Electronic Design

Jamie Smith, Head of Strategic Growth, JKI, https://www.jki.net

Securing Mission-Critical LabVIEW Apps for the Cyber Resilience Act

Why static analysis and secure development practices matter more than ever.

For more than three decades, National Instruments technologies and LabVIEW have helped engineers design, validate, and test some of the world's most important systems. LabVIEW applications are found throughout aerospace and defense platforms, medical devices, semiconductor manufacturing equipment, industrial automation systems, autonomous vehicles, scientific research facilities, and countless test and measurement environments.

Throughout my career, I have worked on systems ranging from medical devices and autonomous agricultural equipment to advanced manufacturing platforms. Today at JKI, we continue to support organizations that rely on LabVIEW to develop and test mission-critical products. These are systems that directly affect safety, reliability, national security, and economic productivity.

As software continues to become a larger percentage of the value and functionality of modern systems, the expectations placed on development organizations are changing. It's no longer sufficient for software to simply work. Organizations must demonstrate that software is trustworthy, secure, maintainable, and developed using disciplined engineering processes.

This shift is being driven by both increasing cybersecurity threats and evolving regulations such as Executive Order 14028 in the United States and the European Union's Cyber Resilience Act (CRA). Together, these initiatives are changing how organizations think about software quality, software assurance, and cybersecurity throughout the product lifecycle.

For LabVIEW development teams, this raises an important question: How do we apply modern software assurance practices to graphical programming environments that were never originally designed with today's cybersecurity requirements in mind?

The Growing Importance of Software Assurance

Historically, engineering organizations focused their testing efforts on validating functionality against requirements, ensuring the systems meet all environmental, functional, and performance design goals.

These questions remain critical, but we need more fundamental testing of SW components as today's products increasingly incorporate network connectivity, cloud integration, third-party software components, remote update capabilities, and complex supply chains. Such systems introduce cybersecurity weaknesses that lead to operational, financial, safety, and regulatory risks.

Organizations must now identify vulnerabilities before deployment, understand the software components included in their systems, maintain software inventories, and demonstrate secure development practices.

The challenge is that many engineering teams using LabVIEW have access to far fewer security-focused tools than developers working in languages such as C++, Java, Python, or C#.

This is where static analysis becomes increasingly important.

What is Static Analysis?

Static analysis examines source code without executing it. Rather than waiting for vulnerabilities to appear during testing or after deployment, static analysis identifies potential issues directly within the application's source code.

Modern static-analysis tools can detect:

- Hard-coded passwords and credentials
- Race conditions
- Unsafe use of external libraries
- Obsolete functions
- Deadlock conditions
- Command injection vulnerabilities

The earlier these issues are identified, the less expensive they are to fix. This concept is referred to as shift-left, the idea of shifting defect detection and redemption earlier in the development process for efficiency and savings.

For organizations managing large LabVIEW codebases, static analysis provides a scalable way to evaluate thousands of virtual instruments (VIs) and libraries that would be impractical to review manually.

Understanding Common Weakness Enumerations

One of the most important developments in modern software security has been the creation of industry-standard vulnerability taxonomies.

The most widely used is the Common Weakness Enumeration (CWE) framework maintained by MITRE. A CWE is a standardized category of software weakness that can lead to security vulnerabilities.

Examples include:

- CWE-78: OS Command Injection
- CWE-798: Hard-Coded Credentials
- CWE-362: Race Conditions
- CWE-833: Deadlocks
- CWE-477: Use of Obsolete Functions
- CWE-787: Out-of-Bounds Write
- CWE-20: Validate User Input

Rather than inventing proprietary security terminology, cybersecurity tools increasingly map findings to CWE identifiers. This provides a common language that can be shared among developers, security teams, auditors, regulators, and management.

The JKI Security Suite follows this same approach by mapping LabVIEW security findings to industry-standard CWEs. It is important to map the MITRE CWE definitions into a LabVIEW context for evaluation and testing. This allows organizations to evaluate LabVIEW applications using the same security framework they already use for other programming languages.

Security teams can review LabVIEW applications alongside C++, Python, Java, and web applications using a common set of definitions and reporting structures. It allows them to integrate the findings from multiple tools and languages into a standard report and evaluation process.

Bringing Modern Security Analysis to LabVIEW

The JKI Security Suite was leveraged to ensure the LabVIEW applications developed to be used by NASA at their ARC Sine testing facility complied with their 7150.2D SW development process, which includes the need for static analysis. These tools have been generalized to enable software assurance within all LabVIEW-based systems.

The platform performs static analysis of LabVIEW source code and identifies potential vulnerabilities, security weaknesses, architectural concerns, and software-quality issues.

Current security-analysis capabilities include detection of:

- Hard-coded credentials and secrets (CWE-798)
- Command injection risks (CWE-77 and CWE-78)
- Out-of-bounds memory access concerns (CWE-787)
- Race conditions (CWE-362)
- Deadlocks (CWE-833)
- Unsafe loop behavior
- Deprecated and obsolete functions (CWE-477)

Beyond security-specific analysis, the platform also collects software metrics, identifies broken code, evaluates architectural patterns (such as state-machine implementations), audits framework usage, and provides insights into code complexity and maintainability. It will also analyze the existence of unit test frameworks in production applications.

The result is a more comprehensive view of software health that supports both cybersecurity and long-term maintainability.

Software Bills of Materials and Supply-Chain Visibility

Another major requirement emerging from modern cybersecurity regulations is the need for software bills of materials (SBOMs).

An SBOM serves as an inventory of software components included within an application. Much like a manufacturing bill of materials identifies physical parts within a product, an SBOM identifies software dependencies, libraries, packages, and components.

The Cyber Resilience Act, Executive Order 14028, and numerous industry cybersecurity frameworks increasingly emphasize software transparency and supply-chain visibility.

When a new vulnerability is disclosed in a third-party component, organizations with accurate SBOMs can quickly determine whether they're affected and respond appropriately.

For many engineering organizations, creating and maintaining SBOMs for LabVIEW applications has historically been difficult. Modern tooling now makes it possible to automate this process and generate machine-readable SBOMs that integrate into broader software assurance workflows. Both JKI and Emerson are working to provide the SBOM tools needed to maintain these lists of dependencies.

Open Standards Matter

One of the most important design decisions in any security program is whether to adopt proprietary formats or open standards.

Modern engineering systems are rarely developed using a single programming language. A medical device may com-

bine LabVIEW test systems, embedded C firmware, Python scripts, and cloud services. An aerospace platform may involve FPGA code, LabVIEW applications, Linux services, and model-based engineering artifacts, such as SysML v2.

Security information becomes significantly more valuable when it can move easily between tools and organizations. It's important to design security tools around open standards and industry-recognized formats wherever possible. Security findings, vulnerability information, and SBOM data can be integrated into larger software assurance ecosystems rather than existing in isolation. More stakeholders will be able to understand the reports and take steps to mitigate and evaluate the issues.

The use of open and standard formats simplifies regulatory audits, cybersecurity assessments, and enterprise security reviews because LabVIEW applications can participate in the same governance processes already established for other software technologies.

Security in the CI/CD Pipeline

Another major shift occurring across engineering organizations is the adoption of continuous integration and continuous delivery (CI/CD) practices.

Historically, security reviews were often conducted near the end of a project, immediately before deployment. By that point, addressing significant issues could be expensive and disruptive.

Modern DevSecOps approaches seek to move security earlier in the development lifecycle. Security tools must support automation through robust and flexible command line interfaces (CLIs).

By integrating static analysis and SBOM generation directly into automated build pipelines, development teams can identify security concerns early. Teams will often perform scans during code commit or during the build process. Automated analysis provides faster feedback and creates a repeatable audit trail demonstrating that security checks were performed throughout development.

For organizations seeking compliance with emerging regulations, CI/CD integration can transform cybersecurity from a periodic activity into a continuous engineering practice.

Preparing for the Cyber Resilience Act

The European Cyber Resilience Act represents one of the most significant regulatory changes affecting software-enabled products.

While specific obligations depend on product classifications and deployment scenarios, the overall direction is clear. Manufacturers must demonstrate greater control over software security, vulnerability management, software inventories, and secure development processes.

Organizations that use LabVIEW to develop products, or to test products being sold into regulated markets, should begin evaluating how their current development processes align with these emerging expectations.

Static analysis, secure development practices, software inventories, vulnerability management, and automated compliance evidence are rapidly becoming standard engineering requirements rather than optional activities.

The Road Ahead

LabVIEW has earned its place in the engineering world by enabling small teams to solve difficult measurement, control, and automation challenges quickly and effectively. That value proposition remains as strong as ever.

At the same time, the cybersecurity expectations placed on software have changed dramatically. Mission-critical systems now require the same level of software assurance that organizations have long applied to safety, reliability, and performance.

The good news is that the tools and methodologies needed to meet these expectations are becoming available to the LabVIEW ecosystem.

By combining static analysis, CWE-based vulnerability detection, SBOM generation, open standards, and CI/CD integration, engineering organizations can begin applying modern secure software-development practices to the systems they depend on every day.

The result is not only improved compliance with emerging regulations, but ultimately more trustworthy software, more resilient products, and greater confidence in the systems that increasingly power our world.

.