Protected by Design:

# Building Mobile App Defense Into Your CI/CD Pipeline

An Intellyx White Paper for Verimatrix
by Jason 'JE' English, Director & Principal Analyst

Fall 2025

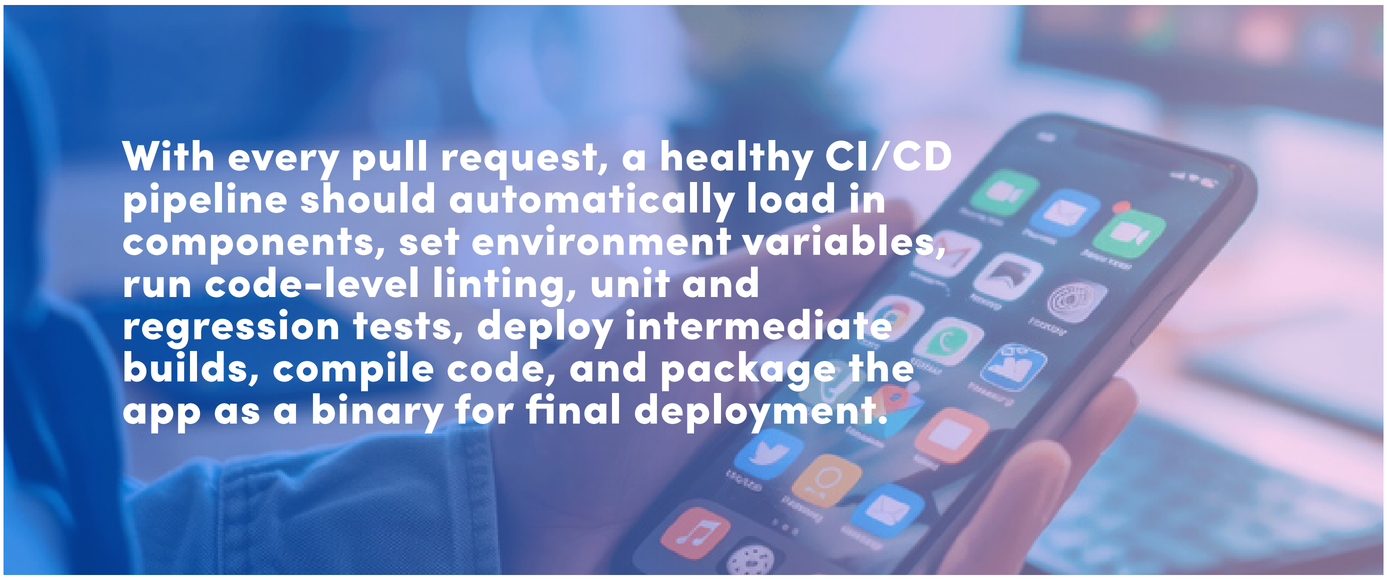
## Mobile apps. They're what hackers crave.

The overwhelming balance of consumer and business application usage now happens through native apps on smartphones—as much as 80-90%, depending on the statistics you look at. The current dominance of this mobile usage model makes native apps the preferred attack surface for bad actors.

In a recent Intellyx whitepaper my colleague Eric Newcomer wrote about 4 distinct types of mobile security, he talks about how mobile application developers can preventatively apply multiple countermeasures using solutions such as Verimatrix XTD within apps and the device itself to make this attack surface much less appealing.

***"The analogy here is of using the moat, alarm system, and safe room to deter intruders at each level. Hackers basically look for easy wins and will attack someone else if they encounter too many barriers,"*** he states in the paper.

Application teams are under constant competitive pressure to deliver new software features to market faster, in order to attract and retain customers. This drives them to adopt agile development practices and DevOps methodologies, which call for automating as much of the continuous integration and continuous delivery, (or CI/CD) pipeline as possible.



**With every pull request, a healthy CI/CD pipeline should automatically load in components, set environment variables, run code-level linting, unit and regression tests, deploy intermediate builds, compile code, and package the app as a binary for final deployment.**

Defense of that app binary itself should also be part of the CI/CD pipeline, but all too often, it is seen as a constraint to rapid delivery and therefore left up to security teams as an afterthought.

Best-in-class mobile development shops directly embed native mobile app protection within the modern CI/CD lifecycle—using familiar platforms like GitHub Actions, GitLab CI, Jenkins, Bitbucket Pipelines, and Azure DevOps.

If you are doing CI/CD for mobile apps, or thinking about how to protect apps without creating additional security bottlenecks in the software delivery pipeline, then this white paper is for you.

## Why mobile app security gets overlooked in the CI/CD lifecycle

In one sense, the process for building security checks into a mobile app should be similar to how developers would secure any web-based or desktop app.

However, because of the need for speed, mobile apps were once under the purview of small 'innovation teams' who iterated quickly to improve runtime quality of experiences and rely on APIs to pass business logic to cloud-based or centralized systems. Early mobile teams who were highly skilled at building responsive front-end experiences in React Native, Flutter, or Xamarin were not used to wearing security hats, and didn't share the release practices of the conventional application development teams.

Now that CI/CD is widely adopted, enterprises understand it is a mobile-first world and they need to accelerate agile releases across the organization. The native app software pipeline can't afford to wait on an external security review or settle for release windows for pentesting, therefore CI/CD practices have matured for mobile development work.

Like any other app under development, the mobile CI/CD tool chain will be built atop integration and release orchestration platforms like Jenkins or GitHub Actions, and include unit and regression tests, code linting tools, OS and container builds, and more recently, code-level testing (SAST/DAST) routines to look into code-level issues.

Now, leading vendors and open source projects have stepped up with mobile application security testing (or MAST) tools such as MobSF, to ensure that the build isn't misconfiguring infrastructure, and that code internals aren't exposing any known vulnerabilities.

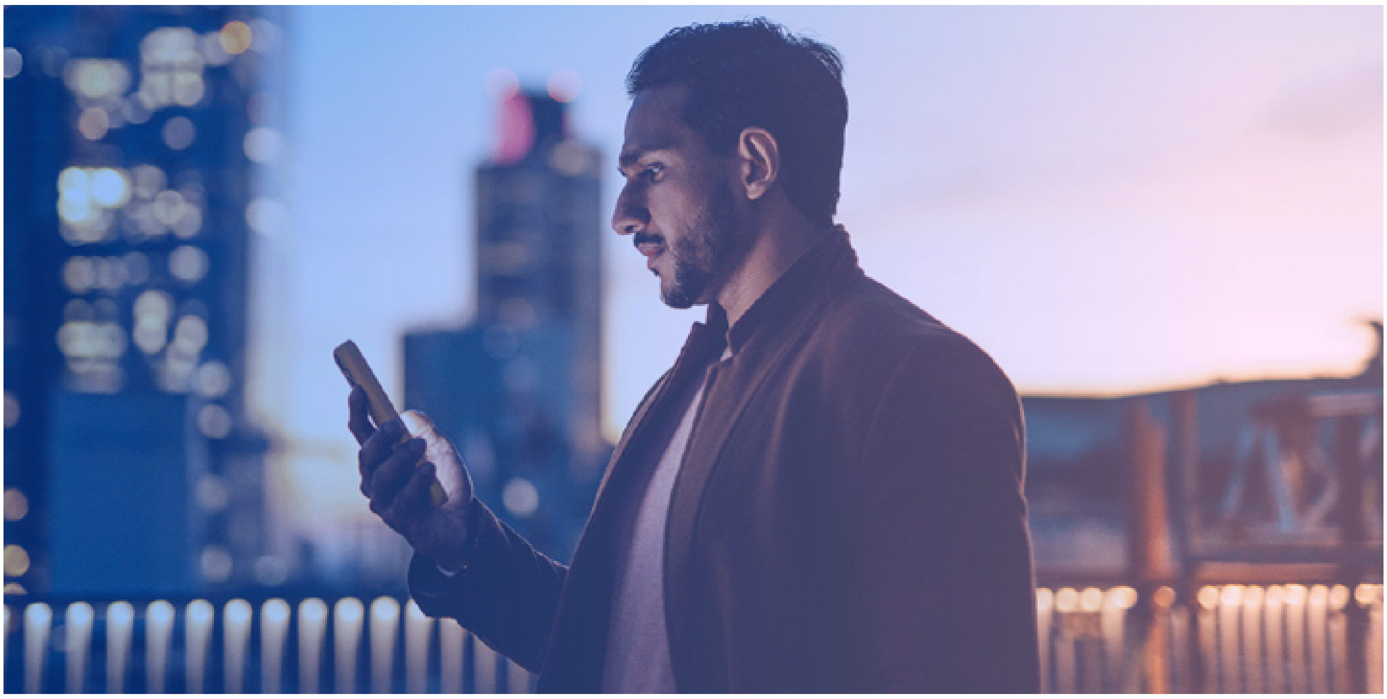
But there's still a key final step missing. Running so many code and component-level scans won't be able to predict what happens after that mobile app is packaged up. Once an app binary is remitted to app stores, it is out there on its own in the world.

## Exploring the mobile app exploits

Leading stores like Google Play and Apple App Store carry vulnerable and exploited apps way more often than you might expect. With millions of apps, some released and updated weekly or daily, there's no way we can expect a mega-marketplace to thoroughly scan them all, beyond perhaps a quick compatibility check.

We also can't assume that the latest Android or iOS operating system will protect against all runtime application attacks. Never mind that only 40% of Android users will even update within a year, and iOS users who think they are safe based on the Apple commercials often defer system updates until they are forced to. Since these architectures are so well studied, hackers have had plenty of time to explore ways around OS-level protections through the apps themselves.

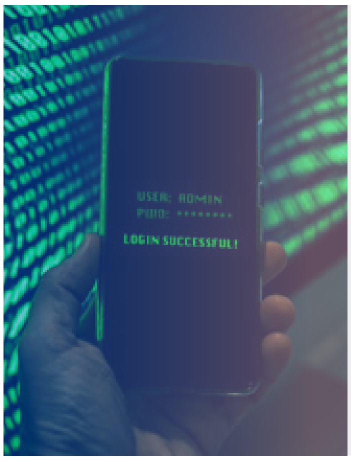
*Let's take a tour through some of the security exploits that are unique to native mobile apps…*



### Screen recording and snapshots

The coolest mobile apps use a smartphone's camera, microphone, GPS and other sensors to deliver features truly tailored to the customer, and can record user screenshots or recordings for social sharing. Think, an insurance app that allows you to take pictures of the damage to a car on the side of the road, or perhaps a store locator that pinpoints the nearest locations that have inventory of the item you are looking for. Or any number of social media, streaming and video chat applications.

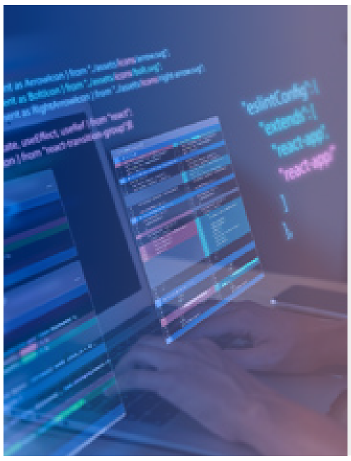
From the perspective of a user, if an application asks for access to the camera, or location data, or even the screen, most people automatically tap "OK" or "OK while using app" to continue on with their work. Often, the app's access to local hardware resources is embedded by developers through a plugin or open source package that could have been corrupted to route private data to a less reputable third party.



### Session takeover, copy-paste, and remote operation

New takeover exploits are often enabled by adjacent applications that can snoop on the user's screen in order to copy keystrokes, or look within unencrypted fields and data exchanged with external applications.

From there, the user's credentials or personal identifiers can be used by hackers or bots to log in as the user, or socially engineer deeper access. In addition, some support app routines allow outsiders to remotely operate the screen controls or take over a user session



### Developer mode and debugger access

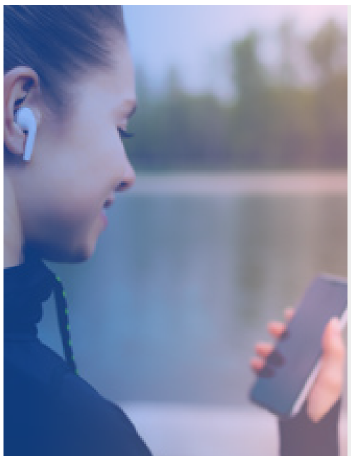
Developers that want to deliver resilient, high-performance apps use developer mode and debuggers so they can drop in on a safe pre-production version of the app to investigate errors and lagging user experience issues. All too often, these direct avenues to introspection and control are not properly disabled and remain accessible in production.

On Android, in addition to developer mode, debuggers can be activated by USB cable or a TCP connection, whereas some iOS apps could still contain a debugger if the user selects the “trust this developer” setting on app install.

### Code tampering and reverse engineering

If bad actors can unobfuscate, view, or modify app code, or run the app in an emulator, they can also tamper with the package and quite easily spoof its functionality and reproduce an impostor version of the app.

Since the reproduction might look legitimate to end users in an app store or download site, it can be used to distribute malware and steal user data and funds, as well as damaging the reputation and revenue potential of the original publisher.



### Third-party accessory attacks

Both iOS and Android-based devices have peripheral vulnerabilities—external keyboards and Bluetooth accessories such as earbuds and webcams. These come with driver software or update the host smartphone with packages that can allow a third party to register access to the user's keystrokes, possibly listen in on an unencrypted audio stream, or even snoop in on video calls and messages. With AI's ability to comprehend text, images, and voices, the possibilities are endless.

## Fitting app security into the CI/CD flow

Adoptees of the DevOps movement should tirelessly automate as much of the development, integration, code-level security checks and deployment processes as possible in order to deliver new features to market. Therefore, mobile app security needs to be an integral part of the CI/CD pipeline without slowing velocity.

Mobile developers already know where the "crown jewels" are within an app's workflow, and so do bad actors. At these points, a session is just starting up, the user is entering a password or data in a field, or private data is being displayed.

If developers use appropriate measures for handling secrets, and embed thorough unit tests, code and configuration checks, and MAST into the CI/CD pipeline, that will provide a solid foundation for protecting app internals against known threats and exploits from a massive library of OWASP definitions.

But what happens when all of these checks are done, and the application is compiled and packaged for final remittance to the app stores? Once permissions and session data are abstracted within a native mobile app, it's nearly impossible to control what happens to it in the wild, where hackers can employ debuggers, jailbroken phones, and ride-along malware apps.

**We need a holistic approach that doesn't depend on developer awareness of thousands of threat definitions and domain blacklists to protect the app.**

Besides, these can only cover the known threats and add burdensome load and time when checking apps at the end of the CI/CD pipeline, as well as burdensome runtime performance issues every time a user launches the app.

Developer net adoption is everything when it comes to embedding mobile app security countermeasures. The process should be as simple as adding a line in a Terraform config or YAML document, or developers will continue to punt on native app security.

## Bringing the Sec to DevSecOps with Verimatrix XTD

DevOps is all about collaborating and automating the software pipeline to achieve the shortest possible path to release, and since you can't successfully release an app without security, pros who know now call it DevSecOps.

Verimatrix XTD Enterprise Suite offers a solution that wraps the finished app at compile time with polymorphic encryption and countermeasures for detecting and protecting against runtime exploits, making it nearly impossible for bad actors to snoop into code, components, or user sessions.

But of particular interest is how easily Verimatrix fits native app protection into the CI/CD software supply chain. While the company offers expert support for difficult cases, here's a quick step-by-step they provided that most developers can follow in a self-service manner, another key tenet of DevOps practices.

**How to embed app security into the CI/CD lifecycle with Verimatrix XTD**

**1. Generate API Key**
- Go to the XTD SaaS portal → Settings > API Key Manager
- Download the api-key.json file

**2. Configure CI/CD Tool**
- Choose your CI/CD platform (e.g., GitHub Actions, Jenkins, GitLab)
- Install the Verimatrix CLI (vmx-aps) or use the GitHub Action

**3. Build Your App**
- Then compile your Android or iOS app as usual

**4. Protect the App**
- Use the CLI or GitHub Action to upload the app and apply protection

**5. Download Protected App**
- The protected app is returned and ready for testing or release

**6. Detect and Response in XTD Dashboard**
- Monitor app activity and threat detections in the XTD dashboard
- Take response actions as needed

**7. Iterate**
- For each new release, protect the app again using the same or updated protection settings.

*Source: Sami Urtti, Product Management Director, Verimatrix XTD*

For maximum effectiveness, Verimatrix suggests that development teams re-run the CI/CD process every 12 weeks or so, even if there is not a new release, to allow the in-app protection to stay up to date with evolving threats. This adds little overhead, as the app protection process typically takes less than 10 minutes to complete.

## Scenarios for using Verimatrix XTD Enterprise Suite

Different companies have very different priorities for protecting their mobile apps, but they share one key trait—they don't want to talk about the real security shortcomings that led them to select a solution. So like a compiled and protected app, we'll provide three brief but anonymous playbooks to describe what real-world customers are doing without specifically identifying anyone.

| | **Media** | **Financial** | **Gaming** |
|---|---|---|---|
| **Problem** | A leading entertainment provider was looking to crack down on viewer piracy and prevent IP theft of its proprietary streaming technology and application experience by competitors. A new 'Freemium' service appeared with an eerily similar player experience that mirrored their own. | A financial institution had an all-hands-on deck moment when a hacker triggered a background screen grab of a customer's login information and phone number in order to circumvent two-factor authentication and empty out a bank account. | A popular game producer was starting to see inexplicable performance lags as players complained of in-game crashes. App store ratings were starting to decline below 4.0 stars, and some reviewers were even mentioning problems that couldn't be reproduced by the developers. |
| **Solution** | By inserting Verimatrix XTD Protect in their CI/CD pipeline, they were able to automatically seal their binary against debuggers and developer mode attempts to reverse engineer their player functionality, and if a user attempts to run their app on a rooted device, the app just shuts down, while XTD Detection and Response reports the incident back to the team within their SIEM threat monitoring dashboard. | Because they had to be compliant with regulations requiring data sovereignty, they installed a local version of Verimatrix XTD Enterprise Suite behind the corporate firewall, on their internal Jenkins development servers. Once XTD wrapped the next build package of the banking app, a wave of overlay attacks was detected. The Verimatrix User Identity Tag allowed a warning to be sent to possibly affected users, asking them to contact the bank directly, while XTD cryptographically eliminated the ability for any adjacent apps to recognize passwords in a screen grab, as well as preventing future session takeovers. | After installing Verimatrix XTD Detection and Response, they discovered someone was tampering with code at the point of in-game purchases and adware, as well as when game sessions were being recorded or shared, resulting in lost users and even a spoofed similar version of their game on app stores. By wrapping their binary with XTD Protect within the next CI/CD release cycle, further outside code tampering and inspection were prevented, and they immediately ordered the spoofed and adware editions of their game they had detected taken down. |

**The Intellyx Take**

If you're building mobile apps at DevOps speed, then now is the time to secure them at scale— without impacting the release timeline.

Native app security should be a mobile development enabler rather than a roadblock, so it absolutely needs to be done in a way that not only shields and protects the app against a host of runtime exploits, but simultaneously makes the CI/CD pipeline as automated and frictionless as possible.

Securing mobile apps against an ever-growing set of attack vectors is hard enough already, so let's not make it any harder to adopt than it needs to be. Verimatrix XTD offers DevSecOps teams a way to achieve mobile app security at the last mile without advanced knowledge, by inserting preventative measures into the delivery pipeline by design.

## About the Author

**Jason "JE" English** is Director & Principal Analyst at Intellyx. Drawing on expertise in designing, marketing and selling enterprise software and services, he is focused on covering how agile collaboration between customers, partners and employees accelerates innovation.

A writer and community builder with more than 25 years of experience in software dev/test, cloud computing, security, blockchain and supply chain companies, JE led marketing efforts for the development, testing and virtualization software company ITKO from its bootstrap startup days, through a successful acquisition by CA in 2011. He co-authored the book *Service Virtualization: Reality is Overrated* to capture the then-novel practice of test environment simulation for Agile development. Follow him on LinkedIn.

## About Verimatrix

**Verimatrix** (Euronext Paris: VMX, FR0010291245) secures the apps that power our digital world. The Verimatrix Extended Threat Defense platform (XTD) delivers next-generation app protection for Android, iOS, embedded, and desktop apps—blending layered shielding, adaptive AI-driven threat intelligence, real-time detection and response, and whitebox cryptography to block attacks before they happen. Built for modern development, XTD integrates seamlessly with CI/CD pipelines and SIEMs—boosting security without slowing innovation. For 30 years, global leaders have relied on us to build trusted, seamless digital experiences—without compromise. Backed by ISO-certified security, 150+ patents, and award-winning localized services in 9 countries, we protect industries like telecom, hospitality, banking, e-commerce, healthcare, and automotive from relentless cyber threats. The apps we trust need protection. Verimatrix delivers.

Visit **www.verimatrix.com/cybersecurity**