Software Engineering Institute
Carnegie Mellon University

methodpark

CYBERSECURITY CONSIDERATIONS FOR VEHICLES

Mark S. Sherman, Software Engineering Institute
Jens Palluch, Method Park
December 2015

Historically, cybersecurity has been mainly considered for software of computer systems and networks. In recent years, the number of electronic control units (ECUs) in modern vehicles, as well as the amount of software and importance of software used in vehicles, has been increasing. Modern vehicles contain up to 100 ECUs, which are connected to fulfill their respective functions.[1] Many of these ECUs are safety related, such as airbag control, adaptive cruise control, and the braking system. Because ECUs within a vehicle are interconnected, the security of the safety-related ECUs depends strongly on the security of the other ECUs. Cybersecurity needs to expand to include vehicles.

The Latest Wakeup Call: Hacking a Jeep Cherokee

Although it has been shown in the last years that the ECUs in a vehicle can be successfully attacked, few corrective actions have been taken because previous attacks required physical access to a vehicle. But today, some of the ECUs in a vehicle have connections to the outside via cellular connections, thereby providing an attack surface that does not require physical access. For example, infotainment systems could provide access to the Internet by connecting a vehicle to the rest of the world.

Valasek demonstrated a hack to remotely control a Jeep Cherokee.[2] It was possible for the attacker to control the steering and braking systems by sending commands via the infotainment system. In that case, the safety was especially compromised through insecure systems and it forced Fiat Chrysler to recall 1.4 million vehicles.[3]

The Jeep experiment illustrated how security vulnerabilities could lead to a loss of safety. Security vulnerabilities could also have an impact on performance or other quality properties. As stated by the former president of the European Commission, José Manuel Barroso: “There is no safety without security and vice-versa.”[4]

---

[1] C. Miller, C. Valasek: A Survey of Remote Automotive Attack Surfaces. Available: illmatics.com/remote%20attack%20surfaces.pdf

[2] http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

[3] http://www.wired.com/2015/07/jeep-hack-chrysler-recalls-1-4m-vehicles-bug-fix/

[4] http://europa.eu/rapid/press-release_SPEECH-12-227_en.htm?locale=en

The deployment of Smart Transportation further connects vehicles with each other and the transportation infrastructure. The expanded connectivity enlarges the remotely accessible attack surface. Therefore cybersecurity has to be strongly considered for vehicles.

Back-End Costs Can Be Enormous

Recalls of vehicles can lead to enormous costs. Fiat Chrysler had to reserve $761 million for future recall costs in North America resulting in a $339 million loss in the third quarter of 2015. Further, the company had to pay a $105 million penalty in July 2015 for violating laws in performing 23 recalls—prior to the Jeep recall—with 11 million vehicles involved.[5]

Most recalls today are safety related, but the Jeep experience suggests that the number of security-related recalls will increase.

Organizational Resiliency

The automotive community has accepted that safety concerns are a basic part of the design, construction, and maintenance of a vehicle. Recent history compels the industry to recognize that security must also be built into the software of vehicles. Just as organizations must incorporate safety concerns into their organizational culture, processes, and behaviors, organizations must add security concerns into the fabric of the automotive business.

Security must be integrated within the organization. Every member of this organization needs to be aware of security. The management must take over responsibility and drive the implementation of security. This includes not only defining security policies, roles, and responsibilities, but also providing enough experienced resources and appropriate tools.

People at many levels and in many functions within an organization need to be trained for increased awareness of security. Role-specific training is necessary especially for the methods of a secure software development lifecycle, to establish the capability to build secure vehicle software.

To build up organizational resiliency it is helpful to implement an information security management system. ISO 27001 defines the requirements for such a system. Nowadays some of the original equipment manufacturers (OEMs) in the automotive industry require that their suppliers have in place such an information security management system in compliance with ISO 27001. Finally, a certification of the information security management system according to ISO 27001 is visible evidence of the implementation of organizational aspects and demonstrates that the organization takes care of security.

---

[5] http://www.bloomberg.com/news/articles/2015-10-28/fiat-chrysler-profit-rises-35-as-jeep-suv-deliveries-climb

Developing Secure Software for Vehicles

The software being developed for vehicles should be created in a way that makes the vehicle resilient to cyberattacks. We call this process for developing resilient software the Secure Software Development Lifecycle.

Building resilient software requires attention to all elements of the development lifecycle.[6] Studies have shown that more than 80% of current practices omit security concerns in one or more phases of the development lifecycle, yet a system is only as secure as its weakest link.

The lifecycle has three overlapping, iterative phases:

- requirements
- engineering and development
- deployment and operations

These elements are illustrated in Figure 1.

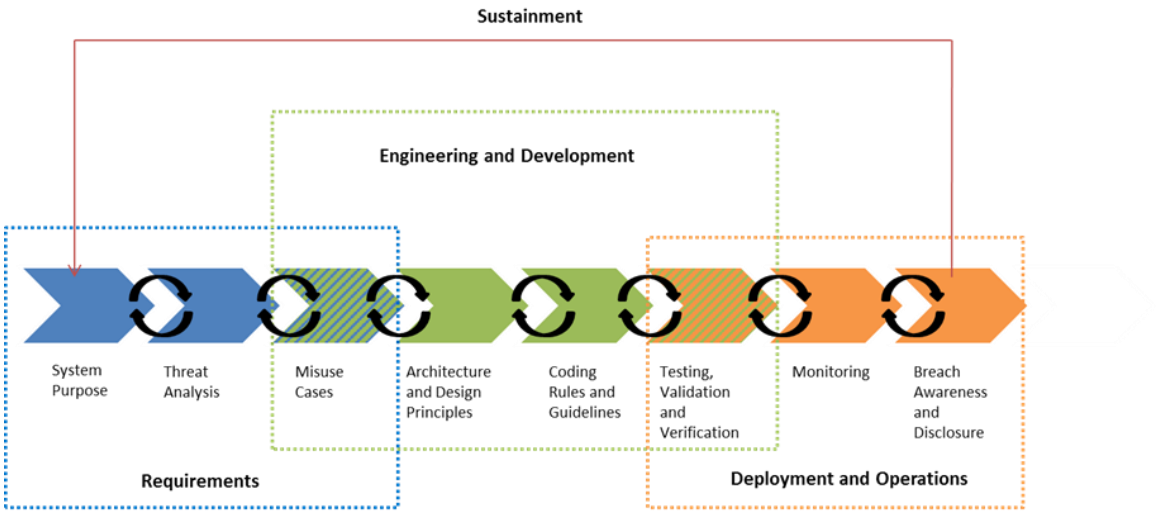

Figure 1: The Secure Software Development Lifecycle

[6] Forrester Consulting, "State of Application Security," January 2011.

Requirements

The requirements phase starts with an understanding of the multiple automotive systems that are to be supported by the software, such as the infotainment system or braking system. The description should cover what the system is supposed to accomplish from several viewpoints:

- physical layout
- process flow
- data flow and access
- human actors, their roles and access
- operational procedures
- network layout

From the description of the system and the system composition, developers can then extend the "use cases" for expected operation of the system with "misuse" cases. Misuse cases describe how the system might be used outside of its expected function. Taken together with the actors who could carry out the misuse and the actors' motivations, developers can then construct a threat model. For example, security concerns could include the possibility of an individual standing on the side of the road pointing a pen laser at a LIDAR unit on a vehicle. Or criminals could access a vehicle through a dealer's real-time maintenance network to target a desirable vehicle for acquisition and disassembly.

The system description also serves as the basis for describing the attack surface, that is, how an actor can gain access to the system to carry out the misuse cases. For example, a high-end vehicle can have up to 50 antennas, each of which could be a point of entry.[7]

Engineering and Development

Constructing the threat model provides a bridge from the requirements phase to the engineering and development phase.

In the engineering and development phase, the threat model and attack surfaces are used to start creating security requirements. For example, some surfaces, such as those communicating with unauthenticated cell towers, are more accessible to unknown parties and require a defensive-in-depth engineering methodology. Some surfaces protected by physical constraints might need less security checking—a reduction that is desirable for coping with other vehicle constraints, such as those for space, power, or weight.

The security requirements can help developers make architecture and then design decisions. A skilled architect can use the requirements to make recommendations on system decomposition, containment and redundancy strategies, encryption of data in motion or at rest, and implementations of authentication mechanisms and authorization policies.

---

[7] https://www.cst.com/webinar14-10-23~?utm_source=rfg&utm_medium=web&utm_content=mobile&utm_campaign=2014series

The implementation of the design should follow the appropriate programming hygiene. Very flexible languages like C are used to implement many systems in vehicles. Developers who have not focused on writing secure C code have historically introduced unintended ambiguities in their programs, which have been exploited to generate buffer overflows, cross-site scripting, and other common attack methods. A practical way to minimize such attack methods is to implement a discipline of programming. Much like MISRA focused on providing guidance for building software for safety-sensitive deployment, the CERT Secure Coding Standards provide programming guidance for security concerns[8]. Publications, education, and automated tools are all available to assist in implementing the discipline.

Secure Agile Development

The development team, which will typically use Agile development processes, also needs to integrate security processes into their Scrum and DevOps environment. The integration can be carried out in a non-disruptive way that makes security the natural consequence of development rather than a distractive requirement, as summarized in Figure 2.

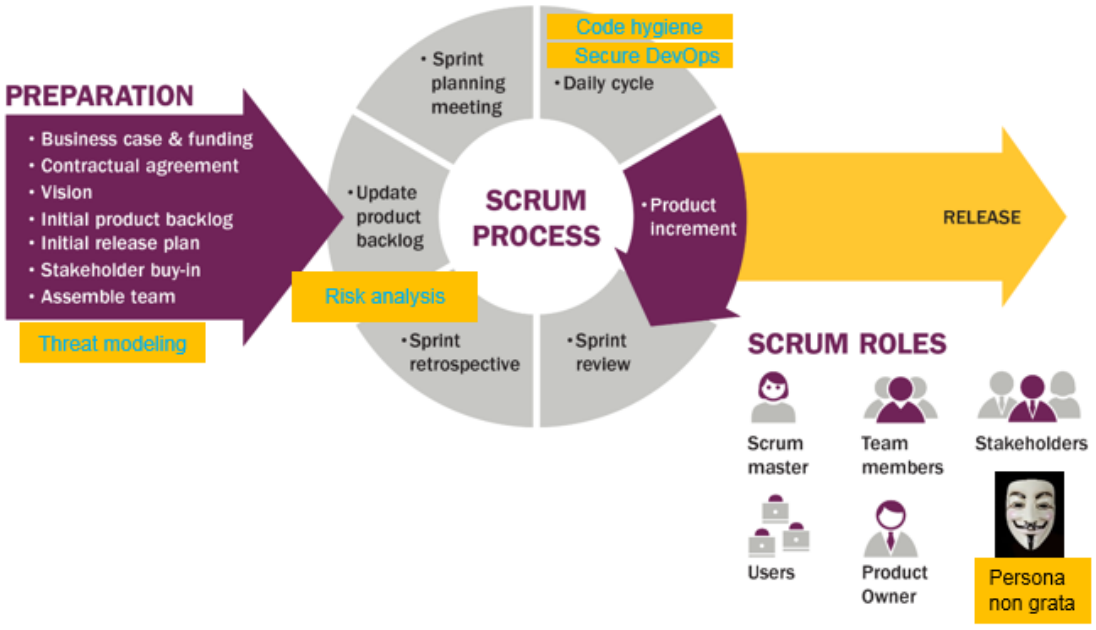

Figure 2: Secure Agile Development with SCRUM

In addition to adopting good programming hygiene—necessary for most quality improvements—secure analysis tools can be added to the automation system. The tools can check for security issues in the code

[8] https://www.securecoding.cert.org/

at the same time that other automated tests are run, encouraging rapid correction of security weaknesses while other defects are being corrected. The earlier development of a threat model naturally leads to the creation and maintenance of stories for misuse. One way to structure these security stories is to add another "persona" or stakeholder representative to the stand ups. The product owner is already representing a collection of stakeholders, from users to administrator to business owners. The addition of a "persona non grata," representing attackers, to the product owner's stakeholders provides a natural way to extend security needs represented in stories. After the stories are defined, they can be prioritized and estimated using established techniques.

Third-Party Software and the Software Supply Chain

Most software, including many of the tens of millions of lines of code in a modern vehicle, comes from a third-party source that must be secure. Secure Agile development can improve software written in house, but in practice, applications can better be described as assembled rather than built. The assembly uses many third-party components, including open source code of unknown provenance. The Heartbleed and Shellshock vulnerabilities have made the larger community sensitive to the potential for weaknesses in open source software. The risks of using third-party, open source software can be mitigated by putting in place procedures that ensure that materials are retrieved from reliable repositories and that inventories of deployment for components are kept.

Deployment and Operations

The final phase in the lifecycle is deployment and operations. This phase focuses on ensuring that the software has gone through an appropriate evaluation for readiness to deploy, and that the operational environment contains the expected security procedures. A complete operational environment would also provide a process for vulnerability disclosure and management when attacks are successful.

An evaluation process could contain many different steps to provide confidence in the software's readiness. Procedural steps, such as making sure that there is a record of the final bill of materials, is critical. As weaknesses are uncovered in components—either third-party or internally developed—fixed components need to be delivered to all affected systems. This can be done only if the location of all components can be identified. Maintenance procedures to deliver the updated component must be defined and tested as well.

Technical steps can be performed to check for system security. Fuzz testing and red teams are two examples of technical steps that organizations can employ.

Operational state and situational awareness are key to maintaining a system's security. The very beginning of the lifecycle starts with an analysis of the operational environment that could lead to tradeoffs and decisions in the architecture, design, and development. The most common assumption is that a system is isolated in some way, whether through network isolation, an "air gap," or by limited physical access. As the automotive world becomes more widely connected, these assumptions change, and the resulting security assumptions violated. In practice, security violations frequently occur in existing code

that is put into a new operational environment. The Jeep example mentioned earlier illustrated this effect. Maintaining operational awareness and evaluating environmental changes are necessary for good software security.

Summary

Software is a powerful enabler of diverse functionality. Connectivity amplifies the value of intelligent vehicles enabled by software. With power comes increased complexity that can be difficult to manage, leading to security vulnerabilities. Instituting an organizational culture that places security needs at the same level of scrutiny as safety concerns is critical because security violations frequently turn into safety violations. The organizational commitment needs to include adoption of secure software development throughout its lifecycle.

About the Software Engineering Institute and Method Park

The Software Engineering Institute (SEI) is a U.S. Department of Defense Federally Funded Research and Development Center supporting research and government projects in software engineering and cybersecurity. The SEI works with government and industry worldwide to build robust, safe, and secure software for enterprises, cyber-physical systems, and mission critical systems.

Method Park, based in Erlangen, Germany, provides consulting in software for safety-critical systems in the automotive industry and the medical technology area, for which the company develops its own software solutions. Method Park brings extensive know-how to fields with high and extremely high safety requirements.

Contact Us

Software Engineering Institute
4500 Fifth Avenue, Pittsburgh, PA 15213-2612

Phone: 412/268.5800 | 888.201.4479
Web: www.sei.cmu.edu | www.cert.org
Email: info@sei.cmu.edu

Method Park
Wetterkreuz 19a, 91058 Erlangen, Germany

Phone: +49.9131.97206.0
Web: www.methodpark.com
Email: info@methodpark.de

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.