ESP IJAST
ESP International Journal of Advancements in Science & Technology
ISSN: 2583-9233/ Volume 4 Issue 3 July-September 2026
Paper Id: IJAST-V4I3P105/Page No: 65-75

Original Article

Detecting Excessive Service Permissions in Containerized Applications through Runtime Behaviour Analysis

Karthik Rao
Cloud Security Researcher, Independent Research, India

Received Date: 25 September 2026 Revised Date: 30 September 2026 Accepted Date: 01 October 2026

Abstract: *Containerized applications are now a core component of today's cloud native infrastructure due to their benefits of portability, scalability, flexibility in deployment and efficient use of resources. As a result, there has been an explosion of container technology; however, this growth has also led to many security concerns. One of the main areas where these concerns lie is in regard to too many permissions being given to services and unauthorized access to system resources. Many times applications will have more permissions than they actually need to run properly. This creates greater opportunities for potential damage caused by compromised services, malicious actions or privilege escalation attacks. Most traditional methods for evaluating service permissions rely on evaluating the configuration of the service (static analysis). This method does not allow for evaluation of how the service executes. Therefore, the static method cannot determine if there are security risks that occur at runtime. This paper proposes a method for identifying excessive service permissions in containerized applications using runtime behavior analysis. The proposed method evaluates how the application executes, the types of system calls made, what resources are accessed, how processes interact with each other and how permissions are utilized at the service level to create a behavioural profile of the application. Once a behavioural profile is created it can be compared to the configured permissions and expected operational requirements of the application to determine if the application is using unnecessary permissions, attempting unauthorized access or executing in a dangerous manner. The proposed method includes continuous monitoring of the application's behavior, mapping of permissions to behaviors, identification of anomalies in behavior and assessment of risk to provide continuous evaluation of the application's security posture. By associating the observed behavior of the application at runtime with the permissions assigned to the application, the proposed model seeks to increase visibility into permissions used by the application, decrease unnecessary access rights and increase isolation among containers.*

Keywords: *Container Security, Runtime Behavior Analysis, Excessive Permissions, Least Privilege, Containerized Applications, System Call Monitoring, Privilege Escalation, Anomaly Detection, Cloud Native Security, Access Control*

I. INTRODUCTION

Container technology is a major development in current software engineering and cloud computing. It allows developers to bundle applications with all necessary dependent libraries, files, etc., so that they can run on various operating systems. Dockers and Cabernets have made the use of container architecture much simpler for organizations. They help make application deployment, orchestration, scaling, and infrastructure management easier. More organizations are using container-based environments to develop micro services, distributed applications, CI/CD, and cloud-native services. Containerization provides many benefits to organizations; however, it also creates several security risks related to application isolation, resource allocation, authentication, and authorization.One of the main security concerns with using containerized applications is that you are giving too much permission to the services. Too much permission can occur when you give an application, container, process, or service access to more things than what is needed for them to do what they were created to do. An example would be a web application which only needs to access certain areas of the file system but is given greater than necessary file system permissions; or an application that has been given root access (privileged execution) when it doesn't need to have that level of permission. Another example would be a micro service which only communicates with other micro services but is configured with broad network access. Giving these kinds of excessive permissions increases the number of ways that someone can attack your system, and creates opportunities for malicious behavior once an application has been compromised.

The principle of least privilege serves as a fundamental framework for resolving this issue. The principle states that each application component should be granted only the minimum amount of privilege needed to perform its assigned duties. First, application requirements can vary widely as the application progresses through different stages of development, deployment, scaling, and runtime execution. Second, container images may contain default privilege settings. Third, the orchestration configuration may provide additional capability. Lastly, application dependencies can require permissions that

This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/2.0/)

are not readily apparent at the time of initial deployment. Thus, administrators often provide more extensive permissions to avoid potential disruptions in service delivery. This results in the establishment of long-term vulnerabilities in security.

Conventional security assessment methodologies focus on examining container configuration, deployment manifest, access control policy, security context, and image level setting. Static analysis identifies potentially insecure configurations such as execution with elevated privilege, excess Linux capability, un-restricted file system access, and excessive resource permission. However, configuration based assessment cannot determine if assigned permissions are truly necessary for the executing application. It is possible that a container has been granted many permissions but will not utilize those permissions at any point in time during the execution of the application. On the other hand, it is possible for an application to access resources in an unexpected manner or execute in a suspicious manner even though the initial configuration of the container was quite restrictive. Thus, simply assessing permissions based on configuration may result in an incomplete view of the security exposure.The use of Runtime Behavior Analysis (RBA) provides another avenue for overcoming some of the drawbacks associated with using only static configuration settings. Rather than being dependent on static configuration settings, RBA monitors the interactions of applications with the execution environments they run in. Some of the most common behavioural indicators monitored by RBA are: System Calls, Process Creation, File System Operations, Network Connections, Resource Access Requests, Privilege Related Operations, and Interactions with Kernel Interfaces. The results of monitoring these behavioural indicators will allow developers to understand what permissions an application is exercising when it runs. When the results of monitoring are correlated with the permissions that have been granted to the application, then security controls will be able to determine which permissions are regularly exercised, which permissions are rarely exercised, and which permissions may represent anomalous activity.

System call monitoring in containerized environments offers special value due to the way that applications use operating system resources via kernel mediated interfaces. Monitoring system calls like opening files, executing processes, modifying permissions, and making network connections provides insight into execution behaviors. Runtime security products can collect execution events and associate those events with container identities, process information, and deployment configurations. While collecting execution events is essential, context-based analysis of the application's function, its expected execution behavior, the responsibility of the service, and the association of the activity to the permissions of the service is also necessary for effective threat detection.The next big challenge with this type of analysis is identifying what constitutes legitimate behavior by the application and what constitutes a use of privileges that is either suspicious or simply unneeded. Applications running as micro services will often exhibit varied behaviors based upon incoming requests, the nature of the workload being processed, the dependencies of other services, and the current operational state of the system. This means that an application can show several different ways of executing correctly. A privilege that appears to be unneeded at a given point in time may actually be required to support the application when it experiences a heavy workload or undergoes maintenance. Simple approaches to identifying excessive privileges by assuming that all privileges not used during a certain period are unnecessary could result in false positives and possible disruption to application operations. For reliable identification, there needs to be a baseline for behavior that represents acceptable performance of the application under various operational scenarios.

The growing sophistication of container orchestration makes it even more important to continuously evaluate permissions. Cabernets based environments can have many interconnected services, dynamically generated containers, service accounts, and policy for workloads. Permissions can come from different places such as the configuration of the container; the capabilities of the OS; the ownership of files on the system; service identification; and authorization at the orchestration level. Therefore, assessing if there are too many permissions will require evaluating how the system is behaving at run time, as well as the context in which the permissions exist. This assessment should also include evaluating changes in the lifecycle of containers, updates to applications, variations in workload, and newly added dependencies between services.The focus of this study was to detect excessive service privileges within containerized applications using runtime behavior analysis. We proposed a behavior-based security assessment methodology which correlated the assigned privileges with the observed behaviors of the application to identify excessive privileges and potential dangerous access behaviors. Our proposed conceptual methodology included; collecting runtime events, building behavioural profiles, mapping privileges, identifying anomalies, and assessing privilege risks. This was intended to create a formal connection between what an application was permitted to do and what the application did while running. Creating this formal connection would allow for better informed privilege refinement and improve adherence to the least privilege principle.

The main objective of my work is to create an analytical framework for detecting runtime permissions. This framework would be based on both the permissions configured by the user (static configuration) and how the application operates at runtime. My approach differs from other studies which have focused solely on the analysis of permission configurations; I believe it is necessary to monitor application activity continuously, and interpret the context of these actions

in order to detect potential misconfigurations or vulnerabilities. In addition to creating a new methodology for identifying violations of permission policies, I recognize the importance of reducing false positives, maintaining application performance, and being able to respond to changes in operations. Ultimately, I hope that my research will contribute to creating a better understanding of how we can effectively manage permissions in a dynamic environment, such as containers. I believe that this will lead to improved isolation among applications, decreased exposure to attacks, and enhanced security throughout our modern cloud native infrastructure.

II. RESEARCH PROBLEM AND MOTIVATION

The growing popularity of containerized application development has led to an increased need for effective ways to manage the use of unnecessary service privileges in order to reduce risk of unauthorized access to resources, privilege escalation, and other possible security vulnerabilities, while maintaining continuous availability of applications. While current security standards for containerized applications offer configuration-based controls, these controls may not accurately reflect the level of privilege that is required by applications as they run. It is common practice to give services broad privileges to facilitate easy deployment, compatibility, and/or to meet unpredictable operational needs. However, this type of practice can lead to a risk of unauthorized access to resources, privilege escalation, and other possible security vulnerabilities in a containerized environment.A major research problem exists in evaluating whether the permissions granted to an executing program align with its valid operational behavior. Although static assessment techniques can recognize possible vulnerabilities through configuration issues, these methods cannot consistently evaluate if each permission granted is required based on the variety of workloads that will occur.

Likewise, real-time monitoring can record application actions; however, the data obtained from the monitoring must be evaluated relative to the responsibility of the service being monitored, the context in which it operates, and the current access controls implemented. If there is no such evaluation of the relationship between the activity being performed by the application and the operational context of the application, then it is possible that the security mechanism may produce excessive alarms or inappropriately suggest the removal of permissions needed to execute the application in a valid manner.The dynamic nature of cloud-native environments is also a major concern. Cloud-native environments can have container instances created, terminated, updated, or scaled on an automated basis as well as having applications change their behaviors based on workload demands and dependency relationships. Therefore, permission assessment mechanisms must take into account how the execution patterns of applications are evolving and be able to create accurate and consistent behavioural profiles over time. Additionally, permission assessment mechanisms should be able to differentiate between privileges that are actually not required by an application, and privileges that are infrequently used but necessary during extreme circumstances.

This research is intended to address these issues through a runtime behavioural-based methodology for identifying excessive permissions. The proposed method will associate the observed behavior of the system (i.e., system calls, file system interactions, process activities, and network operations) with the permissions provided to each containerized service. Based on the associated behaviors, behavioural baselines may be used to identify unused permissions, unauthorized access attempts, and deviations from normal operational practices. Assessments generated using the methodology may enable administrators to prioritize permission related risks and develop more restrictive security policies.The main goal is to find a middle ground for how we enforce security and maintain operationally reliable systems. The research will use both runtime observations and permission analysis to minimize excessive access rights, increase visibility into potential threats, and better support the concept of least privilege in containerized applications.

III. PROPOSED FRAMEWORK FOR RUNTIME BEHAVIOUR-BASED PERMISSION DETECTION

The suggested approach is based upon continuous monitoring of actual application activity at runtime (i.e., when the application is running) to determine if a container has too much service authorization. A link can then be established between the amount of service authorization given to each container and the amount of resources used by the running processes within each container. Each indicator provides insight into how the application utilizes operating system resources and whether the authorization provided to the application matches its legitimate operational needs. The suggested approach is made up of three related components: runtime behavior analysis, authorization usage analysis, and excessive authorization identification.

Each component contributes to a structured analysis process allowing security personnel to gain knowledge of how applications behave and identify authorizations that are not required. Additionally, the suggested approach accounts for variations in workload and dependencies among applications so that the likelihood of incorrect authorization assessments is minimized. By utilizing all three components, the suggested approach seeks to increase transparency regarding authorizations, reduce unneeded authorization rights, and facilitate appropriate utilization of the principle of least privilege within dynamic containerized environments.

A. Runtime Behaviour Monitoring

As long as applications are executed in a containerized environment, this phase will continuously monitor the actions of those applications and gather data on how they use resources. Monitoring data includes but is not limited to: system call activity; file access activity; process creation activity; network connection activity; and interaction with other operating system services/resources. When a monitored action takes place, the action is documented along with the identity of the container, process, and service that initiated the action. There are numerous methods available for collecting monitoring data. Some examples include using Linux kernel APIs or utilizing runtime security software to track execution activities while minimizing operational disruptions. Once monitoring data has been gathered, it can be used to develop behavioural profiles that describe what typical application behavior is during periods of differing workload.

B. Permission Utilization Analysis

The Permission Utilization Analysis (PUA) provides a view into how well an organization's configured access controls align with actual container application usage. PUA assesses whether the assigned permissions for each containerized service are truly needed to meet the operational needs of that service. The assessment includes evaluations of file system access, Linux capabilities, process privileges; network access control lists (ACLs), etc. PUA analyses all of the observed runtime events within a given timeframe, and compares them to the configured access control settings. This analysis identifies permissions that may be extraneous to meeting the operational requirements of the containerized services.

C. Excessive Permission Identification

Excessive permission identification utilizes behavior profiles and usage of permissions to identify possible unused or harmful permissions. This component recognizes permissions which are not utilized during representative periods of execution, permissions which exceed the required services, and unexpected attempts to use protected resources. A risk assessment process can then prioritize results based on the sensitivity of the permission, importance of the resource, and potential security consequences. Risk assessment can be categorized as essential, required under certain conditions, unnecessary, and high risk. Administrators responsible for security can utilize these categories to evaluate current permissions policies and establish the proper limitations for the utilization of permissions. All recommendations made by the risk assessment process must be verified through application compatibility testing prior to being enforced so as to avoid any disruption of service and maintain legitimate service operations.

IV. RUNTIME BEHAVIOUR ANALYSIS AND PERMISSION DETECTION MECHANISM

The use of runtime behavior analysis is crucial for detecting excessive permission levels in container-based applications. Runtime behavior analysis allows us to observe how our applications interact with the operating system during their operation. We will be using this data to compare what the application does to the permissions granted to it by the container configuration, security policy, and service level access control. Both types of observations can help you assess whether the permissions assigned to your services are appropriate.

Table 1: Runtime Behaviour Indicators and Their Permission Assessment

Runtime indicator	Observed activity	Permission assessment
System calls	Kernel-level operations	Identify unnecessary capabilities
File system access	File reading and writing	Evaluate file permissions
Process execution	Process creation and execution	Detect excessive execution privileges
Network activity	Incoming and outgoing connections	Assess network access requirements
Resource access	Access to sensitive system resources	Identify unauthorized or unnecessary access
Privilege utilization	Elevated permission usage	Evaluate privilege necessity

The system also utilizes a contextual assessment to prevent misinterpreting legitimate permissions as excessive. Although some application functions may only be performed rarely, they are still required for the purposes of maintenance, disaster recovery, or extreme workload situations. Therefore, permission recommendations should account for a typical time frame of operation as well as alternative operational scenarios. Potential risks can then be ranked based on the level of privilege required to access them, the possible consequences of the risk, and how often the risk has been observed in use. Once the administrator reviews the recommendations, he/she can create appropriate restrictions using container security settings. In addition to continuously monitoring for changes in application behavior, deployment configuration, and service

dependency, continuous monitoring allows the detection mechanism to adapt to new behaviors. management in cloud native application environments.

V. IMPLEMENTATION METHODOLOGY AND DETECTION STRATEGIES

The implementation methodology provides a formal process to evaluate excess service privileges by analyzing how applications behave during runtime in container-based applications. The methodology being presented uses container monitoring, execution-event collection, behavioural profiling, privilege mapping and risk assessment to determine if there is a gap in what privileges have been assigned to an application versus what an application requires. This method is intended to run on modern container-based platforms such as Dockers-based systems and Cabernets-managed infrastructure. Prior to running the application, container configuration data and service-level permission policies are reviewed to understand the current security posture of each application. After reviewing the current security posture of each application, execution-event data is collected from the runtime environment. Execution-event data includes; system calls, file system operations, inter-process communications, network activity and access to sensitive resources. This execution-event data is then used to create behavioural profiles of legitimate application operation under various load conditions.

The process compares the user's behavior patterns to the permissions they have been assigned to determine if any of those permissions are unneeded, too broad, or could pose a risk. In addition, there is an assessment of the context in which the application is being used (i.e., its function, how often it runs, changes in workload, etc.) Instead of automatically revoking permissions, the methodology makes data-driven recommendations that can be reviewed by administrators and then refined through controlled policy development. The next three sections describe the key implementation steps and how each contributes to successful detection of permissions.

Behavioural profiling examines collected execution events to create a baseline of typical operations for each containerized service. This process of profiling identifies the operations that are most commonly executed, the resources that are accessed, where communications are directed, and any other execution needs that are unique to the application. Once identified, these observed behaviors are compared against the security settings applied to the service to see if the assigned permissions accurately reflect how the service behaves. A good example would be a data processing service that needs to read from specific data sets, communicate internally within the company, but does not need full access to the host file system. Furthermore, historical execution information can help prevent false positives in classifying legitimate permissions as redundant because of infrequent execution.

A. Risk Classification and Permission Optimization

The ultimate implementation phase assesses the excessive permissions and the possible security risks that could result from them. This detection system will look for unused permissions; excessively wide access rights; unusual interaction between resources; and suspicious activities regarding privileges. The findings that are identified by this system will then be categorized based upon how necessary they are to the operation of the system and the level of security risk that they pose. An example would be if a user had unrestricted access to the file system versus having an unused permission that was tied to a non-critical application resource. In the former case, the risk associated with this user having unrestricted access would likely be higher than the risk associated with him/her having an unused permission. A risk assessment process can take into consideration the sensitivity of the privilege; the importance of the resource; what the observed usage of the resource is; and what may happen if someone unauthorized accessed the resource. Based on what is found through this assessment process, the framework will create recommendations for refining permissions. After reviewing the recommendations and performing compatibility testing, administrators can implement appropriate restrictions via container security policies. Regular re-assessment will help ensure that the permission settings for applications continue to be consistent with changing application needs while maintaining service availability and operational reliability.

VI. SYSTEM ARCHITECTURE AND OPERATIONAL WORKFLOW

The architectural design of the proposed runtime permission detection system based on behavioural characteristics of running applications provides a clear view of how the system will work and what parts it contains. It shows how containerized applications interact with runtime monitoring tools, behavioural analysis software, permission assessment software, and security administrators. The architectural design allows for continuous monitoring of the application while running, as well as continuous assessment of whether the service permission(s) granted to the application match its current operational needs. The architectural design uses both runtime event data and contextual permission data to find unnecessary permissions and possibly hazardous access patterns. In addition to being able to integrate with container management systems, the architectural design allows for integration with other existing security enforcement tools.

The process of operation starts with finding the current active containers, and checking the permission settings for each container. Then, the runtime monitoring components will collect data about the application's activity, which includes

what applications are being run, how long they are running, and what actions those applications are taking. After collecting the data, the monitoring components send it over to the behavioural analysis component. The analysis component creates a profile for every application, and compares the applications' actual behavior with its allowed behaviors. When the analysis component finds a mismatch between an application's behavior and its allowed behaviors, it notifies the detection component. The detection component determines whether the mismatch is a security concern, and if so, what the risk associated with it is. Once the detection component has completed its evaluation, it creates a list of suggested actions that an administrator can take to limit access to sensitive resources, and then submits them to the administrator for review. This system is designed to provide on-going monitoring and flexibility in limiting access to sensitive resources while minimizing disruption to applications' operations.

A. Proposed System Architecture

The suggested design for the system will include several components that work together to monitor, process, evaluate and analyze how containers execute. The Container Application Layer is what is currently executing (the service) and the workload it has to complete. The Runtime Monitoring Layer collects data such as system calls, file system operations, process activity and network interaction. The Event Processing Component takes all the collected data and creates associations between observations and the container identity and execution context.

The first step of the behavioural analysis component is to determine what constitutes "normal" operation of a system through a process of identifying repetitive tasks performed by an application as well as how that application utilizes available resources. Once established, the permission correlation component then compares those identified trends against defined access rights, such as file system permissions, Linux capabilities, and/or relevant service-level privileges. Based upon both the identified trend(s) and the level of sensitivity associated with the privilege being analyzed, the detection engine classifies the potential excessive permissions as low, medium or high risk. At this point, the security recommendation component displays any identified issues and makes recommendations regarding potential changes in permission settings to administrators.

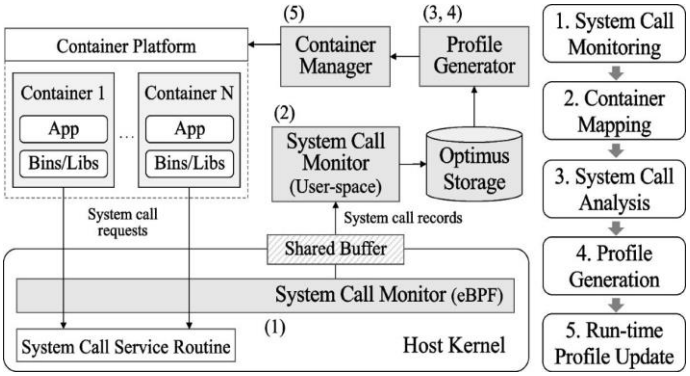


Figure 1: Proposed Architecture of Runtime Behaviour-Based Excessive Permission Detection

B. Operational Workflow and Security Evaluation

The continuous assessment of permissions on active containers during the operational workflow of the proposed framework is performed using several steps. First, the active containers are identified and their security configurations (capabilities assigned to each container, file system permissions, and service level access policies) are retrieved. Next, events that occur in applications running inside the containers are collected by the runtime monitoring systems and forwarded to the event processing component. Finally, the collected data is processed and organized to create meaningful behavioural records.

The analysis phase involves comparing actual use of applications by users to anticipated operational behavior and set up user permission levels. The framework will find permissions that seem excessive, overly broad, or used in an unusual manner. These findings will be evaluated for potential security risk and for potential operational issues. This evaluation produces an assessment of permission usage and possible security vulnerabilities for administrators to review.The last step provides suggestions for how to optimize permissions. Administrators are able to review the risks that were found, confirm what applications need to do, and implement the necessary restrictions using container security configurations. At a later date, administrators will monitor the application while it runs to determine if the new policy still allows the application to function properly. This on-going loop of reviewing and updating allows for continuous improvement in the area of security, and also reduces the probability of having unneeded access rights available in containerized environments.

VII. PERFORMANCE EVALUATION AND SECURITY ASSESSMENT

The evaluation of the proposed runtime behavior based permission identification framework will focus on assessing how well it can identify excess service permissions in an application environment, while preserving the operational performance of containerized applications. A systematic evaluation must include an assessment of the detection accuracy; the level of overhead associated with monitoring; how effective it is at identifying permissions; and its ability to adapt to changes in the applications' workload. These elements are relevant since any runtime security mechanism must protect against unauthorized actions without causing excessive delays in processing or impacting the availability of the application. This can lead to uncorrected vulnerabilities in the system. The next four subsections will describe four key evaluation areas to determine the effectiveness and practical applicability of the proposed framework.

A. Detection Accuracy and Identification Effectiveness

Detection accuracy is the degree to which the framework is capable of identifying excessive permissions within a containerized application. The evaluation compares the identified permission issues with a reference classification created based upon configuration inspection and application requirements. A true positive identifies a legitimate permission issue; a false positive misidentifies a legitimate permission as excessive; and a false negative misses a legitimate permission issue. Precision, recall, and F1-score provide quantitative measures of detection effectiveness. They will be used to assess whether using runtime behavioural evidence provides better permission assessments than those obtained from configuration-based inspection alone. To evaluate consistency of detection performance, testing should utilize various types of container workloads to demonstrate consistency in detection across different application environments.

B. Runtime Monitoring Overhead

The runtime overhead of continuous monitoring will assess the extra computational overhead that is needed to continuously monitor application behavior as well as processing execution events. Continuous monitoring of system calls, files system activities, network activity and process interaction can have a negative impact on the performance of a container. Therefore, the assessment needs to determine the CPU use, memory consumption, latency in event processing as well as application response times during both monitored and non-monitored states. Additionally, different monitoring configuration options could be investigated to determine how the amount of event collections is related to resource consumption. The implementation of continuous monitoring must be designed such that it minimizes unnecessary event collection but still provides adequate information for reliable assessment of permissions. Monitoring efficiency is critical for deploying continuous monitoring in resource constrained and large scale container orchestration environments.

C. Permission Utilization and Risk Classification

The purpose of Permission Utilization Analysis is to determine if the privilege levels assigned to each service align with the services' actual operational needs. To do this, the framework uses three criteria to classify permissions based on: 1) the frequency at which the privilege is used; 2) how necessary the privilege is to perform a task; and 3) what are the consequences if the privilege is abused.

Table 2: Permission Utilization and Risk Classification

Permission category	Observed behaviour	Risk level	Recommended action
Essential	Regularly required	Low	Retain
Conditional	Required under specific workloads	Medium	Validate and retain
Unnecessary	Not required by observed operations	Medium	Review and restrict
Excessive	Broader than service requirements	High	Reduce privileges
Suspicious	Associated with abnormal access activity	Critical	Investigate immediately

The following table illustrates a sample classification model that will assist organizations in conducting a systematic review of permissions. The classifications will provide administrators with a way to differentiate between those privileges which are essential to a service's function, those which are needed under certain circumstances, those which are not needed, and those which present a significant risk. Administrators should evaluate permissions over representative workload periods since many privileges that are never exercised in normal operations may be required in exceptional situations. By evaluating permissions contextually, administrators can make more informed decisions about when to remove permissions and thus reduce the likelihood of disrupting applications.

D. Scalability and Adaptability Assessment

Scalability testing needs to assess single-container deployments, as well as multiple-container applications, and applications managed through Cabernets. The data presented in this table is a proposed model for classification, and it is not

based on actual measured experimental results. Performance measures will need to be added to the data after testing and implementation.

VIII. SECURITY IMPLICATIONS AND PRACTICAL APPLICATIONS

The proposed runtime-based behavior-based permission detection approach offers significant security advantages to organizations that deploy containerized applications throughout cloud-native infrastructure. . Organizations who reduce excessive permissions will be able to decrease the potential consequences of compromised containers, unauthorized access to resources, and attempts to escalate privileges. Additionally, organizations will increase the strength of application isolation by ensuring that each service operates in a security boundary with the appropriate restrictions.The practical use cases of this framework include micro service architectures, enterprise-wide cloud computing platforms, distributed computing systems, and infrastructure managed by Cabernets. Additionally, the assessment of database container privilege levels and file system accesses can help organizations evaluate unnecessary privileges. This evaluation process will assist organizations in developing more comprehensive security policies, as well as providing more uniform management of permissions across all of their distributed applications.

Continuous compliance and security governance are two additional significant implications of this study. Using permission utilization records, organizations will be able to perform their own internal security audits, determine what is wrong with the configuration of their systems and show how they have made improvements to enforce "least-privilege" policies. Monitoring applications at run-time offers additional support for examining unusual application behavior and evaluating whether or not unauthorized actions were taken. In conjunction with current container security platforms, the new method can help enhance vulnerability scanning, image analysis, access control policy enforcement and runtime threat detection.However, when implementing this approach in practice, organizations will need to take into account various factors such as monitoring overhead, variations in behavior, privacy concerns, and application compatibility. In addition, organizations should not restrict privileges solely based upon the fact that a particular privilege has gone unused over a short time frame. Organizations will need to assess the recommendations for restricting privileges against typical workloads and atypical operational conditions. Runtime behavior-based permission assessment is an effective method to improve container security by providing continuous visibility into the container's environment; analyzing context surrounding the container's actions; and enabling organizations to optimize permissions in a more informed manner. Therefore, this approach enables organizations to move away from static permission configurations and toward adaptive and behavioural-driven security management.

IX. CHALLENGES, LIMITATIONS, AND FUTURE RESEARCH DIRECTIONS

The adoption of runtime-based behavior detection for permission control creates many technical and operational considerations which must be taken into account when designing an efficient container security solution. Although runtime monitoring offers a great deal of information regarding application behavior, the proper interpretation of event data collected from the runtime environment will depend upon the developer's understanding of the application's intended functionality, execution requirements, and workload characteristics.

Applications running as containers tend to run in very dynamic environments where service interactions, resource demands and execution patterns vary continuously. Variations in behavior profiles can make it difficult to separate normal activities from potential threats. Additionally, continuous monitoring can add significant overhead to system performance especially in large scale systems with hundreds or thousands of containers and micro services. Finally, another area of concern is determining permissions that are seldom used but require a level of access for special operational scenarios. Restricting those permissions could lead to decreased application reliability and service availability. An effective method for detecting permissions will need to strike a balance between providing effective security, minimizing monitoring costs, and maintaining service availability. The subsequent sections will address major issues and suggested methods for improving detection capabilities for future studies and practical implementations.

A. Challenges in Runtime Behaviour Interpretation

The challenge in interpreting runtime behavior lies in the fact that application execution patterns vary greatly based upon workload conditions, user requests, service dependencies and deployment configurations. For example, a specific permission may never be used under normal operating conditions; however, this same permission could be very important during maintenance or recovery activities.

As a result of this variability, short monitoring intervals can create incomplete behavioural profiles. In addition, there is difficulty in determining whether a legitimate system call was made for legitimate reasons versus possibly malicious reasons. This is due to the fact that similar system calls may be performed for different application purposes. Finally, dynamic container creation and termination also add complexity to continuously identifying application activity. To address

all of these challenges, it will require contextual event analysis, representative workload observation, and adaptive behavioural profiling mechanisms that can support legitimate execution variations without producing false positive security alerts.

B. Limitations of the Proposed Approach

The suggested method has several constraints that could impact its performance in actual deployments. To begin with, monitoring at run-time is reliant upon having access to and being able to collect accurate execution events. If events are missed or if the monitoring record is incomplete, then the assessment process will be affected. Additionally, if you identify unused permissions it doesn't mean that these permissions are not needed; there can be times when applications need to exercise privileges that are rarely used due to unique application requirements. Furthermore, attackers can intentionally emulate normal application behavior in order to avoid being detected. Also, the amount of monitoring overhead may grow as you deploy more containers and collect more execution events. Lastly, when assessing permissions on different types of operating systems, container runtimes, and orchestration platforms there may be a requirement for adaptation based upon the specific implementation.

C. Future Research Directions

Future work on this model will enhance it through the use of machine learning methods to recognize complex pattern of execution and deviation from normal behaviors. The model will incorporate adaptive profiling techniques that will allow continuous updates of applications requirements as the application's workload, software version, or service dependency changes. Research into explanation-based artificial intelligence (AI) can assist in explaining why permissions-related alerts occur and what can be done to optimize them. Integrating the model with ebb-based monitoring, Cabernets security policies, and automated policy validation will make the model more scalable and more compatible with other systems. Researchers can examine how collaborative permission assessment can be used among micro services to determine if there are any unnecessary privileges in either a single service or an entire application. Evaluations using real-world container workloads, benchmark data sets, and simulated attacks will provide further support for the models ability to accurately detect, scale, and monitor overheads. All of these enhancements can help develop more intelligent, adaptive, and reliable container permission management systems.

X. ADVANCED SECURITY CONSIDERATIONS AND RESEARCH EXTENSIONS

A. Integration with Container Security Technologies

The overall effectiveness of permission management may be enhanced through the integration of runtime behavior analysis with other container security technologies. Many existing technologies, including but not limited to Dockers security configurations, Cabernets security contexts, Linux capabilities and sec comp profiles, offer ways of controlling application access to system resources. The proposed framework can complement those technologies through its examination of real-time execution activity and identification of differences between what has been configured as privileged and what is needed for the application to function properly. Integration of the two types of technology can give administrators a clearer picture of their container security posture. Also, combining runtime observation data with the enforcement mechanism(s) that are already in place can provide administrators with a better opportunity to validate policies, manage privileges consistently and create greater separation among different applications operating in a distributed environment.

B. Intelligent Behavioural Pattern Recognition

The use of intelligent behavioural pattern recognition will allow for the ability to better detect unusual application behavior. By utilizing machine learning methods, the system will have access to all of the historical execution records, which will enable the system to find recurring patterns of legitimate service operations. The system can then use these patterns as a reference point to determine whether there is any unexpected system calls, unusual file system interaction, or abnormal network activity occurring. In addition, the use of classification and anomaly detection algorithms can help separate normal execution variations from potential malicious activity. Additionally, adaptive learning mechanisms can continually update the behavioural profile of an application, as it goes through legitimate operational changes. While the use of intelligent analysis is a great tool for security sensitive container environments, the proper training data, model validation, and explain ability must be taken into consideration before implementing such technology.

C. Continuous Permission Optimization

Continuous Permission Optimization is used as a method of keeping access controls up-to-date with changes in the way an organization uses applications over their entire lifespan. Container-based applications are often updated with new versions of software, modified configurations, altered workloads, and migrated to different infrastructure environments. As a result of these activities, there will be instances where additional permission controls are required or old permission controls are no longer needed. To ensure that the right level of access control exists at all times, a process should exist to allow for continuous evaluation of how container-based applications operate in comparison to the permissions that have

been granted. In addition to being able to provide organizations with continuous evaluations of access controls, there also needs to be a mechanism for validating access controls and deployed access controls in a manner that does not negatively impact operations. By providing organizations with a means to continuously evaluate and manage access controls, they will be able to support least-privilege enforcement on an on-going basis and reduce the amount of excess permissions that accumulate within dynamic container-based environments.

D. Distributed Container Security Management

Many modern applications are comprised of many micro services that are distributed throughout several containers, nodes, and orchestration clusters. In addition, each service has its own unique resource requirements, communication relationships, and security responsibilities. Therefore, performing permission assessments individually will likely result in overlooked risks due to interactions among the interconnected components. A distributed assessment mechanism can assess the relationship between service identities, runtime behaviors, and communication patterns to determine if too much access is being granted across application boundaries. In addition to a distributed assessment mechanism, centralized security dashboards and policy management systems can enhance the administrative visibility. These capabilities may also allow for consistent permission governance, synchronized security monitoring, and scalable protection across large containerized infrastructure.

XI. CONCLUSION

The rapid expansion of the use of containerized applications in today's modern cloud native environment has made it essential to manage permissions appropriately as well as continuously monitor runtime security. When services are granted excessive permissions, this represents a major security risk since it allows services to have more access than what is truly required for them to function. As a result, there is an increased likelihood that if someone were able to gain unauthorized access to a system, compromise an application, or escalate their privileges, the negative impact could be greater. This paper discusses some of the issues that exist when trying to determine which permissions are excessive, and provides a method for using the runtime behavior of applications to find those excessive permissions in containerized applications.

The developed framework creates a linkage between granted permissions and the actual behavior of applications running in the environment, by utilizing continuous run-time monitoring, behavioural profiling, permission association, and risk categorization. Through analysis of system calls, file system actions, process executions, network communications, and pattern recognition of resource usage; the model provides a basis for determining excessive permissions and possible threats posed by excessive access. Additionally, the model utilizes contextual evaluation to differentiate between permissions that are necessary for operation versus those that are only needed under certain conditions, and ultimately, excessive access. Lastly, the model's design and implementation demonstrate how run-time monitoring can be used as an alternative or additional method of conducting traditional security assessments based upon configuration.The study further identified significant assessment metrics such as accuracy of detecting unauthorized use of permissions; overhead incurred during the monitoring process; usage patterns of permissions; ability to scale; and adaptability to changing environments. In addition, the study examined several operational issues related to differences in behavior; partial record keeping for monitoring purposes; identification of false positives; and interoperability with other systems. Therefore, the findings indicate that optimal use of permissions can be achieved through representative workload observation, validating policies appropriately and on-going evaluations rather than simply deleting privileges due to insufficient execution data.

Future studies could be conducted to further develop this method with the use of machine learning based behavioural recognition, explained anomaly detection, adaptive permission recommendations, and by integrating the method with advanced container security technologies. An experiment with a Dockers and Cabernets environment would allow for evaluation of the effectiveness of the method in terms of detection, resource usage, and operational reliability under real world workloads.Overall, runtime analysis of how an application behaves in production environment is a way forward in terms of enhancing the security of applications running within containers, through providing better visibility into the permissions applied to those applications and enabling the enforcement of least-privileged access. The current research proposal presents a conceptual framework to develop adaptive, scalable, and behaviourally-based permission control systems, which will support more secure and reliable cloud-native computing environments.

XII. REFERENCES

[1] Abed, A.S., Clancy, T.C. and Levy, D.S. (2015) ‘Applying moving target defines to cloud security’, Proceedings of the IEEE International Conference on Cloud Computing, pp. 72–79.

[2] Casalicchio, E. and Iannucci, S. (2020) ‘The state-of-the-art in container technologies: Application, orchestration and security’, Concurrency and Computation: Practice and Experience, 32(17), e5668.

[3] Lasagnes, C., Trestioreanu, L.A., Jolly, C. et al. (2020) ‘The rise of ebb for non-intrusive performance monitoring’, 2020 IEEE/IFIP Network Operations and Management Symposium (NOMS). Doi: 10.1109/NOMS47738.2020.9110434.

[4] Coombe, T., Martin, A. and Di Petro, R. (2016) 'To Dockers or not to Dockers: A security perspective', IEEE Cloud Computing, 3(5), pp. 54–62.
[5] Fournier, G., Afghani, S. and Bureau, S. (2021) 'Runtime security monitoring with ebb', SSTIC 2021 Proceedings.
[6] Gawk, S., Doan, T.-P. and Jung, S. (2023) 'Container instrumentation and enforcement system for runtime security of Cabernets platform with ebb', Intelligent Automation & Soft Computing, 37(2), pp. 1773–1786. Doi: 10.32604/iasc.2023.039565.
[7] Parkas, O., Koi, R., Dong, N. and Mahmud, R. (2025) 'A container security survey: Exploits, attacks, and defences', ACM Computing Surveys, 57(7), Article 170, pp. 1–36. Doi: 10.1145/3715001.
[8] Kithara, H., Gahanna, K. and Watanabe, Y. (2020) 'Highly-scalable container integrity monitoring for large-scale Cabernets cluster', Proceedings of the IEEE International Conference on Big Data.
[9] Merkel, D. (2014) 'Dockers: Lightweight Linux containers for consistent development and deployment', Linux Journal, 2014(239), Article 2.
[10] Moabite, R., Allman, J. and Koru, M. (2015) 'Hypervisors vs. lightweight virtualization: A performance comparison', 2015 IEEE International Conference on Cloud Engineering, pp. 386–393.
[11] National Institute of Standards and Technology (2017) Application Container Security Guide. NIST Special Publication 800-190. Gaithersburg, MD: NIST.
[12] Pal, C. (2015) 'Containerization and the Papas cloud', IEEE Cloud Computing, 2(3), pp. 24–31.
[13] Pal, C., Brogan, A., Solana, J. and Jamshidi, P. (2019) 'Cloud container technologies: A state-of-the-art review', IEEE Transactions on Cloud Computing, 7(3), pp. 677–692.
[14] Rostamipoor, M., Gravamina, S. and Polychromies, M. (2023) 'Confine: Fine-grained system call filtering for container attack surface reduction', Computers & Security, 132, Article 103325. Doi: 10.1016/j.cose.2023.103325.
[15] Souppaya, M., Morella, J. and Scarf one, K. (2017) Application Container Security Guide. NIST Special Publication 800-190. Gaithersburg, MD: National Institute of Standards and Technology. Doi: 10.6028/NIST.SP.800-190.
[16] Sultan, S., Ahmad, I. and Demetrious, T. (2019) 'Container security: Issues, challenges, and the road ahead', IEEE Access, 7, pp. 52976–52996.
[17] Yang, S., Kang, B.B. and Nam, J. (2024) 'Optimums: Association-based dynamic system call filtering for container attack surface reduction', Journal of Cloud Computing, 13, Article 71. Doi: 10.1186/s13677-024-00639-3.
[18] Balaji, R. (2024) 'Threat-Model-Driven Zero-Trust Authorization for Microservice APIs in Kubernetes-Based Cloud-Native Applications', ESP Journal of Engineering & Technology Advancements, 4(4), pp. 223–234. doi: 10.56472/25832646/JETA-V4I4P128.
[19] Xavier, M.G., Naves, M.V., Rossi, F.D., Ferret, T.C., Lange, T. and De Rose, C.A.F. (2013) 'Performance evaluation of container-based virtualization for high performance computing environments', 2013 21st Euro micro International Conference on Parallel, Distributed, and Network-Based Processing, pp. 233–240.
[20] Linux Kernel Organization (2026) Control Group v2. Available at: Linux kernel documentation (Accessed: 30 October 2026).
[21] Linux Kernel Organization (2026) Extended Berkeley Packet Filter (ebb) Documentation. Available at: Linux ebb documentation (Accessed: 30 October 2026).
[22] Linux man-pages project (2026) Capabilities (7): Overview of Linux capabilities. Available at: Linux capabilities manual (Accessed: 30 October 2026).