# BlackBerry Jarvis - How It Began

Adam Boulton
Chief Technology Officer
BlackBerry Technology Solutions

*“BlackBerry® Jarvis is a powerful binary static analysis tool that can help automakers secure their software supply chain. BlackBerry Jarvis inspects binary files in an easy, quick, scalable, and cost-effective way, and delivers deep insights into the quality and security of software components.”*

This is where BlackBerry Jarvis is today; an industry leading SaaS tool focused on supporting organizations to build safe and secure software. This article explores the journey, starting as the focal point to mature our own internal security engineering practices through to becoming a leader in binary static analysis.

## BlackBerry Product Security

BlackBerry is synonymous with security because security is, and always has been, at the forefront of everything BlackBerry does. The recent acquisition of Cylance for $1.4bn continues to demonstrate how heavily BlackBerry invests in software security. Securing such a massive product portfolio across the globe, that spans multiple technologies, for security and safety critical systems, requires continual investment. More than 15 years ago, a centralized software security team was built, with a focus on securing BlackBerry products. The initial priority was on our high-risk and flagship products and over time the scope widened to cover all products. In addition to securing our own internally developed products, we also needed to secure our software supply chain. The security group contained experts in the fields of security code reviews, reverse engineering, exploit development, penetration testing, incident response, certifications and capability development.

It is no secret that building a highly dedicated and skilled team to undertake security engineering is a very difficult initiative. A common problem that software companies encounter is that the complexity of solutions grows exponentially resulting in diminishing quality. In order to support aggressive growth in our codebase and to sustain quality, we began to invest heavily into our internal capabilities, building a suite of solutions that support the daily activities of the product security team. These tools were needed to address very particular challenges, including the ability to identify areas of risk in software components, prioritize where manual effort is spent, and the scale of efforts. Ultimately, our goal is to support and enable software product teams so that software security does not become a cumbersome and time-consuming blocker.

Development teams need support to meet their timelines and this becomes extremely challenging when taking on the complex facet of software security.

# Source code reviews

Source code reviews have historically been one of the most common ways of performing security assessments (when you have access to the source code, of course!) and have several benefits. Reading source code is a skill more widely available than reverse engineering and, secondly, source code reviews are easier than reviewing compiled components. This is the nature of the beast; source code is written and reviewed by humans whereas compiled components are not.

There are issues associated with source code reviews, however, and these get worse as the complexity of the project increases. An ideal situation is to perform the code review after the point of code freeze, as there is no point reviewing source code if it is to significantly change over the course of the next few weeks or months. In reality, this rarely happens. Unfortunately, the norm is that security assessments happen prior to the code freeze. This means the assessment essentially becomes redundant shortly after changes are introduced. There are additional complications because the source code is only a human readable form of what the product might become; source code is not a 1:1 representation of what will actually be built. A source code review is unable to capture what will actually get compiled and bundled into the final product. Through the wide variety of compiler configurations, including optimizations and compiler defenses, and compiled libraries, to which you have no source code access and build scripts, it is impossible to capture through source code reviews what the final solution will really become.

# Binary reviews

An approach to dealing with the issues mentioned above (resource constraints, source code access, developer interactions, accuracy and relevance of an assessment) is to examine the final binary produced by the build system, and then feed back to the development team any findings that were deemed important. There are several advantages of this approach, with the most important being the security team gets to review the actual binary that external attackers will have access to. This is critical as often security vulnerabilities are introduced at the build stage and wouldn’t necessarily be caught by a code review.

The BlackBerry security team developed a system that would allow binaries to be uploaded, scanned, and scrutinized for common security issues. Initially a few basic checks were included, for example hardcoded private keys, but over time the number of checks grew as did their sophistication and complexity.

# The requirement for automation

The beauty of automated solutions is the ability to provide scale and consistency of assessments.  A recurring theme when performing security reviews was the repetition of certain issue classes. It got to the point that when performing reviews on certain types of components, it would almost be a surprise if the security guidance documentation had been followed entirely. This is not a criticism of the development teams; this is simply the reality of modern development. There are too many issues to think about and expecting development teams to focus on security issues as deeply as a dedicated security team is nice in theory, but impractical in practice.

Expecting the development teams to stop what they are doing and spend time addressing security issues can often be very challenging, given the focus will typically remain on building to customer requirements. To cut down the time spent identifying, recording and providing guidance on the ‘low hanging fruit’ issues, tools developed by the security team were made available, so developers could run them before security engagements even started. This allowed them to identify and fix the simpler issues before the security team was fully engaged.

This approach helps the product development team to avoid the accrual of security debt over time. Maximum efficiency is achieved when this process is fully integrated into the CI pipeline, aligned with the current development workflow. To support this objective, the solution must be lightweight, fast and accurate so development teams can remain focused on customer deliverables.

# Closing Thoughts

Hopefully you can relate to the journey that BlackBerry has taken, and continues to take, to ensure the most secure products in the market. These approaches can be taken by any organization that wants to create and/or enhance their own capabilities. It is critical that the processes, the expertise and the tools are implemented to drive continuous improvement in software security and quality.

Look for future articles highlighting the unique capabilities of BlackBerry Jarvis, which will help you outpace attackers and maintain a competitive advantage in your chosen markets.

In the meantime, you can review the BlackBerry Jarvis product brochure or email our experts at jarvis@blackberry.com