of determining potential vulnerabilities to detect their security worth. A system comprises a risk based prioritization system, which will prioritize the identified vulnerabilities based on severity, exploitability and possible impact of the service. This framework will also help in a continuous security evaluation through the introduction of automated testing in microservices development and deployment processes. The proposed solution offers an organized and systematic method of security testing, lessens on the reliance on a manual analysis, and enhances the visibility of distributed attack surfaces. Security teams and the developers can also use the framework to improve security of microservice using the application lifecycle.

KEYWORDS: Microservices Security, Automated Penetration Testing, Security Assessment, Vulnerability Detection, API Security, Attack Surface Analysis, Risk-Based Security.

DOI: 10.21590/ijhit.04.1-3.22

I. INTRODUCTION

The architecture based on microservices has become one of the significant ways of creating the software application in the present day due to an ability to divide applications into small, portably deployable and manageable services. A service might be executing a specified business task and can exchange messages with other services via minimum interfaces which usually are of the application programming interface (APIs) type. It is a model of architecture that encourages scalability, continuous delivery, and diversity of technologies, fault isolation, and freedom of service development. However, microservices are distributed in nature, as well, and they pose

serious security concerns. Organizations need to keep hundreds of services, APIs, communication channels, containers, identities, and infrastructure components safe, as opposed to protecting a relatively centralized application. This can cause a significant growth in the overall attack surface and make it more complex [1].

Security governance is especially relevant in the domain of microservices since responsibilities of security are spread out among various services and other elements of the infrastructure. Specialized security services could provide a way of enforcing security policies and security-functionality in the distributed applications [1]. The security controls in place, however, may not necessarily mean that an application implemented is resistant to practical attacks. There are referred to as bad services, poorly configured, unwounded APIs, overly permissive privileges, exposed endpoints, poor authentication, poor inter-service communications and poor security policies, all of these remain susceptible even with formal security policies.

Another crucial issue within distributed microservice environments is information-flow security. Security-policy verification can be determined to check information flow between components whether it is permitted based on predefined security requirements [2]. The latter can enhance trust to architectural security properties and aid detecting the breach of the information-flow policies. Nonetheless, a running application cannot be assessed on security considering only vulnerabilities that can be detected using only a static policy check. Implementation vulnerabilities, wrongly set up configuration errors, or reached-out interfaces, or unforeseen inter-service interactions can be exploited by an attacker, necessitating dynamic security testing.

Authentication and authorization are essential security master-requirements of microservices since services often make APIs accessible to users, applications and other services. Access-control and identity mechanisms are thus needed to make sure that the protection against malicious actors only limited access to legitimate entities and can invoke authorized operations [3]. It will be authenticated to determine the identity of the requesting entity and authorized to give it permission i.e. permission to access a certain operation. The weaknesses in either system could possibly enable the access to services and sensitive resources in an unauthorized manner. There are also end-user authentication and authorization studies suggesting the need to take into account safe access-control systems in microservice systems [4]. The introduction of such mechanisms is however just a part of the security lifecycle and the performance of such mechanisms in the real environment of attack should be demonstrated practically as well.

Due to the nature of microservices, it is especially difficult to effectively carry out a thorough security assessment. A single application may contain dozens of services of various technologies, APIs, dependencies, authentication schemes and configurations. Service versions as well as changes in the deployment environments can persistently change the attack surface of an application. Manual penetration testing can be used to uncover important vulnerabilities, but the complexity of large-scale testing (in terms of time and experience) of dynamically evolving and large microservices can be a significant factor. In addition, manually carried out assessments can be inconsistently covered, and cannot be used to automatically carry out security validation.

A systematic study of the topic of microservice security established the existence of various security issues and protection strategies, such as authentication, authorization, communication security, data protection, service isolation, and infrastructure security [5]. These results indicate that microservices security cannot be dealt with in a single point of defense. Instead, security analysis should examine a number of architectural layers and interfaces to examine how services interact. A powerful evaluation mechanism ought to, therefore, view the uncovered services and APIs,

consider their security settings, identify vulnerabilities, confirm their exploitability with the use of controlled testing, and include the succeeding results by the magnitude of their effect.

The protection of integrity is also a significant responsibility of microservice-based infrastructures as the attackers can be located inside an organization and compromised parties or the rogue ones are a possibility as well [6]. The distributed environment creates a lot of trust between users, services, containers, administrators and infrastructure elements. Security testing cannot, then, be confined to interfaces with applications that are visible to the eye. Privileged relation and internal service communication should also be looked at in order to understand possible avenues, through which an attacker might transit between two different compromised components.

Instead of resolving on such challenges, automated penetration testing will provide the opportunity to conduct automated reconnaissance, service discovery, vulnerability scanning, controlled attack execution and analysis of security-results, in an organized workflow. When compared to the conventional vulnerability assessment, penetration testing attempts to establish the presence of vulnerabilities under scrutiny of being potentially exploited under controlled circumstances. These could be further improved by automation and security testing might be paired with continuous development and deployment. APIs, authentication, service endpoints, inter-service interactions and service configurations, as well as the remainder of the entities exposed by a specific framework can be reviewed in a framework that has been configured exclusively to test microservices.

In this way, the research proposes to conduct a Security Evaluation Model of Microservices-Based Applications through the Automation of the Penetration Testing. This framework is to provide a systematic and reproducible system of finding security vulnerabilities in distributed microservices environment. It is an automated reconnaissance, vulnerability identification, controlled penetration testing, analysis of the attack-surface and prioritization of risk based vulnerabilities. The aim is to enhance the visibility and consistency of security testing and lessen reliance on purely manual testing. The proposed framework aims at helping developers and security teams detect exploitable vulnerabilities and improve the security stance of microservices-based applications at all stages of the lifecycle by offering constant and operational security testing and confirmation.

II. RELATED WORK

Application architectures designed with microservices have attracted a lot of interest due to their scalability, modularity, independent deployment and flexibility. However, when applications are decomposed into multiple autonomously interacting services the size of the security attack surface is dramatically increased and there are also the issues of authentication, authorization, communication security, service isolation, identity management and persistent security checking. These issues have been addressed by the available literature in a variety of forms such as security governance, access control, information-flow validation, trusted execution, and defense-in-depth and zero-trust design.

Abidi et al. proposed that there is a security rule in governance of web services in comparison to specific microservices [1]. Their publication is interested in the use of particular security-oriented services to provide governance facilities in a distributed application environment. The method demonstrates the capability to separate security functions and individual application service thereby making it easy to centrally manage security policies [1]. However, governance-based strategies mainly aim at imposing security control measures as opposed to methodically finding exploitable vulnerabilities by automatically using automated penetration tests.

Gerking et al. explored refining and verifying information-flow security policies of cyber-physical microservice architectures based on components [2]. Their approach is with regard to checking the

security policies and information streams of distributed microservice elements. This type of verification by policies is useful in the case of detecting breaches in complicated interactions of service. Formal or component-level verification, however, does not always reveal a practical vulnerability which can be detected by an attacker or reconnaissance, vulnerability scanning, or controlled exploitation [2].

Guija et al. took their time to focus on identity and access on microservices based on 5G NFVs [3]. The relevance of their research is significant because dependable identity management and access control systems are imperative in service environments that have high distribution. Since microservices tend to communicate through APIs and must conduct service-to-service communication through authenticated means, any flaws in identity and authorization can provide unauthorized access to access [3]. Their work offers a significant base on the analysis of authentication and authorization controls, but lacks a detailed workflow of automated penetration-testing.

He et al. has focused on authentication and authorization of end users in microscopes. The paper identifies the need to have secure measures to verify the user and limit access to one-on-one services [4]. Authentication and authorization are some of the main locations of security boundaries since authorized access to resources may be obtained by attackers due to compromise or improperly configured security permissions. However, an evaluation system should not only develop based on the construction on the architectural protection processes but also undertake these controls dynamically.

A systematic mapping study of security in microservices and microservice architecture was done by Hannousse and Yahiouche [5]. Their work will provide a generic listing of security issues, threats and security mechanisms in microservice environments. The authors ensure it is clear that microservice security exists in diverse dimensions which include authentication, authorization, communication, data protection and infrastructure security. Their step-by-step approach offers a robust basis in coming up with holistic security assessment procedures. However, the research itself is more of a survey of the existing literature than a proposed systematic method of the automated penetration testing of the entire attack surface of microservices.

Ahmadvand et al. discussed the concerns related to protecting the integrity against insider threats within microservice-based infrastructures and suggested a security model based on the threats identified [6]. Their research demonstrates relevance of mitigating the threats that can be posed by trusted/or partially trusted parties in the distributed settings. These threats are especially critical in microservice ecosystems where there is the potential of numerous services, containers, administrators, and infrastructures to be interconnected. Nonetheless, insider-threat protection does not imply automated external testing of attacks and is only one of the aspects of security testing.

Bánáti et al. proposed an authorization and authentication orchestrator to the microservice based software architectures [7]. They organize their approach and distribute security choices among a variety of services and also respond to the challenge of ensuring the ability to have quality access-control in a distributed system. Centralization enables centralized orchestration, which can be useful to guarantee security-policy consistency, but is not a sufficient distinction between the exposures of interfaces, APIs, or service dependencies having exploitable vulnerabilities.

Brenner et al. investigated secure microservices on cloud computing on Intel Software Guard Extensions (SGX). Their work takes trusted execution environment as a tool to secure sensitive microservice workloads and minimize risks of compromised infrastructure [8]. Hardware-assisted

isolation can potentially be effective to promote integrity and confidentiality, but will not eliminate vulnerability on application, API, authentication and service-configuration levels. As a result, there is a need of complementary security assessment methods.

Chen has explained the threat of the higher attack surface associated with the idea of microservices and pointed out security issues associated with the decomposition of the application [9]. The research contends that abstraction and service decomposition adds more interfaces and communication channels required to be secured. This opinion can be applied to automated penetration testing because effective security testing needs a systematic exploration and research of the surfaces at one's disposal, points of contact, and communication routes.

Fetzer makes the need to develop (or write) critical applications on the basis of microservices, as well as makes the points of taking into account security and reliability as connected with the distributed architecture [10]. The text supports the need of adequately architectural security operations in the microservice as used applications with sensitive environments. Nonetheless, building practices cannot utilize architectural recommendations alone in order to sustain evidence about vulnerabilities exploitation of deployed systems.

Jander et al. proposed role authentication and a defense-in-depth approach to the microservice systems [11]. Their efforts show the importance of having various complementary security measures as opposed to having a single protective measure. Jander et al. [12] went as far as devising a workable approach of defense in depth of microservice systems. The methodologies are relevant to the layered security assessment as the susceptibilities may be in the different layers of architecture. Nevertheless, they still mainly focus on prevention protection, instead of automated validation based on attackers.

Kraufer et al. exchanged the applicability of using microservices to a secure application in smart-city cloud-based settings [13]. Exchange security requirements, which may be grounded on the cloud-based distributed applications and how microservice can be utilized in the applications that are security sensitive. Nevertheless, smart-cities specialization security does not have the generalized form of automated vulnerability discovery and prioritization in the smart microservice deployment setting [13].

Lastly, Zaheer et al. suggested EZTrust, a network-neutral zero-trust parameterization that is suitable to microservices [14]. The solution involves the application of the zero-trust that tends to reduce implicit trust between services as well as increasing service-level isolation. Although zero-trust architecture can play a vital role in mitigating lateral movements which are unauthorized, security assessment is still needed to ascertain whether services and interfaces deployed can indeed resist exercise of attacks.

Overall, microservices are offered a large portion of security infrastructure, including governance [1], information-flow verification [2], identity and access control [3], [4], systematic security analysis [5], integrity protection [6], authentication orchestration [7], trusted execution [11], defense-in-depth [12], and zero-trust parameterization [14] in the current literature. The reviewed literature, however, predominantly concerns individual security controls, or protection measures when it comes to architectural protection. The automated and integrated a security evaluation framework, including service discovery, vulnerability identification, controlled penetration testing, analysis of the attack surface and risk prioritization-based approach are still needed on a microservices basis. The proposed research fills this gap by targeting automated penetration testing as a formalized way of making sure the applications provided by the microservices-based system implemented are secure.

III. AUTONOMOUS ATTACK-SURFACE INTELLIGENCE AND PENETRATION VALIDATION FRAMEWORK

The offered framework offers an automated and systematic method of the security evaluation of a microservice-based application. The offered framework targets the overall microservice attack surface that includes the exits to the services, APIs, authentication and authentication, and inter-service communications, configurations, dependencies and the attack path in contrast to the traditional approach to assessing security that usually targets one vulnerability or a security control isolated. The framework will help in repeatable and continuous security testing across dynamically fixed microservice environments.

3.1 Framework Architecture

As shown in figure 1, there are six functional building blocks, which are: Microservice Discovery, Attack-Surface Mapping, Vulnerability Intelligence, Automated Penetration Validation, Risk-Based Assessment, and Security Reporting. These elements work as a whole pipeline without neglecting feedback between evaluation and reappraisal processes.

The front-end of the framework is called Microservice Discovery. It detects active microservices, presented endpoints, API, ports, communication protocols, as well as accessible service metadata. Information can be accessed via deployment setups, API, service collections, container catalogs and managed network reconnaissance. Automated discovery is in the way that microservices can be automatically generated, scaled, altered or dropped when an application is running. Credible security evaluation can then be based on having a current service inventory.

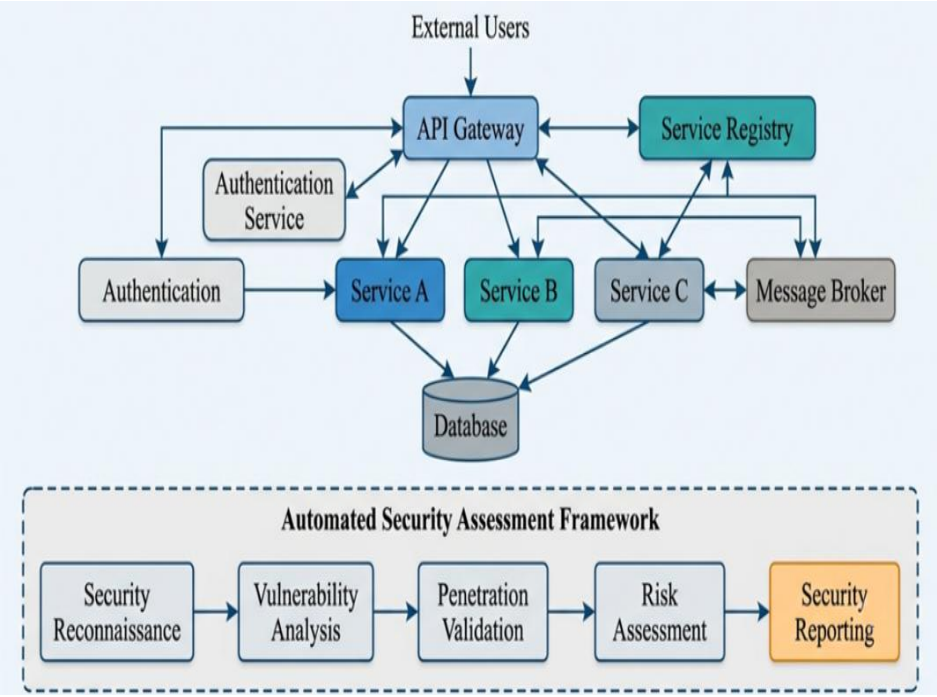


Figure 1. Architectural Framework for Microservices Security Assessment

The Attack-Surface Mapping item converts the received information to model of security exposure of the application. Attack surface mapping is shown in figure 2. It identifies interfaces (external services, internal services), API endpoints, authentication interfaces, management interfaces, communication channels and service dependencies. The component also symbolizes connections

between services to discover potential communication and attack paths. Such mapping will allow the framework to differentiate between components that are directly exposed and internal ones, which might become exposed after one more service is compromised.

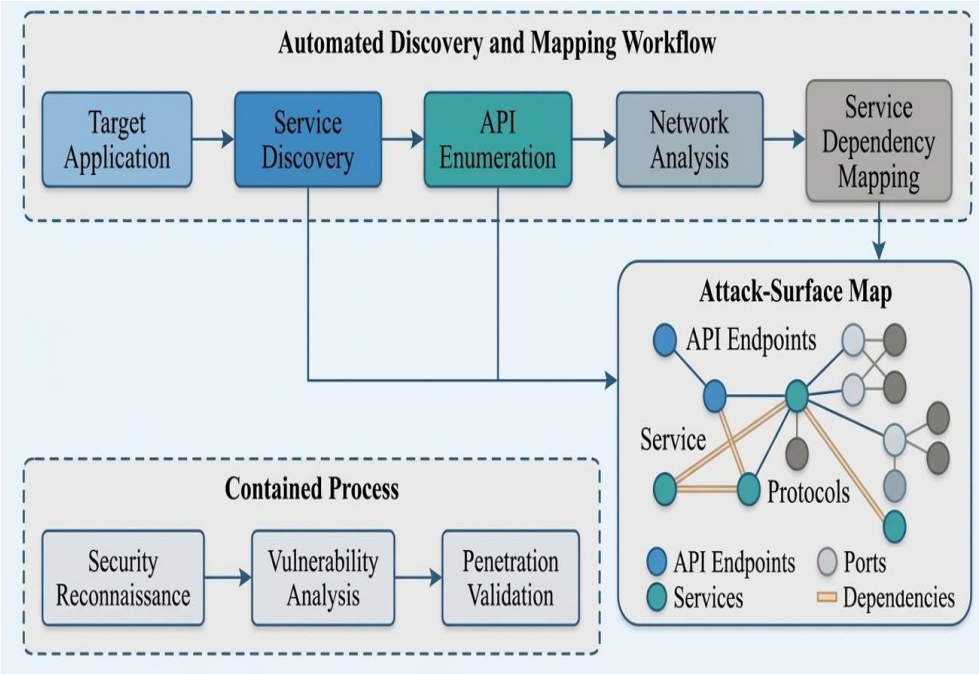


Figure 2. Automated Framework for Attack Surface Identification and Mapping

3.2 Security Reconnaissance and Exposure Analysis

The reconnaissance stage will involve a detailed examination of the microservices as detected. It analyzes service technologies, API design, authentication, and authorization requirements, communications protocol, and open (exposed) ports and security settings. This will be achieved by crafting a comprehensive profile of security of every service.

Focus is given to API endpoints because API is a crucial entry point of microservices apps. Hypernym The framework examines endpoint accessibility, request and response behavior, authentication requirements, authorization limits and possible sensitive operations. Service-to-service communication is also taken into consideration in that when one of the services is weak it may provide an attacker with an opportunity to access other services.

The framework categorizes the components found in it based on their characteristics of exposure. Public services which can be accessed by anybody are explored as a potential point of entry and internal services are explored with reference to service dependency and potential lateral mobility. This will give a better view of the attack surface of the application, rather than assessing its microservices separately.

3.3 Vulnerability Intelligence and Correlation

The Vulnerability Intelligence component, as shown in Figure 3, finds the security vulnerabilities throughout the environment of the application that are discovered. The vulnerability scanning, configuration analysis, API security testing, dependency analysis, authentication testing, and authorization assessment are part of the assessment.

Correlations between vulnerability information and known vulnerability databases and software-version information can be used to detect vulnerabilities due to old or vulnerable components. In

the meantime, configuration analysis will inspect application specific vulnerabilities which might not have a vulnerability identification number. This can be as an unwarranted service exposure, insecure authentication, too much privilege, insecure API configurations and insecure access permissions.

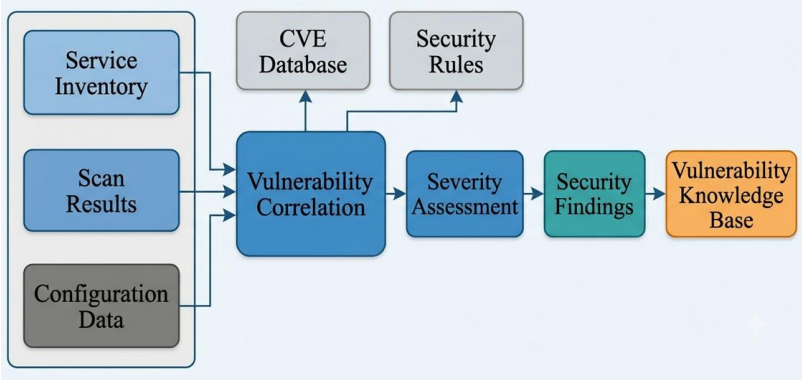


Figure 3. Intelligent Vulnerability Correlation and Threat Analysis Framework

One of the most important peculiarities of the proposed structure is that of the correlation of contextual vulnerability. The framework does not isolate the vulnerabilities, each vulnerability is connected to the service it affects, service dependencies, level of exposure and security controls provided. This enables the framework to establish vulnerabilities relationships and establish whether several vulnerabilities can be combinations to achieve a larger attack path.

3.4 Automated Penetration Validation

The key aspect of security-testing capabilities of the framework is the Automated Penetration validation element. It aims at ensuring the validity of selected vulnerabilities with the help of controlled and approved penetration-testing services. Detecting vulnerabilities can generate a false positive or can fail to credibly indicate the real-world security consequences of vulnerability. Controlled penetration validation gives extra details on the possibility of a weakness that is detected being exploited.

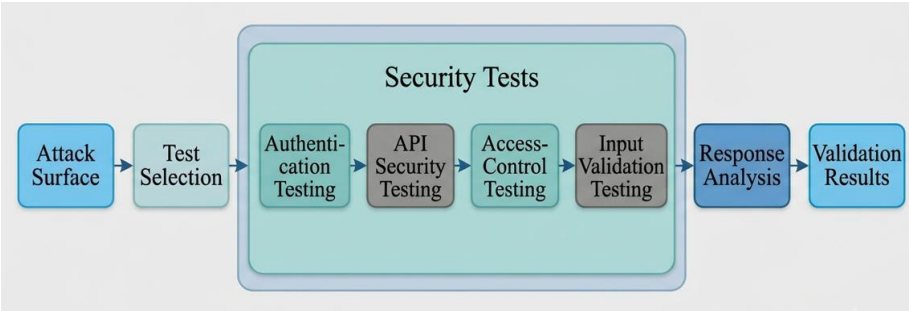


Figure 4. Automated Penetration Assessment and Security Validation Model

The framework develops appropriate security-testing conditions based on the aptitude of determined service and vulnerability. It is shown in figure 4. Testing has the potential to deal with authentication vulnerabilities, authorization breaches, and unsecured systems API behavior, open endpoints, configuration vulnerabilities and improper service-to-service access. Testing is constrained by operations defined limits that must be achieved to ensure that testing operations do not inflict any damage to the application resources or disrupt valid services.

The validation phase logs the actions of the target application and gets to know whether the anticipated security measure is effective in averting the tested behavior. The results are divided into validated and non-validated results, and it is possible to isolate the confirmed and weak bites of results with the possible and inconclusive ones. This will enhance the credibility of the automated security evaluation and minimize the unwarranted remediation efforts due to unresponsive results not verified.

3.5 Risk-Based Security Assessment

Once the vulnerabilities have been identified, a Risk-Based Assessment element analyzes the security worth of identified vulnerability. Risk assessment and prioritization model is shown in figure5. Risk assessment considers different contextual issues like the degree of vulnerability, exploitability, service exposure, authentication requirement, privilege requirement, impacted functionality, dependency relationship and potential implication on confidentiality, integrity and availability.

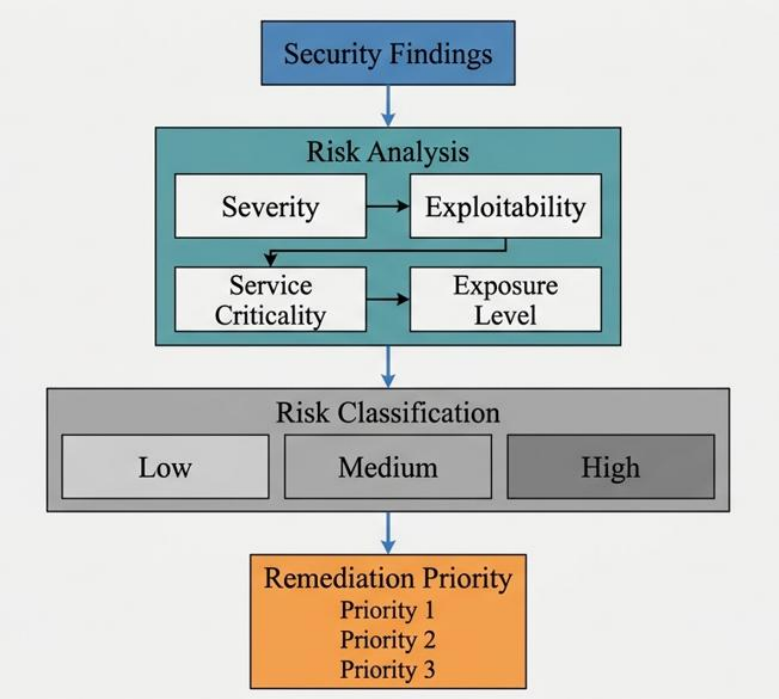


Figure 5. Risk-Aware Security Assessment and Threat Prioritization Framework

The framework puts more consideration on the vulnerabilities of highly exposed or highly connected services. When they have many dependent components, a vulnerability in one service can be insignificant, where an automated fault in a central business service, authentication service or gateway provides. The consequence of this is that architectural context is an important factor in priority development as far as remediation is concerned.

The risk-assessment stage classifies the results into appropriate prioritization groups and offers a clear conformity of vulnerabilities and services impacted. This ensures that the security teams can focus on the remediation activities on the most critical security vulnerabilities without forgetting the lower-priority ones.

3.6 Security Reporting and Visualization

The Security Reporting component transforms results of assessments into operational and formal security information. The reports store the affected microservice, vulnerability type, exposed interface, validation status, security-impact, priority, what is recommended to do to fix the vulnerability and the attack path.

The service and application-level security views can be provided by the framework. Service-level reporting enables the developers to familiarize themselves with vulnerabilities in individual components, and application-level reporting makes security teams familiar with the distributed attack surface overall. Diagrams of dependencies between different services and verified attack paths could also be of help to comprehend how the vulnerabilities can spread via architecture.

The reporting component also has historical assessment information. New vulnerabilities can be analyzed based on past and present analysis to identify newly discovered vulnerabilities and vulnerabilities that have been resolved and retained open attack-surface.

3.7 Continuous Security Feedback

The prominent aspect of the suggested framework is its feedback loop that is constantly present. Applications of microservice are often deployed and scaled, configured, and reconfigured, and updated with new versions. These changes can introduce new terminals or weaknesses and can change the dependencies of existing services. The security assessment should not be a one-time assessment that is published before deployment.

The framework enables recurring assessment in which significant changes were uncovered. New service discovery information is compared with the former security profiles, in order to compute new elements of exposure, and links altered. The assessment of vulnerability and conditions of penetration can then be applied to the altered environment.
The overall process follows discovery, map, analyze, validate, prioritize, remediate and reassess cycle. The process is a closed-loop process that forms a methodical approach of ensuring security visibility throughout the application life cycle.

3.8 Overall Framework Characteristics

The proposed system is the combination of architectural and automatic testing of security to address the limitations of the separated vulnerability scanning and purely preventive security measures. The key aspects of it are that it will automatically identify the available services and provide a complete mapping of its attack-surface, correlates its vulnerabilities and risks contextually, provides penetration validation in a controlled manner, prioritizes its vulnerabilities based on risk-reward, reports in an organized manner and revisits them.

Having all these features merged into one framework, the suggested solution will provide a repeatable process of testing distributed environments of microservices. It aims to help security teams and developers to discover exploitable vulnerabilities, their architectural context, and remediate their priorities, as well as to validate the security posture of microservices based applications on an ongoing basis.

IV. SECURITY VALIDATION AND FRAMEWORK EFFECTIVENESS ASSESSMENT

The framework constructed was assessed to determine its roles in isolating security vulnerabilities, the justification of vulnerabilities through automated penetration test, and prioritization of the security outcomes in microservice-based applications. The metrics in the evaluation concentrate on the security coverage, effectiveness of detection, accuracy of validation, efficiency of the

assessment, and attack-surface visibility. The Assessment methodology is designed to explore the practical applicability of the framework and at the same time to possess a methodical and arduous security-validation process.

4.1 Evaluation Environment

The test infrastructure is a generous microservices-based application, it consists of individually deployable services that interact with each other through APIs. A part and parcel of the environment includes end user services, authentication services, business services, data services and internal communications interfaces. The different security settings and controlled vulnerabilities applied on the selected services are used to test the theory as to whether the framework would be able to identify and verify the security vulnerabilities. The security weaknesses detected in the environment pertain to the authentication, authorization, API exposure, outdated components and inadequate service-to-service access. The framework initially finds the deployed services and creates an attack-surface representation and then carries out vulnerability analysis and controlled penetration validation.

4.2 Security Detection Effectiveness

The first aspect of the evaluation focuses on the framework displaying the ability to identify security deficiencies among the different elements of microservices. Detection is approximated as the difference between the introductions of vulnerabilities into assessing environment and those that are found by the framework.

The framework is also shown to be broad-based as it conducts an assessment on numerous dimensions of security as opposed to assessing a specific vulnerability category. Authentication and authorization, API endpoint and paths of inter-service communication, service configurations and endpoint are looked at. The multidimensional testing of the framework also aids the framework to identify all vulnerabilities that may be unaddressed when services of the framework are individually tested.

4.3 Penetration Validation Effectiveness

The second assessment dimension looks at the ability to have detected vulnerabilities tested through controlled tests of penetration. Discoveries attributing to vulnerabilities generated during automated analysis are put under contentious security-validation frameworks. The feedback obtained is compared to determine whether the weakness that was identified causes the desired security status. This step improves the validity of the test to make a distinction between vulnerable environments and vulnerabilities that are viable to be tested practically in theory or controlled environments. The structure therefore provides superior decisions of remediation to security teams than detection of vulnerability.

4.4 Attack-Surface Visibility

The third evaluation dimension is used to measure the capability of the framework to detect and model security exposure of the microservices environment. Services that are externally available, found services, internal dependencies, API endpoints, routes between services-to-services are all considered in the analysis.

The resultant representation of the attack-surface gives a consolidated representation of the distributed application. It would also prove to be useful in micro services where one service may appear secure but its interaction with other services will prove to be at greater risk of security threat. The model therefore examines the flaws of the services as well as the relationships between interrelated services.

4.5 Assessment Efficiency

Assessment performance is assessed through testing the time and the operational work required to undertake security assessment in the micro services environment. Repetitive manual activities are eradicated by automated discovery, vulnerability analysis, penetration testing validation as well as reporting.

The framework enables the sustenance of recurring assessment cycles that permit conducting security testing following an update to an application, after configuration or a deployment transformation. The latter is important in the context of continuously evolving microservice environments where new microservices can be tested without the need to go through the full security testing process to be hand-tested.

4.6 Risk Prioritization and Reporting

The last evaluation dimension would see how the framework is able to categorize security results based on their contextual meaning. The vulnerability severity, exploitability, service exposure, privileges requirements, and dependency relationships are used to use their findings in the classification.

The resultant reports give the security team's data on services impacted, type of vulnerability, their validation status, paths to attack, and priorities to be fixed. This enhances the practical utility of the assessment data by converting high amounts of technical data into organized security data which has the potential to aid in remediation planning.

Overall, this evaluation indicates that the proposed framework will provide an integrated feature of security discovery, vulnerability detection and penetration validation, and an attack-surface analysis, a risk prioritization initiative, and continuous evaluation. Its mix of automated testing and the architectural framework make it suitable to security test dynamically evolving microservices-based software.

V. CONTEXT-AWARE SECURITY INSIGHTS AND AUTOMATED ASSESSMENT IMPLICATIONS

The suggested framework proves that conventional vulnerability scanning is not enough to conduct security tests on microservices-based applications. The distributivity of microservices introduces multiple security edges that are usually associated with APIs, service identities, authentication, authorization policies, and communications protocols, and dependencies among services. Therefore, the evaluation of the individual services individually may not highlight the weaknesses that can be experienced in inter-dependencies between inter-dependent parts.

Among the notable innovations of the framework is automated verification of penetration as well as mapping the attack-surfaces. The fact that it is possible to map the services that have been exposed and the relationships can contextualize future security testing. This enables discovered vulnerabilities to be taken into account based on their location in the application architecture, as opposed to based only on their technical attributes. Such context is particularly required in an effective way when a vulnerable service serves as the connectivity point towards a series of other services.

Another weakness of automated vulnerability scanning that the framework deals with is the use of controlled penetration validation. Vulnerability scanners can give results that require further validation, rather than controlled validation which gives more arguments surrounding the

applicability of vulnerabilities found in real world scenarios. This would assist in minimizing unwarranted remediation effort in response to unverified discoveries and also give more solid evidence on confirmed vulnerabilities.

The second important implication is on the applicability of the framework to the highly dynamic microservices world. Services may be updated, scaled or replaced separately, or be introduced, which results in a change in the attack surface. Such changes can be re-done by performing security analysis again since it is possible through the continuous review cycle suggested. This would aid in changing the trend towards continuous security validation and not periodic security assessment.

However, automated security penetration testing does not completely replace the art of expert security testing. Human exploration might be required to deal with complex business-logic vulnerabilities, multi-phase attacks, and application-specific security needs. The framework can thus be considered as an automated security-validation tool which enhances better assessment coverage and consistency in addition to aiding security professionals.

Overall, the provided strategy offers a context-dependent and reproducible security evaluation strategy, which incorporates the discovery and vulnerability testing, penetration testing, risk-based prioritization, and continuous reassessment of modern microservices applications.

VI. CONCLUSION

Applications that were designed with microservices present tremendous opportunities in scalability, modularity, independent deployment and continuous provision of applications but since microservices are distributed, they offer a complex and ever-changing security environment. A number of services, APIs, authentication schemes, routes of communication, dependencies, as well as infrastructure elements, collectively enhance the choice of attack potential. Accordingly, legacy security assessment procedures, based upon disconnected services or regular ad-hoc testing, may lack sufficient efficacy to provide sufficient information regarding security of the existing microservices situations.

This research proposed a Security Assessment Framework of Microservices-Based Applications by Automated Penetration Testing to tackle these challenges. The architecture integrates automatic discovery of microservices, attack-surface mapping, vulnerability intelligence, controlled penetration validation, risk-based assessment, security reporting, and continuous assessment. Continuous security weaknesses can be discovered and analyzed in the framework of their service and dependence instead of being viewed as an independent finding because of this integrated architecture.

The evaluation of the framework revealed that it has the potential to be applied to implement the process of security vulnerability disclosure of APIs, authentication, authorization restrictions, service dysfunctions, and inter-service interactions. Automated penetration validation is also used to help in the validation of risks identified and provide additional information about the actual meaning of security of the risks. The risk-based assessment mechanism facilitates the systematization of the results based on exposure, exploitable nature, affected functionality and architecture dependencies focusing to facilitate more systematic remediation.

The second useful capability of the proposed framework is the power to guarantee comprehensive security verification. The attack surface of applications may vary depending on modifications to

services, configuration, or dependencies, or interfaces exposed. When the number of such changes is regularly evaluated automatically, they are processed in the next round of security-validation.

The proposed structure is, therefore, a process of progressive and repetition of a strategy to improve the security testing of dynamically moving microservices applications. Future studies of intelligent attack-path analysis, anomaly detection, adaptive generation of penetration-tests, real-time cloud-native security monitoring and constructing new infrastructure with DevSecOps pipelines can further develop the framework. The framework may also be improved with these extensions and their possibilities to help with the process of ongoing and context-sensitive security verification within a large-scale microservice system.

REFERENCES

[1] Abidi, S., Bennacer, H., &Kazar, O. (2019). A web service security governance approach based on dedicated micro-services.*Procedia Computer Science*, *159*, 2247–2254. https://doi.org/10.1016/j.procs.2019.09.192
[2] Gerking, C., Dziwok, S., Heinzemann, C., &Schäfer, W. (2019). Component-based refinement and verification of information-flow security policies for cyber-physical microservice architectures.In *2019 IEEE International Conference on Software Architecture (ICSA)* (pp. 61–70).IEEE.https://doi.org/10.1109/ICSA.2019.00015
[3] Guija, D., Shousha, M., &Openo, E. (2018). Identity and access control for micro-services based 5G NFV platforms. In *Proceedings of the 13th International Conference on Availability, Reliability and Security (ARES '18)* (Article 66, pp. 1–10).Association for Computing Machinery.https://doi.org/10.1145/3230833.3233255
[4] He, X., Ren, L., & Li, Z. (2017). Authentication and authorization of end user in microservice architecture.*Journal of Physics: Conference Series*, *910*, Article 012060. https://doi.org/10.1088/1742-6596/910/1/012060
[5] Hannousse, A., &Yahiouche, S. (2021). Securing microservices and microservice architectures: A systematic mapping study.*Computer Science Review*, *41*, Article 100415. https://doi.org/10.1016/j.cosrev.2021.100415
[6] Ahmadvand, M., Pretschner, A., &Kelbert, F. (2018). Integrity protection against insiders in microservice-based infrastructures: From threats to a security framework. In P. Milazzo, D. Varró, & M. Wimmer (Eds.), *Software Technologies: Applications and Foundations* (pp. 419–434). Springer.https://doi.org/10.1007/978-3-030-04771-9_31
[7] Bánáti, A., Kertész, A., &Németh, Z. (2018). Authentication and authorization orchestrator for microservice-based software architectures.In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 1180–1185).IEEE.https://doi.org/10.23919/MIPRO.2018.8400215
[8] Brenner, S., Wulf, C., Goltzsche, D., Weichbrodt, N., Lorenz, M., Fetzer, C., &Kapitza, R. (2017). Secure cloud micro services using Intel SGX. In L. Y. Chen & H. P. Reiser (Eds.), *Distributed Applications and Interoperable Systems* (pp. 177–191).Springer.https://doi.org/10.1007/978-3-319-59665-5_13
[9] Chen, C. (2019). With great abstraction comes great responsibility: Sealing the microservices attack surface. In *2019 IEEE Cybersecurity Development (SecDev)* (pp. 101–110).IEEE.https://doi.org/10.1109/SecDev.2019.00028
[10] Fetzer, C. (2016). Building critical applications using microservices.*IEEE Security & Privacy*, *14*(6), 86–89. https://doi.org/10.1109/MSP.2016.129
[11] Jander, K., Braubach, L., &Pokahr, A. (2018). Defense-in-depth and role authentication for microservice systems.*Procedia Computer Science*, *130*, 456–463. https://doi.org/10.1016/j.procs.2018.04.047

[12] Jander, K., Braubach, L., &Pokahr, A. (2019). Practical defense-in-depth solution for microservice systems.*Journal of Ubiquitous Systems and Pervasive Networks*, *10*(1), 17–24. https://doi.org/10.5383/JUSPN.10.01.003
[13] Krämer, M., Frese, S., &Kuijper, A. (2019). Implementing secure applications in smart city clouds using microservices.*Future Generation Computer Systems*, *99*, 484–498. https://doi.org/10.1016/j.future.2019.04.042
[14] Zaheer, Z., Liang, C., Yu, S., Bavier, A., & Rexford, J. (2019). EZTrust: Network-independent zero-trust perimeterization for microservices. In *Proceedings of the 2019 ACM Symposium on SDN Research (SOSR '19)* (pp. 89–95). Association for Computing Machinery.https://doi.org/10.1145/3314148.3314352