Security Patterns for Automotive Systems

Betty H.C. Cheng, Bradley Doherty, Nick Polanco, and Matthew Pasco
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824 USA

***Abstract*—As automotive systems become increasingly sophisticated with numerous onboard features that support extensive inward and outward facing communication, cybersecurity vulnerabilities are exposed. The relatively recent acknowledgement of automotive cybersecurity challenges has prompted numerous research efforts into developing techniques to handle individual threat vectors, the pathways and threat surfaces by which an attack can be realized. Security design patterns have been developed for many application domains (e.g., enterprise systems, networking systems, and distributed systems), but not much has been explored for automotive systems. This paper introduces a collection of security design patterns targeted for automotive cybersecurity needs. We leverage and extend the *de facto* standard template used to describe design patterns to include fields specific to the automotive domain and SAE J3061 cybersecurity development guidelines.**

I. Introduction

As autonomous systems and features become further integrated into vehicles, software and electronics will account for over 50% of the design overhead [1]. However, the current software and its respective sub-systems have been demonstrated to be vulnerable to cyber threats in controlled testing environments [2], [3], [4]. These types of attacks demonstrate a clear increase in threat vectors and attack surfaces for autonomous features and systems, while illustrating the potential for impacting human safety. In order to support systematic and reusable development practices focused on automotive cybersecurity needs, this paper introduces automotive-focused security design patterns to address cybersecurity vulnerabilities associated with inward facing (i.e., intra-vehicle) and outward facing (i.e., inter-vehicle) communication.

The connectivity of modern vehicles to each other, third party systems, and consumer devices increases the volume and pathways for possible attacks. The increasing automotive security vulnerabilities motivate the development of prevention, detection, and mitigation techniques that take into consideration automotive-specific constraints. Formalizing and abstracting common problem and solution strategies (i.e., designs) into design patterns facilitates rigorous development practices and promotes design reuse [5]. Common software security patterns enable developers to rigorously harden a system against security vulnerabilities [6]. While these security patterns, along with security solutions such as encryption and secure network protocols, have helped to harden many software systems, the unique architecture of automotive systems and constraints on system performance [7] necessitate application-specific design strategies.

In order to leverage successful security patterns, while also addressing the automotive constraints, we have specialized existing security patterns to account for automotive systems' unique architecture and constraints. This framework for hardening security design and assessing security features can be applied to current automotive systems comprising numerous ADAS (automated driver-assistance systems) elements and extended to the emerging fully-autonomous vehicles. Automotive-specific security patterns enable developers to explicitly and rigorously address security as part of the development process for automotive systems in a proactive fashion. We introduce a template to describe the patterns that extends previously-developed security patterns template [8], [9] to include fields, classification schemes, and information specific to the automotive domain [10]. In addition to the textual descriptions, we also include UML (Unified Modeling Language) [11] class and sequence diagrams to respectively describe the structure and the behavior of a given solution strategy.

The automotive cybersecurity patterns leverage and extend previously-developed security patterns to address vulnerabilities specific to automotive systems, with solution strategies that adhere to constraints and contexts relevant to automotive systems and relevant operating environments. As a preliminary step, we performed an extensive review of the research literature (i.e., state of the art), technical reports from regulation and standards bodies (e.g., [12], [10]), and obtained input from our industrial collaborators (i.e., state of the practice). From these sources, we identified threat vectors and attack surfaces, as well as techniques and/or solution strategies to address the respective vulnerabilities. We distinguish vulnerabilities and solutions that apply to onboard (inward) facing communication (e.g., wheel speed sensor and throttle position sensor signals sent to power train control module) versus outward facing communication subsystems (e.g., telematics units and navigation systems that communication with cell towers and global positioning systems, respectively). Based on the state of the art and state of the practice reviews, as well as existing security patterns [6], [8], [9], we collated the information to define a set of automotive cybersecurity

978-1-7281-5125-0/19/$31.00 ©2019 IEEE
DOI 10.1109/MODELS-C.2019.00014

patterns. In order to facilitate their use, we provide a number of descriptors for each pattern that are commonly used in security development, including threat types using Microsoft's STRIDE framework [13], Viega and McGraw security principles [14] promoted by a given pattern, and automotive cybersecurity development guidelines as described in SAE J3061 [10].

The paper presents an ongoing effort with international industrial collaborators working in automotive cybersecurity, including both Tier 1 suppliers and OEMs (original equipment manufacturers) to develop a repository of security patterns.[1] These patterns are being incorporated into development practices that focus on security and its impact on safety for autonomous vehicles, involving different levels of autonomy. The remainder of this paper is organized as follows. Section II discusses related work in the area of security and design patterns. Section III overviews the vehicle's cyber-threat surfaces. Section IV discusses related state of the art security solutions in automotive systems. Section V discusses the use of Security Patterns in automotive systems. Section VI provides a closer look at two sample patterns, and Section VII summarizes the work and outlines future work.

II. Background

Software design patterns make it easier to reuse successful systems and architectures [15]. By using a known solution to a common problem, developers are able to benefit from successful designs and lessons learned from other developers. The original design patterns by Gamma et al. [15] describe a design pattern with four essential parts: the name of the pattern, the problem addressed by the pattern, the solution strategy, and the consequences of the pattern. Following this approach, several hundred abstract patterns have been implemented for general use in software systems (e.g., [9], [16], [17]).

Security Patterns

While the design patterns developed by Gamma et al. are intentionally abstract with the goal of being applicable to a wide range of problems, domain-specific constraints need to be considered for the design. In the case of security patterns, threats to a system need to be monitored using specific security mechanisms for the specific context [6]. Following a similar format for describing a design pattern, security patterns are also given a descriptive name, the security problem addressed (e.g., a class of specific threats), a solution strategy to prevent, detect, and/or mitigate the security vulnerability, and a set of consequences [6].

Research in the field of security patterns has been active for almost two decades [16], [18], [19]. The focus has been on capturing higher level security mechanisms and organizing them into abstract patterns, which has guided cross-application security patterns [20], [21]. The aforementioned work underscores the wide use of security patterns throughout different software systems, including distributed systems [22], enterprise systems [6], [17], and, more recently, cloud computing systems [6], [9]. Security patterns apply to all phases of the software development process, such as requirements gathering, design, and implementation [19]. Within a collection of security patterns, several abstract patterns exist, indicating more specializations of patterns (e.g., [6], [17], [23]). A study by Ito et al. [16] surveys trends in security pattern research over the past decade.

Similar to general design patterns [15], security patterns have been described using specific fields to capture use-cases, consequences of design, etc. We leverage and extend a specific template developed by Wassermann and Cheng [8] and then refined by Konrad et al. [24] that captures important areas of a security pattern, namely: **Name**, **Alias**, **Intent**, **Applicability**, **Motivation**, **Structure**, **Consequences**, **Known Uses**, and **Related Patterns**.

III. Automotive Cybersecurity Vulnerabilities

In this section, we overview *attack vectors* of automotive systems (i.e., interfaces or paths that an attacker uses to exploit a vulnerability [12], such as an open cellular or Bluetooth network. Of particular interest is the internal communication system, the Controller Area Network (CAN Bus) [25] that acts as a medium for a variety of ECUs on a vehicle. We also overview other attack vectors according to the type of access needed to perpetrate an attack (e.g., physical, remote).

CAN Bus

The CAN Bus is the major networking communication infrastructure used within an automotive system. It relies on a broadcasting protocol that allows for any ECU attached to the CAN Bus to both send and receives messages as predefined ECU behavior, independent of the CAN Bus [26]. The CAN Bus is designed to be lightweight and fast, and consequently lacks common communication protocol features such as authentication or encryption. While data on a CAN Bus may carry identifying information to be interpreted by the intended receiving ECU, the CAN itself does not identify particular data frames, instead relying on a prioritization scheme of message arbitration to control access to the CAN bus [27].

Potential security drawbacks to such a communication model include vulnerabilities posed by the lack of message authentication, such as spoofing and injection attacks. An injection attack makes it possible for an adversary to manipulate the functionality of an ECU [28]. Because the CAN Bus has no default protocol for encrypting data frames on the network, the risk of compromising sensitive data poses another security concern [26], [29]. Finally, the

[1]Our project sponsors involve international researchers and developers, as well as customers, who have provided feedback regarding the patterns.

CAN message arbitration system can make it vulnerable to denial of service (DoS) attacks through an attacked ECU, such as the tire pressure monitoring system (TPMS) [3].

Degree of Access

While numerous automotive threat surfaces have been identified, we broadly categorize them by the type of access an attacker uses: direct (i.e., physical), indirect, and remote [30].

Direct access targets include the CAN Bus, the OBD-2 port, and the media/auxiliary port. The OBD-2 port provides a mechanism for sub-system diagnostics and requires an attacker to be physically plugged in as shown in experimental attacks [3], [31], [4]. The media player and auxiliary ports in a vehicle are often connected to the CAN Bus; and by physically inserting compromised files into the new media device, an attacker can subsequently inject (malware) code onto the CAN Bus [3].

Indirect access to a vehicle refers to compromising a device that can be used as a medium to attack the vehicle's sub-systems. Bluetooth networks are commonplace in modern vehicles, and accessing the vehicle's internal Bluetooth network is feasible through a connected, compromised device [3], [4]. Similarly, if a diagnostic device has been compromised and then attached to the OBD-2 port of a vehicle, then the vehicle becomes vulnerable to a malicious attack [3], [4].

Finally, *remote attacks* have been demonstrated in several scenarios, where the attacker can be physically distant from the target. Through a vehicle's telematics system, such as General Motors' OnStar® or Hyundai's Blue Link®, "white hat hackers"[2] have shown the ability to attack using only an Internet connection [32]. With more vehicles communicating with one another (V2V), with infrastructure (V2I), and other devices (V2X), there is increasing risk for malicious actors to inject malware through vehicular ad hoc networks (VANETs) [33].

IV. Security Solutions for Automotive Systems

Automotive security needs differ from security solutions used for traditional computing-based systems due to the specific performance constraints and requirements imposed by a vehicle's architecture and communication infrastructure. Specifically, automotive systems face challenges with available communication protocols and the use of the CAN bus architecture, limited communication resources, and stringent performance requirements for safety-critical functionality [7], [34]. Despite these limitations, researchers have adapted existing security approaches to comply with automotive system's constraints, with a focus on authentication and encryption [35][36] [37] [38] [39].

[2]Hackers who look for security vulnerabilities for the sole purpose of hardening and improving the systems. https://us.norton.com

As mentioned previously, a common security issue for automotive systems is the inability to authenticate communications. Without the ability to determine what traffic is valid can expose a network to spoofing, DoS, and other types of attacks. This vulnerability can be addressed by using authentication protocols or intrusion detection systems (IDS). However, the traditional approaches for these solutions are prohibitively expensive for an automotive network (e.g., with respect to performance constraints). A method for developing a more lightweight IDS is to base identification of connected ECUs on clock-based fingerprinting with predictive algorithms to determine expected clock-skews of ECUs and subsequently detect spoofed messages [35]. Similar to the clock-based IDS, is the concept of a module-specific firewall. Rizvi et al. [36] propose a firewall-like program installed on individual ECUs to examine incoming packets in order to determine if the message should be released to the ECU for processing based on a set of rules and accepted subjects.

The VeCure framework [40] proposes another solution to address the lack of authentication in the CAN Bus protocol [40]. VeCure addresses message authentication by having ECUs transmit an authentication frame with the message being broadcast. At the same time, ECUs are segmented into trust groups where outward facing ECUs are given low trust. The high trust group ECUs are given a key to verify message senders as a protection mechanism from potentially spoofed outward facing nodes.

Vai et al. [37] examine a framework for both data encryption and message authentication in similarly constrained cyber physical systems. Using a co-processor that performs cryptographic primitives in hardware as an independent cryptographic module, using the least squared method (LSM), the researchers specify a communication bus security protocol where the cryptographic module behaves like an ordinary ECU attached to the bus. Each ECU on the communications channel has a digitally-signed configuration file containing rules for communication with other ECUs, as well as at startup. The LSM validates the configuration files and establishes secure communication channels for the ECUs using the session keys.

A similar use of cryptography can be applied to VANETs as a method of authentication and privacy preservation [38]. The LESPP system uses lightweight symmetric encryption and message authentication code (MAC) generation for message signing. It also contains a MAC regeneration for verification in order to authenticate senders. Moreover, privacy is protected as only the system's key management center can expose a vehicle's real identity.

In order to prevent physical tampering to systems, the use of tamper resistant designs can prevent malicious actors from making unsafe changes to a system [39]. The researchers propose the use of system-on-chip design and non-detachable connections to prevent ECU hardware alteration, or at least detect and provide evidence of tampering.

V. Security Patterns for Automotive Systems

This section overviews the template used to describe the security patterns, as well as the security principles and automotive security development guidelines used to characterize the patterns. In addition to the template description, we overview the Viega and McGraw security principles [14], the STRIDE threat categories [13], and the SAE J3061 development guidelines for automotive security [10], all of which is incorporated into the pattern descriptions.

Automotive Security Pattern Template

The automotive security patterns are described in a template similar to that used by Konrad et al. [24].

- **Pattern Name** - The name and classification of a pattern as either behavioral or structural.
- **Intent** - How the pattern can be used and what security problem it addresses.
- **Motivation** - Background on the security problem, and a mechanism to solve, including examples and security principles that are related.
- **Properties** - STRIDE properties [13] that are addressed through this pattern.
- **Applicability** - Indicates the type of solution proposed: prevention, detection, and/or mitigation of a given security vulnerability.
- **Structure** - Includes the UML class diagram for the pattern and the related descriptions of participants and collaborations.
- **Behavior** - Includes a UML sequence diagram and descriptions of an illustrative scenario with the pattern.
- **Constraints** - Denotes constraints placed on the pattern implementation stemming from the limited resources and real-time constraints of the automotive system.
- **Consequences** - Discusses the effects of the pattern on the areas of Accountability, Confidentiality, Integrity, Availability, Performance, Cost, Manageability, and Usability following the template from Wasserman and Cheng [8].
- **Known Uses** - Discusses examples of this pattern's use in an automotive setting.
- **Related Patterns** - Any design patterns or security patterns that relate to the given pattern.

Security Properties Using STRIDE Threat Framework

STRIDE is a component of Microsoft's Security Development Lifecycle [13] framework used to categorize threats to a system. Along with threat categorization, STRIDE incorporates a set of information technology properties that correspond to the given threats. Table I outlines the STRIDE threats, the corresponding information technology properties, and security questions related to the properties.

Using the STRIDE framework in the pattern template assists developers in connecting the threats and their corresponding properties to patterns. Our industrial collaborators suggested the use of the STRIDE system in the pattern descriptions given their common use in industrial contexts. The STRIDE descriptors further facilitate the identification of appropriate patterns to use for a given problem/solution context.

Guiding Principles

Following a template similar to that used by Konrad et al. [8], [24], several security principles are used to help motivate the problems that the security patterns are intending to solve. The following security principles from Viega and McGraw [14] are incorporated into the template:

- **V1 - Secure the Weakest Link** - Securing the most vulnerable piece of an application(e.g. network access points or interfaces).
- **V2 - Practice Defense in Depth** - Promote several layers of security throughout an application, which will ensure redundant security if one layer fails.
- **V3 - Fail Securely** - In the case of a system failure, ensure that unauthorized access to the system or information is not possible.
- **V4 - Follow the Principle of Least Privilege** - Give actors in a system the lowest degree of access/clearance necessary to carry out a given task.
- **V5 - Compartmentalize** - Keep sub-systems separate from each other so that a security failure in one will not compromise the others.
- **V6 - Keep it Simple** - Using unnecessarily complicated procedures or systems may lead to unforeseen security vulnerabilities.
- **V7 - Promote Privacy** - Ensure personal information and sensitive data is only accessible by those with ownership or given consent.
- **V8 - Remember that Hiding Secrets is Hard** - Security critical information such as passwords are not easily kept safe and secret. Therefore, designers must be wary of compromised accounts/subsystems affecting the whole system.
- **V9 - Be Reluctant to Trust** - Sub-systems should not rely on other sub-systems being secure and must enforce security against all actors that interface with them.
- **V10 - Use Community Resources** - Community resources are often more secure than individual solutions due to the large number of developers.

We also used guiding principles for developing secure automotive systems defined in SAE Standard J3061 [10] to more closely relate the security patterns to the industry standards for automotive systems. Our industrial collaborators also indicated that applying J3061 would potentially facilitate more industrial collaborations between development organizations. While the principles share common principles/objectives with those from Viega and McGraw [14], J3061 captures the principles in the specific context of the automotive system. They are as follows [10]:

- **J1 - Protect Personally Identifiable Information and Sensitive Data** - Ensure personal information and sensitive data is only accessible by those with ownership or given consent.
- **J2 - Use Principle of Least Privilege** - Give actors in a system the lowest degree of access/clearance unless necessary.
- **J3 - Apply Defense in Depth** - Promote several layers of security throughout an application. This will ensure redundant security if one layer fails.
- **J4 - Prohibit Software Changes that have not been Thoroughly Analyzed and Tested** - Preventing software

Threat	Property	Security Question
Spoofing	Authentication	Does the system use multi-factor authentication? Does the system enforce secure credential creation, use and maintenance principles?
Tampering	Integrity	Can the system detect and prevent parameter manipulation? Does the system protect against tampering and reverse engineering? Were secure software design principles followed during development, including third-party software?
Repudiation	Non-Repudiation	Does the system verify and log all user actions with attribution?
Information Disclosure	Confidentiality	Does the system follow standard encryption practices to secure connections?
Denial of Service	Availability	Was the system built and tested for high availability (e.g., fuzz testing and load testing)?
Elevation of Privilege	Authorization	Does the system support the management of all users and privileges?

TABLE I: STRIDE Threats and Properties [13]

from being changed without being thoroughly tested prevents unforeseen security flaws in a system.

- **J5 - Prevent Vehicle Owners from Making Unauthorized Changes** - Along with J4, changes to a vehicle that are not planned thoroughly may miss critical security flaws leading to system vulnerabilities.

Current Repository of Automotive Security Patterns

The current repository of automotive security patterns has been drawn largely from automotive security solutions described in recent research literature, where we leverage the intent of previously-developed security patterns [6], [22]. These patterns have been developed with guidance from our industrial collaborators, who have developed additional patterns for internal use, as well as augmented their development process to include the use of these security patterns. Table II categorizes the automotive security patterns to date in the context of their applicability (i.e., prevention (Prev), detection (Det), and/or mitigation (Mit) of security vulnerabilities) and related security principles/guidelines as defined by Viega and McGraw (prefixed with 'V') and by the SAE J3061 automotive cybersecurity guidelines (prefixed by 'J'). Next, we include a brief description for each of the patterns in the repository, including citations to known realizations in the automotive domain.

- Authorization provides a structure that facilitates access control to resources [37].
- A Blacklist pattern intends to keep track of the traffic of potentially malicious addresses in a network. Nodes in the network use the *Blacklist* to block traffic originating from the malicious nodes [41], [42].
- The DDoS Redundancy pattern is intended to make a resource or network more resilient to a Distributed Denial of Service attack (DDoS) by providing redundant resources in case a resource becomes inundated with service requests [43], [44].
- Pattern Firewall designs a structure that allows for network traffic to be filtered by a set of predefined rules to prevent malicious intrusion (e.g., securing ECUs individually [36]).
- The Multi-Factor Authentication pattern is used to provide a redundant security measure in authenticating an actor or a message. By enforcing an additional level of authentication, malicious actors can be prevented from attacking if a node or single credential is compromised [45].
- Multilevel Security is intended to provide a mechanism for handling access in a system with various security classification levels [40].
- Signature IDS provides a mechanism for detecting anomalies in network traffic by using a base-line characteristic of the (communication) traffic [35], [46].
- Symmetric Encryption is used to encrypt messages (between vehicles and with infrastructure) so that only a sender and receiver can read the contents [38].
- A Tamper Resistance-based module deters unauthorized changing of a system by preventing alterations, or preserving evidence of alteration [39].
- The Third Party Validation pattern is intended to provide validation of messages broadcasted in a given network. If a spoofed message is sent to a node and the receiving node cannot validate the message with a trusted node somewhere else in the network, then the receiving node can disregard the message [44].

VI. Sample Security Patterns

Due to space constraints, this section provides only two sample patterns that illustrate the use of the automotive security patterns template. One pattern is applicable to securing ECUs involving inward facing communication using a signature-based IDS, and another pattern makes use of blacklists to ensure secure outward facing communication. Underlined Helvetica refers to pattern names, inline italics refers to UML diagram elements (participants), and courier font refers to security principles/guidelines.

A. Signature Based IDS

The Signature based IDS is a structural based security pattern.

1) Intent: The pattern provides a mechanism for detecting anomalies in network traffic by using a base-line characteristic of the traffic.

2) Motivation: On an open network, there is a need for detecting malicious traffic as soon as it occurs to prevent damage to a system [6]. In an automotive network such as that used with the CAN, there is no mechanism for detecting malicious traffic in the protocol, such as a spoofed ECU. By detecting deviations from a baseline behavior, such malicious activity can be detected. SAE J3061 emphasizes the need for comprehensive responses to cybersecurity incidents [10]. A component of this response

TABLE II: Applicability and Viega McGraw [37]/J3061 Principles of Select Patterns [10]

Pattern Name	Applicability	V1	V2, J3	V3	V4, J2	V5	V6	V7, J1	V8	V9	V10	J4	J5
Authorization	Prev				X	X		X					
Blacklist	Mit/Prev		X			X				X			
DDoS Redundancy	Prev/Mit		X	X		X							
Firewall	Prev/Det	X			X					X			
Multi-Factor Authentication	Prev		X			X				X			
Multilevel Security	Prev/Mit				X	X		X	X	X			
Signature IDS	Prev/Mit/Det									X			
Symmetric Encryption	Prev							X		X			
Tamper Resistance	Prev/Det/Mit			X	X							X	X
Third Party Validation	Det/Mit							X		X			

is the ability to detect intrusions and respond quickly. The pattern supports the principle of `Reluctance to Trust` [14] as authenticity is always verified before a message is allowed to pass.

3) Properties: The Signature based IDS can be used to satisfy the Authorization property, and the Integrity property.

4) Applicability: The Signature based IDS is applicable to attack Detection, attack Prevention, and attack Mitigation.

5) Structure: The Signature based IDS intercepts the access request for a service (Figure 1). The *Event Processor* processes the request to parse the relevant *Signature Information*. The *Attack Detector* then tries to match the *Signature Information* against the known signature information to determine if the *Signature* is known. If the *Signature* is unknown, then the appropriate *Response* is issued [6].

6) Participants:

- *IDS:* Intercepts messages and performs proper action given a *Response*
- *Event Processor:* Obtains *Signature* information from message
- *Attack Detector:* Checks *Signature* against known signatures
- *Signature Information:* Has ID of known actor and the corresponding *Signature*
- *Signature:* Identifying characteristic of actor
- *Response:* Communicates signature results back to *IDS* for message handling

7) Collaborations: The *IDS* has a one-to-one relationship with the *Event Processor* and forwards requests to be processed. The *Event Processor* collaborates in a one-to-one relationship with the *Attack Detector* that obtains processed data from the *Event Processor*. The *Attack Detector* includes the *Signature Information* of known entities. The *Attack Detector* has a one-to-one relationship with the *Response* object that formulates the response given the results of the *Attack Detector*. Finally the *IDS* has a one-to-one relationship with the *Response* object that forwards the correct response to the IDS.

8) Behavior: A message is sent from a source to an intended destination node (Figure 2). The Signature based IDS intercepts the message and determines if the *Signature* is known to the system. If the sender is unknown, then an appropriate *Response* is raised [6].

9) Constraints: As an IDS analyzes traffic before the messages are sent to the intended receiver, there is overhead incurred when using the pattern. The solution provided by Cho and Shin [35] utilizes a predictive algorithm implemented in hardware to decrease the processing time in order to minimize the overhead.

10) Consequences: Table III describes the consequences of using the Signature based IDS pattern.

11) Known Uses: An example of a lightweight IDS for CAN bus was proposed by Cho and Shin [35]. Here, the system used clock-based finger-printing and predictive algorithms to determine expected clock-skews of ECUs, thus enabling the detection of spoofed messages and authentic ones.

12) Related Security Patterns: The pattern is a specialization of the Abstract IDS pattern and is related to the *Firewall* pattern, another access control pattern for networks [6].

13) Supported Principles: Principals supported by the Signature based IDS include Principle 9 (`Be Reluctant to Trust`), as the IDS intercepts messages to verify the identity of the sender before allowing it to access protected resources, and Principle 3 (`Fail Securely`).

B. Blacklist

Blacklist is a structural based security pattern.

1) Intent: A Blacklist intends to monitor the traffic of potentially malicious addresses in a network. Nodes in the network use the Blacklist to block traffic originating from the malicious nodes.

2) Motivation: Communication with external entities is becoming increasingly integrated into modern automotive systems. Today's cars communicate with smart infrastructure, receive over the air updates from manufacturers, and exchange information with other vehicles on the road.

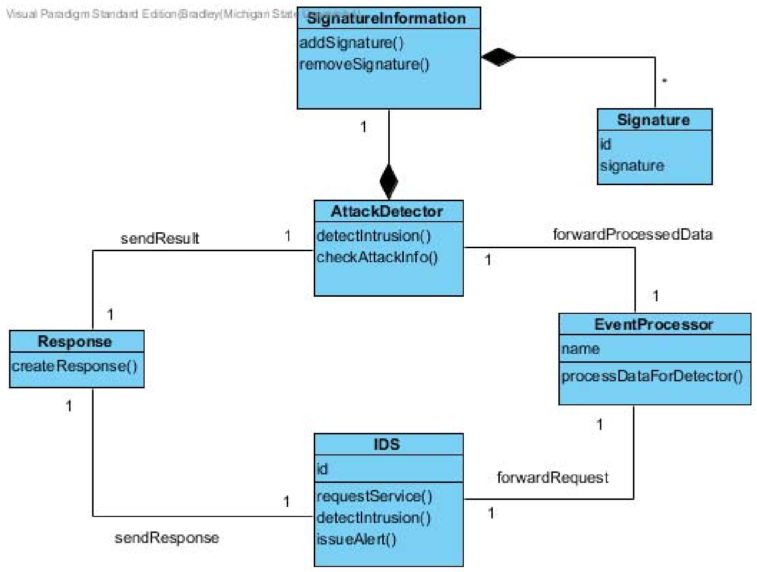

Fig. 1: Class Diagram for Signature IDS Pattern

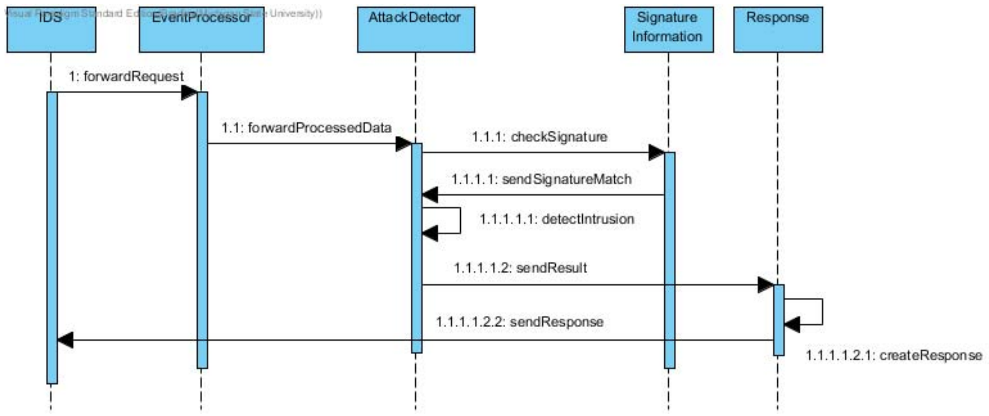
IDS
EventProcessor
AttackDetector
Signature Information
Response
1: forwardRequest
1.1: forwardProcessedData
1.1.1: checkSignature
1.1.1.1: sendSignatureMatch
1.1.1.1.1: detectIntrusion
1.1.1.1.2: sendResult
1.1.1.1.2.2: sendResponse
1.1.1.1.2.1: createResponse

Fig. 2: Sequence Diagram for Signature IDS Pattern

While these technologies allow for improved performance and user experience, the increased reliance on networks leaves a vehicle susceptible to attack from a malicious user on the network. The inability to identify compromised nodes poses a security risk to an automotive system. VANETs allow vehicles and smart infrastructure to share information about current driving conditions within a specific broadcast range. Due to the routing protocols upon which VANETs rely, a malicious node can join a VANET and proceed to attack the network by intercepting the communication between vehicles, dropping packets, and changing or fabricating packets [47]. Applications such as the Post Crash Notification [48] can be deployed on VANETs to detect an attack such as injections of false messages that can lead to potentially dangerous responses by the vehicles on the network.

3) Properties: The Blacklist can be used to satisfy the Authentication property, the Authorization property, and the Non-Repudiation property.

4) Applicability: A Blacklist is applicable to the Prevention and Mitigation of an attack.

5) Structure: The Blacklist structure of a system can be captured in terms of their relationships (Figure 3). An entity sending a message is captured by a *Client* object. The entity secured with a Blacklist is represented as a *Service* object. The interface between the two objects is a *Checkpoint* object. And finally, the Blacklist captured with the *Blacklist* object.

6) Participants: The following section describes the participating classes in the pattern structure.

- *Client:* An actor requesting access to a node, or *Service* object.
- *Service:* The intended protected object. A *Service* has access to a system's message processing.

TABLE III: Consequences of Signature Based IDS Pattern

Accountability:	The *Signature based IDS* allows a system to be accountable for the authenticity of the traffic sent on it's network.
Confidentiality:	Not addressed.
Integrity:	By preventing unrecognized agents from communicating on a network, the *Signature based IDS* protects the network from misuse, and the agents attached to the network from attack
Availability:	The *Signature based IDS* may prevent availability on a network as overhead on authentication may lead to under use.
Performance:	The system may take a performance hit in validating the message signatures as it may require significant overhead.
Cost:	Additional hardware to process signatures may incur a cost.
Manageability:	Allows a system to more easily manage the actors that may utilize a network.
Usability:	The usability may decrease as only a small subset of actors may be known to a network.

- *Blacklist:* The system's list object for tracking bad addresses.
- *Checkpoint:* The interface through which all communication with a *Service* is achieved, and where the *Blacklist* is checked.

7) Collaborations: A given *Client* will attempt to communicate with a *Service* through a *Checkpoint*. Consequently, there is an association between both the *Service* and the *Checkpoint* and the *Client* and the *Checkpoint* that acts as the interface. There is also an association between a *Checkpoint* and the associated *Blacklist*.

8) Behavior: A Blacklist (Figure 4), is a solution that can partially fulfill the relevant security requirements that are unmet in the problem statement. A Blacklist maintains a list of addresses within a network that have exhibited inappropriate behavior. When a packet from a blacklisted address arrives at a node, the node simply drops the packet. In the context of a VANET, nodes will not process or forward any messages that originate from a blacklisted address, and routing algorithms will not include the blacklisted nodes in calculating routes of the packets.

9) Constraints: Real-time constraints are important to consider in the context of the Blacklist pattern as message processing incurs overhead as well as consumes additional resources per message.

10) Consequences: See Table IV.

11) Known Uses: An example of a Blacklist being deployed in an automotive setting is given by Daeinabi and Rahbar [41]. Here the authors propose a system in which a select number of nodes in a VANET are tasked with monitoring behavior of the other nodes, where the overall network can dynamically change over time. If one of the monitoring nodes detects abnormal traffic from a given node, then it increases a distrust value for that node. Meanwhile, all nodes in the network maintain a Blacklist. If a node is given a distrust value beyond a certain threshold by the monitoring nodes, then that node is reported and blacklisted for the other nodes in the network.

12) Relevant Security Principles: The Blacklist leverages `Practice Defense in Depth`, `Reluctance to Trust`, and `Compartmentalize`.

13) Related Patterns: The Blacklist is related to traffic filtering patterns such as the Firewall pattern, and the Signature-Based IDS.

VII. Conclusion

As software and external communication become further integrated into modern automotive systems, it is critical that security is incorporated explicitly into the design and development process. By preventing, detecting, and mitigating attacks, automotive systems can continue to provide a better customer experience while also ensuring safety. With the use of automotive-focused security patterns, automotive software developers can incorporate known solutions to security problems into their systems more easily.

In this paper, we discussed security pattern concepts in relation to automotive systems and described the initial set of patterns in our repository. Moving forward, we are working with our industrial collaborators to expand the security pattern repository. We are also exploring how the security patterns can be incorporated into a security and safety-focused development processes, similar to that proposed by Amorim et al. [49]. Finally, with the upcoming release of the ISO/SAE 21434 Automotive Cybersecurity standard due for release in 2020 [50], [50], [51], we will update the security patterns to incorporate the appropriate terminology modifications, process development guidelines, risk assessment strategies, and the interactions with safety standards defined in ISO26262 [52].

Acknowledgments

This work has been supported in part by grants from NSF (CNS-1305358 and DBI-0939454), Ford Motor Company, General Motors Research, and ZF; and the research has also been sponsored by Air Force Research Laboratory (AFRL) under agreement numbers FA8750-16-2-0284 and FA8750-19-2-0002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL), the U.S. Government, National Science Foundation, Ford, GM, ZF, or other research sponsors. We are grateful to B.

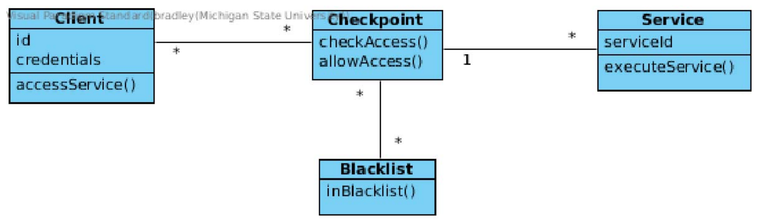

Fig. 3: Class Diagram for Blacklist Pattern

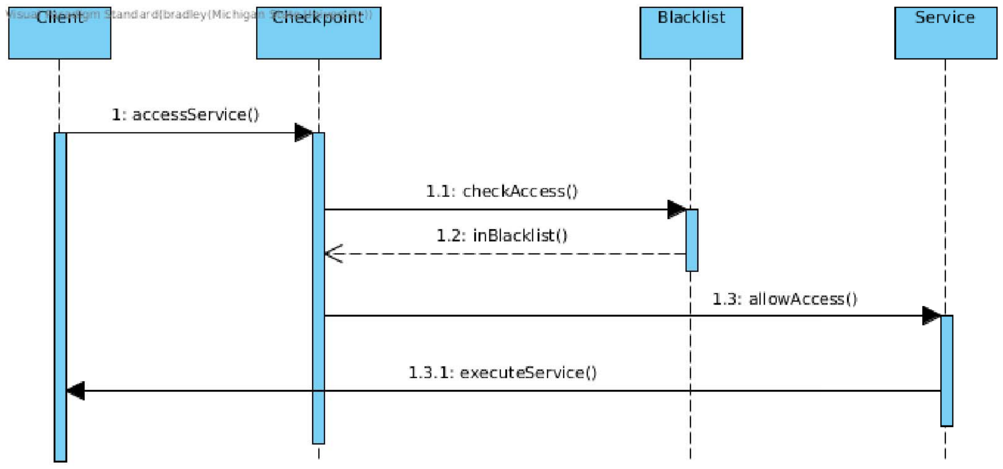

Fig. 4: Sequence Diagram for Blacklist Pattern

TABLE IV: Consequences of Blacklist Pattern

Accountability:	Preventing misbehaving nodes from participating in a network improves the ability of the network to hold malicious actors accountable.
Confidentiality:	Preventing nodes that are known to misbehave from accessing data sent between nodes in the network provides a higher degree of data confidentiality.
Integrity:	Preventing nodes that are known to misbehave from receiving and possibly modifying data sent between nodes in the network provides a higher degree of data integrity.
Availability:	Depending on the blacklisting protocol the Blacklist may prevent harmless nodes from participating in the network, thereby reducing availability.
Performance:	Performance may be affected by the overhead incurred by using the network Blacklist pattern. Performance improvements may occur however as the resources consumed by misuse are reduced .
Cost:	Not applicable.
Manageability:	By setting blacklist rules, manageability of a network can be improved.
Usability:	Depending on the blacklisting protocol, legitimate users may be prevented from accessing the services.

Czerny, I. Hawkes, B. Murray, and other industrial collaborators for their valuable feedback on this work.

References

[1] A. Weimerskirch, “Automotive and industrial data security,” in *Cybersecurity and Cyber-physical Systems Workshop*, 2012.

[2] C. Miller and C. Valasek, “Remote exploitation of an unaltered passenger vehicle,” *Black Hat USA*, vol. 2015, 2015.

[3] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, *et al.*, “Comprehensive experimental analyses of automotive attack surfaces.,” in *Proceedings of the 20th USENIX Conference on Security*, SEC’11, 2011.

[4] S. Parkinson, P. Ward, K. Wilson, and J. Miller, “Cyber threats facing autonomous and connected vehicles: future challenges,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 2898–2915, 2017.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, “Design patterns: Abstraction and reuse of object-oriented design,” in *European Conference on Object-Oriented Programming*, pp. 406–431, Springer, 1993.

[6] E. Fernandez-Buglioni, *Security patterns in practice: designing secure architectures using software patterns*. John Wiley & Sons, 2013.

[7] C.-W. Lin, B. Zheng, Q. Zhu, and A. Sangiovanni-Vincentelli, “Security-aware design methodology and optimization for automotive systems,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 21, no. 1, p. 18, 2015.

[8] R. Wassermann and B. H. C.. Cheng, “Security patterns,” tech. rep., Department of Computer Science and Engineering, Michigan State University, East Lansing, Michigan 48824, 2003. (A shortened version appeared in the Proceedings of Requirements Engineering “Using Security Patterns to Model and Analyze Security Requirements” (with S. Konrad, L. Campbell, and R. Wassermann), IEEE Workshop on Requirements for High Assurance Systems,

(RHAS03), September 2003, Monterey, California.).

[9] E. B. Fernandez, N. Yoshioka, and H. Washizaki, “Patterns for security and privacy in cloud ecosystems,” in *Evolving Security and Privacy Requirements Engineering (ESPRE), 2015 IEEE 2nd Workshop on*, pp. 13–18, IEEE, 2015.

[10] V. C. S. E. Committee, “J3061 cybersecurity guidebook for cyber-physical vehicle systems,” tech. rep., SAE International, 2016.

[11] G. Booch, J. Rumbaugh, and I. Jacobson, *Unified Modeling Language User Guide, The (2Nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional, 2005.

[12] National Highway Traffic Safety Administration and others, “Cybersecurity best practices for modern vehicles.” Report No. DOT HS, 2016.

[13] Microsoft Corporation, “Security development life cycle,” jun 2018.

[14] J. Viega and G. McGraw, *Building Secure Software: How to avoid security problems the right way*. Addison-Wesley, 2002.

[15] E. Gamma, *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[16] Y. Ito, H. Washizaki, M. Yoshizawa, Y. Fukazawa, T. Okubo, H. Kaiya, A. Hazeyama, N. Yoshioka, and E. B. Fernandez, “Systematic mapping of security patterns research,” in *Proceedings of the 22nd Conference on Pattern Languages of Programs*, p. 14, The Hillside Group, 2015.

[17] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating security and systems engineering*. John Wiley & Sons, 2013.

[18] D. M. Kienzle, M. C. Elder, D. Tyree, and J. Edwards-Hewitt, “Security patterns repository version 1.0,” *DARPA, Washington DC*, 2002.

[19] N. Yoshioka, H. Washizaki, and K. Maruyama, “A survey on security patterns,” *Progress in informatics*, vol. 5, no. 5, pp. 35–47, 2008.

[20] E. B. Fernandez, H. Washizaki, and N. Yoshioka, “Abstract security patterns,” in *Proceedings of the 15th Conference on Pattern Languages of Programs*, p. 4, ACM, 2008.

[21] E. B. Fernandez, N. Yoshioka, H. Washizaki, and J. W. Yoder, “Abstract security patterns for requirements specification and analysis of secure systems.,” in *WER*, 2014.

[22] A. V. Uzunov, E. B. Fernandez, and K. Falkner, “Securing distributed systems using patterns: A survey,” *Computers & Security*, vol. 31, no. 5, pp. 681–703, 2012.

[23] C. Dougherty, K. Sayre, R. Seacord, D. Svoboda, and K. Togashi, “Secure design patterns,” Tech. Rep. CMU/SEI-2009-TR-010, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2009.

[24] S. Konrad, B. H.C.. Cheng, L. A. Campbell, and R. Wassermann, “Using security patterns to model and analyze security requirements,” *Requirements Engineering for High Assurance Systems (RHAS’03)*, vol. 11, 2003.

[25] S. Corrigan, “Introduction to the controller area network - texas instruments,” Tech. Rep. SLOA101, Texas Instruments, 2016.

[26] O. Avatefipour and H. Malik, “State-of-the-art survey on in-vehicle network communication “can-bus” security and vulnerabilities,” *International Journal of Computer Science and Network*, vol. 6, pp. 720–727, 2017.

[27] J. Kaiser and M. Mock, “Implementing the real-time publisher/subscriber model on the controller area network (can),” in *Object-Oriented Real-Time Distributed Computing, 1999.(ISORC’99) Proceedings. 2nd IEEE International Symposium on*, pp. 172–181, IEEE, 1999.

[28] D. K. Nilsson, U. E. Larson, F. Picasso, and E. Jonsson, “A first simulation of attacks in the automotive network communications protocol flexray,” in *Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems CISIS’08*, pp. 84–91, Springer, 2009.

[29] A. Van Herrewege, D. Singelee, and I. Verbauwhede, “Canauth-a simple, backward compatible broadcast authentication protocol for can bus,” in *ECRYPT Workshop on Lightweight Cryptography*, vol. 2011, 2011.

[30] S. Mitra and B. H.C.. Cheng, “Survey on cybersecurity for automotive systems,” tech. rep., Computer Science and Engineering, Michigan State University, East Lansing, MI, 2017.

[31] C. Miller and C. Valasek, “Adventures in automotive networks and control units,” in *DEF CON 21 Hacking Conference. Las Vegas, NV: DEF CON*, 2013.

[32] J. Pagliery, “Chryslers can be hacked over the internet,” Jul 2015.

[33] R. Brooks, S. Sander, J. Deng, and J. Taiber, “Automobile security concerns,” *Vehicular Technology Magazine, IEEE*, vol. 4, no. 2, pp. 52–64, 2009.

[34] F. Koushanfar, A.-R. Sadeghi, and H. Seudie, “Eda for secure and dependable cybercars: challenges and opportunities,” in *Proceedings of the 49th Annual Design Automation Conference*, pp. 220–228, ACM, 2012.

[35] K.-T. Cho and K. G. Shin, “Fingerprinting electronic control units for vehicle intrusion detection.,” in *USENIX Security Symposium*, pp. 911–927, 2016.

[36] S. Rizvi, J. Willet, D. Perino, S. Marasco, and C. Condo, “A threat to vehicular cyber security and the urgency for correction,” *Procedia Computer Science*, vol. 114, pp. 100–105, 2017.

[37] M. M. Vai, R. I. Khazan, D. M. Utin, S. R. O’Melia, D. J. Whelihan, and B. R. Nahill, “Secure embedded systems,” tech. rep., MIT Lincoln Laboratory Lexington United States, 2016.

[38] M. Wang, D. Liu, L. Zhu, Y. Xu, and F. Wang, “Lespp: lightweight and efficient strong privacy preserving authentication scheme for secure VANET communication,” *Computing*, vol. 98, no. 7, pp. 685–708, 2016.

[39] M. Wolf and T. Gendrullis, “Design, implementation, and evaluation of a vehicular hardware security module,” in *International Conference on Information Security and Cryptology*, pp. 302–318, Springer, 2011.

[40] Q. Wang and S. Sawhney, “Vecure: A practical security framework to protect the can bus of vehicles,” in *Internet of Things (IOT), 2014 International Conference on the*, pp. 13–18, IEEE, 2014.

[41] A. Daeinabi and A. G. Rahbar, “Detection of malicious vehicles (dmv) through monitoring in vehicular ad-hoc networks,” *Multimedia tools and applications*, vol. 66, no. 2, pp. 325–338, 2013.

[42] T. Zhang, H. Antunes, and S. Aggarwal, “Defending connected vehicles against malware: Challenges and a solution framework,” *IEEE Internet of Things Journal*, vol. 1, pp. 10–21, Feb 2014.

[43] A. Rawat, S. Sharma, and R. Sushil, “VANET: Security attacks and its possible solutions,” *Journal of Information and Operations Management*, vol. 3, no. 1, p. 301, 2012.

[44] G. Samara and Y. Al-Raba’nah, “Security issues in vehicular ad hoc networks (VANET): a survey,” *arXiv preprint arXiv:1712.04263*, 2017.

[45] G. Yan, D. Wen, S. Olariu, and M. C. Weigle, “Security challenges in vehicular cloud computing,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 284–294, 2013.

[46] U. E. Larson, D. K. Nilsson, and E. Jonsson, “An approach to specification-based attack detection for in-vehicle networks,” in *2008 IEEE Intelligent Vehicles Symposium*, pp. 220–225, June 2008.

[47] F. Picconi, N. Ravi, M. Gruteser, and L. Iftode, “Probabilistic validation of aggregated data in vehicular ad-hoc networks,” in *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, pp. 76–85, ACM, 2006.

[48] M. Ghosh, A. Varghese, A. A. Kherani, and A. Gupta, “Distributed misbehavior detection in VANETs,” in *Wireless Communications and Networking Conference, 2009. WCNC 2009.*, pp. 1–6, IEEE, 2009.

[49] T. Amorim, H. Martin, Z. Ma, C. Schmittner, D. Schneider, G. Macher, B. Winkler, M. Krammer, and C. Kreiner, “Systematic pattern approach for safety and security co-engineering in the automotive domain,” in *Computer Safety, Reliability, and Security*, pp. 329–342, 08 2017.

[50] C. Schmittner, G. Griessnig, and Z. Ma, “Status of the development of iso/sae 21434,” in *25th European Conference, EuroSPI 2018, Bilbao, Spain, September 5-7, 2018, Proceedings*, pp. 504–513, EuroSPI, 01 2018.

[51] H. Hunjan, “ISO/SAE 21434 automotive cybersecurity engineering.”

[52] “Road vehicles — functional safety — part 2: Management of functional safety.” ISO 26262-2:2018(E).