# Security Testing of an OBD-II Connected IoT Device

Gustav Marstorp and Hannes Lindström

***Abstract*—The Internet of Things (IoT) is a rapidly growing network. As society begins to trust the devices in the IoT with increasingly complex tasks, issues regarding the security of these devices are of high priority. An example of an IoT-device in which failure of security could be fatal, is the Telia Sense. Telia Sense is an OBD-II dongle which together with a mobile application connects a car to a smartphone.**

**In this paper, the discoveries that was made during security testing of Telia Sense will be discussed. The system was investigated through a black box perspective. Primarily, a model of the system was produced. Threats were then identified, ranked and tested accordingly.**

**No major vulnerabilities were found. The results all indicated that Telia Sense is a well secured system. The main reasons to this is the fact that the device has very limited functionality and its communications are bounded. Even though no major vulnerabilities were found, this paper can still be used as a guide for future testing of security in IoT devices.**

## I. Introduction

### *A. Background*

Cyber security is an issue of increasing importance, especially considering the rate of which the IoT is growing. The Mirai malware, which affected countless IoT-devices, is what enabled some of the largest DDoS-attacks in history [1]. Despite this, cyber security can still be overlooked by the manufacturers in an attempt of getting their product out on the market as quick as possible. This lack of security is the reason that ethical hackers exist. Ethical hacking is the concept of attempting to expose security vulnerabilities, and thereafter informing the system owners in order to make the system more secure.

This project investigated the security of one specific IoT system: Telia Sense. Telia Sense consists of an OBD-II adapter which together with a mobile application can be used to connect a car to a smartphone.

### *B. Telia Sense*

Telia Sense is an IoT system consisting an OBD-II dongle and a mobile application [2]. Together, the two makes it possible to connect a car to a smartphone. The OBD-II dongle also functions as a Wi-Fi hotspot. The device is manufactured by ZTE with the model name VM6200S.

The mobile application shows information about the registered car. Both physical information as dimensions and survey period but also status of the car. By collecting data from the dongle, the app sends warnings when the battery level is low, emissions are too high or if the engine lamp indicates defects. It is also possible to access the car's location as well as information on previous trips [2].

### *C. OBD-II*

The OBD-II port is a 16-pin connection port, often times placed below the steering wheel. Its primary use was to give independent repair shops and car dealers access to download diagnostic data and run tests, for example on emission [3]. Today there is a growing market of devices that utilize OBD-II in order to give owners access to the same data through their smartphone. In 1996 the OBD-II port was made mandatory for all cars in the United States. In 2001 the same standard was introduced to all gasoline fueled cars in Europe [3].

### *D. CAN*

Through the OBD-II port it is possible to get direct access to the Controller Area Network (CAN). For over 30 years, CAN has been the standard for internal networks in passenger cars. CAN was not designed to be secure from intrusion and had no reason to be, since the only way to access it was through physical access [4]. Connected to this network are electronic control units (ECUs). These units are what connects the mechanical functions of the car to the electronic control system; this includes vehicle operations such as the throttle, breaks, steering and also simpler functions like the locks [3]. On a CAN bus, messages are broadcasted to all nodes. The CAN frame includes a destination field and every node ignores data that is not addressed to them. However, information about the source is not included in the frame. This means that the receiving node can not know where the message comes from and if it is trustworthy or not [3].

### *E. Project goal*

The main goal of the project was to investigate potential flaws in security of the Telia Sense system. The question asked was: how secure is Telia Sense, and is it possible to access the CAN network by hacking the unit?

## II. System Model

In order to be able to carry out sufficient security testing of a system, a model of the system is of much use. Testing of the device is carried out from a black-box perspective; this means that very little prior information is given regarding the system architecture and what security measures have been implemented. Because of this, some testing had to be done in order to make sure that the model was as true to reality as possible. This map, which can be seen in figure 1, of the system could then be used as a starting point in the task of identifying attack vectors and creating a complete threat model.

To identify which servers are being used, a proxy was set up to intercept traffic between the app and the internet.

## III. Threat analysis

### A. Identifying threats

A threat analysis was carried out based from the system model, and the threats were categorized according to the STRIDE model. STRIDE is an acronym that stands for Spoofing of identity, Tampering with data, Repudiation, Information disclosure, Denial of service and Elevation of privileges [5].

Considering the limited extent and goal of the project, some categories of threats were of higher priority. In order to send arbitrary CAN messages from the dongle, an attack involving elevation of privilege or tampering with data seemed most likely. The main threats, those which seemed most likely to lead to a working exploit, were identified and listed in tables I to V.

TABLE I
THREAT 1

| | |
|---|---|
| Threat description | Attacker gets access to a root shell on the device, then uses this to execute commands. |
| Threat target | Internet interface of the Sense dongle & Wi-Fi hotspot |
| Attack techniques | Attacker manages to compromise a service running on an open port. |

TABLE II
THREAT 2

| | |
|---|---|
| Threat description | By analyzing the mobile application, the attacker gains access to information about the system, e.g. backdoor accounts, configuration files, source code and private keys. |
| Threat target | Telia Sense mobile application |
| Attack techniques | Through deconstruction and analysis of the app, an attacker discovers files or code which contains sensitive information. |

TABLE III
THREAT 3

| | |
|---|---|
| Threat description | By analyzing the firmware, the attacker gains access to information about the system, e.g. backdoor accounts, configuration files, source code and private keys. |
| Threat target | Firmware of the Sense dongle |
| Attack techniques | By obtaining, deconstructing and analyzing the firmware, an attacker discovers files or code which contains sensitive information. |

TABLE IV
THREAT 4

| | |
|---|---|
| Threat description | Attacker succeeds in performing a buffer overflow attack on the device. |
| Threat target | Sense dongle. |
| Attack techniques | Attacker manages to overrun buffer boundaries by tampering with data that is sent to the server |

TABLE V
THREAT 5

| | |
|---|---|
| Threat description | Attacker succeeds in performing a command injection attack on the device. |
| Threat target | Sense dongle. |
| Attack techniques | Attacker manages to insert and execute commands by tampering with data that is sent to the server |

### B. Rating threats

The threats were rated using the DREAD rating system. DREAD stands for:

- **Damage potential:** How great is the damage if exploited?
- **Reproducibility:** How easy is it to reproduce the attack?
- **Exploitability:** How easy is it to attack?
- **Affected users:** Roughly how many users are affected?
- **Discoverability:** How easy is it to find the vulnerability?

The rating system of DREAD is 1-3. 1 is low risk, 2 is medium risk and 3 is high risk. The final risk is then ranked where 5-7 is low, 8-11 is medium and 12-15 is high risk [5]. The result of this is displayed in table VI.

TABLE VI
DREAD

| | Threat 1 | Threat 2 | Threat 3 | Threat 4 | Threat 5 |
|---|---|---|---|---|---|
| **D** | 3 | 2 | 2 | 3 | 3 |
| **R** | 2 | 3 | 2 | 2 | 3 |
| **E** | 2 | 2 | 2 | 1 | 1 |
| **A** | 3 | 3 | 3 | 1 | 1 |
| **D** | 3 | 2 | 2 | 1 | 1 |
| **Total** | 13 | 12 | 11 | 8 | 9 |

## IV. Theory

### A. Port scanning

To combat the possibility of remote attacks of the device, it is very important that its internet interface is secured properly. One basic task that can be performed during reconnaissance of a remote system is port scanning. Port scanning can be performed in multiple ways and by using different protocols, however, the primary method is the same. By attempting to connect to a port and then analyzing the response (or lack of response), conclusions can be drawn about whether the port is open or closed. Open ports indicate a running network service and, to an attacker, a possible point of entry. Because of this, it is recommended to avoid publicly open ports if they are not required.

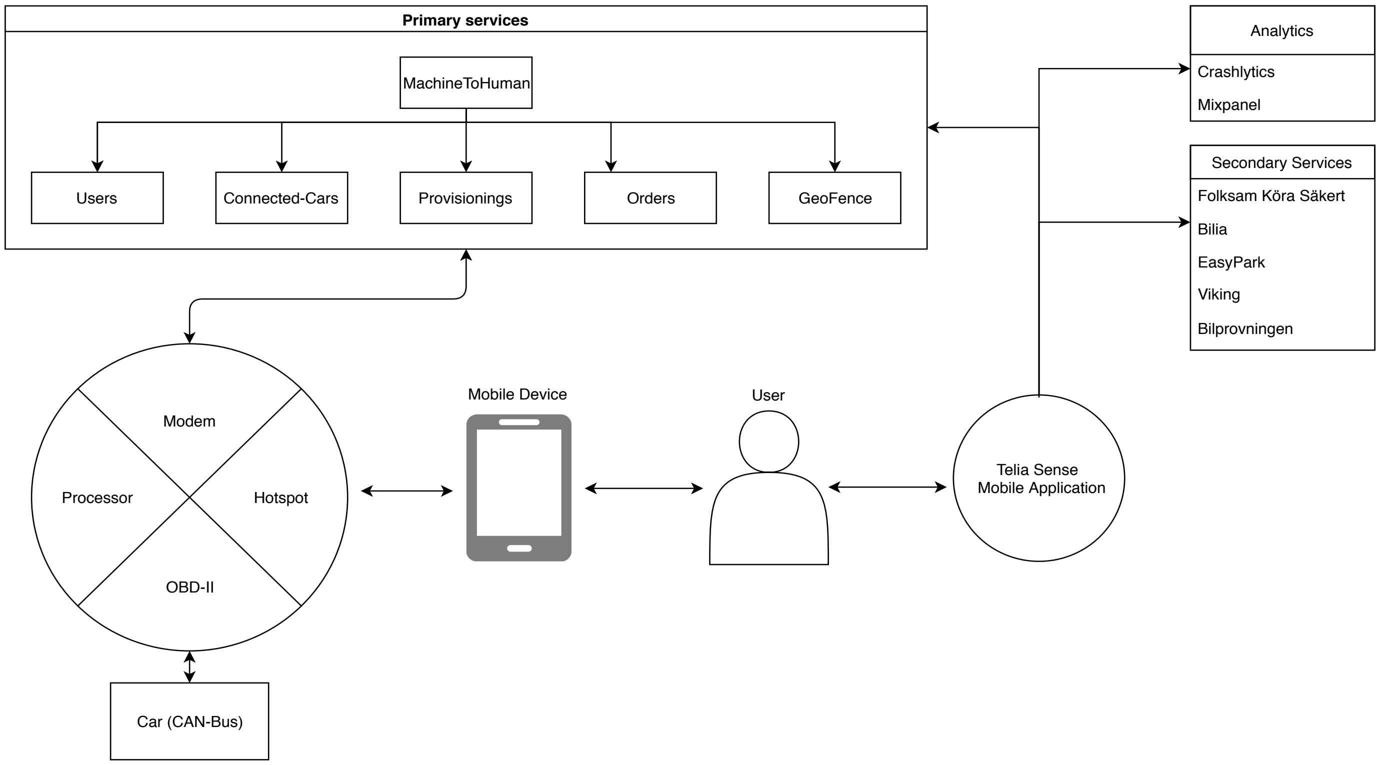

Fig. 1. Basic map of the Telia Sense system.

The information gathered from a port scan is not restricted to finding open ports. By using different types of probe packets, additional information can be gathered. For example, OS-fingerprinting and version detection of services can be performed. This could improve an attackers possibilities of finding vulnerabilities.

Despite the simplicity of a port scan, there are countless examples of exploits made possible because of services on publicly open ports. One notable example is discussed in [6]; in this case the researchers found multiple open ports, one of which they were able to compromise. The service that they found was bound to the open port was a D-Bus message service which is used for inter-process communications. By using this exposed service, a command injection vulnerability was discovered. However, this vulnerability was not even needed since the service already included a feature to execute arbitrary shell commands. This further goes to show the importance of not leaving key services exposed to a potential attack.

## *B. Mobile application*

The analysis of the mobile application can be divided into two parts: static analysis and dynamic analysis.

Static analysis is a method used to study a program without executing it. The purpose is to detect errors or weaknesses in the program. Usually an automated tool is used to make the process easier. These tools go through the code and looks at app permissions, browsable activities and other additional functions. Examples of poor usage of coding standards and potential vulnerability are flagged. This could, for example, be hardcoded sensitive information such as private keys or IP-addresses [7].

Android apps are usually written in Java source code and then compiled to Java bytecode. Bytecode is computer object code, designed for efficient execution by a program often referred to as a virtual machine. The virtual machine translates each generalized machine instruction into a specific machine instruction, instructions that the computer processor understand. In android machines this virtual machine is called android runtime (ART). ART uses bytecode as input. The format for these files are DalvikEXecutable (.dex) or Optimized Dalvik Executable (.odex) [8].

To perform static analysis of an app, the program code need to be readable for humans. When downloading an Android package kit (APK), which is the package file format for android apps, this is not the case. The APK consist of a lot of different files: program code, assets, resources, certificates and a manifest file. The program code, which in the APK is in .dex or .odex format, is very interesting when investigating security. To make the code readable for humans, it needs to be decompiled. First to Java bytecode and then to Java source code, in which it was originally written.

The second part of analyzing an application is the dynamic half. During dynamic analysis, data storage and server communications are investigated in runtime.

How data is stored in an app is important. Storing critical data in an unsecure manner can lead to negative consequences. A commonly used way of saving permanently small collections of key value-pairs is with the SharedPreferences API [9]. The file is stored within the app's data directory. By gaining root level access it is possible to get the data stored in the SharedPreferences file where sensitive information could be found.

The communication part of the dynamic analysis can be done by setting up a proxy. All HTTPS traffic between the app and server can then be displayed and even tampered with. By executing all of the functions in the app at least one time, it is possible to map which server are used for what purpose [7]. Inputs can also be tested for command injection and buffer overflow vulnerabilities.

### C. Firmware

When it comes to controlling an IoT device, the firmware and the hardware that it is running on is fundamental. This, combined with the fact that security in firmware is often times overlooked by developers, is why it is of great interest to an attacker. Firmware analysis is often times performed with the aim of locating: passwords, private keys, vulnerable services, configuration files, backdoors or source code [10]. The amount of work required to be successful in this task can vary; some manufacturers and industries puts more effort into complicating the reverse engineering process than others.

The first step that is required in order to start analyzing the firmware is, naturally, obtaining it. This can be achieved in multiple different ways. For instance, it can be downloaded directly from the vendor, it can be proxied during an update and finally: Firmware can be dumped from the hardware [10].

### D. Hardware

When it comes to hacking IoT-devices, especially devices like Telia Sense, exploits that require access to the hardware is not going to be the major thing to worry about. Despite this, it is still important that security researchers do not forget to explore this aspect. Information about the hardware can be crucial in order to succeed in executing a software exploit. Communication interfaces that were intended for debugging and programming can be of help to a hacker, for instance, by presenting runtime information. In some cases, these interfaces might even present a root shell. Some of the main communication interfaces to pay attention to are: USB, UART, SPI, I2C and JTAG [11].

## V. Method

### A. Port scanning

Port scanning of the Telia Sense dongle was executed using multiple different approaches. What differed between these test was not only the content of the test probes, the source of the scans also connected to the dongle through different interfaces. The setups that were used includes the following:

- Source of scan connected directly to the Sense's hotspot (scanning the local IP-address assigned to the gateway)
- Source of scan located on a different network, scanning the Sense over the internet using its global IP-address.
- Source of scan connected to the hotspot of one Sense, scanning another Sense over the internet using its global IP-address.

For all of these configurations, the ports were probed using both UDP and TCP. The program that was used for the scanning was Nmap: one of the most popular network scanning softwares.

### B. Mobile application

In this project, static analysis was performed by using the automatic tool Mobile Security Framework (MobSF). The Telia Sense APK was downloaded and then inputted to MobSF, which then decompiled the app and analyzed its content. In the program interface, components as activities, services, receivers and providers were listed. It also analyzed API usage, app permissions, browsable activities and additional functions. Code sections that indicate the possibility of a vulnerability or hardcoded sensitive information were flagged. These sections were then reviewed manually.

In order to perform the dynamic analysis of the Telia Sense, Burp Suite was used. Burp Suite is a graphical tool for security testing. By setting up a proxy, all the data traffic between the app and servers could be observed. By executing all the functions inside the app and analyzing the messages that are being sent to the servers, information about how the app and servers work can be gained. To complement this information, the app's data storage was analyzed. The SharedPrefs file and its content was scanned during runtime. This was made possible by ”rooting” the Android phone, which gives access to root level directories and files.

By using the proxy in Burp Suite, messages to the server were intercepted and edited after they had left the phone. This allowed for the ability to test how the server responded to illegitimate messages and modified values of variables. In order to detect what was being forwarded to the physical unit, the UART read console was observed at the time of testing. The main focus of this testing was to look for the possibility of injection or buffer overflow attacks, (targeted at the dongle, not the servers). In both tests, settings regarding the hotspot configuration were changed in the application. This would then trigger a message to be sent to the Sense device, and not stop at the server.

In the case of testing for buffer overflow vulnerabilities, this was performed by modifying the SSID and password string variables to be of a very large size.

Injection vulnerabilities were also tested for by editing the SSID and password fields, this time the variables were modified to include special characters and system commands in different forms. The UART read console was then studied in hopes of responses from the device that could indicate that system commands were being executed.

### C. Firmware

The firmware of Telia Sense was not available for download online. Also, the device communicates with the server over an encrypted 4G-connection, this means that proxying or mirroring was not an option. Because of this, an attempt of acquiring the firmware was made by dumping the contents of the flash memory that the device boots from. This was done by using a TL866CS, a universal programmer/reader. In order to do this, the flash memory had to be desoldered from the PCB.

Further analysis of the flash-dump was performed by using Binwalk. Binwalk is a penetration testing tool which can be used for scanning binary files for embedded files and source code.

### *D. Hardware*

Disassembling the Sense-dongle and removing metal shielding exposed a USB-C port. By using the data sheets of the processor [12] in combination with a multimeter, additional JTAG and UART interfaces could be located. Unfortunately, it was not possible to establish a connection with the USB port from a computer. It was also not possible to communicate with the device over the JTAG interface. However, readings of the UART port were successful at 115200 baud.

## VI. Results

### *A. Port scanning*

Table VII shows the result of the port scans were UDP- and TCP-probes were used.

TABLE VII
RESULTS FROM PORT SCANNING

| | **Probing with TCP** | **Probing with UDP** |
|---|---|---|
| Source of scan located on a different network, scanning the Sense over the internet using a global IP-address | Port 53 open The rest of the ports were open—filtered. | All ports were either closed or open—filtered |
| Source of scan connected directly to the Sense's hotspot (scanning the the local IP-address assigned to the gateway) | All ports were filtered | All ports were either closed or open—filtered |
| Source of scan connected to the hotspot of one Sense, scanning another Sense over the Internet using its global IP-address | All ports were filtered | All ports were either closed or open—filtered. |

### *B. Mobile application*

The results of the static analysis were limited. In table VIII, the issues that MobSF discovered in its code analysis are listed. However, by reviewing the code sections affected by hand, no actual threats were discovered.

The analysis of the traffic between the app and server, in combination with the data found in the SharedPrefs file lead to a understanding of how the application worked and what variables were being passed on to the servers. This understanding was mainly used in testing for injection and buffer overflow vulnerabilities in the device.

The modified messages that contained very large strings (to test for the possibility of buffer overflows) were not accepted by the server. Since the servers controlled the variables for size, this was not a feasible attack.

However, the servers did accept special characters in strings. These modified messages were then forwarded to the device. The UART readings on the device, did not indicate any successful injection of commands. This attack technique was then set aside as well.

TABLE VIII
MOBSF CODE ANALYSIS

| **Issue** | **Severity** |
|---|---|
| This App uses Java Hash Code. It's a weak hash function and should never be used in Secure Crypto Implementation. | High |
| App can read/write to External Storage. Any App can read data written to External Storage. | High |
| Files may contain hardcoded sensitive informations like user-names, passwords, keys etc. | High |
| IP Address disclosure | Warning |
| App creates temp file. Sensitive information should never be written into a temp file. | High |
| The App uses an insecure Random Number Generator. | High |
| This App may have root detection capabilities. | Secure |

### *C. Firmware*

By dumping the contents of the flash drive, we were presented with a binary file. By translating this file into ASCII and reading its contents, it could be concluded that major sections of the file were compressed or encrypted. An entropy analysis was performed using Binwalk, and the results of this are shown in figure 2. This entropy analysis seems to confirm the assumption that the firmware is compressed or encrypted.

The file was also scanned for signatures that indicate file systems or partition headers etc., however, none were found.

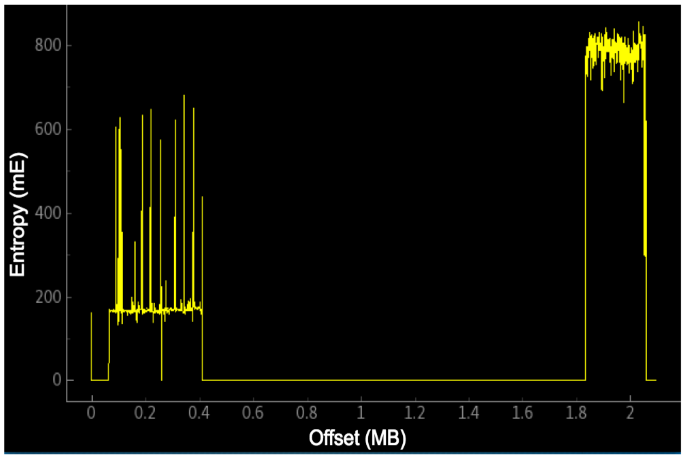

Fig. 2. Entropy graph of the binary file which was dumped from the device.

### *D. Hardware*

The UART communication interface presented a read console which printed information regarding the device's status and hotspot settings, for instance. It was, however, only a read console and commands could not be executed.

## VII. Discussion

### *A. Port scanning*

The port scans all indicated that the system was well secured. No vulnerable services were discovered, and nearly all ports were classified as filtered. This means that the packets were dropped somewhere in the process, most likely due to a firewall. This is an excellent countermeasure to take in order to make port scanning less rewarding to an attacker. Even though potentially vulnerable services are running on the device, utilizing different rules for filtering their traffic can complicate the process for a hacker. Not only does the attacker now have to compromise the service itself, but also find a way to bypass the firewall. This could be very time consuming and could, for example, require spoofing.

### *B. Mobile application*

The result of the statical and dynamical testing of the mobile application did not present any obvious vulnerabilities. It did, however, help in getting an understanding of how the system worked. This could in turn, be of help when testing for vulnerabilities against command injection and buffer overflows. Any time a user gets the chance to input data into a program, it is important that this input is verified. This is handled by Telia well in the mobile application. However, by modifying the HTTPS messages, some potential minor flaws were discovered. For instance, special characters were able to be sent to the device even though they were not supported. In this case it did not turn out to be a problem, but a simple verification on the server-side could stop these messages from being forwarded at all.

### *C. Firmware*

The content that was dumped from the flash memory did not reveal much information. The developers seems to have encrypted the firmware in order to make the reverse engineering process much harder. This is definitely a reliable way of making attacks harder to perform. However, some might argue that a secure firmware or application does not need to be obfuscated.

### *D. Hardware*

None of the identified hardware communication interfaces presented the user with a shell command line and only the UART interface presented any data at all. This is good since it counters the possibility of many hardware hacks, it also complicates the information gathering process for attackers.

## VIII. Conclusions

In conclusion, the Telia Sense system is very well secured over all. What makes it so secure is mainly the fact that the device has very limited functionality and its communications are bounded. The app is very well separated from the actual Sense dongle and it only communicates with the server. When the communications are limited to one channel and the functions that are handled are few, attackers are very limited in finding a vulnerability. This was proven in the case of the Telia Sense.

Furthermore, because of the fact that the device is connected to the Internet over an encrypted 4G connection, it is going to be very hard for a hacker to compromise this channel of communication. This combined with the fact that the firmware is hard to acquire makes it very difficult for an attacker to get an idea of how the device functions at all. This makes it much harder to attack.

Another important aspect of security is the ability to update the firmware when vulnerabilities and bugs have been discovered. Updates are carried out over the air (OTA), and once again: this encrypted communication is hard for an attacker to compromise.

If the work on this project was to be further continued, a deeper analysis and exploration of the firmware would be performed. Without additional information about the system, it is going to be hard to manage to find any vulnerabilities.

## Acknowledgment

The authors would like to thank supervisor Pontus Johnson for his guidance and support during the entire project.

## References

[1] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *Proceedings of the 26th USENIX Security Symposium*, 2017. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[2] (2018, april) Gör bilägandet till en barnlek. Telia Sverige AB 556430-0142 Box 50077, 973 22 Luleå. [Online]. Available: https://www.telia.se/privat/bredband/tjanster/teliasense#pageSection_01

[3] C. K. Dan J. Klinedinst, "On board diagnostics: Risks and vulnerabilities of the connected vehicle (white paper)," in *'Architecture of OBD-II and CAN'*. Carnegie Mellon University Software Engineering Institute 4500 Fifth Avenue Pittsburgh, PA 15213-2612: CERT Division, April 2016, p. '2'.

[4] R. Currie, "Hacking the can bus: Basic manipulation of a modern automobile through can bus reverse engineering (white paper)," in *' Introduction'*. SANS Institute InfoSec Reading Room, May 2017, p. '2'.

[5] A. Guzman and A. Gupta, "Iot penetration testing cookbook," in *'IoT Threat modeling'*. 35 Livery Street Birmingham UK: Packt Publishing Ltd, November 2017, pp. '33–35'.

[6] D. C. Miller, , and C. Valasek, in *'Remote Exploitation of an Unaltered Passenger Vehicle'*. 303 Second Street South Tower, Suite 900 San Francisco, CA 94107: Black Hat, August 2015.

[7] A. Guzman and A. Gupta, "Iot penetration testing cookbook," in *'Exploiting IoT Mobile Applications'*. 35 Livery Street Birmingham UK: Packt Publishing Ltd, November 2017, pp. '172–222'.

[8] (2017, November) Art and dalvik. 1600 Amphitheatre Parkway Mountain View, CA 94043 USA. [Online]. Available: https://source.android.com/devices/tech/dalvik/

[9] (2018, April) Save key-value data. 1600 Amphitheatre Parkway Mountain View, CA 94043 USA. [Online]. Available: https://developer.android.com/training/data-storage/shared-preferences?authuser=4

[10] A. Guzman and A. Gupta, "Iot penetration testing cookbook," in *'Analyzing and Exploiting Firmware'*. 35 Livery Street Birmingham UK: Packt Publishing Ltd, November 2017, pp. '70–83'.

[11] ——, "Iot penetration testing cookbook," in *'IoT Device Hacking'*. 35 Livery Street Birmingham UK: Packt Publishing Ltd, November 2017, pp. '223–255'.

[12] (2017, April) Stm32f105xx stm32f107xx. [Online]. Available: http://www.st.com/content/ccc/resource/technical/document/datasheet/e4/f3/1a/89/5a/02/46/ae/CD00220364.pdf/files/CD00220364.pdf/jcr:content/translations/en.CD00220364.pdf