IOActive®

Research-fueled Security Services

\ WHITE PAPER \

Commonalities in Vehicle Vulnerabilities

2018 Remix

Josh Hammond
Jerel Culliss

September 2018

Contents

With the connected car becoming commonplace in the market, vehicle cybersecurity continues to grow more important every year. At the forefront of security research, IOActive has amassed real-world vulnerability data illustrating the general issues and potential solutions to the cybersecurity threats today's vehicles face.

This paper provides an overview of how we approach threat modelling and test methodologies at IOActive. Detailed findings follow, including the impact, likelihood, overall risk, and remediation of vulnerabilities IOActive consultants have discovered over the course of thousands of testing hours. This analysis is a follow-up to our paper on vehicle vulnerabilities published in 2016. The goal of this paper is to deliver current data and discuss how the state of vehicle security has progressed.

Introduction

This paper is a follow-up to IOActive's 2016 report[1] on vehicle vulnerabilities. The goal of this paper is to revisit the topic using data from the past two years (2016, 2017) and to compare this information to previous findings to analyze how the industry is progressing.

Vehicle cybersecurity is a focused, growing area of security research for IOActive. In 2016 and 2017, IOActive consultants performed over 6,000 hours of work on vehicle hardware systems. This does not include time spent on backend systems, mobile applications, and web interfaces, all of which are becoming much more common in this industry.

IOActive's experience puts us in a unique position to provide valuable insight into common struggles, failures, and solutions in the automotive and related transportation and component industries. Our research uses hard data taken from vulnerability assessments of real-world vehicle systems. We have conducted enough of these assessments to properly anonymize the sources of this information and extract valuable "big-picture" data.

We begin with a discussion of threat modeling, attack vectors, and attack methodologies to explain how we discovered these vulnerabilities. We then cover categorization and walk through a vulnerability evaluation. Finally, we look at the data and discuss common vulnerabilities and trends.

[1] https://ioactive.com/securing-the-connected-car-commonalities-in-vehicle-vulnerabilities/

Threat Modeling the Connected Car

Understanding the attack surfaces of the connected car is an important first step. This means noting possible ways to attack a target, which might be the entire vehicle or just a component therein. A threat model does not focus on attack *methods*, but rather looks at possible attack *vectors*.

The modern connected car tends to have a wide variety of interfaces that allow for user interaction. These commonly include Bluetooth, cellular, WiFi, and USB as well as interfaces that vary based on manufacturer. Our focus at IOActive is on practical threats; our analysis starts with exposed interfaces and attempts to identify real, exploitable vulnerabilities. Rather than digging through an entire codebase looking for every possible overflow, we focus on where an attacker could get data into the system and use that to look for the most likely attacks.

Categorizing Vulnerabilities

To provide meaningful quantitative analysis of its findings, IOActive uses a likelihood combined with impact approach to scoring. For each individual finding, the assessment team assigns two ratings: one for impact and another for likelihood; that is, the likelihood the given vulnerability will be exploited. Each vulnerability is then assigned a rating of Critical, High, Medium, Low, or Informational—corresponding numeric scores range from 5 (Critical) to 1 (Informational). Table 1 explains each rating in terms of score, impact, and likelihood.

Table 1. Rating and score as applied to impact and likelihood

Rating and Score	Impact	Likelihood
Critical (5)	Complete compromise of component or potential safety concerns if exploited.	Vulnerability can be exploited remotely. Vulnerability is easily discovered or already has publicly available information.
High (4)	May allow for partial control of component, disclose sensitive personal information, or disable functionality if exploited.	Vulnerability can be exploited from nearby. Vulnerability can be exploited with limited skills and information.
Medium (3)	May disclose technical details, compromise telematics communications, or disrupt user experience if exploited.	Vulnerability can be exploited with limited physical access. Vulnerability can be exploited by a skilled attacker.
Low (2)	Compromise is not sensitive or damaging on its own but could be useful in exploiting other vulnerabilities.	Vulnerability can be exploited with extensive physical access. Vulnerability can be exploited by an attacker with limited insider knowledge or significant skills and experience.
Informational (1)	Poor programming practice or design decision that may not represent an immediate risk on its own.	Vulnerability can be exploited with unreasonable time, effort, or resources. Vulnerability can be exploited with sensitive insider information.

IOActive assigns aggregate risk scores to identified vulnerabilities; specifically, the impact score multiplied by the likelihood score. For example, a vulnerability with high likelihood and low impact would have an aggregate risk score of eight (8); that is, four (4) for high likelihood multiplied by two (2) for low impact. The Aggregate Risk Score determines the finding's Overall Risk Level, as shown in Table 2.

Table 2. Overall risk levels and corresponding aggregate scores

Overall Risk Level	Aggregate Risk Score (Impact multiplied by Likelihood)
Critical	**20–25**
High	**12–19**
Medium	**6–11**
Low	**2–5**
Informational	**1**

Example Categorization

The following hypothetical example is based on real vulnerabilities IOActive has discovered. We offer this step-by-step description as a way to get a better feeling for our rating process. In this example we'll be looking at a head unit that allows media files to be loaded and played via USB.

As a standard part of testing, we would fuzz the interface with different media types, looking for errors in the media parsers. Let's say that we found an issue parsing an mp3 file that resulted in memory corruption and code execution as the media player user.

Impact

This finding would not directly allow for control of the vehicle or complete control of the unit. The media user has limited access and doesn't directly access any critical systems. This could be detrimental to the user experience and could be used as a foothold for further attacks. This vulnerability would be rated medium (3) impact.

Likelihood

This vulnerability requires physical access to the device but doesn't require access for an extended period of time. An attacker would need some skills and expertise to find this vulnerability and develop an exploit. This vulnerability would be rated medium (3) likelihood.

Overall Risk

With an impact of 3 (medium) and a likelihood of 3 (medium), the overall risk rating would be medium (9). This vulnerability does not represent a severe risk on its own but a skilled attacker

could develop an exploit that would allow someone with limited access to gain a foothold on the device.

Remediation

Remediation of this issue would vary based on the root cause. If this vulnerability was found in a media library that was out-of-date, the recommendation may be to update the media library. If this vulnerability was in an open source library, it may be feasible to patch the codebase. The specific details of our recommendations can vary based on how much context and insight we have into the system.

Common Vulnerabilities and Analysis

Given how IOActive finds and classifies vulnerabilities, we can now look at the raw data. We present the data organized by:

- Impact
- Likelihood
- Overall Risk
- Attack Vector
- Vulnerability Type
- Remediation Difficulty

Impact

Looking at the data for 2016 and 2017, most vulnerabilities tend to be medium impact. These types of vulnerabilities would result in things like personal information disclosure or compromised network connections, but generally would not result in persistent, privileged access to the system.

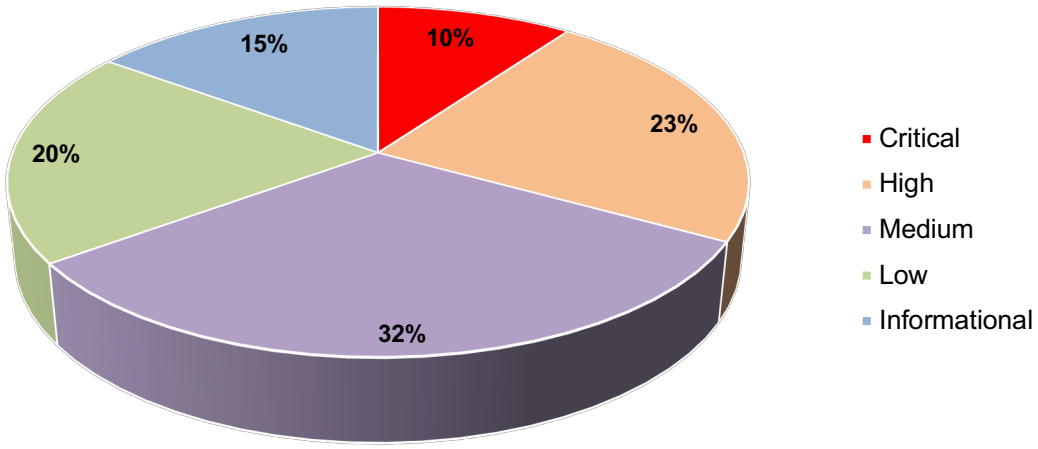

Figure 1. Impact Ratings 2016-17

This represents a significant drop in the proportion of critical-impact vulnerabilities from our previous report. Critical-impact vulnerabilities have decreased 15 percentage points, while the distribution of medium- and low-impact vulnerabilities has increased. This is likely the result of better security awareness and user separation. We've seen significant growth in the design of vehicle systems to incorporate security from the start. This includes making sure that the processes that handle data are running with limited privileges, which helps lower the impact of the most likely attacks.

Likelihood

In our data, most vulnerabilities fell into medium- and low-likelihood categories. This means that most vulnerabilities could either only be exploited by advanced attackers or may require another compromise to be exploitable.

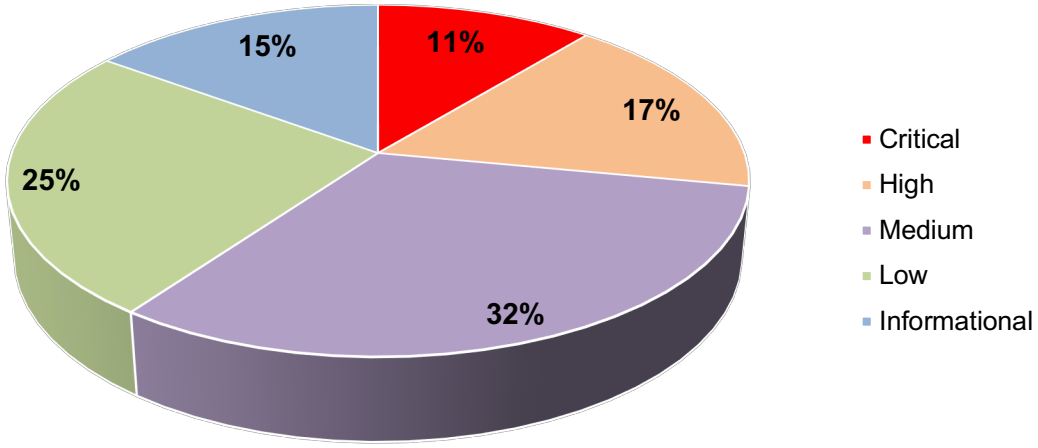

Figure 2. Likelihood Ratings 2016-17

Compared to the previous report, high-likelihood findings have shifted towards critical, and medium-likelihood vulnerabilities have skewed down towards low. This complex interaction is probably the result of a variety of factors driving the vulnerabilities in different directions. We've seen security architecture improve significantly but we've also seen an expansion in the number and scope of remote services that could be leveraged to attack the system.

Overall Risk

The overall risk ratings for our findings were mostly medium and low; however, a significant number of medium- and low-risk vulnerabilities does not necessarily mean that there is not significant risk. Individually, these vulnerabilities may not be severe but they may still be harmful when exploited together or could end up being harmful if another, presently unknown, vulnerability is discovered.

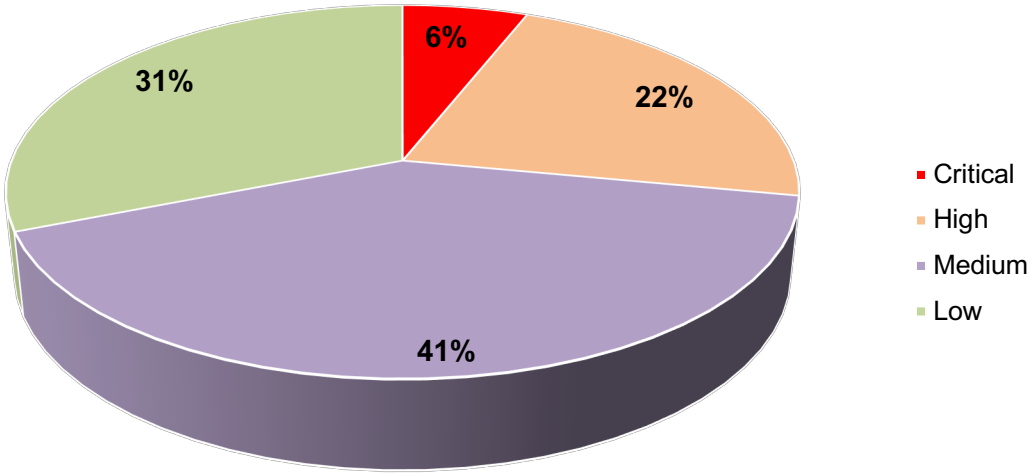

Figure 3. Overall Risk Ratings 2016-17

Compared to the previous report, risk ratings are tending to decrease. This is especially notable in the critical-risk category with a 16 percentage point decrease. This is indicative of the overall security improvements we've seen in the automotive industry.

This data does not include purely informational risk findings (both impact and likelihood are informational), as these are not considered to present a significant risk.

Attack Vectors

Attack vector categories are useful when evaluating how an attacker could approach a system. The most common attack vectors for the vulnerabilities IOActive discovered are local and network. Local attacks require that an attacker already has a foothold on the system. This generally lowers the likelihood of the attack but often represents an attacker's ability to elevate privileges or otherwise manipulate the system once they have gained access. The network attack vector tends to represent the highest exposure and is often a major focus area when testing.

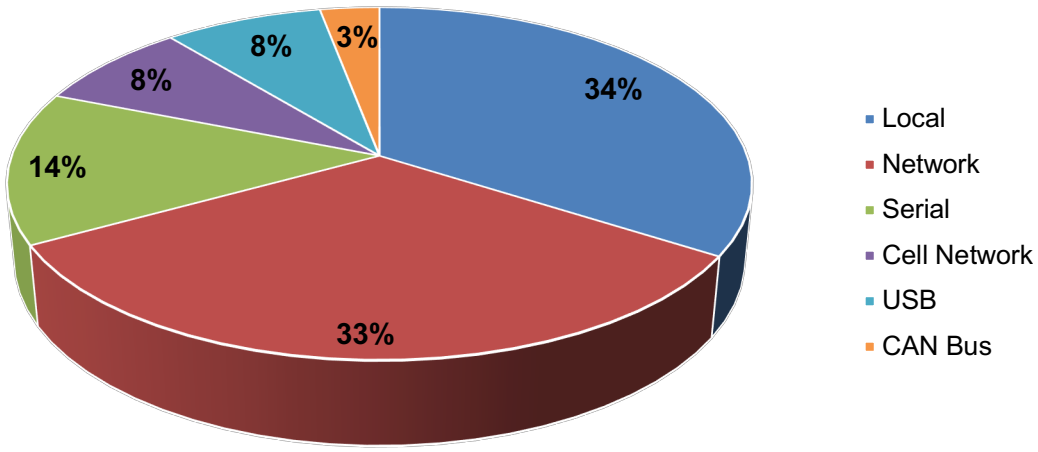

Figure 4. Attack Vectors

When compared to the previous report, there has been a notable rise in serial attacks. These attacks require physical access to the device and can include reading and modifying firmware, reading data between components, and taking advantage of debugging and test features left in the hardware.

The large increase in local and serial attacks can be attributed to a shift in testing approaches. As security has become a more prevalent concern, more companies are providing documentation and debugging access to help identify vulnerabilities inside their systems. The automotive industry is also taking more of an interest in lower-level security features, like secure boot, which is reflected in the areas we end up testing.

Vulnerability Types

As part of our analysis, we divide the data into different types of vulnerabilities. This helps us identify common issues and determine how these vulnerabilities come about in the first place. The most prevalent vulnerability type in our data set was coding logic errors. These errors come from bypassing the logic of the program rather than exploiting a technical flaw in how data is handled. For instance, if a firmware image header can result in copying too much data into a buffer, it's memory corruption. But if a directory traversal when unpacking the firmware puts files in a location that is not authenticated, it's a logic error.

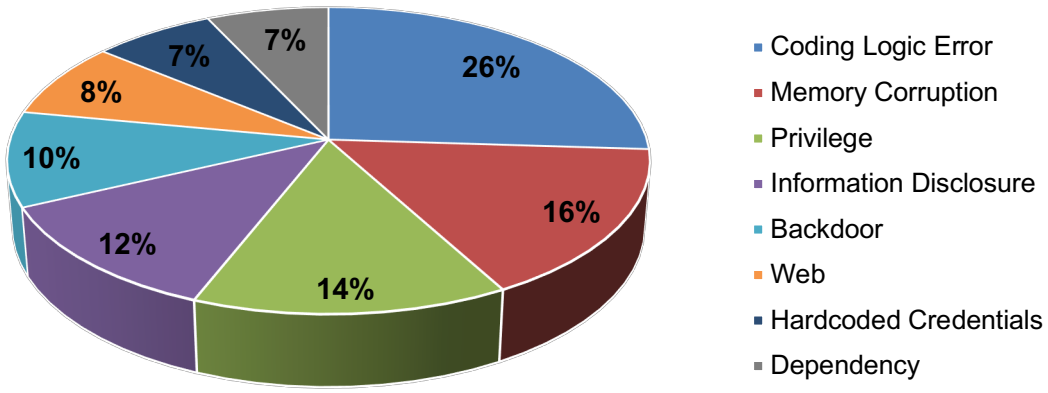

Figure 5. Vulnerability Type

The biggest change since our previous report is the increase in the percentage of coding logic errors. As security architecture and secure development practices improve, this area is expected to represent a larger portion of errors.

Critical Impact Remediation

Part of our standard rating includes an estimated level of effort to remediate. Low-effort fixes may involve patching a buffer overflow or enabling a feature in the device's configuration. High-effort changes may require substantial modifications to the design of the vehicle's communication systems. It's important to note that the level of effort is based on how difficult we at IOActive think it would be to develop and deploy a fix. This doesn't account for specific issues in the automotive industry, like the increased level of effort for pushing an update, especially for vehicles without remote firmware updates.

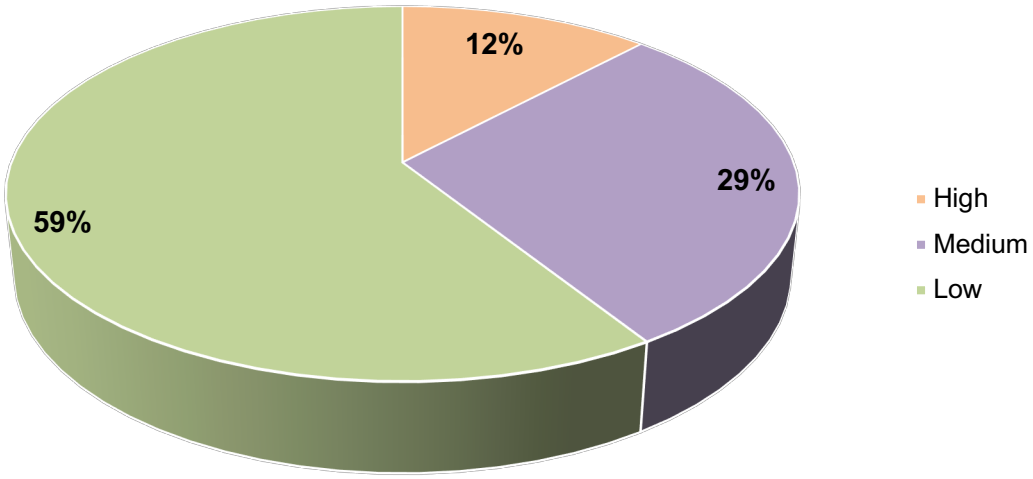

Figure 6. Effort to Fix

Compared to the previous report, the effort required to fix issues has gone up. Previously, low-effort fixes accounted for 77% of critical-impact vulnerabilities; now they only account for 59%. Most issues are still relatively easy to fix but the decrease is likely due to better practices resulting in fewer low-hanging bugs in the first place.

Ounce of Prevention

As part of every assessment, IOActive provides recommendations for each vulnerability on how best to fix or mitigate the issue. Part of our research was broadly categorizing those fixes to give a general sense of where these issues stem from and how they can be prevented.

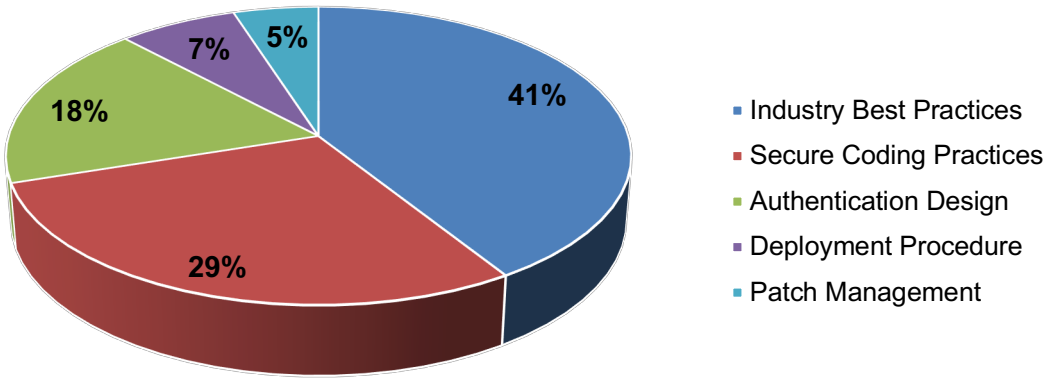

Figure 7. Remediation Categories

The largest category by far was industry best practices. These are issues that could be solved by following common guidance from groups such as the Auto-ISAC and OWASP. These tend to be issues like not authenticating data, not encrypting and authenticating network traffic, and not filtering user inputs.

The next largest category is secure coding practices, such as using insecure functions and not checking return values. These can mostly be fixed with strong implementation guidelines and enforcing banned functions.

Authentication design may be the most difficult category to fix. These are issues that come from the design of the system, where strong controls are lacking in the system architecture. Fixing these may involve significant changes in how services communicate and how the system is accessed.

Deployment procedure issues seem to be less common and mostly involve not disabling debugging features before releasing a product.

Finally, patch management generally involves out-of-date software. This category likely ended up as the least common because multiple unpatched pieces of software are often grouped into one finding, which states that the system software should be updated in general.

Conclusion

Since our previous report, IOActive's research and hands-on experience has revealed several trends:

- In general, vulnerabilities have decreased in both impact and likelihood.
- The most common attack vectors are internal software components and network-connected applications.
- Hardening of local interfaces appears to be improving.
- The most common vulnerability types are logic errors, as traditional memory corruption attacks are becoming less common.

Based on our findings, the best path forward is to continue diligently applying industry best practices for secure design and enforcing strong secure coding practices to help prevent easy-to-fix bugs in the first place.

