GEOTAB
management by measurement

15 Security Recommendations for Building a Telematics Platform Resilient to Cyber Threats

Alex Sukhov, Embedded Systems Developer
November 14, 2016

The connected vehicle provides countless new benefits, namely safety, efficiency, and convenience. However, connecting vehicles to the internet has opened up concerns over cybersecurity and sparked an industry-wide discussion.

Among others, these concerns have been expressed by the FBI, NHTSA, and NAFA.[1,2,3] In particular, the FBI has recommended that "vehicle owners should check with the security and privacy policies of the third-party device manufacturers and service providers, and they should not connect any unknown or untrusted devices to the OBD II port." In a similar vein, NAFA recommends that "fleet managers should have policies in place to ensure that only secure devices are connected to the port." We offer this blog in part as a starting point for this conversation.

It can be challenging, of course, for vehicle owners and fleet managers to ascertain what exactly they should look for when trying to assess the security policies and measures related to their telematics platform. The purpose of this blog is to provide guidance in this regard and to invite further discussion as to how the telematics industry as a whole can and should keep advancing security.

In the past, the telematics industry has largely relied on the 3G wireless networks and newer cellular infrastructures which do provide a measure of security. Yet, GSM has shown us that cellular infrastructure, like any other connected system, is not immune to breaches[4] and that more is needed. Having spent considerable time and effort studying the issue and conversing with other experts, we put forward these 15 security recommendations for a resilient telematics platform.

1

Implement Secure Data Transfer

Implementing socket data encryption provides data privacy regardless of the state of the cellular network or any other intermediate connection medium. Implementing authentication will verify that the received/transmitted data sources and destinations are what you think they are. See NIST recommendations for acceptable encryption and authentication algorithms.[5]

Use standard libraries where possible. Writing your own algorithms is time consuming and is very difficult to implement correctly. If there are elements in the system where standard approaches do not meet the requirements or are not compatible, make sure to have a third party (or parties) perform penetration testing. It is much better to have issues identified and fixed before exploits for them are developed.

2

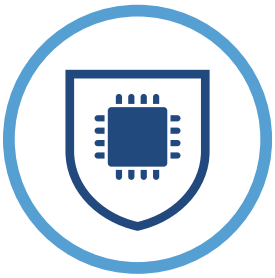
Digitally Sign Updates

Digitally signing application updates is a crucial element in telematics device security. Data communication breaches, however dangerous, limit an attacker to operating within the feature set of the device. Often, the most dangerous attacks on embedded systems require the injection of a malicious application or firmware image. This can allow an attacker to activate system elements which were not intended by design or are restricted by design (i.e. for safety reasons). Signing application updates allows devices to verify that the updates have come from a trusted source. For more information on acceptable algorithms to use for the signature, review these NIST recommendations.[5] It's important to implement a secure internal signature process and key storage location to minimize internal threats.

3

Enable Hardware Code Protection

If the microcontroller supports it, the ability to read firmware code from the device should be disabled. Enabling code protection on the microcontroller greatly limits an attacker's ability to reverse engineer devices as the attacker no longer has easy access to code. This is a great tool in protecting against low skill/resource threat actors but provides little benefit against medium (or higher) skill/resource threat actors. There are companies that will provide code extraction from various code-protected processors for a fee. This is an excellent example of why you should always assume that the attacker has full knowledge of your system implementation.



4

Assume Your Code Is Public

Security implemented through obscurity is a very fragile solution. Security elements should be designed assuming the attacker has full knowledge of your system and already has full access to the code. Even if internal machines and repositories are currently considered in a secure state, this may not have been true in the past or may not be true in the future. A former employee may have had the opportunity to copy the system's code to their home machine.

5

Use Cryptographically Strong Random Numbers

Random number generation is used by many algorithms used in security. It is important that the source of random numbers can provide cryptographically strong random numbers. If the generated random numbers are not cryptographically strong ("not random enough"), the strengths of the algorithms which are using these random numbers can be drastically weakened.

Individualize Security-Critical Data

Security-critical data such as encryption keys and authentication tokens should be unique for each device. If a single device is compromised, it should never compromise any other device or part of the ecosystem. Embedded systems in particular are notorious for using one key across many devices.[6]

6

7

Use Different Keys For Different Roles

Compromising one element of security logic should never affect another. Different keys should be used for different roles. For example, the same key should not be used for socket communication as for the application signature.

Monitor Metadata

Being able to detect and react early to suspicious activity is crucial in minimizing damage done by malicious actors. Actively looking for errors or trends in debug information can reduce or prevent the damage from an attack. There are many cloud processing services available such that even large amounts of performance data can be monitored in real time. Creating services which notify appropriate people when errors or anomalies are detected is essential in identifying attacks early.

8

9

Disable Debug Features

Debug modes and data are an essential part of development, troubleshooting and functionality verification. They are also great tools for detecting anomalies within a system. However, debug-related logic can often be overlooked from the security perspective. Debug logic which contains security-critical information should not be accessible in production software builds.

Perform Third-Party Auditing

All security-relevant components of the system should be audited appropriately. Disclosing your code to a professional third-party company for review should be welcomed as your system should be designed assuming an attacker has full knowledge of the code base. It is far better to have vulnerabilities discovered and fixed in private than not. Identified vulnerabilities with unacceptable risk should have countermeasures implemented within a reasonable time.

10

11

Limit Server Access

Internal account hierarchy should be implemented allowing access to backend servers/features to only those individuals who need them. Multi-factor authentication is an extremely powerful tool for access control. There are various types of hardware or cell phone apps that can be used for this purpose. Login records to servers are essential in forensic analysis of suspicious account activity.

Apply Secure Design Practices

Security should be considered at design stages and not added as an afterthought. Apply the principle of least privilege — ensure that each element of the system permits access to only those who need the access. Do not trust any system inputs to limit the number of ways in which an attacker can probe your system.

12

13

Implement Support for Software/Firmware Updates

It is not realistic to assume that a system is perfectly secure at any point in its lifetime. Security-related issues will arise and there should be a process in place to fix them. The ability to update software/firmware is an essential security feature in a connected system. Having the capacity to perform updates quickly is invaluable in mitigating zero-day threats. It is crucial that the manufacturer is responsible for maintaining the firmware on the device. The end user should not be relied upon to ensure their device is updated and secure — updates need to be pushed automatically to all devices in the field. The manufacturers of the telematics (or other IoT) devices should be accountable for their own security patches.

Verify and Test

The system code base is constantly changing. Have a formal development process. Peer reviewing each change will catch countless issues in advance before they have the opportunity to do damage. Unit testing is also crucial in ensuring security logic remains functional. Unit testing should be set up at both code and hardware levels for comprehensive coverage.

14

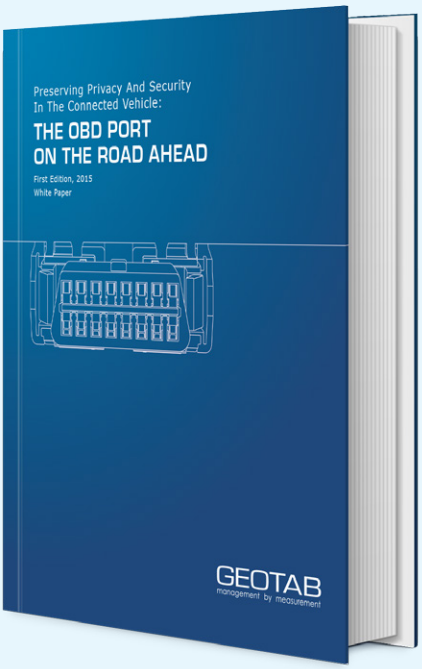
15

Develop a Security Culture

Even the most securely designed systems can be compromised through their operators. All staff with network access (not only security developers!) should be routinely trained and tested in secure internet usage practices. This includes resilience to phishing attacks, use of strong passwords, awareness when clicking URLs, and paying attention to security certificates. Training alone is a good start, but getting staff interested in security and being mindful of it is far more powerful.

To learn more, read the white paper: "Preserving Privacy and Security in the Connected Vehicle: The OBD Port on the Road Ahead" — available at https://www.geotab.com/vehicle-privacy-security/

While we do believe that these recommendations will go a long way in making a telematics platform resilient to cyber threats, learning and improvement will be critical to keeping systems and users secure. This must become an industry wide effort. Therefore, if you have thoughts or ideas on ways to communicate security awareness please contact us at: testdrive@geotab.com



References

1. Federal Bureau of Investigation, "Alert Number I-031716-PSA: Motor Vehicles Increasingly Vulnerable to Remote Exploits," Mar. 17, 2016. [Online] Available: https://www.ic3.gov/media/2016/160317.aspx.

2. National Highway Traffic Safety Administration, "25-16: U.S. DOT issues Federal guidance to the automotive industry for improving motor vehicle cybersecurity," Oc. 24, 2016. [Online] Available: https://www.nhtsa.gov/About-NHTSA/Press-Releases/nhtsa_cybersecurity_best_practices_10242016.

3. NAFA Fleet Management Association. "Fleet Management and the Connected Vehicle," Oct. 2016. [Online] Available: http://www.nafa.org/download.php?f=832.

4. G. Cattaneo, G. De Maio, and U.F. Petrillo, "Security Issues and Attacks on the GSM Standard: a Review," *Journal of Universal Computer Science*, vol. 19, no. 16 (2013), 2437-2452," Oct. 2013. [Online] Available: http://www.jucs.org/jucs_19_16/security_issues_and_attacks/jucs_19_16_2437_2452_cattaneo.pdf.

5. E. Barker, "National Institute of Standards and Technology Special Publication 800-57 Part 1, Revision 4. Recommendation for Key Management," Jan. 2016. [Online] Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf.

6. I. Arce et al., "Avoiding the Top 10 Software Security Design Flaws," IEEE Computer Society, Nov. 13, 2015. [Online] Available: http://cybersecurity.ieee.org/blog/2015/11/13/avoiding-the-top-10-security-flaws/.

About Geotab

Geotab is a leading global provider of advanced, end-to-end telematics technology that helps businesses manage vehicles by extracting accurate, actionable intelligence from real-time and historical trips data. Collecting over 600 million data points daily, Geotab makes benchmarking data accessible to improve productivity, optimize fleets, enhance safety and achieve stronger regulatory compliance.

www.geotab.com

GEOTAB

management by measurement

www.geotab.com