Ethical hacking of IoT devices: OBD-II dongles

[redacted]

Abstract— **The subject area of this project is IT security related to cars, specifically the security of devices connected through a cars OBD-II connector. The aim of the paper is to see the security level of the AutoPi OBD-II unit and to analyze where potential vulnerabilities are likely to occur when in use. The device was investigated using threat modeling consisting of analysing the architecture, using the STRIDE model to see the potential attacks that could be implemented and risk assessments of the attacks using the DREAD model. After modelling the system, attempts of implementing attacks, with the basis in the threat modelling, were carried out. No major vulnerabilities were found in the AutoPi device but a MITM attack on the user was shown to be possible for an attacker to succeed with. Even though no major vulnerability was found IoT devices connected to cars might bring security concerns that needs to be looked into by companies and researchers.**

Sammanfattning— **Ämnesområdet för detta projekt är IT-säkerhet relaterad till bilar, mer specifikt säkerheten gällande enheter som kopplas in i en bils OBD-II-kontakt. Syftet med uppsatsen är att bedöma säkerhetsnivån på en OBD-II-enhet av modell AutoPi och att analysera var potentiella sårbarheter kan finnas i systemet. Enheten kommer att undersökas med hjälp av hotmodellering som består av att analysera arkitekturen, använda STRIDE-modellen för att upptäcka potentiella attackmetoder samt bedöma riskerna för attackerna med hjälp av DREAD-modellen. Efter det steget görs attackförsök utifrån resultaten från hotmodelleringen. Inga större sårbarheter hittades i AutoPi-enheten men en MITM-attack på användaren visades vara möjlig för en angripare att lyckas med. Även fast inga större sårbarheter hittades kan IoT-enheter kopplade till bilar medföra säkerhetsbrister som företag och forskare måste se över.**

Index Terms— **Hacking; OBD-II; Threat model; IoT security; AutoPi**

I. INTRODUCTION

Security is the state of being free from danger or threat[1] and to be free from these one must know where these can occur in a system. In Internet of Things (IoT) security there are multiple factors that can be liable for damage such as usage, environment, connection and many more. Thus, a secure system will have considered as many as possible of these factors and have a solution for them. Since computer systems became available for the general population to use, cyber security needs has seen a increase in its demand, protecting the users from unknowingly leaking information.

A. Security concerns in IoT devices

The Internet of Things is all embedded computer systems with internet connection, the number of which is increasing in recent years. There are many examples of IoT devices from connected door locks that can be opened from a smartphone[2] to smart fridges[3] that let users view the contents of the fridge remotely through a camera. However, there has been multiple security concerns with IoT systems. Some devices are not built with enough emphasis on security which may lead to widespread problems, such as the Mirai botnet[4]. This botnet infected IoT devices that had hardcoded login credentials, something which could have been avoided by the vendors, and used the hacked IoT devices to launch DDoS attacks. With the rise of IoT its security needs to be explored, especially in cyber-physical systems where an attacker potentially could harm people. However, research into these matters are being conducted, such as the security of connected power grids [1].

In recent time the car industry has entered the IoT sphere with cars and accessories being internet connected, leading to incidents such as the remote hack of a Jeep in 2015[5]. Following these kinds of hacks research into vehicular security has gained more traction. Methods for modeling and assessing risks are being developed, such as vehicleLang [2].

B. Goal

The goal of this project is to evaluate the security and privacy of the AutoPi[6] by trying to find vulnerabilities in its architecture. The result is an answer to the question: Is the AutoPi secured from the threats discovered by the threat modeling process? If not, how can it be exploited and what are the impacts?

C. Scope

The scope of this paper is the AutoPi unit including its local WiFi network. This means that the SaltStack back-end and AutoPi's web service is not included in the scope and is open to being explored, with permission from the owners of the infrastructure, if one wishes to continue the research.

II. THEORY

A computer system is built upon many different technologies and techniques. To be able to analyze a system there has to be a general knowledge about founding structures of the system one wish to analyze.

[1] https://en.oxforddictionaries.com/definition/security

[2] https://www.yalehome.com/en/yale/yalehome/residential/yale-real-living/assure-lock/yrl-assurelock-touchscreen/

[3] https://www.samsung.com/us/explore/family-hub-refrigerator/overview/

[4] https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/

[5] https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

[6] http://AutoPi.io

A. Threat modeling

Threat modeling is the process of modeling a system to help identify and rank threats in its architecture. There exist multiple tools that automate the process of threat modeling, such as SecuriCAD [3] by foreseeti[7] and Microsoft's Threat Modeling Tool[8]. The insights gained from the results of a threat model can be utilised for different things. The threat model can both be used by an attacker and a defender to gain insights of where system vulnerabilities might reside. Some examples on how threat modeling can be used, is simulations with vehicles threat modeling [4] and automatic threat-modeling of networks [5].

There are multiple threat modeling methods to choose from. In this research two different methods were used to identify and assess the threats after modeling the system's architecture. The STRIDE [6] method was used to identify threats. In the STRIDE method threats are divided into six categories: Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service and Elevation of privileges. After threats have been identified with the help of STRIDE, the DREAD [7] method is used to rank the severity of the threats. DREAD works by assigning a score from one to three in four different categories: Damage, Reproducibility, Exploitability, Affected users and Discoverability. The score for each category is then added to give the threat its final score. By using DREAD one can prioritize which threats are most critical and how to spend time researching the security of a system more effectively.

B. OBD-II and IoT

IoT car dongles are IoT devices the user plugs into their car's diagnostics port, usually located by the driver's seat or in the glove box, to expand the car's functionality. Some cor [redacted] net [redacted] adv [redacted] to [redacted] in [redacted] IoT car dongles plug in to the car's diagnostic port for power and connectivity to the car's Electronic Control Unit(ECU). The diagnostic port or on-board diagnostic system (OBD-II) is a standard [8] mandated in every vehicle manufactured or sold in USA and Canada after 1996 and 2001 in Europe [9]. The OBD-II port supports multiple protocols that communicates with the car's different internal systems, enabling IoT car dongles to read these internal messages and sometimes even send its own messages on the internal buses. The dongles' connectivity to the car's internal system combined with connectivity to the internet may allow remote access to internal car systems if the dongle doesn't have correct security mechanisms in place.

C. CAN

The Controller Area Network(CAN) [10] is one of the buses supported by the OBD-II standard, allowing different embedded computer systems in a car communicate. CAN bus messages are broadcasted on the bus which means they reach every connected node which makes a malicious node a security concern. By default there is no authentication or encryption of CAN messages, only error detection, which leads to concerns with man-in-the-middle(MITM) and message replay attacks as well as just sending crafted malicious commands.

D. AutoPi

The AutoPi is a car dongle that is designed and developed in Denmark. It is marked as being the first extendable IoT platforms for one's car. The AutoPi uses a RaspberryPi running a Linux operating system. The AutoPi has built in functions for WiFi, 3G/4G connection, Bluetooth, Gps, Speakers, Accelerometer, HDMI output and also GPIO pins. The dongle connects to the car using the OBD-II. The AutoPi is connected to AutoPi's cloud service and can execute commands remotely that interact with the car. The AutoPi device is identified using a serial number which consists of hexadecimal digits and four hyphens in the following format: xxxxxxxx-xxxx-xxxx-xxxxxxxx, where x represents a hexadecimal digit.

The main demographic for AutoPi is car enthusiasts who wants to make their car smarter in different ways, as can be seen in the AutoPi community[9], but also have some DIY electronics and programming knowledge.

[redacted] ysical device a 4G [redacted] ternet. The web interface is used for accessing the physical device remotely. The web interface fetches data from the back-end and can send commands to the back-end which are forwarded to the physical device. The back-end handles data transfers and uses a software called SaltStack[10] to send commands, updates and queries to the AutoPi device. SaltStack is used to automate as well as efficiently scale configurations and control of many IT systems in a secure way. SaltStack works by having a Master which is a central control server and having minions, in this case the AutoPi device, connect back to it for instructions[11]. A simple architectural overview is shown in Figure 1.

[7] https://www.foreseeti.com/

[8] https://docs.microsoft.com/en-us/azure/security/azure-security-threat-modeling-tool

[9] https://community.AutoPi.io/

[10] https://www.SaltStack.com/

[11] https://docs.SaltStack.com/en/latest/topics/tutorials/walkthrough.html

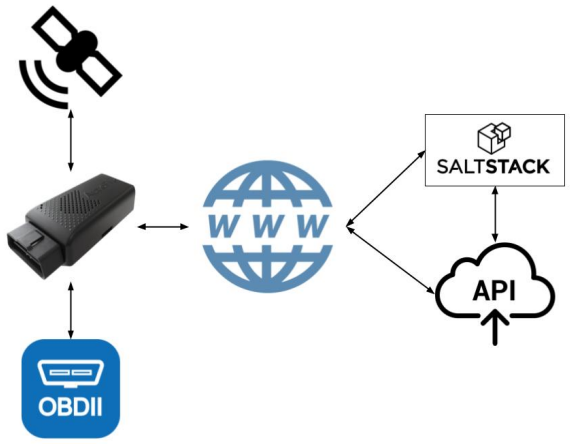

Fig. 1. AutoPi communication flow

III. METHOD

Multiple methods are used in order to test the security of the device architecture. First a system model is created as a base for the threat modeling. The threats uncovered in the threat modeling process are then ranked and prioritised. When threats have been established penetration tests are performed to see if proper counter measures are taken. The outcome of the threat modeling and attacks is presented in the results chapter.

A. Threat modeling

The method used in this research is described in "IoT Penetration Testing Cookbook" [11, p. 42]. First assets of the AutoPi system were identified and used to create a model of the system to be used when identifying probable ways of attacking the system. Then, threats were identified using the STRIDE method and successively five threats were chosen using DREAD.

B. Man-In-The-Middle

By putting an attacker-controlled device between a user and the back-end server a Man-In-The-Middle (MITM) attack tests the security during transport. This attack was carried out using the Kali Linux[12] penetration testing platform in a lab WiFi network. The steps needed to carry out the attack are described in "IoT Penetration Testing Cookbook" [11, p. 78-82] and are easy enough for a novice attacker to follow. In short, the attack works by the MITM downgrading the target user to HTTP instead of HTTPS and then proxying the traffic to the back-end server. This allows the attacker to see all communication between the user and back-end server, if the attack is successful.

C. WiFi attack

Attacking WiFi networks is a well documented area [12]. Our method of attack was to first evaluate the security configuration of the WiFi and then, if viable, conduct one of the attacks documented in literature.

[12]https://www.kali.org/

D. Portscanning

To scan the ports of the device the popular network scanning tool Nmap[13] is used. Nmap provides many different modes and techniques to find open ports and fingerprint services running on a device. Three different settings were tested: TCP scan, UDP scan and Service Detection scan. These three settings were tested on the AutoPi's WiFi hotspot network as well as on its public internet-facing interface.

E. Disclosure of versions

Manual analysis of software and configurations in the AutoPi system is used to try to find software versions and configurations with public vulnerabilities. By searching Google and vulnerability databases for the services and their version numbers one can find out if any public vulnerabilities exist in the AutoPi. Only services reachable from outside of the AutoPi are tested in the manual analysis, for example network services.

F. Firewall rules analysis

To evaluate the security of the firewall rules manual analysis is used. By reading the firewall configuration file one can tell how the firewall separates the networks of the device and which forwarding rules are used for communication between different networks. The AutoPi uses iptables to control the Linux firewall and therefore has a config file with the rules available.

IV. THREAT MODEL

In this chapter the results of threat modeling is presented, to give an overview of the system and ways into it but also to show how priorisations were made.

A. Identify Assets

In this section every asset of the AutoPi system is identified and described.

1) AutoPi Dongle: The dongle plugs into the cars OBD-II connector and can read messages from the cars ECU. The dongle connects to a cloud service over 3G/4G network and has a local WiFi hotspot that allows configuration and internet connection. It runs software that controls many functions and is open for users to do a lot with.

2) Radio Communication: The device supports wireless communication over Bluetooth/BLE, WiFi and 3G/4G. It has a GPS module for position data.

3) Firmware: Configurations and communication with OBD-II.

4) Integrated RaspberryPi: Connects to the OBD-II chip as well as all other modules. Runs a web server interface that allows local configuration.

5) Cloud service/web app: The AutoPi dongle communicates with the auto pi cloud service where live commands can be executed, firmware can be updated, GPS tracking and many more features that can be implemented. The cloud saves historic data such as GPS information plus user defined data points.

[13]https://nmap.org/

6) Dongle hardware: 2x USB ports, HDMI out, GPIO pins, Speaker, Accelerometer. The device connects to a car through the car's OBD-II port.

7) RaspberryPi OS: Runs a full Linux distribution.

B. Architecture Overview

To fully grasp the system and its information flows an architecture overview was made. The overview describes what kinds of information flows exist in the system and their purpose.

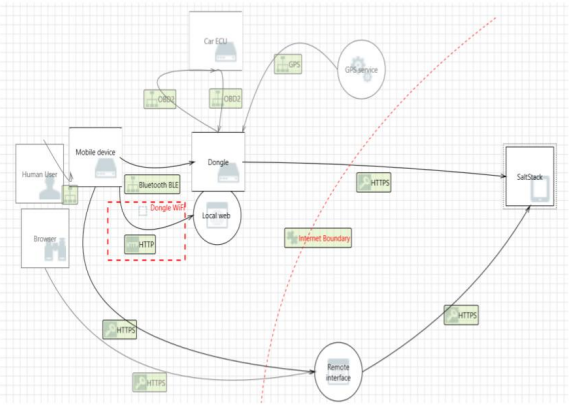

Fig. 2. AutoPi System Model

1) Communication protocol, WiFi 802.11: Local hotspot for the dongle.

2) Communication protocol, HTTP: Local configuration interface uses clear text protocol.

3) Communication protocol, HTTPS: Cloud interface on encrypted text protocol. Used for communication between SaltStack and dongle as well as remote web app and SaltStack.

4) Communication protocol, Bluetooth low energy 4.1: Not activated by default. Configurable by user.

5) Communication protocol, 3G/4G: Connect dongle to the internet, for example to access cloud services.

6) GPS: Dongle has GPS module for location data.

7) OBD-II Port: Port for communication for car internal system. Using the OBD-II standard protocols.

8) Web application (Local): Used for initial configurations and for basic setups. Has console for AutoPi API command executing.

9) Web application (Cloud): Cloud interface to remotely access AutoPi and view statical data. Remote access lets the user execute AutoPi API commands. Lets you see GPS data.

10) SaltStack: Open source technology used by manufacturer to deploy updates and configurations to the device. All commands and data is encrypted.

11) Linux OS: Runs multiple services with open ports.

C. Entry Points

The entry points of a system are the points in which an attacker could interact with the system. A description of the entry points and their use is given.

1) Local web app: Application running on the dongle connected to with local WiFi connection. Sends traffic over HTTP. Has console using SaltsStacks API. If connected to the WiFi the local web app requires no authentication for login.

2) Dongle: The dongle has multiple communication interfaces. An OBD-II interface for communication with a car. 3G/4G modem for internet connectivity. A WiFi interface used for having a local WiFi to configure the dongle and to have internet connectivity(if 3G/4G is connected). The device has a few services running on its internal network reachable through the Access Point (AP) but not reachable from the internet. WiFi interface also allows connection to other APs for internet connectivity. Dongle has built-in GPS chip. Bluetooth interface is present but not turned on by default.

3) Remote Interface: The remote interface is a web app that lets user connect configure and view data provided from the dongle remotely by logging into an account. The dongle is registered to the account with the serial number. Has API for communicating with SaltStack.

4) SaltStack: Used to push commands, updates and configurations to the AutoPi dongle. AutoPi remote interface sends commands through the SaltStack API who then sends it to the dongle.

5) Protocols: Uses HTTP and HTTPS for communication with cloud and the two different web interfaces.

6) Wireless Connections: Uses Bluetooth 4.1/BLE, 4G/3G and 802.11 WiFi.

D. Identifying Threats (STRIDE)

Threats are identified using the STRIDE method.

- **Spoofing identity**
  - Spoof login to the remote interface.
  - Spoof ownership a dongle to connect it to an attackers account.
- **Tampering with data**
  - Tampering with contents of cookies and tokens to perform unauthorized actions.
  - Tampering with dongle ID when sending requests to SaltStack or intercepting valid requests to stalt stack.
  - Creating requests that get routed to the dongle's internal network, i.e. bypass firewall rules.
- **Repudiation**
  - See if all user actions are logged and authenticated.
- **Information disclosure**
  - Disclosure of dongle ID in requests, such as in URLs or JSON data.
  - Disclosure of credentials through in web interfaces.

  - Version and configuration info disclosure as a result of badly configured services.
  - Disclosure of sensitive data by MITM of HTTPS connection.
  - Accessing cookies/tokens by XSS or alike on the remote web interface.
  - Open ports and unsecured services on the AutoPi device.
- **Denial of service (DoS)**
  - DoS attack on the local dongle WiFi and see if it clogs the network.
  - DoS logged in user on remote interface by sending bad requests to the API, causing invalidation of user's token.
- **Elevation of privilege**
  - Try to elevate from SaltStack minion to master.
  - Try to get access to execute commands as user without the dongle ID.
  - Privilege level when executing commands on the dongle, i.e. if an account breach gives root access.

E. Documenting Threats

Each threat is documented and rated. Ratings for each threat is presented in Table 1.

Threat #1:

- **Description:** MITM of connection between user and remote interface.
- **Target:** AutoPi customer.
- **Attack techniques:** Set up malicious computer and intercept the traffic between the two parties.
- **Countermeasures:** Only allow for HTTPS traffic and inform users of the importance of HTTPS.

Threat #2:

- **Description:** Unsecured and open ports on the AutoPi.
- **Target:** AutoPi device.
- **Attack techniques:** Port scanning.
- **Countermeasures:** Close all unused services as well as ports and secure the used ones. Perform configuration auditing.

Threat #3:

- **Description:** Disclosure of configurations and versions of software with public vulnerabilites.
- **Target:** Services used by AutoPi.
- **Attack techniques:** Code review of open source code and config to potentially find known vulnerabilities, analysis of the OS.
- **Countermeasures:** Services should continually be updated. Configurations should be correct.

Threat #4:

- **Description:** WiFi attacks.
- **Target:** WiFi interface on the AutoPi device.
- **Attack techniques:** WiFi interface on the AutoPi device.

TABLE I
DREAD RANKING OF THREATS

	Threat 1	Threat 2	Threat 3	Threat 4	Threat 5
D	3	3	2	3	3
R	1	2	3	2	3
E	3	2	1	2	2
A	1	3	3	3	3
D	3	3	2	2	2
Tot.	11	13	11	12	13

TABLE II
PORTSCAN: PUBLIC INTERFACE

Type	Open Ports	Comments
TCP scan	None	All ports are filtered.
UDP scan	None	All ports are filtered.
Service Detection	None	The scan indicates that the host is up and reachable but all ports are filtered.

  - **Countermeasures:** Use randomized and unique password for WiFi access.

Threat #5:

- **Description:** AutoPi's internal network exposed.
- **Target:** AutoPi device.
- **Attack techniques:** Reviewing firewall rules to see if it's possible to route outside packets to the AutoPi's internal network.
- **Countermeasures:** Correct firewall rules.

V. ATTACK RESULTS

This chapter presents the results of attacks performed on each threat found in chapter IV.

Threat #1: The MITM attack was successful. By being on the same network as a user it is possible for an adversary to act as a Man-In-The-Middle and therefore getting access to all data entered by a user on the AutoPi Cloud interface. Firefox and Chrome browsers show warnings that the connection has been downgraded to HTTP but Internet Explorer show no such warnings. By being able to access data entered by a user the AutoPi device is compromised in the form of AutoPi Cloud credentials as well as the Dongle ID connected to the account a user logs into. However, this attack targets the user, browser and transport protocols. This is not a problem unique to the AutoPi but rather a general attack.

Threat #2: Table II and III show outcomes of the two rounds of port scanning. All ports not in the tables have a state of either closed or filtered.

TABLE III
PORTSCAN: INTERNAL WIFI

Type	Open Ports	Comments
TCP scan	22, 53, 80, 9000	More info in service detection scan.
UDP scan	53, 67	The Nmap output indicates that port 53 is related to DNS and that port 67 runs a dhcp server.
Service Detection	22/SSH, 53/dnsmasq 2.7.6, 80/lighthttpd 1.4.45, 9000/Werkzeug httpd 0.14.1	Multiple successful fingerprints of services.

Threat #3: As seen in Table III a couple of service versions were uncovered in the Nmap scan. Below is the results of the manual analysis of services reachable from outside the AutoPi.

- **OpenSSH**
  - Version: 7.4p1
  - Public vulnerabilities: No.
  - Comment: SSH server running on internal hotspot.
- **dnsmasq**
  - Version: 2.7.6
  - Public vulnerabilities: Yes.
  - Comment: dnsmasq runs on internal hotspot. Multiple public vulnerabilities such as DoS and potential code execution[14].
- **lighthttpd**
  - Version: 1.4.45
  - Public vulnerabilities: Yes
  - Comment: Lighthttpd runs on internal hotspot. One potential vulnerability[15].
- **Werkzeug httpd**
  - Version: 0.14.1
  - Public vulnerabilities: No.
  - Comment: Werkzeug running on internal hotspot.
- **wpa_supplicant**
  - Version: 2.4
  - Public vulnerabilities: Yes.
  - Comment: Software used when the device connects to WiFi networks. Multiple public vulnerabilities such as DoS and potential code execution[16].
- **hostapd**
  - Version: 2.4
  - Public vulnerabilities: Yes.
  - Comment: Software used to set up the AutoPi hotspot. Multiple public vulnerabilities such as DoS and potential code execution[17].

Threat #4: The WiFi hotspot broadcasted by the AutoPi device uses WPA2 authentication in Pre-Shared Key(PSK) mode. According to documentation in the area it is possible to brute-force the PSK to the network by sniffing packets from the setup of a connection between a client and the WiFi access point [18]. However, the AutoPi hotspot uses a 13 character password where 12 of the characters are varying hexadecimal characters, as described in Chapter II. This means that the password has 12^{16} different permutations, making it infeasible to brute-force within reasonable time. Therefore, the attack was aborted in the evaluation stage.

Threat #5: When manually reviewing the firewall configuration of the AutoPi the following observations were made.

- The internal WiFi does not accept input from the 4G internet-connection unless packets have the state ESTABLISHED or RELATED. This means that communication to a host on the AutoPi hotspot is possible if it's initiated by that host, which is necessary for any host on the internal AutoPi hotspot to be able to reach the internet.
- Ports 22/TCP, 53/TCP UDP, 67/UDP, 80/TCP and 9000/TCP are open on the internal AutoPi hotspot, all other ports are closed.
- There are no restrictions on outward connections from the AutoPi hotspot.

VI. DISCUSSION

As one could tell from reading the results, no major or critical vulnerabilities were found. However there are some aspects of the AutoPi that can be hardened to make it more secure.

A. MITM

As described in the results of the MITM attack it has potential to compromise a user's AutoPi by stealing credentials to the AutoPi web interface. To be able to carry out the attack the attacker and user would just have to be on the same WiFi network, which the attacker can achieve by, for example, logging into the same coffeeshop WiFi network or by setting up a rogue hotspot. The effectiveness of the attack depends mostly on the user's general knowledge of computers and what browser is used. A well-informed user probably wouldn't enter any credentials when the connection isn't HTTPS whereas a less-informed one might. The fact that some browsers display explicit warnings about entering

[14]https://www.cvedetails.com/vulnerability-list/vendor_id-8351/product_id-14557/Thekelleys-Dnsmasq.html

[15]https://www.cvedetails.com/vulnerability-list/vendor_id-2713/Lighttpd.html

[16]https://www.cvedetails.com/vulnerability-list/vendor_id-12005/product_id-29296/W1.fi-Wpa-Supplicant.html

[17]https://www.cvedetails.com/vulnerability-list/vendor_id-12005/product_id-22495/W1.fi-Hostapd.html

[18]https://www.evilsocket.net/2019/02/13/Pwning-WiFi-networks-with-bettercap-and-the-PMKID-client-less-attack/

credentials on an HTTP connection limits the chance of success of the attack but one should note that certain browsers don't display warnings. With access to the credentials an attacker would have full control of the dongle as well as saved data which could be history of driven routes for example.

B. Port scan

The results of the portscan did not show any particular security concerns. A remote attacker cannot, from the findings of this project at least, access anything on the AutoPi from its public internet interface if no prior foothold on the device is gained. As seen in Table III there are multiple services open on the AutoPi's hotspot. These open services are both vital for the functionality of the AutoPi but also inside one of the AutoPi's trusted zones, its hotspot. If a malicious actor were to access the hotspot they would already have access to the SSH which has hardcoded credentials as well as the local web interface which allows code execution and changing of settings.

C. Disclosure of versions

In the AutoPi there are outdated software running with known vulnerabilities of different degrees. Of the programs running two of them can be used to communicate outside of the AutoPi's trust zones, wpa_supplicant and hostapd. The wpa_supplicant program is a common Linux utility used to connect to WiFi hotspots and the hostapd program is used for the AutoPi to also act as a hotspot. An example of concerns with having this kind of software outdated is the fact that there exist multiple Common Vulnerabilities and Exposures(CVEs) for the versions running. An example of an unpatched CVE in wpa_supplicant is CVE-2015-1863 which through manipulating the SSID in certain WiFi frames can cause DoS and possibly execution of arbitrary code.

D. WiFi hacking

AutoPi's WiFi was not found to be vulnerable to any particular known attacks against WiFi from our studies of it and the pre-installed credentials are complex enough to make bruteforcing infeasible. However, it seems strange to derive the SSID and password to the WiFi from the dongle ID. The dongle ID is quite a vital part of the identity of an AutoPi device it seems like it would be more logical to use a fully randomised value as SSID and password for the WiFi. For example, if an AutoPi user wishes to connect friends or family to the WiFi hotspot one would have to trust them with a significant part of the dongle ID.

E. Firewall analysis

The analysis of the firewall rules of the AutoPi confirms the result of the portscan. By segmenting the WiFi interface and public internet interface the device is hardened in regards to remote attacks. The fact that it sets no restrictions on communication in outward direction from its internal hotspot is vital to make it usable by users browsing the internet from the hotspot. However, it also contributes to making it easy for a compromised user device to "phone home" to establish malicious connections. But, configuring tighter firewall settings outward would severely reduce the usability of the AutoPi's hotspot.

F. Critical components

The dongle ID, described in the theory chapter, is used to connect a physical device to an account in the cloud service, in communication with the back-end as well as for the local WiFi hotspot. In other words, the dongle ID is a vital part of securing a device and is supposed to be kept a secret. If an attacker would get access to the dongle ID they would be able to connect the physical device to their own account(as long as it's not already connected to two accounts) as well as access the AutoPi device's WiFi hotspot.

Other parts of the AutoPi system are also critical for its security, SaltStack and the cloud interface. Since the SaltStack service relays commands between the cloud interface and device an attacker gaining access would likely be able to replicate and send commands themself. As for the cloud interface, if an attacker could gain access to a specific user account using a flaw in the system the device would be completely in control of the attacker. These two services that are vital for the way the AutoPi works still increases the system complexity and attack surface, making them critical to be secured.

G. Other devices

Brief research was made on another device and came to the conclusion that it is a very insecure unit. The unit comes with a hardcoded password which was on the extreme low in terms of security. But what is alarming with the device is that there seems to be a possibility to write external commands to this device. A service such as pythonOBD[19] makes the process easy. Researchers has then showed that if you know the specific car CAN message then you can use functions that is not supposed to be enabled such as turning of lights [13]. So the security detail in device lie upon the fact that it is to slow to snap up the individual can messages. But if someone already know these then there could be potential for an attack. A security percussion for the specific device would be to simply remove it from the OBD-II port. We did not explore the function if connections would turn on the device even if the car was off so to assume the worst case scenario ones best solution is to simply remove the unit from the OBD-II port when not in use.

H. Impact

A successful attack on the AutoPi would mean that one has direct access to send commands into the car, making it display unwanted behavior for example open doors. If the consumer uses a SIM card then one would also be able to find the car with its GPS settings. Therefore a potential intruder could check where the car is and wait until it sees that the engine is turned off (car owner is not in car) and then lock

[19]https://python-obd.readthedocs.io/en/latest/Command%20Tables/

the door up, start the car and drive away. If the hacker has even more malicious intent then he/she can send commands while the consumer is driving and thus having a potential to cause serious harm to the everyone surrounding the cars proximity.

VII. CONCLUSION

From the information gathered, the AutoPi has been made with some degree of security in mind. The attacks performed in this report showed no extreme level of exploitability but this project was also limited by the scope so there is a bigger surface that a hacker could explore when attacking and also due to limited time, knowledge and resources there is also potential for continued penetration testing research on the unit.

Since no prior agreement was made with AutoPi the research focused solely on the actual device rather than the infrastructure behind the scenes, because of legal reasons. Therefore, the results in this project doesn't guarantee anything about the security of the back-end parts of the system, such as the Cloud Interface, SaltStack or login mechanisms.

As demonstrated with the MITM attack the user is a potential attack vector. By tricking the user an attacker could gain access to the user's account. The problem causing the viability of this attack isn't in the design of the AutoPi but is rather a general problem of informing users of what is secure and what is not.

The AutoPi device places a lot of trust in the security of WiFi, once inside the local WiFi a user or attacker has access to the inner workings of the device with ease. This trust poses a potential threat to the security of the device should the WiFi not be as secure as thought. Also this puts more security into the hands of the user.

With physical systems like cars connected to the internet more studies of the security and its importance must be made by both companies and researchers. The potential outcome of hacks against cyber-physical systems are severe and should not be treated as fiction.

ACKNOWLEDGMENT

The authors would like to thank Pontus Johnson and Robert Lagerström for their guidance, help and feedback during the project.

REFERENCES

[1] A. Vernotte, M. Välja, M. Korman, G. Björkman, M. Ekstedt, and R. Lagerström, "Load balancing of renewable energy: a cyber security analysis," *Energy Informatics*, vol. 1, p. 5, Jul 2018.

[2] S. Katsikeas, "vehiclelang: a probabilistic modeling and simulation language for vehicular cyber attacks," Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2018.

[3] M. Ekstedt, P. Johnson, R. Lagerström, D. Gorton, J. Nydrén, and K. Shahzad, "Securi cad by foreseeti: A cad tool for enterprise cyber security management," in *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*, pp. 152–155, Sep. 2015.

[4] W. Xiong, K. Fredrik, and L. Robert, "Threat modeling and attack simulations of connected vehicles: A research outlook," 02 2019.

[5] P. Johnson, A. Vernotte, M. Ekstedt, and R. Lagerstrom, "pwnpr3d: An attack-graph-driven probabilistic threat-modeling approach," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*, (Los Alamitos, CA, USA), pp. 278–283, IEEE Computer Society, sep 2016.

[6] R. Khan, K. McLaughlin, D. Laverty, and S. Sezer, "Stride-based threat modeling for cyber-physical systems," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pp. 1–6, Sep. 2017.

[7] A. Omotosho, B. A. Haruna, and O. M. Olaniyi, "Threat modeling of internet of things health devices," *Journal of Applied Security Research*, vol. 0, no. 0, pp. 1–16, 2019.

[8] R. Malekian, N. R. Moloisane, L. Nair, B. T. Maharaj, and U. A. K. Chude-Okonkwo, "Design and implementation of a wireless obd ii fleet management system," *IEEE Sensors Journal*, vol. 17, pp. 1154–1164, Feb 2017.

[9] European Parliament Council of the European Union, "Directive 98/69/ec," Oct 1998.

[10] ISO, "Road vehicles – controller area network (can) – part 1: Data link layer and physical signalling," tech. rep., International Organization for Standardization, 2015.

[11] A. Guzman and A. Gupta, *IoT Penetration Testing Cookbook: Identify Vulnerabilities and Secure Your Smart Devices*. Packt Publishing, 2017.

[12] R. Guo, "Survey on wifi infrastructure attacks," *International Journal of Wireless and Mobile Computing*, vol. 16, p. 97, 01 2019.

[13] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*, pp. 447–462, May 2010.