1.	Record Nr.	UNINA9910555069303321
	Autore	Oka Dennis Kengo
	Titolo	Building secure cars : assuring the automotive software development lifecycle / / Dennis Kengo Oka
	Pubbl/distr/stampa	Hoboken, New Jersey : , : Wiley, , [2021] ©2021
	ISBN	1-119-71077-4 1-119-71078-2 1-119-71076-6
	Descrizione fisica	1 online resource (xiii, 304 pages) : illustrations
	Disciplina	629.272
	Soggetti	Automotive telematics - Security measures
	Lingua di pubblicazione	Inglese
	Formato	Materiale a stampa
	Livello bibliografico	Monografia
	Nota di contenuto	Cover -- Title Page -- Copyright -- Contents -- Preface -- About the Author -- Chapter 1 Overview of the Current State of Cybersecurity in the Automotive Industry -- 1.1 Cybersecurity Standards, Guidelines, and Activities -- 1.2 Process Changes, Organizational Changes, and New Solutions -- 1.3 Results from a Survey on Cybersecurity Practices in the Automotive Industry -- 1.3.1 Survey Methods -- 1.3.2 Report Results -- 1.3.2.1 Organizational Challenges -- 1.3.2.2 Technical Challenges -- 1.3.2.3 Product Development and Security Testing Challenges -- 1.3.2.4 Supply Chain and ThirdParty Components Challenges -- 1.3.3 How to Address the Challenges -- 1.3.3.1 Organizational Takeaways -- 1.3.3.2 Technical Takeaways -- 1.3.3.3 Product Development and Security Testing Takeaways -- 1.3.3.4 Supply Chain and ThirdParty Components Takeaways -- 1.3.3.5 Getting Started -- 1.3.3.6 Practical Examples of Organizations Who Have Started -- 1.3.3.7 -- 1.4 Examples of Vulnerabilities in the Automotive Industry -- 1.5 Chapter Summary -- References -- Chapter 2 Introduction to Security in the Automotive Software Development Lifecycle -- 2.1 VModel Software Development Process -- 2.2 Challenges in Automotive Software Development -- 2.3 Security Solutions at each Step in the VModel -- 2.3.1 Cybersecurity Requirements Review -- 2.3.2 Security Design Review -- 2.3.3 Threat

Analysis and Risk Assessment -- 2.3.4 Source Code Review -- 2.3.5 Static Code Analysis -- 2.3.6 Software Composition Analysis -- 2.3.7 Security Functional Testing -- 2.3.8 Vulnerability Scanning -- 2.3.9 Fuzz Testing -- 2.3.10 Penetration Testing -- 2.3.11 Incident Response and Updates -- 2.3.12 Continuous Cybersecurity Activities -- 2.3.13 Overall Cybersecurity Management -- 2.4 New Technical Challenges -- 2.5 Chapter Summary -- References -- Chapter 3 AutomotiveGrade Secure Hardware.
3.1 Need for Automotive Secure Hardware -- 3.2 Different Types of HSMs -- 3.3 Root of Trust: Security Features Provided by Automotive HSM -- 3.3.1 Secure Boot -- 3.3.2 Secure InVehicle Communication -- 3.3.3 Secure Host Flashing -- 3.3.4 Secure Debug Access -- 3.3.5 Secure Logging -- 3.4 Chapter Summary -- References -- Chapter 4 Need for Automated Security Solutions in the Automotive Software Development Lifecycle -- 4.1 Main Challenges in the Automotive Industry -- 4.2 Automated Security Solutions During the Product Development Phases -- 4.2.1 Static Code Analysis -- 4.2.2 Software Composition Analysis -- 4.2.3 Security Testing -- 4.2.4 Automation and Traceability During Software Development -- 4.3 Solutions During Operations and Maintenance Phases -- 4.3.1 Cybersecurity Monitoring, Vulnerability Management, Incident Response, and OTA Updates -- 4.4 Chapter Summary -- References -- Chapter 5 Static Code Analysis for Automotive Software -- 5.1 Introduction to MISRA and AUTOSAR Coding Guidelines -- 5.2 Problem Statement: MISRA and AUTOSAR Challenges -- 5.3 Solution: Workflow for Code Segmentation, Guideline Policies, and Deviation Management -- 5.3.1 Step 1: Segment the Codebase into Different Categories/Components Based on Risk -- 5.3.2 Step 2: Specify Guideline Policies (Set of Guidelines to Apply) Depending on Risk Categories -- 5.3.3 Step 3: Perform the Scan and Plan the Approach for Prioritization of Findings -- 5.3.4 Step 4: Prioritize Findings Based on the Risk Categories and Guideline Policies and Determine How to Handle Each Finding, e.g. Fix or Leave as Deviation -- 5.3.5 Step 5: Follow a Defined Deviation Management Process, Including Approval Steps -- 5.3.6 Step 6: Report on MISRA or AUTOSAR Coding Guidelines Compliance Including Deviations -- 5.4 Chapter Summary -- References.
Chapter 6 Software Composition Analysis in the Automotive Industry -- 6.1 Software Composition Analysis: Benefits and Usage Scenarios -- 6.2 Problem Statement: Analysis of Automotive Software OpenSource Software Risks -- 6.2.1 Analysis Results -- 6.2.1.1 zlib -- 6.2.1.2 libpng -- 6.2.1.3 OpenSSL -- 6.2.1.4 curl -- 6.2.1.5 Linux Kernel -- 6.2.2 Discussion -- 6.3 Solution: Countermeasures on Process and Technical Levels -- 6.3.1 Fully Inventory OpenSource Software -- 6.3.2 Use Appropriate Software Composition Analysis Approaches -- 6.3.3 Map OpenSource Software to Known Security Vulnerabilities -- 6.3.4 Identify License, Quality, and Security Risks -- 6.3.5 Create and Enforce OpenSource Software Risk Policies -- 6.3.6 Continuously Monitor for New Security Threats and Vulnerabilities -- 6.3.7 Define and Follow Processes for Addressing Vulnerabilities in OpenSource Software -- 6.3.8 How to Get Started -- 6.4 Chapter Summary -- References -- Chapter 7 Overview of Automotive Security Testing Approaches -- 7.1 Practical Security Testing -- 7.1.1 Security Functional Testing -- 7.1.2 Vulnerability Scanning -- 7.1.3 Fuzz Testing -- 7.1.4 Penetration Testing -- 7.2 Frameworks for Security Testing -- 7.3 Focus on Fuzz Testing -- 7.3.1 Fuzz Engine -- 7.3.2 Injector -- 7.3.3 Monitor -- 7.4 Chapter Summary -- References -- Chapter 8 Automating Fuzz Testing of InVehicle Systems by Integrating with Automotive Test Tools -- 8.1 Overview of HIL Systems -- 8.2 Problem Statement: SUT Requires

External Input and Monitoring -- 8.3 Solution: Integrating Fuzz Testing Tools with HIL Systems -- 8.3.1 WhiteBox Approach for Fuzz Testing Using HIL System -- 8.3.1.1 Example Test Setup Using an Engine ECU -- 8.3.1.2 Fuzz Testing Setup for the Engine ECU -- 8.3.1.3 Fuzz Testing Setup Considerations -- 8.3.2 BlackBox Approach for Fuzz Testing Using HIL System.
8.3.2.1 Example Target System Setup Using Engine and Body Control Modules -- 8.3.2.2 Fuzz Testing Setup Using Duplicate Engine and Body Control Modules -- 8.3.2.3 Fuzz Testing Setup Considerations -- 8.4 Chapter Summary -- References -- Chapter 9 Improving Fuzz Testing Coverage by Using Agent Instrumentation -- 9.1 Introduction to Agent Instrumentation -- 9.2 Problem Statement: Undetectable Vulnerabilities -- 9.2.1 Memory Leaks -- 9.2.2 Core Dumps and Zombie Processes -- 9.2.3 Considerations for Addressing Undetectable Vulnerabilities -- 9.3 Solution: Using Agents to Detect Undetectable Vulnerabilities -- 9.3.1 Overview of the Test Environment -- 9.3.2 Modes of Operation -- 9.3.2.1 Synchronous Mode -- 9.3.2.2 Asynchronous Mode -- 9.3.2.3 Hybrid Approach -- 9.3.3 Examples of Agents -- 9.3.3.1 AgentCoreDump -- 9.3.3.2 AgentLogTailer -- 9.3.3.3 AgentProcessMonitor -- 9.3.3.4 AgentPID -- 9.3.3.5 AgentAddressSanitizer -- 9.3.3.6 AgentValgrind -- 9.3.3.7 An Example config.json Configuration File -- 9.3.4 Example Results from Agent Instrumentation -- 9.3.4.1 Bluetooth Fuzz Testing -- 9.3.4.2 WiFi Fuzz Testing -- 9.3.4.3 MQTT Fuzz Testing -- 9.3.4.4 File Format Fuzz Testing -- 9.3.5 Applicability and Automation -- 9.4 Chapter Summary -- References -- Chapter 10 Automating File Fuzzing over USB for Automotive Systems -- 10.1 Need for File Format Fuzzing -- 10.2 Problem Statement: Manual Process for File Format Fuzzing -- 10.3 Solution: Emulated Filesystems to Automate File Format Fuzzing -- 10.3.1 System Architecture Overview -- 10.3.2 Phase One Implementation Example: Prepare Fuzzed Files -- 10.3.3 Phase Two Implementation Example: Automatically Emulate Filesystems -- 10.3.4 Automating User Input -- 10.3.5 Monitor for Exceptions -- 10.4 Chapter Summary -- References.
Chapter 11 Automation and Traceability by Integrating Application Security Testing Tools into ALM Systems -- 11.1 Introduction to ALM Systems -- 11.2 Problem Statement: Tracing Secure Software Development Activities and Results to Requirements and Automating Application Security Testing -- 11.3 Solution: Integrating Application Security Testing Tools with ALM Systems -- 11.3.1 Concept -- 11.3.1.1 Static Code Analysis - Example -- 11.3.1.2 Software Composition Analysis - Example -- 11.3.1.3 Vulnerability Scanning - Example -- 11.3.1.4 Fuzz Testing - Example -- 11.3.1.5 Concept Overview -- 11.3.2 Example Implementation -- 11.3.2.1 Defensics -- 11.3.2.2 codeBeamer ALM -- 11.3.2.3 Jenkins -- 11.3.2.4 SUT -- 11.3.2.5 Implementation Overview -- 11.3.3 Considerations -- 11.4 Chapter Summary -- References -- Chapter 12 Continuous Cybersecurity Monitoring, Vulnerability Management, Incident Response, and Secure OTA Updates -- 12.1 Need for Cybersecurity Monitoring and Secure OTA Updates -- 12.2 Problem Statement: Software Inventory, Monitoring Vulnerabilities, and Vulnerable Vehicles -- 12.3 Solution: Release Management, Monitoring and Tracking, and Secure OTA Updates -- 12.3.1 Release Management -- 12.3.2 Monitoring and Tracking -- 12.3.2.1 Solutions in Other Industries -- 12.3.2.2 Solutions in the Automotive Industry -- 12.3.2.3 Example Automotive SOC Overview -- 12.3.2.4 Example Automotive SOC Workflow -- 12.3.2.5 Newly Detected Vulnerabilities in OpenSource Software - Example -- 12.3.3 Secure OTA Updates -- 12.3.3.1 Identify

Vulnerable Vehicles Targeted for OTA Updates -- 12.3.3.2 Perform Secure OTA Updates -- 12.3.3.3 Target Systems for OTA Updates -- 12.3.3.4 Overview of Secure OTA Update Process for ECUs -- 12.3.3.5 Standardization and Frameworks for OTA Updates -- 12.4 Chapter Summary -- References -- Chapter 13 Summary and Next Steps -- Index.
EULA.