WHITE PAPER

HOW IOT IS MAKING SECURITY IMPERATIVE FOR ALL EMBEDDED SOFTWARE

WHY EMBEDDED SOFTWARE DEVELOPMENT NEEDS TO CHANGE AND WHAT ORGANIZATIONS CAN DO TO IMPROVE SOFTWARE SECURITY WHILE REDUCING DEVELOPMENT TIME

www.programmingresearch.com

PRQA Programming Research

THE DEMAND FOR EMBEDDED SOFTWARE IS EXPLODING

We now live in a world of interconnected things — where humans are interacting with a variety of machines and devices that are in turn communicating with other machines and devices — and it is here to stay. Studies project that by 2020, the number of connected devices will grow to between 45 and 60 billion. This growth is not driven by the anticipated increase in human population, but rather by the unprecedented growth of connected devices, including those we use every day such as refrigerators, cars, thermostats, and lights as well as critical infrastructure and operational technologies such as those found in transportation systems and on the factory floor.

> *We believe that every industrial company will become a software company.*
>
> GE CEO and Chairman Jeff Immelt

Manufacturers in the appliance, automotive, consumer electronics, and medical device industries are rapidly expanding the use of embedded devices powered by software, making smarter products and adding new features and capabilities. Analysts expect numerous other industries to embrace the Internet of Things (IoT), and companies in these industries will require software for their smart, interconnected devices. To meet the growing demand for software and to keep up with rapidly changing business and consumer trends, developers are under pressure to write and reuse more code than ever, to deliver newer and better features, and to do it all faster.

THE NEED FOR SECURITY IS GROWING SHARPLY

As developers produce more and more software to power new IoT products, they introduce new risks and bring to market devices vulnerable to security attacks. Cutting-edge hackers are acutely aware that many of the security procedures and applications in use today have been designed to defend against attacks on personal computers, not mobile and embedded systems.

Security problems often stem from the need to accelerate development and bring new products to market ahead of the competition. A majority of security vulnerabilities are a result of coding errors that go undetected in the development stage. In fact, several recent studies have all identified coding issues as the primary cause of exploitable vulnerabilities. For example, Carnegie Mellon's Computer Emergency Response Team (CERT) found that 64% of vulnerabilities in the CERT National Vulnerability Database were the result of programming errors. Similarly, the U.S. National Institute of Standards and Technology (NIST), which tracks cyber-security issues, found that the application layer is among the leading attack vectors, accounting for 92% of the vulnerabilities reported.

FIAT CHRYSLER AUTOMOBILES IS RECALLING 1.4 MILLION CARS AND TRUCKS TO UPDATE SECURITY TO BLOCK POSSIBLE HACKING ATTEMPTS.

Fiat Chrysler Automobiles, under pressure from federal regulators, said Friday it will recall 1.4 million cars and trucks to protect them from cybersecurity attacks just days after Wired magazine revealed that a Jeep Cherokee could be hacked remotely.

The National Highway Traffic Safety Administration has launched an investigation so it can closely monitor the recall.

Detroit Free Press July 24, 2015

Issues that can introduce vulnerabilities include corrupted data, overflows, dangling pointers, and uninitialized data. These issues all contribute to attack surfaces, which hackers can exploit once the flaws are discovered.

For traditional business and personal computing devices such as smartphones, servers, and personal computers, methods have been established to update software and fix newly found bugs. While updates are not always timely, they are at least easy to perform because these devices are usually on networks and have widely used operating systems. It is significantly more complicated to address software defects and issues on embedded devices, which aren't always connected to a trusted source and may require on-site visits by technicians to update the firmware. In many cases, the end-user might have to bring his device (for example, an implanted pacemaker or an automobile) to a facility where software can be properly updated.

As a result of these limitations, security problems in embedded devices can have an enormous business impact. For example, when an automobile manufacturer needs to perform an onsite software update, additional costs include outreach to customers to notify them, training for mechanics to fix the problem, and significant labor costs to actually perform the update. The company also runs an increased risk of losing unsatisfied customers to competitors and difficulties in attracting new customers.

Moreover, with increased privacy concerns and the many rules and regulations for data protection, including Health Insurance Portability and Accountability Act (HIPAA) and Payment Card Industry Data Security Standard (PCI DSS) regulations in the U.S., security issues can lead to lawsuits, penalties, and fines in addition to reputational damage.

Cyber-based attacks that were once solely in the realm of movies and TV fiction are now uncomfortable realities and must be taken seriously. With attacks such as Stuxnet and the remote control hacks of Chrysler's Jeep in the news, it is only a matter of time before product manufacturers face the legal consequences due to a product that becomes compromised. With so many people depending upon devices for health and protection, there is a very real possibility that someone will be injured or killed by compromised embedded software in an airplane, automobile, pacemaker, or fire control system.

HACKERS ARE TARGETING "SOFT" INFRASTRUCTURE

Because security for personal computers is improving, hackers are increasingly looking for "softer" targets.

Medical devices, which far outnumber hospital PC workstations, are usually the softest targets in a hospital network, lacking firewalls, malware protection, strong encryption, or even recent security patches or Operating System (OS) updates. Medical devices are increasingly leveraging IP and common OS platforms that enable them to utilize large software libraries and communicate more easily. But in the rush to establish common platforms and network these devices, security concerns have been poorly addressed.

Source: Embedded Computing Design

SECURE BY DESIGN

Although incorporating security features such as encryption and password protection will help to safeguard access to devices and data, such features are insufficient when the application code contains defects that render it vulnerable. So while architects strive for more secure features and designs, the best approach for securing embedded software applications is to find and address coding issues at an early stage and then deliver high-quality, defect-free code. To do this, developers need tools that can help them ensure that the code they write is free from known weaknesses and follows proven guidelines and standards.

UNIQUE CHALLENGES OF SECURING EMBEDDED APPLICATIONS

Developers have long been concerned with the quality of the software they create and have processes in place to detect and eliminate defects that adversely affect quality. Many organizations, however, have not yet adopted strategies for ensuring the security of the software they create. Because fixing issues in a deployed embedded device is both costly and difficult, addressing both quality and security problems in the early stages of development is imperative.

Compared to their counterparts who develop software for traditional devices, including computers and smartphones, embedded developers have far more variables to consider. Embedded developers face the unique and almost impossible challenge of gaining a deep understanding and proficiency in multiple combinations of operating system, platform, I/O interface, and language. Traditional developers work on a limited number of platforms, enabling them to become more familiar with specific security issues and the areas in which common software vulnerabilities can occur and be prevented. In contrast, embedded developers often work on a variety of platforms, each of which might handle data storage and memory usage in a different way. New platforms are introduced frequently, making it almost impossible for embedded developers to thoroughly understand the unique vulnerabilities of each OS/platform/language/interface combination.

SECURITY IS A PROCESS NOT A FEATURE

Because many developers lack security training, it's no surprise that they have difficulty finding security problems during code reviews. However, organizations can mitigate this weakness by incorporating coding standards into their development process and adhering to guidelines from organizations such as CERT (including the Common Weakness Enumeration, or CWE, standards) to avoid known vulnerabilities.

By employing an advanced static analysis platform, organizations can compensate for their developers' lack of security experience and eliminate the inefficiency and ineffectiveness of manual code reviews to analyze and review millions of lines of code. Automated static code analysis tools can detect coding defects and security vulnerabilities at an early stage of embedded software development. Characteristics of advanced static testing tools include:

- Customizable and finely tuned parsing capabilities to identify security issues in an application.
- Diagnostics and reporting of potential defects and remediation.
- Testing for compliance with current security standards.

ENABLING EARLY DETECTION AND PREVENTION

Most developers work under immense time pressures and are evaluated on their ability to produce quality software. Training a development team to develop secure software inevitably takes time away from development activities and will put additional strain on deadlines. While some organizations look to quality assurance teams to perform security tests, at some point developers still need to make modifications to the code.

In his study "Software Defect Origins and Removal Methods," Capers Jones finds that "Due to low defect removal efficiency at least eight forms of testing are needed to achieve reasonably efficient defect removal efficiency." He adds, "Testing by itself without any pre-test inspections or static analysis is not sufficient to achieve high-quality levels," later concluding, "Pre-test inspections and static analysis are synergistic with testing and raise testing efficiency."

Static analysis tools, when configured properly and used in the early stages of development, enable developers to spot known vulnerabilities and identify weaknesses, so they can take corrective actions immediately. This approach makes it possible to identify most defects in early development when they are easiest to eliminate, instead of during late-stage tests and deployment where they are orders of magnitude more costly and time-consuming to correct.

INTRODUCING STATIC ANALYSIS

Static code analysis tools, integrated into a developer's integrated development environment (IDE) and incorporated into a team's existing workflow, enable the early and automated detection of key security issues and vulnerabilities. A process in which developers run frequent analytics on their code (either on the desktop or via a server-based model) provides developers with the quick feedback they need to make corrections as the code is being written. This approach not only enables developers to find vulnerabilities, it also provides guidance on how to correct coding errors to prevent defects. As a result, developers can immediately take corrective action to deliver code that is more secure and more reliable.

Static code analysis tools can address a variety of security issues, including unvalidated user input, buffer overflows, code injection, unprotected data, insecure APIs, resource and memory leaks, race conditions, and dereferencing Null pointers, as well as errors related to memory allocation, resource management, and use of uninitialized data.

Rather than rely on manual code reviews to find these issues, teams can use these tools to automate the detection of potential security vulnerabilities in the source code. Static analyzers give developers a consistent mechanism to identify, fix, and manage security vulnerabilities. Over time, these tools also serve to enhance developers' knowledge of secure coding practices.

Two of the most prominent security initiatives related to secure software development are the CWE database project and the CERT C coding standard. The CWE database includes security issues related to multiple programming languages and can be applied to a broad range of application types, including web, desktop, mobile, and embedded. In contrast, the CERT C coding standard is focused specifically on the C language, which is the language most widely used in embedded software application development. An increasing number of organizations are making adherence to these guidelines and standards a requirement for both internal development organizations and outsourced application development vendors.

Development organizations can now employ static analyzers that include built-in checkers for CWE related coding errors and that support the CERT C standards, enabling developers to address errors before code check-in. Static analysis tools can automatically aggregate information from the whole development team about what errors are being found and fixed, giving teams a better understanding of defect reduction trends. These tools can be used to generate an organization-wide view of how well defect elimination efforts are working and provide critical insights that enable management to rapidly identify areas within the code base that have the greatest risk.

SUMMARY: FASTER AND MORE SECURE DEVELOPMENT

A renewed emphasis on software security is needed to address vulnerabilities in the proliferation of devices dependent on software, including medical devices, appliances, entertainment systems, automobiles, and the stand-alone sensors deployed as part of an organization's IoT initiative.

Identifying and eliminating the defects and standards violations that lead to insecure code at the development stage is easier and less costly than updating a product when vulnerabilities are discovered after its release. This early identification is made possible by static code analysis tools integrated into developers' IDEs and seamlessly incorporated into their development workflows.

With static analysis tools, developers can detect security problems at the early stages of software projects before code is compiled, executed, or tested, and capitalize on the opportunity to improve the quality and security of the software they write for embedded devices.

Performing checks to ensure their software complies with CWE guidelines and CERT standards gives developers an objective measure of their work. The checks also provide managers and organizations a means of documenting that their software complies with established security standards and requirements.

Static analysis reduces the number of iterations needed to produce software, cuts testing and development costs, speeds development, and enables the delivery of more secure software.

ABOUT PRQA

Established in 1985, PRQA is recognized throughout the industry as a pioneer in static analysis, championing automated coding standard inspection and defect detection, delivering its expertise through industry-leading software inspection and standards enforcement technology.

PRQA static analysis tools, QA·C and QA·C++, are at the forefront in delivering MISRA C and MISRA C++ compliance checking as well as a host of other valuable analysis capabilities. All contain powerful, proprietary parsing engines combined with deep accurate dataflow that deliver high fidelity language analysis and comprehension. They identify problems caused by language usage that is dangerous, overly complex, non-portable or difficult to maintain. Additionally, they provide a mechanism for coding standard enforcement.

CONTACT US

PRQA has offices globally and offers worldwide customer support. Visit our website to find details of your local representative.
Email: info@programmingresearch.com
Web: www.programmingresearch.com