CHALMERS |  UNIVERSITY OF GOTHENBURG

Wireless Security in Road Vehicles

Improving Security in the SIGYN System

Master of Science Thesis in Computer Science and Engineering

MARKO JUSUFOVIĆ
MATTHIAS NILSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, November 2009

Wireless Security in Road Vehicles
Improving Security in the SIGYN System

MARKO JUSUFOVIĆ
MATTHIAS NILSSON

Examiner: ERLAND JONSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden, November 2009

Abstract

Remote wireless diagnostics and firmware updates are emerging technologies in the vehicular industry. They have many benefits, such as the ability for a manufacturer to perform firmware updates on a large population at the same time, without having to get vehicles to visit a workshop. Previous systems have relied on wired connections for performing these actions, and therefore the vehicle has not needed a lot of protection. By introducing wireless capabilities in the vehicles, vehicle manufacturers now face new risks which must be prevented or mitigated.

We have studied such a wireless system, SIGYN, currently being developed by Volvo Personvagnar and Volvo IT in collaboration with Chalmers. We have written a reference client, which will be used for testing. We have examined security requirements and implemented security mechanisms to show how security in SIGYN could be improved.

We have found a number of measures to increase security in the system and have implemented some of them. However, our work is only one step on the way to a fully secure system. Thus we finally provide a discussion of which further actions need to be taken.

Keywords: Vehicle, wireless security, remote diagnostics

Sammanfattning

Trådlös fjärrdiagnostik och dito mjukvaruuppdateringare är båda teknologier som just nu utvecklas av fordonsindustrin. Det finns många fördelar med dessa, som att fordonstillverkaren ges möjlighet att uppdatera mjukvara i en hel population av fordon på en och samma gång utan att fordonen behöver uppsöka en verkstad. Tidigare system har förlitat sig på fasta anslutningar för att genomföra sådana ingrepp och därför har behovet av att skydda fordonen varit mindre. Genom att lägga till trådlösa anslutningsmöjligheter i fordonen ställs tillverkarna inför nya risker som behöver förhindras eller reduceras.

Vi har studerat ett sådant system, SIGYN, som utvecklas av Volvo Personvagnar och Volvo IT i sammarbete med Chalmers. Vi har skrivit en referensklient för att använda i testning. Vi har undersökt säkerhetskraven och implementerat säkerhetsmekanismer för att visa hur säkerheten i SIGYN kan förbättras.

Vi har hittat ett antal åtgärder för att öka systemets säkerhet och vi har implementerat några av dem. Dock är vårt arbete endast ett steg på vägen mot ett säkert system. Därför diskuterar vi slutligen vilka ytterligare åtgärder som krävs.

Nyckelord: Fordon, trådlös säkerhet, fjärrdiagnostik

Preface

This Master's thesis at the Department of Computer Science and Engineering at Chalmers University of Technology and Gothenburg University was conducted at Volvo IT in cooperation with Diadrom Systems AB. The thesis was supervised by Per Dahlberg, Ph.D. in informatics and head of system development at Diadrom, and Erland Jonsson, professor in Computer Security at Chalmers.

We would like to thank both Per and Erland for their invaluable assistance, helping us with anything from lending us hardware to giving us helpful suggestions when we were stuck. Special thanks goes to Dennis Nilsson for taking time to give us feedback on our work, to Karl Hagås at Columbitech for help with their VPN software and to Marko Vainio at Volvo IT for providing workspace at Volvo IT throughout our thesis work. Last but not least, a very special thanks to Johan Wikman at Volvo IT for tirelessly putting up with all our questions and for being a great support.

We have in our thesis used code written by IBM and Volvo IT when developing our reference client.

We have carried out the thesis work together, sharing acquired knowledge between us. Hence it will be difficult for us to single out parts of the work that only one of us performed, with two exceptions. Marko wrote the GUI part of our reference client himself and Matthias was responsible for finalizing the thesis report.

Contents

Abbreviations

AES Advanced Encryption Standard

CAN Controller-Area Network

DoS Denial of Service

ECU Electronic Control Unit

EMI Electro Magnetic Interference

FFD Full Function Device

GPRS General Packet Radio Service

GUI Graphical User Interface

HIDS Host-based intrusion detection system

IDS Intrusion Detection System

IEEE Institute of Electrical and Electronics Engineers

IPSec Internet Protocol Security

LR-WPAN Low-Rate Wireless Personal Area Network

MAC Media Access Control

MQTT MQ Telemetry Transport

PDA Personal digital assistant

RFD Reduced Function Device

RSA Rivest, Shamir, Adelman (encryption algorithm)

SIGYN Software In Global Yielding Networks

S/MIME Secure / Multipurpose Internet Mail Extensions

TLS Transport Layer Security

TPD Tamper-proof Device

TPM Trusted Platform Module

VANET Vehicular Ad-hoc Network

VPN Virtual Private Network

VTTP Vehicle Task Telematics Protocol

WTLS Wireless Transport Layer Security

Chapter 1

Introduction

The vehicular industry wants to add wireless capabilities to road vehicles, in order to provide everything from streaming multimedia and infotainment to software updates and remote diagnostics. This introduces a new attack vector and with this, the in-vehicle network that has until now been largely protected from everything except physical access attacks is suddenly very vulnerable. As with most emerging technologies, where focus is on the technological progress, security has been overlooked.

This thesis will examine the security research in this field and the state of a current industrial project for remote software updates and diagnostics. We will give suggestions on how the security recommendations in the research could be incorporated in further development of the project as well as suggestions on areas that need more focus in order to further strengthen the system.

1.1 Wireless systems in the automotive industry

During the last years road vehicles have become more and more computerized. Today a vehicle contains a number of *electronic control units* (ECUs), which are responsible for anything from opening windows to controlling automatic brake systems. Each ECU runs specific software, and as with all software there is a need to be able to update it, whether it is to fix bugs or to add new features. In order to perform such an update, the customer must visit an authorized service station, where the service station employee sets up a wired connection to the vehicle. The new software is downloaded to the car and installed on the specific ECU, overwriting the old software. At the service station it is also possible to perform diagnostic tests on the ECUs, and thereby e.g. identifying causes for malfunctions. Since this pro-

cedure can be both time consuming and inconvenient, vehicle manufacturers have started to examine the possibility of wireless updates instead.

There are a number of benefits with both remote wireless software updates and remote wireless diagnostics, such as being able to perform software updates on whole populations of vehicles at the same time and being able to diagnose errors before a vehicle arrives at the service station, thus cutting down time spent on locating the error at the service station. This means a higher availability for the vehicles and less time spent running faulty software. There are, of course, economic incentives for developing this technology, mainly because it will mean increased efficiency and that work previously done by people can be done by computers instead.

For instance, the OnStar[1] service by General Motors has an Automatic Crash Response function, where sensors in the vehicle activate in a crash and automatically alert an operator and also sends details of the crash collected by the built-in sensors. OnStar also provides vehicle diagnostics, where a driver can receive information on e.g. current oil levels or tire pressure as well as air-bag or engine issues. Another application is streaming multimedia and infotainment systems, which can supply the passengers with a range of movies or real-time weather reports. Related to this are ad hoc networks, where vehicles relay information such as road conditions or accidents to each other (see Section 4.1).

However, when adding wireless capabilities to vehicles the manufacturers need to make a new security assessment, since the added capabilities will introduce new attack vectors and risks exposing the in-vehicle network to attackers. Previously, the CAN bus connecting the ECUs in the vehicle has largely been protected from attackers, since these needed physical access to the vehicle. Implementing wireless capabilities makes it possible to perform remote attacks on the vehicle. Without additional security, an attacker could sniff the connection between the vehicle and the portal and steal proprietary data, manipulate the software sent to the vehicle or plant viruses in the vehicle's ECUs. Since the vehicles will connect to a portal to download the updates, this portal will also become a possible target for attacks, such as Denial of Service attacks or intrusion attempts.

1.2 Remote diagnostics and software updates

When more and more of the features in our cars are controlled by computers more and more of the maintenance, repairs, etc. have to be done with

[1] http://www.onstar.com

the help of computers. The vehicles have become so complicated that finding out what is wrong with them is close to impossible without the help of computers. Also, as with all other software, the software in vehicles need to be updated. Sometimes to fix a bug, sometimes to add new features, sometimes to enhance the old functionality. Now these things are done by mechanics in workshops but vehicle manufacturers and researchers are looking into the possibilities to do them over wireless networks.

Remote diagnostics can be used to find problems before they are activated. By seeing e.g. that a component is starting to become worn, it can be replaced before it breaks. It can be used to detect errors in the construction of a vehicle model by looking at problems frequently occurring and to find out which components are unreliable. Other areas of use are statistics of e.g. driving behaviour, and surveillance of employees. The remote aspect is of more use to both vehicle manufacturers and companies with a large vehicle population, such as transport companies, than to individual drivers. The corporations have a much higher demand on availability and the possibility to track error trends while for individual drivers, (semi-) annual check-ups at the local workshop will probably suffice.

Remote software update over wireless networks can save work (and thus money) for both the car owner and manufacturers. Since the vehicle does not have to be brought in to a workshop for the software to be updated, both the work of the vehicle owner (bringing the car to the workshop and waiting for the update to be finished) and the work of the mechanic (plugging in a computer to the car and making the update) will become unnecessary. Remote updates will also make it possible for one person to update many vehicles at the same time, as opposed to a mechanic that connects a computer to vehicles one at a time. The process can also become automated, thus minimizing the work required by the driver, and even completely transparent, i.e. removing the driver from the update process altogether.

There are also issues specific to remote updates. An example is what happens if a vehicle misses an update. Even if the automatic updates makes it a lot more likely that a vehicle will be updated, there will probably still be some that are missed, e.g. because they have been standing unused in a garage for a long time. This means that we need to keep track of which vehicles have which versions of the different software for the different ECUs and update in a consistent way so that a vehicle does not have incompatible software versions installed at the same time.

Chapter 2

Aim

We will study a system for remote software updates and diagnostics called SIGYN, currently under development by Volvo Personvagnar and Volvo IT. During the first part of the SIGYN project, the focus was mainly on the technical implementation, which meant that the security aspects were somewhat overlooked. Because of this, the security of the system is not what it should be.

There are a number of scientific articles concerning security in systems for wireless diagnostics in vehicles. By studying the literature, our goal is to decide which of the security recommendations we find will be applicable to SIGYN and if these recommendations are possible to implement. To the extent that this is possible, we will try to implement them in a lab environment. We will also decide whether these measures are enough to protect SIGYN or if there are additional steps that need to be taken in order to secure the system.

We will write a proof-of-concept client in Java, trying to make it as platform independent as possible. The client will communicate using MQTT and should be able to transfer files to and from a broker. This client will also be used when we implement security measures.

2.1 Limitations

Since SIGYN is under development, the system we have looked at is far from finished. A number of features are missing and some are only partly implemented.

We do not have access to the actual hardware that will be used in the vehicle or the portal, we do not even know what it will be. Instead we will use standard PCs for our implementations. This means that the software

we use, at least on the client side, will probably not work in the real system.

We will not examine physical attacks on the vehicle, e.g. side-channel attacks such as Power Analysis, since this is outside the scope of this thesis.

Chapter 3

Technical Background

In this chapter, we provide the reader with the necessary background in order to understand the subject at hand. Section 3.1 is a summary of the work done by Dennis K. Nilsson while working on SIGYN as a Ph.D. student from Chalmers. In Section 3.2 we discuss the plausibility of using strong cryptographic protocols in embedded systems with restricted hardware and in Section 3.3 we briefly talk about protecting sensitive information in vehicles using tamper-resistant storage. We conclude this chapter by discussing the threats against SIGYN and the implications of a system compromise in Section 3.4.

3.1 Academic research related to SIGYN

This thesis builds mostly on the works of Dennis K. Nilsson, who while working as a Ph.D. student at Chalmers participated in the SIGYN project. Nilsson has written a number of articles on the subject of security in vehicular environments [4], [5], [6]. We will present a part of this work here to give the reader the necessary background knowledge to better understand what we have done and why.

3.1.1 Architecture

Figure 3.1 shows a conceptual image of the wireless infrastructure. It is divided into three parts: *portal*, *communication link* and *vehicle*. This is a centralized architecture, with the portal being the central node. Some assumptions are made:

- The portal communicates with a number of vehicles at any time (a one-to-many relationship).

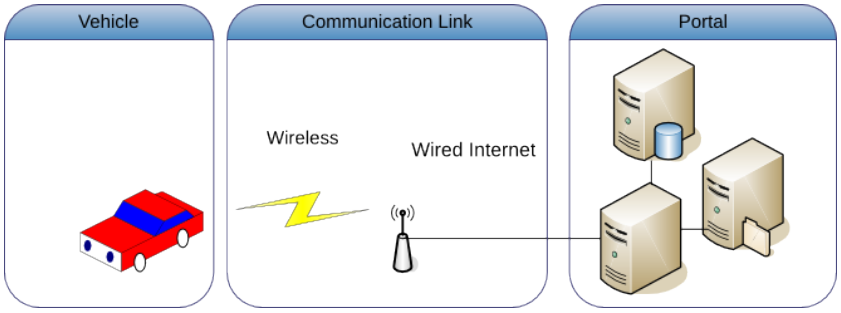

Figure 3.1: Conceptual model of the wireless infrastructure

- The portal keeps track of each vehicle with regards to installed software, cryptographic keys and so on.
- The portal is not considered a bottleneck, i.e. it is powerful enough to handle any number of simultaneous connections.
- There is no communication between vehicles.

3.1.2 Attacker model

Just like Raya et al. in [7], Nilsson categorizes the attacker as either *insider* or *outsider*. The insider is an authenticated user and can, in addition to any actions the authorized user can perform, also mount attacks from inside the system. The outsider on the other hand, is considered an intruder and can only attack the system from the outside. Such attacks may include eavesdropping, interception, modification or injection of messages sent over the communication link. After a successful intrusion, however, an attacker might gain access to the internal network of either the vehicle or the portal, and can then mount attacks as an insider.

3.1.3 Desired security properties

Nilsson identifies the following desired security properties for wireless diagnostics and software updates in vehicles:

Confidentiality

Since the software which will be installed in the vehicle is proprietary, it should be kept confidential. This concerns both the storage as well as the transmission of the software. Also, diagnostic requests, diagnostic responses and stored diagnostic data should be protected.

Integrity

It is important to prevent that the software to be installed in the vehicle is modified, since it controls safety and security-critical features. The vehicle should also verify that the correct software has been received.

Authentication

To prevent impersonation of either the portal or the vehicle, mutual authentication needs to be implemented. Data authentication is also important to allow the vehicle to verify that the software comes from a trusted source.

Freshness

The protocol must ensure that messages are fresh, in order to prevent replay attacks, such as the resending of a diagnostic request instructing the vehicle to unlock the door or turn off the head lights.

Resilience to lost packets

Since packet loss is common in wireless communication, there is a need to handle this in a secure way. Also, protection against denial-of-service attacks is needed to preserve the availability of the communication link.

3.1.4 Risk assessment

Using the attacker model (Section 3.1.2), the desired security properties (Section 3.1.3) and some additional assumptions (such as offline distribution of cryptographic keys and that rekeying will not be considered), Nilsson performs a risk assessment and lists risks for each of the three regions.

Portal

- Impersonation: By impersonating the portal, an attacker may issue diagnostic requests or install software on the vehicle, which will believe that it originated from the real portal. This gives the attacker the opportunity to execute arbitrary code on the vehicle.
- Intrusion and Denial of Service: A successful intrusion by an outside attacker may potentially give him or her equal access to that of an insider, e.g. access to proprietary or sensitive data and the ability to execute more serious attacks. Denial-of-service attacks can prevent users from upgrading their software.

- Execution of Arbitrary Code: By either impersonation or a successful intrusion, an attacker can send arbitrary commands to the vehicle. These commands can potentially cause damage to the vehicle or injury to people, e.g. triggering the airbag in the middle of the highway.
- Disclosure of Information: By executing a successful intrusion, an attacker may gain access to private customer information as well as proprietary software.

Communication Link

- Traffic manipulation: Increased exposure brought on by the added wireless capabilities results in a higher risk of an attacker injecting, modifying or replaying messages sent over the communication link. This gives the attacker the possibility to change the instructions sent to the vehicle, and e.g. telling the vehicle to unlock the door rather than checking if the door was locked. It is also possible for an attacker to eavesdrop on the traffic and to extract privacy-related data from the link.
- Denial of Service: Successful DoS attacks can result in software updates failing or diagnostics reporting erroneous results. This can potentially cause damage to the vehicle or injury to a person.

Vehicle

- Impersonation: By impersonating a vehicle, an attacker can set up a connection to the portal which could lead to the attacker gaining access to confidential data and proprietary software that the portal intended to send to the impersonated vehicle.
- Intrusion and Denial of Service: A successful intrusion could potentially allow an outsider attacker gain equal access to that of an insider, thus providing access to sensitive and proprietary data as well as the possibility of executing more advanced attacks. DoS attacks against e.g. the wireless interface could prevent the vehicle from upgrading software.
- Execution of Arbitrary Code: By either impersonation or a successful intrusion, an attacker can update the ECUs in the vehicle with modified software, thus allowing the attacker to control functionality of the ECUs.

- Disclosure of Information: By attacking a vehicle physically (by e.g. differential power analysis), it may be possible to extract, among other things, authentication and encryption keys stored in the ECUs. These keys can then be used for impersonation or eavesdropping on encrypted communication.

3.1.5 Recommendations

For each risk detailed in 3.1.4, Nilsson defines a corresponding security requirement as well as describing the achieved security each requirement brings.

Portal

- Preventing impersonation: Certificates should be used to establish identity and the public key of the portal should be installed in the vehicles during manufacturing. There should also be a method to revoke certificates, should these become compromised. To prevent spoofing and modification of data sent by the portal, it should e.g. be signed using the portal's private key, which allows vehicles to verify the signature using the already installed corresponding public key. This prevents an attacker from impersonating the portal to set up a communication link to a vehicle.

- Protection against intrusion and Denial of Service: The portal should follow *best practices* for firewalls, Intrusion Detection Systems (IDSs), logging and DoS attack prevention. A firewall and an IDS can prevent or alert on intrusion attempts and logging can give valuable forensic evidence after a successful intrusion. Protection against DoS attacks ensures the availability of the portal.

- Preventing execution of Arbitrary Code: Some commands are dangerous when executed remotely, such as triggering the airbag. These commands need to be well-defined before deployment. In order to prevent the execution of such commands, they should be filtered using e.g. a blacklist to make sure that neither diagnostic requests nor software contain them. Since such commands should be available when using a wired connection, simply removing them altogether is not an option. This will prevent the generation of diagnostic requests or software containing dangerous commands in the portal, and hence these commands will not be executed by the vehicles ECUs.

- Secure Storage and Communication: Proprietary software binaries and private customer information should be encrypted by the portal using a strong symmetric cipher, such as AES[1]. Access should be restricted to require proper authentication and authorization. The portal's internal network should be protected using e.g. TLS[2]. This prevents outsider attackers from accessing sensitive data as well as from injecting or altering messages.

Communication Link

- Secure End-to-End Communication: Traffic between the portal and the vehicle should be encrypted and protected against modification by using e.g. TLS. Because of the limited hardware in the vehicle, it is necessary to carefully choose suitable cryptographic algorithms (see Section 3.2). Timestamps should be used in order to guarantee freshness of diagnostic requests and software updates. This prevents eavesdropping, injection, modification and replay of messages.

- Protection against Denial of Service: The protocol must be resistant to packet loss, which can occur due to either communication problems (such as bad reception) or intentional attacks. By using a reliable protocol which can also handle low bandwidth communication and long delays, we protect the availability of the link.

Vehicle

- Preventing impersonation: The vehicle should possess something unique to establish its identity, such as a certificate. These certificates should be installed during manufacturing. This prevents both insider and outsider attackers from impersonating a vehicle.

- Protection against intrusion and Denial of Service: A firewall and an IDS should be installed in order to block certain incoming traffic and to detect and possibly prevent unauthorized access and raise alerts on intrusion attempts. Trace and network logging should also be enabled. This, together with proper DoS protection, prevents or alerts on intrusion attempts and prevents availability attacks. Log data can be used in forensics and to prevent future attacks.

[1] *Advanced Encryption Standard*, an encryption standard adopted by the U.S. government.
[2] *Transport Layer Security*, a successor to SSL.

- Preventing Execution of Arbitrary Code: The vehicle must verify the integrity of received software and diagnostic requests to ensure that they have not been altered. ECUs should only allow updates of signed software and also filter dangerous requests from being executed remotely. This prevents an outsider attacker from generating and modifying software and diagnostic requests as well as preventing an attacker from executing dangerous commands in the ECUs.
- Secure Storage and Communication: In order to store sensitive data, such as encryption and authentication keys, the vehicle should use a *tamper-resistant storage* (see Section 3.3). A trusted platform module should be used. Data traffic on the internal vehicle network should be protected with regards to confidentiality, integrity and freshness. This prevents an attacker from eavesdropping, replaying, injecting and modifying messages sent on the internal vehicle network, as well as preventing extraction of stored data by physically connecting to the ECUs.

3.1.6 Security policies

Wired diagnostics and software updates are typically performed in closed and controlled environments, such as in an authorized service station. Because of this, there has not been much need for security policies. However, the move towards wireless diagnostics and software updates requires the definition of a set of security policies. For actions where several parties are involved (e.g. portal, service station and vehicle owner), it is important to define who is allowed to do what. Here follows a list of examples from [5]:

- Only the portal is allowed to create and sign software.
- The vehicle must verify the authenticity of the received software to verify that it originated from the portal.
- The portal and service stations are allowed to send diagnostic requests.
- The vehicle owner should be able to access the diagnostics results via the portal.

Also, there is a need to define policies for the ECUs as well. In order to do this properly, the ECUs should be classified into safety-critical classes. Policies should be defined for these classes as well as for the ECUs themselves. In [5], Nilsson gives (among others) the following examples:

- A time limit, e.g., 30 minutes, for updating the software on the same ECU should be used.
- Only ECUs that do not affect the maneuverability of the vehicle should be allowed to be updated over-the-air.
- A time limit, e.g., 1 minute, for responding to repeated diagnostics requests should be used.
- Only non-safety critical diagnostics requests are allowed to be sent to safety-critical ECUs.

3.1.7 Trust Model

In [6], Nilsson et al. develops a trust model to identify weaknesses in the communication infrastructure in order to identify security risks. Here follows a non-comprehensive list taken from that article:

- Administrator trust: The administrator should be qualified to perform administration and operation of the applications and system in the portal, as well as perform administration of the applications and system in the vehicle. The administration should therefore know how to perform firmware updates and diagnostics.
- Connection trust: The vehicles connect to the portal in a way that will not compromise the portal or other vehicles connected to it.
- User trust: The user (here meaning the person associated with the vehicle) should not perform operations of the vehicle, such as firmware updates or try to extract credentials from the vehicle.

Administrator trust

Risk: The administrator may not be qualified to operate the portal.
Exploit: Intentional or unintentional actions by the administrator can cause the portal to crash, expose the internal network to outside attackers or issue malicious firmware updates and diagnostics requests which can affect the operation of vehicles connected to the portal.
Mitigation: Ensure that the administrator has received proper training. Reduce trust in administrator by establishing policies and privilege levels for administrative tasks, e.g. require higher privileges for performing firmware updates than for reading diagnostic data from vehicles. Log transactions in order to find and rectify problems caused by inappropriate actions.

Connection trust

Risk: Trust in the connection between the portal and the vehicles could allow an attacker to eavesdrop or modify sensitive data in the communication link as well as injecting (malicious) messages.
Exploit: Traffic over the communication link can be sniffed by an attacker at any point where data passes through the network, e.g. near the vehicle, at a radio tower or on the Internet. Unencrypted traffic can easily be read, which compromises the confidentiality of sensitive data. An attacker could also modify or inject messages, thus altering the instructions to and responses from the vehicle.
Mitigation: Only allow the portal to initiate connections and only allow communication to the vehicle explicitly from the portal. Encrypt the communication between portal and vehicle using session keys and require mutual authentication using link keys for connections between the portal and the vehicles.

User trust

Risk: The user can extract sensitive data from the vehicle, such as cryptographic keys or other sensitive data, as well as updating the firmware in the vehicle.
Exploit: An attacker can extract link keys for impersonation attacks from the vehicle or install performance-enhanced firmware or malicious firmware in the vehicle.
Mitigation: Reduce trust by enforcing a policy of user responsibility to prevent extraction of data. Require proper authentication and authorization for firmware updates to ensure that only the portal can perform the update. Minimize user involvement when establishing communication with the portal.

3.1.8 Design principles and recommendations

Nilsson et al. draws parallels between hierarchical wireless vehicular networks and ZigBee wireless networks, noting that a number of security design principles valid for ZigBee networks [3] are also valid for vehicular networks:

Use Defense-in-depth

By implementing multiple security measures, we ensure that if an attacker is able to circumvent one security measure, additional measures and levels of security still protect the target (in this case the portal and the vehicles). This can be done by separating and isolating the wireless communication link from the internal portal and in-vehicle network, using strong authentication methods and secure end-to-end communication, filtering traffic from unauthorized vehicles and implement monitoring and intrusion detection in the internal networks.

Harden all components in the infrastructure

Divide the infrastructure into components and harden each component to improve security. Each component should be analyzed to discover ways to harden it against attacks and configuration failures. If credentials are shared between applications in the vehicle, we achieve device-level, but not application-level, authentication and security. If application-level security is necessary, the trust in the vehicle must be reduced so that each application trusts only an isolated part of the vehicle, by preventing the credentials from being shared between applications and the applications from trusting each other.

Separate communication link from the wired networks

If possible, make sure that there is no direct connection between the internal network (both in the portal and the vehicle) and the communication link, to prevent an attacker from gaining access to the internal network by compromising the communication link. Use e.g. a firewall or a security gateway to separate the networks. This isolates, segments and controls traffic between the different networks.

Use security features at all levels of the protocol stack

MAC level security should be used for the wired and wireless communication, encryption, authentication and integrity should be utilized in the network and application layers. Enable security features at all levels of the protocol stack.

Maximize the protection of the most valuable asset

In this type of centralized architecture, the portal is the most valuable asset. If the portal is compromised it would undermine the whole trust model, since the portal is responsible for key management and plays a major role in the security architecture, and all vehicles trust the portal. Therefore, to maximize the security of the portal is paramount.

3.1.9 Practical issues

When creating an infrastructure for wireless communication in vehicles, there are some practical issues that need extra consideration. Nilsson et al. divides these into availability issues and general security issues.

Interference

Electromagnetic noise generated by e.g. motors, pumps and fans in the vehicle can interfere with the operation of the wireless communication, thus affecting the transmission signal negatively. This can cause performance and reliability problems for the communication link and extra care is needed when developing the system. The effect of the interference could be reduced by selecting the channel within the current frequency band that is most resilient against electromagnetic interference or by selecting another frequency band. Another method would be to boost the signal by using a higher-gain antenna.

Reliability

The quality and reliability of wireless communications is affected by environmental factors, such as tunnels and tall buildings which block the signal or objects in the transmission path that cause reflections. It may prove to be a challenge to achieve reliable communication because of these issues. We are also forced to consider routing problems for the traffic sent over the Internet and denial-of-service attacks against gateways and the portal.

Security

By following the mentioned recommendations and design principles, many, but not all, of the risks against the system can be mitigated. The security is heavily dependent on available security controls and the practices followed

by the administrators and the users. Since a wireless communication cannot be restricted and controlled in the same extent as a wired link, the signals reflect against objects which enables attackers to silently sniff network traffic. Traffic sent over the Internet is susceptible to the same risks and attackers could gain access to the internal networks through the wireless gateway. These issues present challenges that must be considered.

3.2 The use of strong cryptography in embedded systems

In [2], Levi evaluates the performance of RSA[3] and Elliptic Curve Cryptography (ECC) in WTLS[4]. The conclusion is that stronger RSA keys (2048 and 3072 bit) should not be used due to poor performance. However, stronger ECC curves than those proposed in the WTLS standard performed within acceptable limits and can therefore be used in high-security applications. In [1], Argyroudis et al. shows that SSL and S/MIME introduces a significant overhead, but within acceptable limits, and that the use of IPsec is plausible when calculating the cost of running the cryptographic operations. It should be noted that IPsec was never implemented in the handheld environment, and the results are based on calculations. These articles are from 2003 and 2004 respectively, and there has been more progress in the field since then. We can conclude that the use of strong cryptography is plausible in computationally weak devices, such as embedded units in vehicles. This is good news, since reliable cryptography is a prerequisite when trying to secure information.

3.3 Tamper-Proof Devices and Trusted Platform Modules

In [8], Raya and Hubaux discuss the pros and cons of using Tamper-Proof Devices (TPDs) for securing Vehicular Ad-hoc Networks (VANETs). TPDs are storage devices that are designed to prevent an attacker with physical access to the device from accessing stored information. In order to be effective, there are some requirements the TPDs should meet. They should have their own battery, which should be able to recharge itself using the vehicle. They should have their own clock, which should be securely resynchro-

[3] An algorithm for public-key encryption

[4] *Wireless TLS*, a part of the *Wireless Application Protocol*

nized when e.g. passing a trusted roadside base station. Access should be restricted to trusted people, e.g. authorized service stations. The TPD should detect hardware tampering and, if such an event occurs, erase cryptographic keys stored on the device to avoid compromised keys.

However, there are problems with TPDs. They are too sensitive for road conditions, since they can not handle e.g. light shocks or extreme temperatures. They are also very expensive, they can cost several thousand dollars. An alternative to TPDs could be Trusted Platform Module (TPM), which can resist software attacks but not sophisticated hardware tampering. As we have seen in Section 3.1.5, Nilsson suggests using a TPM for secure storage.

Neither TPDs nor TPMs are completely secure. TPMs, for instance, are susceptible to *cold boot attacks* where cryptographic keys can be read from memory after a cold reboot. The use of tamper-resistant storage is only one part of an overall defence-in-depth strategy.

There will be a need to weigh security against cost and practicality, and this will affect how the unit used in SIGYN will be designed.

3.4 Threats

There are threats against all computer systems, especially against a big commercial system like SIGYN. There are a lot of things to gain in compromising this system, especially since you can potentially compromise all Volvo vehicles at the same time. From the simple prestige of having managed an intrusion into a major corporation's system, via obtaining valuable information (e.g. proprietary binaries and driving characteristics of vehicle owners) to theft of vehicles and bodily harm. We will now discuss some of the potential attackers and their goals.

3.4.1 Possible attackers

- Hackers: For the classical hacker who does it for the challenge, a new type of system is likely a desirable target. Hacking a new type of system presents new challenges and, if you are successful, something to brag about. The motivation here is probably related to that of an explorer or a mountaineer; being the first to succeed with a difficult task will generate reputation and status.

- Competing corporations: There is a lot of useful information that could be gathered through an attack, e.g. Volvo's software solutions

in the car, update and diagnostics procedures, security solutions, common errors in cars and information about owners of Volvo cars, like driving characteristics. However there are other ways of getting hold of the software in the vehicles, and competitors already extract software from each others' cars. Perhaps also other companies, like e.g. advertising firms, would have an interest in statistical information about car owners' behaviour, geographical location etc.

- Vehicle owners: An authorized user might want to install other software in the car to get new or improved features. This could be done either by the car owner or by a company offering this kind of "updates". Many of these updates might work perfectly, without any negative side-effects. However, since Volvo has no control over the code, it is impossible to guarantee that it is both compatible with SIGYN and that it is secure. If customized code is allowed, the need to strengthen the portal increases considerably.
- Thieves: Being able to control a vehicle remotely through a wireless network gives new possibilities when it comes to stealing cars. You could e.g. unlock the doors, shut off the alarm and the tracking system, and perhaps even drive the car away remotely.
- Abusive users: A system that lets you both change the software in vehicles and control them directly gives rise to a very serious risk, namely that of bodily harm. Terrorists, maniacs, nations at war and others could, if they compromise the system, cause injury and/or death to drivers, pedestrians and others.

3.4.2 Consequences of compromising the different parts of the system

Compromising the portal leads to a risk for the whole vehicle population. The portal sends out all software updates, diagnostics requests etc. and can send data to all vehicles.

Compromising a vehicle does not mean a direct risk against other vehicles but means that you can e.g., send arbitrary code to the portal which might affect it, unless the portal is configured to withstand malicious data sent from authorized users. But even if compromising a vehicle only means you control that single vehicle the consequences might be very serious since it means a direct threat against peoples' lives.

A Denial-of-Service (DoS) attack against the portal or against a vehicle's network connection is not as serious as DoS-attacks in many other systems

since neither software updates or diagnostics are time critical. As long as they will eventually be carried out it is not a significant problem if they are delayed for a short period of time. A DoS-attack against the internal bus in the vehicle might have disastrous results if it prevents the ECUs that control the vehicle to function properly.

Chapter 4

Related areas

When developing any type of system, we can learn from studying other, related fields. Much of the research we came across concerned another use of wireless capabilities in road vehicles, namely vehicular ad hoc networks (VANETs). In this chapter, we will discuss that as well as ZigBee networks, a type of mesh network where many of the methods to increase security also can be applied to systems like SIGYN.

4.1 Vehicular ad hoc networks

VANETs are based on communication between vehicles and roadside base stations (both vehicle-to-vehicle, vehicle-to-base station). The idea is that the vehicles can pass information such as environmental conditions, accident alerts and other information between them. The networks might also be used for proximity alerts, used to avoid collisions between cars. The system is characterized by high mobility (moving vehicles), high speeds and short connection times (crossing vehicles) and the networks are created "on the fly". The real-time aspect is important. Even though VANETs will contain millions of nodes, the communication will be mainly local. The network will thus be partitioned and made scalable.

VANETs are interesting because they also deal with road vehicles with wireless capabilities. The form of communication differs, but there are still some similarities. For instance, both VANETs and systems for remote diagnostics such as SIGYN can use pre-loaded certificates. Thus we may benefit from studying VANETs when deciding e.g. how to handle certificate revocation. In [8], Raya et al. describes one method of certificate revocation where the *Certificate Authority* sends a revocation messages to the vehicle with the compromised keys. When the tamper-proof device receives this

message it erases all cryptographic keys and stops signing new messages.

Another issue which concern both types of system is protection against Denial-of-Service (DoS) attacks. Raya et al. suggests switching between different channels or communication technologies in order to mitigate DoS attacks, and the same strategy could be used in systems for remote diagnostics, depending on what wireless capabilities have been implemented in the vehicles.

4.2 ZigBee networks

ZigBee is a Low-Rate Wireless Personal Area Networks (LR-WPAN) based on the IEEE 802.15.4 standard. It is a system for ad-hoc networks between low-power devices and thus differs a lot from the SIGYN system. In [3], Masica discusses security challenges and best practices when building ZigBee networks. Despite the difference between the two technologies, many of these are applicable also to a system such as SIGYN.

A ZigBee network consists of a number of nodes, with either a star or a peer-to-peer topology. The nodes are divided into two groups, Reduced Function Devices (RFD) and Full Function Devices (FFD). The difference between these two are that the latter can be elected to do certain special tasks that the network might require (e.g. routing) while the former are exclusively leaf nodes. The special tasks (or special nodes) are divided into the following categories:

- Coordinator - responsible for the creation of the network
- Router - routes packets between nodes
- Trust Centre - a node trusted by all the other nodes in the network and which is responsible for certain security tasks
- Gateway - connecting the ZigBee network to other networks

Many of the security principles and mechanisms that Masica outlines are similar to those that we saw in Section 3.1, such as using a Defense-in-Depth approach, hardening all components, maximizing the protection of the most valuable asset and using out-of-band key loading when possible.

Masica also discusses special implications of using LR-WPANs in industrial environments, such as electro magnetic interference (EMI) from machinery, something that is very much an issue in vehicular environments as well. The effect of EMI can be reduced by carefully selecting which

frequency band and which channel to transmit on, by boosting the signal strength and so on.

Chapter 5

Development of security mechanisms in a wireless vehicular environment

In this chapter we describe how we carried out our work, from gathering the necessary background information through design specification and the development of the reference client to building up a lab environment and implementation of security mechanisms.

5.1 Studies and analysis

We have studied SIGYN's documentation and the VTTP[1] protocol specification as well as academic research on remote diagnostics and related areas (see Chapter 3). The most important research we came across was that of Dennis Nilsson, who wrote a number of articles concerning remote diagnostics and software updates while working as a Ph.D. student on the SIGYN project. These articles, however, are quite general and deal more with the technology as a whole rather than a specific implementation (such as SIGYN). Nevertheless, they provided us with an excellent foundation when analysing SIGYN.

5.1.1 SIGYN explained

SIGYN uses MQTT[2] (MQ Telemetry Transport), which is a publish/subscribe protocol developed by IBM. The portal is both the publisher and the

[1] *Vehicle Task Telematics Protocol*, a proprietary protocol developed by Volvo
[2] http://www.mqtt.org

message broker, and the vehicle is the subscriber (see Figure 3.1 and Figure 5.1 for reference). The role of the message broker is to act as an intermediary, enabling decoupling between publisher and subscriber. MQTT flows over TCP/IP and is aimed at remote sensors and control devices in low bandwidth communication.

The backbone of SIGYN is the Vehicle Task Telematics Protocol (VTTP), which is sent over MQTT. VTTP can be said to be divided into two parts, namely control messages and file transfer messages. Control messages include acknowledgement, connect and disconnect and most importantly *Vehicle Task* and *Vehicle Task Result*. A Vehicle Task consists of instructions that should be carried out by the vehicle. It is also possible to bundle files such as software binaries or scripts with the Vehicle Task. These files will then be downloaded by the vehicle and executed according to the instructions. A Vehicle Task Result is the response from the vehicle which informs the portal of the result of the executed instructions. The vehicle can also send files with e.g. diagnostic data together with the Vehicle Task Result. File transfer messages include acknowledgement, file request and so on. The file transfer uses fragmentation to split the files into smaller pieces, which allows retransmission of small chunks instead of whole files.

5.2 Design and development of a reference client

The first major part of our thesis work was to develop a reference client, which we would later use when we worked with the security aspects of SIGYN.

5.2.1 Designing the client

The design requirements for our reference client were as follows:

- The ability to communicate using MQTT over TCP
- The ability to support a subset of VTTP for communication with the portal
- The ability to send and receive files
- The ability to simulate multiple vehicles
- The ability to easily change settings, e.g. using a configuration file
- The ability to choose host and port to connect to

- The ability to select which files to transmit together with a Vehicle Task Result
- The ability to run on different platforms

The idea of the program is to simulate a vehicle (or a number of vehicles). It should be able to download Vehicle Tasks, "execute" them and send back a Vehicle Task Result to the portal.

Previous implementations of reference clients developed by Volvo required special hardware to run on, but this time there was a request that the client should be made platform independent. The language of choice was Java, since it is platform independent and since there were existing libraries we could use to speed up the development.

5.2.2 Developing the client

We had access to three Java libraries (written by IBM and Volvo), one for talking MQTT, one for talking VTTP and one for sending and receiving files. We started off by writing a very simple program that could communicate with the message broker in the portal using MQTT. When we had a working client that could connect to the broker and subscribe to different topics, we turned our attention towards modifying the file transfer code instead. The challenge here was to get the code to work together with the message broker, and after some changes we were satisfied. Now we had the two cornerstones of our reference client. The next step was to combine these two and make sure the end result followed the VTTP protocol specification, making sure our client sent the correct responses to the portal.

5.2.3 The finished client

The goal was to write a client that outward would behave just like a vehicle. The finished client does this, with one exception. The real vehicle will only stay connected to the portal when there are Vehicle Tasks to download and Vehicle Task Results to transmit. The portal is responsible for informing the vehicle when there are new instructions to collect. Our client stays connected until it is explicitly told to disconnect.

When you tell the client to connect, it is possible to specify broker address and port as well as the number of vehicles to simulate. We then start one thread for each car, which then connects to the broker. Each thread waits for messages from the broker and downloads the Vehicle Task and any additional files according to the VTTP specification. If we simulate a

single car, the user is prompted to select files to transmit back to the portal together with a Vehicle Task Result. If we simulate multiple cars, a list of files to transmit is read from the configuration file. This is done for each Vehicle Task. If there are no more Vehicle Tasks to download, the thread simply waits until a new Vehicle Task is available or until the user tells the client to disconnect from the portal.

Configuration file

We use a configuration file to set some of the values in the program. The file is a text file that resides in the same directory as the program file and it can be edited by the user using a text editor. The configuration file specified client id, properties like number of retries, fragment size etc., directories for files to upload and download, options for logging and the amount of information given to the user. It can specify a default portal address and port (these can still be changed in the program, but the default is already filled in in the input box when you connect). As we saw in the previous paragraph, it is also possible to specify what files to upload when you simulate multiple cars.

5.3 Setting up a lab environment

When we had a finished reference client, the next step was to set up a lab environment where we could work without disturbing other users and where we had control over the different parts. When working with our client, we had tested it against a portal which was part of a Volvo development environment at another location in Gothenburg. Since we could not get access to install software on that machine, we wanted to run a local copy of the portal. However, this was easier said that done. We received a VMWare image with the broker software which we tried running on our desktop computers at Volvo IT. No matter what we tried, we were unable to connect to the portal. After a significant time of searching for a possible cause, we finally realised that it was neither the broker nor our client that was causing the problem. The firewall software Volvo IT had on all machines were blocking the incoming connections. In hindsight, this should have been one of the first things to check. We then realised that neither of us had the administrative privileges needed to change the firewall settings or to turn it off.

We then decided to bring our laptops from home and run the VMWare image on one of them. We hooked them up to a separate LAN using a

router borrowed from Diadrom, since we were not allowed to plug them into the company network. When running the VMWare image, we realised that neither of our laptops were powerful enough, and the end result was unacceptably slow. We upgraded the RAM and after that the image ran smoothly. However, the client was still unable to connect to the broker. The explanation for this was that the broker software on the image was an older, unfinished version. We received a newer image, but this was too large to fit on the laptop's hard drive, and trying to run if from an external hard drive resulted in lousy performance. Per Dahlberg then lent us a desktop computer which we did a clean install of Ubuntu on and after that we were able to run the VMWare image and connect to the broker without problems.

5.4 Implementing security measures

We studied the research by Nilsson (see Section 3.1) and examined which of the suggested security recommendations were applicable to SIGYN and which of these were possible to implement. The security measures we decided to implement were the use of a Intrusion Detection System (IDS), logging and a firewall in the portal and the "vehicle" and a VPN tunnel between portal and "vehicle". We chose to implement IDS, firewall and logging in the "vehicle" even though the real vehicles will have fundamentally different hardware and would not be able to run the same software. Since the implementation of these things in the vehicle does not really say anything about what will work in a real vehicle we have also looked up some different software that can be run on embedded systems. In the end, what decided which security measures we would try to implement was how our lab environment was set up, which hardware we had to our disposal and so on. We felt that the limitations would severely affect our work, but rather than refrain from implementing anything at all, we decided to implement what we could.

5.4.1 Set up

The portal software we used consists of message broker and message queue. The message queue was run on a laptop (number 8 in Figure 5.1) using VMWare, the broker was run on a desktop computer also using VMWare (7). On this computer, but on a different virtual machine, we also ran a server for the OSSEC IDS (6). And on the same virtual machine as the broker we also ran an OSSEC agent, Snort IDS and we used Windows' built-in logging, the Event Viewer. Both these computers were plugged into a

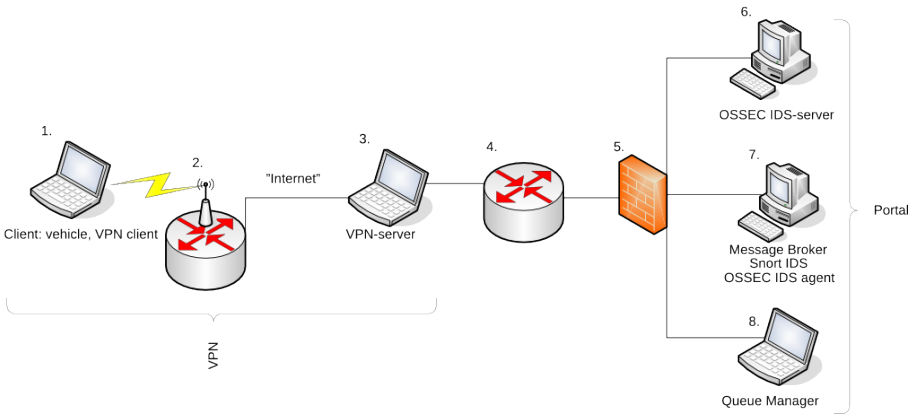

Figure 5.1: Lab setup

router (5) with a built-in firewall which protected them from the outside. The "Internet connection" of this router was plugged into another router (4) to which also a computer (3) running the Columbitech VPN server was connected. This computer had two network cards and the other one was connected to another router (2) to which we had also plugged the "vehicle" (1), which ran our SIGYN client, the Columbitech VPN client, Snort IDS, OSSEC IDS, a ZoneAlarm firewall and Windows logging. This last router represented the Internet in our set up. The client was connected to the VPN server through a VPN tunnel over "the Internet". The VPN server was connected to the portal on its other network interface and so the client had a VPN tunnel to the portal.

The computer running the message queue does not have any IDS, this is because we did not have access to install any software on it.

5.4.2 Software

In this section, we explain which software we chose and why. In some cases, we also list some possible alternatives. When deciding on how we would implement a particular security solution, we looked mainly at free and open source software in order to avoid licence issues and cost. One deviation from this is the VPN software from Columbitech. Volvo IT had previously looked at this, so we could obtain a trial licence and got valuable support from Karl Hagås at Columbitech.

Intrusion Detection Systems

We used OSSEC HIDS and Snort, both of which are open source intrusion detection systems. OSSEC HIDS performs log analysis (which makes it ideal to use in combination with other software like e.g. Snort), file integrity checking, rootkit detection, time-based alerting and active response to attacks on a host (e.g. it can use the system's firewall to dynamically block intruders). It uses a server/agent architecture where agents on the systems monitored are connected to a central server. It has user definable rules which makes it very configurable and dynamic, since a user can write specific rules on what to alert on. Snort is the most popular open source IDS and has all of the capabilities that commercial IDSs have. It performs packet logging and real-time traffic analysis. It does protocol analysis, content searching/matching and blocks or detects a variety of attacks.

Snort and OSSEC were chosen because they are two of the more popular open source IDSs around. They also work well together.

Firewalls

For protecting the portal, we used a Netgear® WNR834B running DD-WRT[3], which is a Linux-based firmware with iptables support. This router was chosen simply because we could borrow one from Per Dahlberg. Another solution would be to put a firewall distro such as m0n0wall[4] or Smoothwall[5] on a dedicated machine and filter all traffic through it. When looking for possible solutions for the vehicle, we have studied firewall distros aimed at embedded systems (such as m0n0wall) as well as Linux-based firmware (such as DD-WRT or OpenWRT[6]). In the end, we settled for the firewall already installed on the laptop than ran the client, which was ZoneAlarm.

Logging

We settled on using the operating system's built-in logging capabilities. This in combination with the IDS and firewall logs should provide sufficient information. It should be noted that the logs need to be protected from tampering, otherwise they will become useless.

[3] http://www.dd-wrt.com
[4] http://m0n0.ch/wall
[5] http://www.smoothwall.org
[6] http://www.openwrt.org

Secure communication

We chose a VPN solution by Columbitech, which aims to be completely mobile, letting the user seamlessly switch between different types of networks, such as Bluetooth, WiFi and GPRS. As we mentioned previously, Volvo IT already had a trial license and had previously done some tests using this software. A similar solution called Lotus Mobile Connect is developed by IBM, and it would have been interesting to compare these two solutions. However, we did not have any opportunity to do that.

Chapter 6

Result

In this chapter we discuss our findings, what their implications are and how these affect systems for remote wireless diagnostics and software updates. We will also discuss the pros and cons of such systems in a security context.

6.1 Findings

We have shown that it is possible to make a Java client for the SIGYN system. It does not really execute any of the code sent from the portal (the portal does not send any code to execute yet either) but the communication part works as it is supposed to. Since Java works on a lot of different systems we have managed to make a client that is more or less platform independent, given that there exists a Java Virtual Machine for the system in question. Running Java code generally requires more powerful hardware than running pure C or assembler programs and this is something that needs to be taken into consideration when designing the hardware that will be used in SIGYN.

Since SIGYN uses a secret proprietary protocol, we cannot discuss it in any detail. There is, however, one issue we believe to be extremely important. In the current version of the protocol, the portal should automatically accept any file transfers from the vehicle. Because of this, the portal *must* take great care if these files are to be executed in order to prevent attackers from running malicious code on the portal.

We have implemented some of the suggested security measures but this does not really say much. On the client side we have used hardware that is very different from the hardware that will be used in the car, so what we could implement on the client side is no real indication about what will

actually work in a real car. When it comes to the portal it is not different from most other servers and can be secured through standard measures, something that was clear from the beginning.

However, studying literature on this subject we have found that there are both cryptographic algorithms and software that enables the implementation of desired security mechanisms (the ones suggested by Dennis Nilsson) on embedded systems. Since we were restricted to standard PC hardware, we could not test this in a satisfying way. Nevertheless, we believe that these security mechanisms can be implemented without much difficulty in the restricted hardware in the vehicles.

When we used our client together with the portal everything ran very slowly. This might have serious implications for the SIGYN systems, regardless of whether it was the client or the portal that slowed things down. If the client was to blame then it might be problematic to run the client software on lightweight, embedded hardware. And if the portal was the problem we might get scalability problems. On the one hand though, we ran the broker on a PC that is a lot "weaker" than the future portal hardware will be, but on the other hand our implementation ran slowly with just one car connected, the real portal will have large amounts of cars connected at the same time.

6.2 Discussion

In this section we will discuss what SIGYN from other large computer networks, the benefits of such a system and who it is aimed at.

6.2.1 Never trust your users

Are the measures we discuss in this thesis enough to achieve acceptable security? We have not been able to do any penetration tests or a thorough analysis and therefore cannot provide any proof. However, one thing that was missing in the texts we have read is that the portal needs to be vigilant about what the vehicle sends to the portal in order to avoid executing malicious code. The academic research is often held at a general level, not focusing on any particular system. This means that it is not enough to simply implement the suggested security measures, since these may overlook issues specific to the system in question. Acceptable security can only be achieved through a thorough analysis and a fundamental understanding of every component in the system.

Anyone with enough money to buy a vehicle can become an authorized user of the system. In other words, an outside attacker can, with relative ease, become an insider. The attacker will then also have physical access to the vehicle, and the range of possible attacks increases even more.

We should assume that anyone with physical access to a vehicle would be able to perform any action that an authorized service station can perform. Although not necessarily feasible for all types of attackers (based on economic strength), this risk cannot simply be ignored. We cannot secure resources outside of our control, we can only try to limit the results of an attack on these resources. Because of this, we should not only secure the portal against outside attacks, but also against misuse of authorized vehicles. That is, we should make sure that a vehicle cannot cause the portal to fail, no matter whether it is controlled by an authorized user wanting to improve the vehicles performance or a malicious attacker who uses the vehicle to launch attacks against the portal. This is a trust issue: we should simply not trust the vehicle enough to allow it to jeopardize the portal.

6.2.2 Benefits vs. cost

Given the difficulty in securing a system for remote diagnostics and software updates, do the benefits still outweigh the additional costs? For the private end user, not having to visit a workshop in order to perform software updates is a rather limited benefit, but a benefit nonetheless. Most updates could just as easily have been performed during yearly checkups. The same goes for remote diagnostics, since personal vehicles seldom have such a high demand for availability. Having your car warn you before you become stranded on the highway is, of course, a convincing selling point, but here we believe that the cost for these additional features will be what tips the scale. For companies with a large vehicle population, however, these benefits are no longer limited. They can instead be what prevents the loss of significant amounts of money due to vehicle "downtime".

One benefit of computerised diagnostics is that you might find an error in the vehicle before it becomes a problem. Instead of discovering that something is wrong with your car when it breaks down on the highway the diagnostics system might be able to alert you that something is wrong before it has any real implications and you might be able to repair it before it becomes a problem. But this does not require a remote connection if the ECUs in the car are able to analyse the data and understand if something is wrong. To be able to do this analysis in the car might mean an increased hardware cost to get enough computing power. But on the other hand, if

the analysis is to be done somewhere else that computing power needs to exist there which will also cost money. However this hardware might be cheaper than the specialised hardware in the cars.

6.2.3 A system aimed at companies rather than consumers

The main benefits do not seem to be for the people who will drive the vehicles but for the vehicle manufacturers, workshops and companies that use a lot of vehicles, and the benefits seem to be mostly organisational and financial. Doing different tasks, such as software updates, diagnostics and information gathering, remotely requires less work than doing them through a wired connection. This saves money. Doing software updates remotely means that vehicle manufacturers do not have to take up space and work power in their workshops for this or pay external workshops to do it. The external workshops will loose money on this but on the other hand they will require less time for repairs when the diagnostics are already done when the vehicle arrives. Faster repairs are a benefit for customers too. Gathering information about e.g. error trends and driving behaviour remotely will save work and money both for vehicle manufacturers and companies that use a lot of vehicles and want to keep track of their vehicles and employees.

Wireless access simplifies gathering information which can be used to e.g. track error trends or driving behaviour, which vehicle manufacturers can then use to improve vehicle design. Transportation companies can use the information e.g. to supervise their employees, or better plan their activity. Drivers on the other hand might not be as thrilled about how easy it is to find out private information about them.

Chapter 7

Conclusion

Security is an important feature of any computer system, but in the case of remote diagnostics and software updates, neglecting security can lead to not only financial damage, but also injury to person or even death. As we showed in Section 3.4, a number of different entities have interest in attacking the system, each with different motives and each with different goals. Even though the worst-case scenarios are less likely, they cannot be ignored. When developing a system such as SIGYN, great thought must be put into making it secure enough. It is impossible to achieve total security in practice, so a trade-off must always be made.

In light of this it might be in place with a discussion about whether this type of system is desirable. All the things that the system does (software updates, diagnostics, surveillance of employees etc.) can already be done without a wireless connection, just not in "real-time". There are probably already structures in place to make software updates in workshops and you can do computer diagnostics using a wired connection, surveillance can be done using e.g. a black box or something similar, etcetera.

7.1 Issues to consider

There are benefits both for vehicle manufacturers and their costumers (both companies with a whole fleet of cars and individuals with a single private car) in a system of this sort. At the same time, the system could be used for e.g. surveillance, which would violate the personal integrity of the drivers.

There are also some things that should be considered before implementing a system like SIGYN. Increased computerization and connectivity means increased risks. It is impossible to achieve total security and the more entry points there are, the bigger the risk of compromise. Using wire-

less networks and sending commands over the Internet to control vehicles makes it much easier to attack them and the more things that are controlled by computers in the vehicles the more things a successful attacker will have access to. It has always been possible to sabotage cars but the possibility of doing it to a lot of cars at the same time and from the comfort of one's own home or another secure location increases the risk that it will happen. The huge amount of information obtainable by compromising a vehicle is something that computerization has brought with it. This also increases the incentive to attack vehicles. It cannot be stressed enough that anyone developing such a system must put a lot of thought into limiting the damage that could be made through remote access, such as isolating critical components that should not be updated remotely.

For the manufacturer, it is important to consider the customers' reactions when you manipulate their vehicles outside of their control and collect information about them. Some people might feel uncomfortable about the fact that the vehicle manufacturer has more control of their own vehicle than they have, even after they have bought it. And with all the information that is possible to extract about vehicle owners and their behaviour it is important that customers feel confident that manufacturers will both be able to protect this information from attackers and not use it in a questionable way themselves.

7.2 Future work

We wanted to make a platform independent client, which is why it was written in Java, but so far we have only tested it on one platform, a PC running Windows XP. We discussed trying it out on a PDA or other mobile device, but could not get hold of one. There are a lot of different systems that can run Java applications and it might be interesting to try out the client on some others, especially a mobile one.

When we connected our client to the message broker everything went very slowly. We have not evaluated this and so do not know what the cause is. We do not know if the client or the portal is to blame, and if it is the client that is the problem we do not know if it is our code or the IBM code we have used that makes it slow. Perhaps it is a combination. This would be good to look into, especially since the client will run on restricted hardware.

We ran the client on a PC when we implemented security measures but the real client in the vehicles will run on some kind of embedded system.

The security measures should be tested on restricted hardware, to verify that they can be used in the vehicles. We have found software for embedded systems that implements the security mechanisms we want, but we have not had the chance to test it.

We have implemented security mechanisms but we have not tested if they actually give the protection they are supposed to. Do the proposed mechanisms offer good enough security? To find out it would be good both to do a penetration test and a thorough security analysis of the SIGYN system. Since the system is still in development and probably will look quite different when it is completed, this will have to wait until a proper test environment has been created.

Bibliography

[1] ARGYROUDIS, P. G., VERMA, R., AND TEWARI, H. D.: Performance analysis of cryptographic protocols on handheld devices. In *In NCA'04* (2004), pp. 169–174.

[2] LEVI, P. A., AND LEVI, A. Performance evaluation of public-key cryptosystem operations in wtls protocol. In *The 8th IEEE Symposium on Computers and Communications - ISCC 2003* (2003), IEEE Computer Society, pp. 1245–1250.

[3] MASICA, K. Recommended practices guide for securing zigbee wireless networks in process control system environments. Tech. rep., 2007.

[4] NILSSON, D. K. *How to Secure the Connected Car*. PhD thesis, 2009.

[5] NILSSON, D. K., LARSSON, U. E., AND JONSSON, E. Creating a secure infrastructure for wireless diagnostics and software updates in vehicles. In *SAFECOMP '08: Proceedings of the 27th international conference on Computer Security, Reliability, and Security* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 207–220.

[6] NILSSON, D. K., LARSSON, U. E., AND JONSSON, E. Security recommendations for a hierarchical wireless vehicular infrastructure. Tech. rep., 2008.

[7] RAYA, M., AND HUBAUX, J.-P. The security of vehicular ad hoc networks. In *SASN '05: Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks* (New York, NY, USA, 2005), ACM, pp. 11–21.

[8] RAYA, M., AND HUBAUX, J.-P. Securing vehicular ad hoc networks. *Journal of Computer Security 15*, 1 (2007), 39–68.