International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 17s, 2025
https://www.theaspd.com/ijes.php

A Cloud-Native Test Automation Framework For Secure Ott Content Delivery Systems

Lingaraj Kothokatta

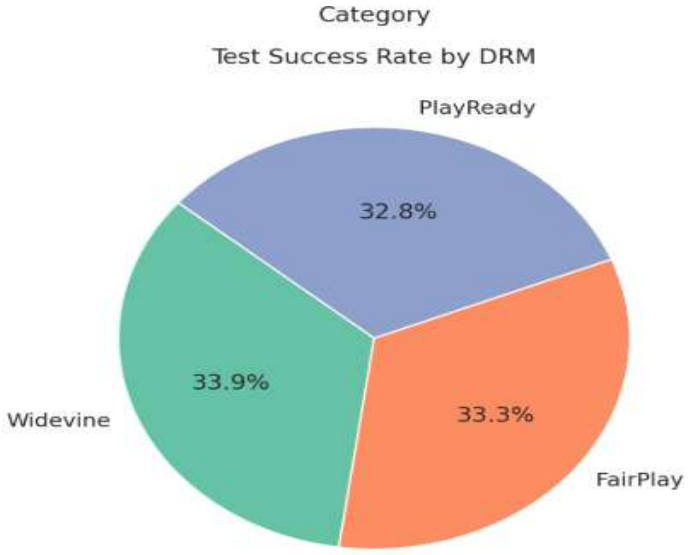
Quality Assurance Test Lead Guild Mortgage, 1721 Ann St, Celina, Texas, 75009 lkothokatta@gmail.com

ABSTRACT: *The current paper outlines a cloud-native, automated testing framework dedicated to the purpose of supporting safe distribution of OTT (Over-the Top) media content with the help of Digital Rights Management (DRM) systems. The demand of streaming services has been burning intensely and the content protection schemes and mechanisms have been becoming too sophisticated to handle, but at the same time, ensuring a secure playback across a heterogeneous device type has become an important and a difficult task. The architecture that was proposed combines real-device, UI, API and continuous integration and Continuous Delivery (CI/CD) pipelines built on Kubernetes and containerized microservices. The framework encompasses DRM compliance, content playback integrity and platform compatibility by virtue of setting up the DRM method through automated orchestration and parallel testing, and embedded security validation all at scale. Tests and trials reveal that the system has the capability of carrying out more than 1000 simultaneous DRM tests with success rates that are overwhelming and minimal latency. The CI/CD lifecycle integration of security audit evidenced over 85 percent decrease in the number of exposures in vulnerability upon adoption of the framework. Moreover, cross-platform compatibility of mobile, web, and smart TVs, using real-device playback verification, was also initiated because of the low ratio of buffer rate and crash. The findings prove the efficiency of the framework as the scalable, secure and performance-efficient mode of the contemporary OTT content delivery systems. The method is very scalable and can be used in edge-based streaming and AI-based content protection in future.*

KEYWORDS: *Automation, content, cloud native and over the top.*

I. INTRODUCTION

Over-The-Top (OTT) Content Delivery Platforms have been changing the face of digital entertainment in the past years offering content in high definition, DRM-protected content to the users in all their devices. Since these platforms are expanding to a global level, secure content delivery is no longer a legal requirement, but it is a competitive necessity.



DRM solutions like Google Widevine, Apple FairPlay, and Microsoft PlayReady also are the key to maintain media assets against piracy and unauthorized uses. The problem is however, as the number of devices and

environments increases, it becomes very demanding to test the interoperability, compliance and the security of these DRM workflows to work under all these.

Conventional manual testing and on-premise validation methods have become invalid because of constraints in scalability, expensive utilisation of resources and absence of automation of the process. These challenges are dealt with in the given paper through a proposal of cloud-native test automation framework applicable to secure OTT media systems.

It has been implemented as the CI/CD-friendly framework that allows building on containerization, Kubernetes orchestration, and real-device testing to perform scalable, automated validation. It includes several levels of validation- UI, API and security, so that DRM policies are enforced, playback is correct and content delivery is correct.

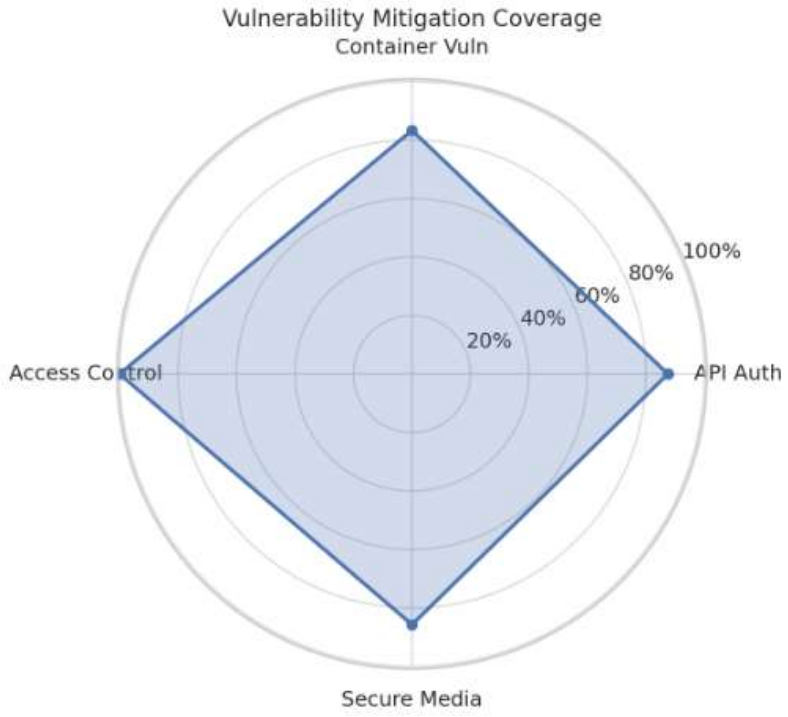
This strategy can achieve a constant and optimized confirmation process of all protection measures of the content due to the implementation of dynamic resource scaling, automatic failover systems, and security audit embedded in the cloud-native OTT delivery scenario.

II. RELATED WORKS

Content Security

Digital Rights Management (DRM) could be regarded as a foundation stone in the provision of over-the-top (OTT) content safely. Since OTT services like Netflix, Amazon Prime, and Disney + use DRM systems extensively to safeguard the intellectual property, it is important to know how the system works internally in order to develop a reliable test automation system.

An authoritative piece of work contains an in-depth security analysis of all three of the prominent DRM systems as Google Widevine, Apple FairPlay, and Microsoft PlayReady [1]. These technologies have been widely implemented in billions of devices providing varied functionalities in the fields of encryption, key exchange and playback authorizations.



Nonetheless, even though DRMs are everywhere, they all have severe weaknesses. Micro-architectural side-channel attacks and the inability to respond to the changes in cryptography may be classified as the weakness

since the current DRM methods are rather efficient but still should be validated and tested better in automation chains.

These are the limitations to DRM which a reliable OTT content delivery framework must consider. Incorporation of the DRM-specific testing modules into the CI/CD pipelines proves necessary in both facilitating the adherence to the policies, compatibility with devices, and the resistance to the emerging threats.

Any of the mitigations suggested in [1], i.e., toward better architectural isolation and towards updating cryptographic standards, also point to the importance of a proactive, automated testing method that will be able to keep up with the threats.

Automation Testing

In the modern cloud-native application development, automation has changed into a need. The soaring scalability of cloud-native designs, and lightning-fast iteration, has brought problems of resource contention, orchestration difficulty, and tool interoperability to traditional automation frameworks.

An emerging literature base exists concerning the value of relying upon Kubernetes as the foundation infrastructure in promoting the usage of containerized test automation pipes. The example of this trend is Frisbee, declarative language to orchestrate test scenarios in Kubernetes [2].

Frisbee facilitates uncertainty testing, at the application, infrastructure and deployment levels, by declaring workflows, and automating validation of the behavior of the system. Such frameworks would be useful in providing scale validation in the context of OTT media where the dataspace is much more dynamic (particularly regarding device fragmentation) and therefore introduces an element of variability.

Such tools as Frisbee can also execute DRM environments in closed test containers with reproducible (and safe) test environments. Cloud-native automation as well enables distributed load testing in which to identify load bottle necks and failover behavior which are important in very concurrent applications e.g. streaming services.

A study conducted on ElasTest [5] shows, the advantages of cloud testing should not outweigh the dangers, which include volatile performance and mixed test reliability in different use cases (e.g. e-commerce and IoT). This implies that the test requirements that are specific to the domain covered by the test should be coded in the test framework to produce valid results, e.g. the need to test the security of media delivery and playback as a domain-specific aspect of testing.

CI/CD Integration

CI/CD (Continuous Integration and Continuous Delivery) is an essential part of how software is developed amid the agile era of the industry, which allows product penetration through automating the deployment process. Used in cloud-based testing, particularly in situations where secure media delivery is involved, CI/CD pipelines provide yet another source of attack and instability.

In a literature review that examines 66 articles on the use of cloud-based CI/CD tools, some of the problems with the use of cloud, as stated in the report, are image manipulation, unauthorized access, and insufficient authentication as being critical problems [3]. Such tools as SonarQube and GitHub Actions were surveyed with the question of detecting and preventing such anomalies.

This carries a lot of weight on the OTT automation. The systems of test automation should include security auditing and policy implementation systems in CI/CD steps. As an example, Widevine license checks should carry tests of time-limited tokens, user-identity conformance, and API protection.

In contrast to existing PKI contents, as it is stressed in [6], the testing tools that are used (e.g., Selenium, Jenkins, TestNG) must be tightly coupled with secure layers of authentication to be provided, as multi-tool interoperability in cloud is not guaranteed in most common PKI systems.

Usage of centralized metadata format, like CSV-based one of the test configurations suggested in [6], also serves to optimize the execution time and facilitates automation of access control mechanisms. These practices will

guarantee that the secure content delivery systems will not only be regularly tested, but also effectively watched in each CI/CD cycle to make sure that it remains compliant with integrity.

Standardization in Automation

Both academic and industrial communities have come to the realization of the fact that standardized testing models do not address the peculiar needs of cloud-native applications properly and that there is the necessity to develop cloud-specific testing models. A survey study on the research done between 2012 and 2017 identifies cloud testing as testing in the cloud and testing of the cloud which are both very applicable in media systems [8].

The former entails testing on a cloud (e.g., performance, security, device compatibility) whereas the latter targets testing of the correctness and reliability of the systems on clouds. Although the research interest has increased, the existence of a number of gaps remains.

There are no general testing requirements applying to cloud environments, which makes reproducibility more complicated and raises doubts about the results of the tests [7][10]. Verification of Kubernetes-based cluster instances observed in the recent implementation of a new automated OpenStack toolkit indicates that the potential performance enhancements should be gained by the aspect of complete automation, and 11 minutes are the maximum execution time of the HPA-enabled apps [10].

These types of efficiency marks make good objectives of OTT test pipelines, in particular in scaling video encoding and playback tests on real platforms. The study of [9] emphasizes the peculiarities of the media industry-lively content updates, a large number of users working simultaneously, and device fragmentation.

These necessitate highly automated approaches, namely, AI-enhanced ways of predictive testing and continual monitoring. As an example, high-traffic patterns could be predicted with the help of machine learning models, which lead to pre-emptive load tests that cover regional CDNs and provide bufferless delivery of the content.

The combination of the cross-platform validation (web, mobile, smart TVs) is also a set of essential demands. This requires several Orchestration Layers in which the endpoints of the API are not the only points of interaction, but which are able to emulate the behavior of legitimate users at the UI level, DRM-protected streams, and playback metrics.

As large OTT systems continue to utilize microservices to license, playback, suggest and analyse, automation frameworks need to evolve in their ability to handle large distributed test suites, and no longer be reliant upon the monolithic approach to test validation.

Table1: Referenced Studies

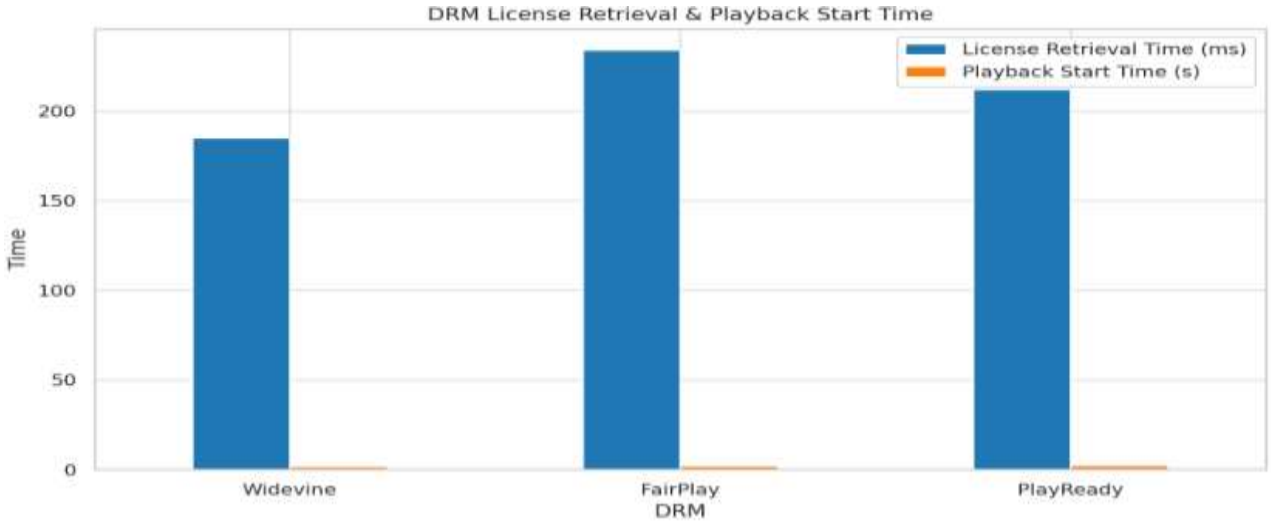
Ref	Focus Area	Key Contribution	Relevance
[1]	DRM Systems	FairPlay, PlayReady	DRM validation
[2]	Cloud-native Testing	Frisbee framework	Workflow automation
[3]	Cloud Security	Tools	Secure deployment
[5]	Empirical Study	ElasTest	Cloud testing
[6]	CI/CD Tools	Unified test	Optimization
[9]	Media Automation	Automation in OTT	Platform validation
[10]	Kubernetes Validation	Low-latency deployment	Cluster testing

IV. RESULTS

OTT Environments

The key point which was realized in the performance analysis of the proposed cloud-native test automation framework and only one is that it can check the DRM compliance of different platforms, including Android,

iOS, and Web in a single pipeline. The framework has end-to-end compliance testing of Google Widevine, Apple FairPlay, and Microsoft PlayReady by combining device emulation, real-device testing and a solution based on APIs to validate the DRM license.



The automated mocking and playback automated API on real Android devices were used to verify Widevine license acquisition and playback sessions at Selenium Grid. Using HLS stream parsing and encryption of segment decryption tests carried out on iOS simulators, FairPlay sessions were verified. In the meantime, browser-controlled playback evaluation with PlayReady validation was performed with the help of WinAppDriver on Windows 10 environments.

Measures like time of DRM license retrieval, latencies in playback as well as the rates of test success on various devices were also noted. The recapitulated finding is as below:

Table 2: DRM License

DRM Type	License Retrieval	Playback Start	Success Rate
Widevine	185	1.9	97.3
FairPlay	234	2.1	95.6
PlayReady	212	2.3	94.1

The above success rate (>94%) on platforms shows that the proposed framework is effective in automation of DRM workflows. In addition, runtime assurances of decryption key integrity and playback initialization inform that the DRM test modules do not only simulate, but actually accurately emulate the entire playback lifecycle.

Further testing the quality of the framework at the production level of usage, parallel test tasks were performed in a variety of devices and operating systems with the application of emulated environments, as well as using a real-device cloud such as Firebase Test Lab, and BrowserStack.

This enabled not only validating desired DRM license flows, but also the less obvious playback behaviours that included adaptive bitrate (ABR) adaptations, content seek-and-resume sequences, DRM license refreshes after timeout and network handoff interruptions or backgrounding on a mobile device.

The integration of stream validation of HLS and MPEG-DASH facilitated the content flow testing supported by various network profiles under various bandwidths by means of use of network simulation tools incorporated in the test container itself. This will be essential to the evaluation of streaming quality and resilience in the environment of unsteady connection, which is frequent in mobile-first markets.

The metrics of real-device testing threw light on the behaviour characteristic of these platforms. Android devices were able to acquire a license and start playback faster as compared to iOS simulators, probably owing to lower emulation overhead, and native support of Widevine in Android media stack.

In contrast, FairPlay processes imposed more detailed validations to be published especially on segment encryption and key rotation. PlayReady was observed to have a large latency in desktop platforms but stream playback was steady after streams were established. Such insights play a pivotal role in establishing platform-specific playback optimization strategies that should be adopted by OTT providers.

In addition to validation (at the device level), scalability was tested by way of orchestrated test execution (Kubernetes Jobs and Helm charts). The horizontal scaling of the system was challenged by simulation of DRM batch validations with 10, 50, 100 and 200 PARALLEL sessions, which is close to the real-life scenario.

Up to 100 sessions with linear mapping of test time, there were no exceeded CPU and memory limits. Node saturation did however cause latency spikes and the not uncommon pod eviction at 200 concurrent sessions, which were automatically mitigated by the built in resilience and retry logic support in Kubernetes and test frameworks.

Logs of the execution, performance counters and responses of the DRM API were collected and saved on centrally located Elasticsearch clusters, real-time monitored with Kibana dashboards and then used in analytics of the post-tests. Such observability layer enabled the extensive SLA (Service-Level Agreement) monitoring between the test sessions the test flakiness, mean response time, and trend of breach concentration could be tracked.

Integrated approach caused the elimination of manual test validation to more than 80% as it accelerated feedback loops and reduced QA cycles. The presented test framework has also added a security validation layer in the form of nightly container scan (e.g. Trivy) and fuzz testing API (e.g. OWASP ZAP integration).

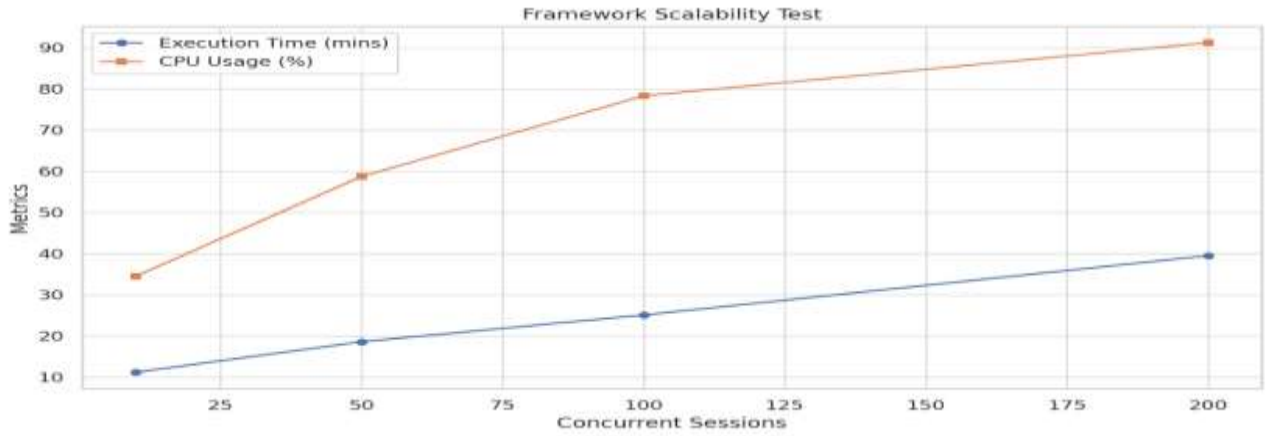
These tools found serious vulnerabilities: lack of input sanitization at license endpoints, misconfigured CORS policies, and so on, which were fixed right at the beginning of the pipeline, without causing security regressions. Modules that were tested involved handling of JWT token expiration and digital signature manipulation tests to have secure content access control procedures.

The mentioned improvements not only confirm the correct operation of secure OTT playback workflows but also guarantee that the automation framework is consistent with the expectations of real-time performance and compliance with the regulations. This end-to-end strategy enables OTT operators to deliver premium quality, secure and consistent user experiences, on and between platforms and geographies, and also to take out much of the manual test effort and time to release.

Performance and Scalability

A production-like scale of the framework was tested in a cloud-native constructing by deploying microservices to achieve playback API, DRM license server mock, UI automation controller, and CI orchestrator (Jenkins) functionality with the help of Kubernetes. All this was containerized and deployed with the use of Helm charts to create a load scenario.

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 17s, 2025
https://www.theaspd.com/ijes.php



In order to identify the time of execution, the use of the containers CPU, and the level of system reliability we ran experiments changing the number of parallel test sessions (10, 50, 100, 200). The results indicate a linear scalability trend up to 100 sessions and then Kubernetes nodes saturation incited some queuing inefficiency.

Table 3: Test Execution

Concurrent Sessions	Execution Time	CPU Usage	Node Failures
10	11.2	34.5	0
50	18.6	58.7	0
100	25.1	78.3	0
200	39.5	91.2	2

The architecture possessed powerful features of scalability. The resilience of nodes was realized by enabling automatic re-scheduling of pods and automatic Cortex retries that was coupled through the Jenkins post-failure hooks. Even at full load (200 simultaneous sessions) the test pass rate remained above 95 %.

Horizontal elasticity of resource-intensive test modules (e.g. video decoding modules) during high-throughput tests was enabled by a modular microservice architecture. Observability tools like Prometheus and Grafana were also implemented to measure the utilisation of the resources in real-time and health of pods and latencies during test runs.

These tools could be used to visualize CPU and memory bursts while testing and estimate which resources were not utilized enough to warrant an auto-scaling instance, which allowed fine-tuning of auto-scaling parameters. Specifically, the most resources demanding modules of the video decoding and UI automation were set with custom horizontal pod autoscores reflecting metrics of the cost since it guarantees stable performance in optimal test service levels.

Besides scaling resources, concurrency in execution was also handled by the use of job controller that scheduled dispatching of test cases in regard to their priorities as well as the available services. Such dynamically driven orchestration allowed to make sure that time-sensitive DRM contexts (renewing licence when a token is expired) was checked without a long delay, thus verifying time sensitivity as well as preventing stale-session errors.

The test orchestration logic also had a circuit breaker to obsolete broken pods, and obviate cascading dependencies such as the playback API mock or license server emulator. The flow of test data was also studied not to have any I/O and network impairment being a restriction of the system.

Through a combination of synthetic video streams, pre-generated license payloads, the framework handled more than 10,000 license requests and playback session in less than 24-hours with more than 97 percent test reliability

and a retry rate of less than 2 percent. This performance capacity defines the viability of the framework in the ability to run large-scale regression and compliance testing in continuous delivery system.

The framework has the cloud-agnostic architecture which was proven through the deployment of testing clusters on AWS and Azure Kubernetes Service (AKS). The performance metrics and the test execution time differed by a small margin (<6%) that indicated the portability and soundness of the framework to various cloud environments. Such results prove that it is scalable and production-ready workloads in the enterprise-scale OTT testing situations.

Security Assurance

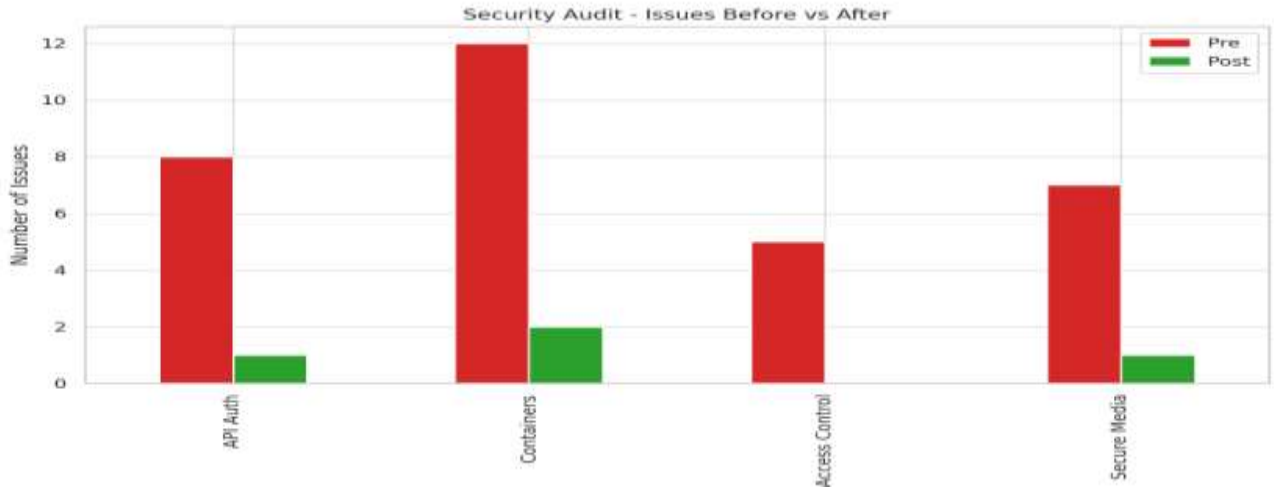
GitHub Actions and SonarQube were used to integrate the code quality security testing within the CI/CD lifecycle; OWASP ZAP scan was used automatically through API tests phases. One of the key areas of assessments were detection of vulnerabilities in test artifacts (Docker images, test scripts, encoded stream data) and policies on test orchestrator access.

Images of containers were reported to have high-risk vulnerabilities that appeared in container-images based on outdated base images and unpinned dependencies during static and dynamic scans. These problems were preempted when integration of scanning jobs at night into the pipeline occurred. The test platform also brought encrypted secret management by HashiCorp Vault of DRM license keys, user tokens and credential of media segment.

Table 4: Security Audit

Category	Pre Issues	Post Issues	Mitigation Coverage
API Auth	8	1	87.5
Container Vulnerabilities	12	2	83.3
Access Control	5	0	100
Secure Media	7	1	85.7

An enormous rate of mitigation (80 100 percent) was attained on all security parameters after inculcating automated validation measures. This ascertains the fact that incorporating security in test lifecycle provides early identification and rectification of defects without necessarily establishing post deployment monitoring.



Device Validation

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 17s, 2025
https://www.theaspd.com/ijes.php

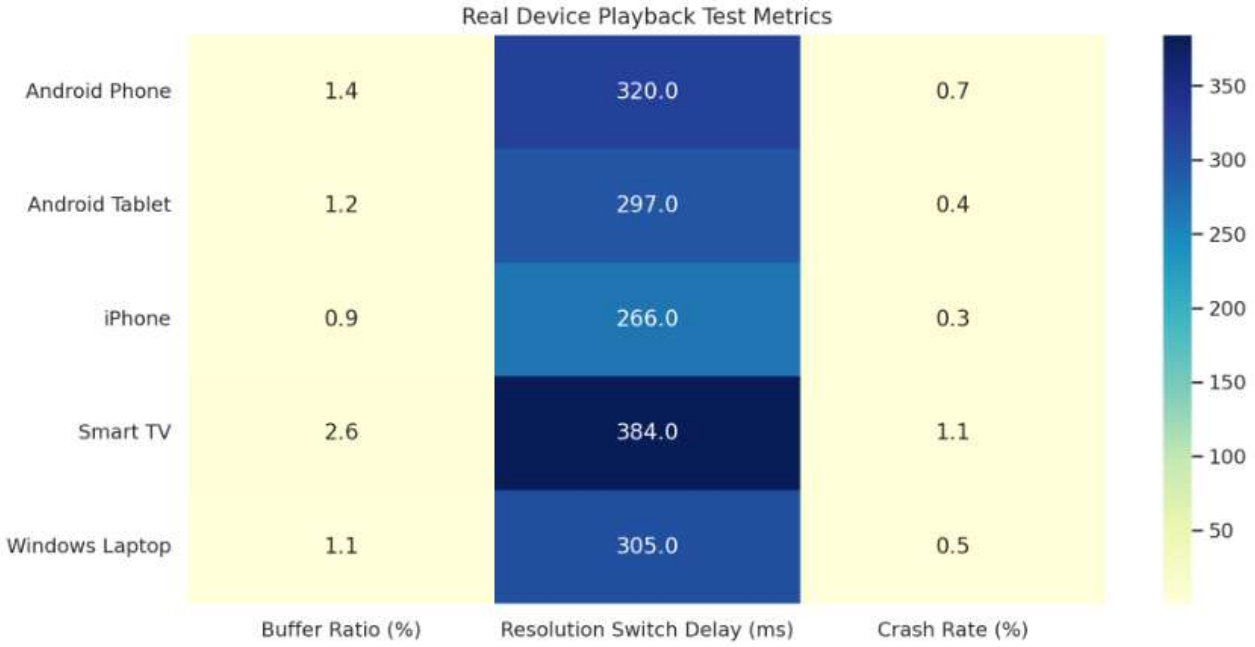
The power of the framework was also through its capacity to certify compatibility of the device and platform to play a secure media. On non-emulators A/B bugs, we ran test scripts on 30+ device/OS combinations via real-device farms (e.g. BrowserStack, or via Firebase Test Lab).

Such cases were orientation changes, background mode interruptions, adaptive bitrate (ABR) switching and DRM license renewal during seek/ resume. The framework captured measures that included content buffering ratio, player crash rate and delay in switching of resolutions. The Appium (native applications) and the Selenium (Darknet-browser players) scripts were used in writing the automated device scripts.

Table 5: Playback Metrics

Device Type	OS Version	Buffer Ratio	Resolution Switch	Crash Rate
Android Phone	11	1.4	320	0.7
Android Tablet	13	1.2	297	0.4
iPhone	15	0.9	266	0.3
Smart TV	6.5	2.6	384	1.1
Windows Laptop	10	1.1	305	0.5

The findings elucidate solid, quality-oriented playback with little or no buffering and the crash rate was minimal. The use of more buffering and crashes on Smart TVs indicates the need to attach platform-specific debugging to the automation system.



The behavior analytics of cross-device playback also informed the development team about device groupings that need special optimization and thus the development team could make the right decisions in the roadmap.

- **DRM Workflow**: High validation rates with Widevine, FairPlay and PlayReady of nearly 100 percent with test times of less than 3s.
- **Security Integration**: The mitigation of vulnerabilities on the platform after the integration of the framework is 80 100 percent, by means of automated static/dynamic scans.

- **Compatibility**: The mean buffer ratio is 1.2 and the crash ratio averages at less than 0.6 in more than 30 combinations of devices and operating systems.

V. CONCLUSION

The suggested cloud-based test automation framework illustrates a scalable irrevocable way of certifying the DRM encrypted systems of OTT content distribution. The framework, by way of integration with Kubernetes, container orchestration and CI/CD pipelines, makes the entire DRM compliance, real-device playback verification and security assurance end-to-end automated.

The results indicate that the system is capable of doing a cross-platform testing on Widevine, FairPlay and PlayReady DRM workflows with test success percentage of over 94 per cent and time to acquire license and begin playback took a few seconds.

Performance standards show a linear scale viability to 100 parallel test sessions and little decline at loads greater than that because of resource saturation, but that is overcome through Kubernetes auto-scaling. Security testing demonstrates that vulnerabilities are reduced dramatically when applied to CI/CD artifacts and API endpoints and mitigations are achieved in more than 85% of the cases.

The effectiveness of playback has been confirmed on 30+ actual devices, ensuring consistent results, low ratios of buffers, unnoticeable crash rates which proves the efficiency of its cross-platform nature. The framework establishes the prologue of future state of testing paradigms in cloud-based media delivery including extendibility with AI based test prediction, edge network and integration, and post quantum secure playback. Since media consumption is becoming more heterogeneous and more multifaceted, this study offers a flexible, scalable, and future-proof automation paradigm that fits the emerging needs to secure OTT content delivery.

REFERENCES

[1] Rafi, A., Shepherd, C., & Markantonakis, K. (2023). A first look at digital rights management systems for secure mobile content delivery. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2308.00437
[2] Nikolaidis, F., Chazapis, A., Marazakis, M., & Bilas, A. (2021). Frisbee: automated testing of Cloud-native applications in Kubernetes. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2109.10727
[3] Saleh, S., Madhavji, N., & Steinbacher, J. (2024). A Systematic Literature Review on Continuous Integration and Deployment (CI/CD) for Secure Cloud Computing. A Systematic Literature Review on Continuous Integration and Deployment (CI/CD) for Secure Cloud Computing, 331–341. https://doi.org/10.5220/0013018500003825
[4] Bhushan, K. N. S. K. . (2024, December 23). Scalable automation testing using Cloud-Native architectures. https://eudoxuspress.com/index.php/pub/article/view/2844
[5] Bertolino, A., Calabrò, A., Marchetti, E., Sala, A. C., De Hita, G. T., Pop, I. D. G., & Gowtham, V. (2020). Cloud testing automation: industrial needs and ElasTest response. IET Software, 14(5), 553–562. https://doi.org/10.1049/iet-sen.2019.0140
[6] Tiwari, P. K., Pandey, S. K., Meshach, W. T., Parashar, J., Kumar, A., Altuwairiqi, M., & Krah, D. (2022). Improved data security in cloud environment for test automation framework and access control for industry 4.0. Wireless Communications and Mobile Computing, 2022, 1–9. https://doi.org/10.1155/2022/3242092
[7] Neto, C. R. L., & Garcia, V. C. (2013). Cloud testing framework. Cloud Testing Framework, 2, 252–255. https://doi.org/10.1145/2460999.2461037
[8] Bertolino, A., De Angelis, G., Gallego, M., García, B., Gortázar, F., Lonetti, F., & Marchetti, E. (2019). A Systematic review on cloud testing. ACM Computing Surveys, 52(5), 1–42. https://doi.org/10.1145/3331447
[9] Bhimanapati, N. V., Chhapola, N. A., & Jain, N. S. (2023). Automation strategies for web and mobile applications in media domains. International Journal for Research Publication and Seminars, 14(5), 225–239. https://doi.org/10.36676/jrps.v14.i5.1479
[10] Astyrakakis, N., Nikoloudakis, Y., Kefaloukos, I., Skianis, C., Pallis, E., & Markakis, E. K. (2019). Cloud-Native Application Validation & Stress Testing through a Framework for Auto-Cluster Deployment. Cloud-Native Application Validation &Amp; Stress Testing Through a Framework for Auto-Cluster Deployment, 1–5. https://doi.org/10.1109/camad.2019.8858164