Penetration Test for Block Harbor

Final Report

4/19/2022

Written by: Jesse Russell, Yahia Khalaf, Hannah Hatala, and Cameron Keesee

Table of Contents

Introduction/Executive Summary

Our team was tasked by Bryan Blancke at Block Harbor Cybersecurity to conduct a penetration test on a 2016 Jeep Cherokee. When this project began, our team had little knowledge on the topic of automotive security, but through a large volume of research and some testing, we were able to learn quite a bit about the functions of ECUs that reside in all modern vehicles. We learned how CAN functions, how it communicates over the vehicle's network, CAN fuzzing (spoofing), and explored multiple tools to analyze the CAN traffic. During the Threat Modeling phase of the investigation, we were able to brainstorm some possible exploits to be used on the attack surfaces (ECUs) in the vehicle module. While we have been unable to find any active exploits that we could successfully attempt, we were able to intercept CAN messages coming from the vehicle's bus which can provide useful information about the vehicle. These messages, however, are in hex format which prevents the team from understanding them. Using a provided database file (DBC) or reverse engineering techniques on the arbitration IDs, the team may have had success decoding these messages into a human readable format. We were unsuccessful in finding this confidential information. This final report was created from our notes and details our findings, methods, issues, and suggestions for how this project could have improved. In addition to the written report, the team will also provide a presentation on the project to elaborate on the details of the report.

Tools

1. Software
    a. Vehicle Spy 3
        i. An enterprise Windows application used to intercept and analyze CAN traffic in a neat and presentable GUI, typically used with a device such as ValueCAN.
        ii. Has the ability to graph data, send/receive messages, fuzz messages, decode arbitration IDs with DBC files, etc.
    b. Kayak
        i. Kayak is similar to Vehicle Spy, but adds multiple features for analyzing CAN data that may be more applicable to certain situations. For instance, one standout feature is the ability to extract GPS data from the CAN bus and plot that data onto a map. This may be useful to track a car's whereabouts and establish a set routine performed by the driver. This is an open source tool available for Linux and Windows systems.
    c. CaringCaribou
        i. Open source CAN traffic analyzer that is very easy to use and can be dropped into a CAN network to gather all sorts of information about the vehicle. Modules in the tool include Unified Diagnostic Services (UDS), Universal Measurement and Calibration Protocol (XCP), Fuzzer, dump, send, raw, etc.
    d. CAN-Utils/SocketCAN

i. A suite of tools that provide a basic framework for looking at the CAN traffic. Often, other CAN analyzers will require this to be installed as a prerequisite.

e. Instrument Cluster Simulator (ICS)

i. A working replica of the instrument cluster inside a vehicle. This allows students to practice CAN bus hacking and controlling the “vehicle” with live feedback. This provides a safe way to make mistakes and learn along the way.

f. Kali Linux

i. A linux distribution with multiple tools intended for penetration testing.

g. WiTech Web Interface

i. A web application running from the WiTech 2.0 device using the usb connector as a network interface. This interface provides an overview of all ECUs, their diagnostic codes, their firmware, and their functionality.

h. TeamViewer

i. Remote desktop application used to conduct work away from the vehicle bench.

2. Hardware

a. WiTech 2.0

i. Connects to the OBD-II port of the vehicle and is used as a diagnostic tool by dealerships to troubleshoot the vehicle.

b. ValueCan

i. A USB to OBD-II device that allows the user to connect the bench or vehicle to a PC for intercepting CAN messages.

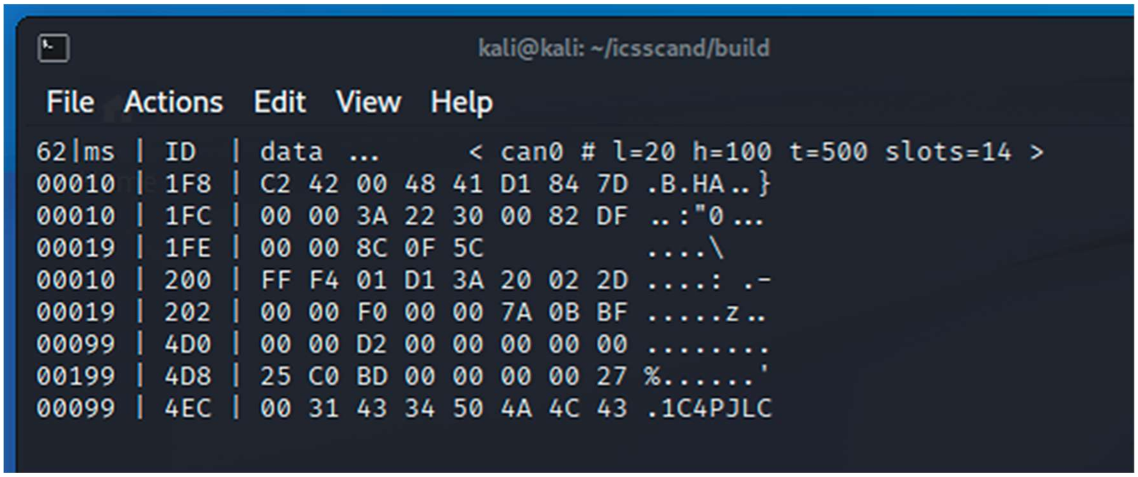
c. Simulated Vehicle Module

Methodology

When we planned our project, we aimed to follow the penetration testing process provided by Block Harbor Cybersecurity to conduct our investigation. This process has 6 tentative phases to keep the investigators' findings organized and to streamline their testing. The phases are as follows:

1. Threat Modeling
2. Passive Reconnaissance
3. Security Defense Checking
4. Active Exploitation
5. Deep Exploitation
6. Report Writing

While we were unable to complete all of its phases, we did learn about how the process works and how it can be utilized in a professional environment. The first phase of this process is to begin threat modeling the attack surfaces in the system. Threat modeling allows investigators to paint a broad picture of what components can be exploited and brainstorm any possible exploits to these surfaces. Our threat model is provided in the next section. During the Threat modeling phase, we conducted research on the CAN bus, automotive security, the penetration testing methodology, and programs we could use to analyze this data (detailed in the Tools section above.) Our team then used the WiTech 2.0 tool to gather information about each ECU and begin our process of gathering reconnaissance on the components provided. This tool, intended for dealership diagnostics, uses the OBD-II connector on the vehicle bus to gather information about various components that the dealer would need to access. Once the tool is set up using the WiTech Setup utility on a Windows desktop, the device can be accessed through a web interface and a provided username and password associated with the device. Accessing the web interface allows the team to look at all of the ECUs present as well as all of their diagnostic codes and firmware versions, which may be useful in the later stages of the penetration test. Our testing process then proceeded with searching for vulnerabilities within the CAN Bus, using ValueCAN. This device enables connectivity from the vehicle module to the Windows workstation through the testing module's OBD II connection. Once the connection has been established. Our team used the program "VehicleSpy 3" to gain access to a user interface displaying the messages transmitted over the CAN bus. At this point, the team was able to intercept the CAN messages flowing throughout the system to gain an understanding of the communications within the module. The reconnaissance continued with a group effort researching the CAN communications through research for the dealer specific CAN database. The CAN databases help to provide translations for the CAN messages to help the investigators understand. Due to security and confidential information, these files are not readily available to access and are kept closed source to prevent sensitive information on the vehicles from getting into the wrong hands. The team was unable to find the 2016 jeep cherokee KL CAN database file, but if we were successful in finding this file, the CAN messages would be presented in human-readable English opposed to only hex messages. We spent a considerable amount of time in Vehicle Spy 3, but upon our lack of success in finding useful information, we decided to change strategies by loading CAN-Utils onto a Kali Linux virtual machine to view and analyze the traffic. We were presented with multiple tools that were a little more helpful in understanding the CAN traffic, but were still unable to decode what these messages were saying. The first of these tools is a command in the SocketCAN suite called candump. This command allows an investigator to view all of the incoming traffic on the device connected to the OBD-II port. However, this tool is unorganized and is simply a raw display of every message being sent. In order to streamline this data and group the packets into specific devices, cansniffer is the tool to use. Cansniffer takes all of the information gathered in candump and organizes the messages by arbitration ID, which identifies the module in the communication, and significantly reduces the length of the list. Pictured below is the reduced list presented by cansniffer.

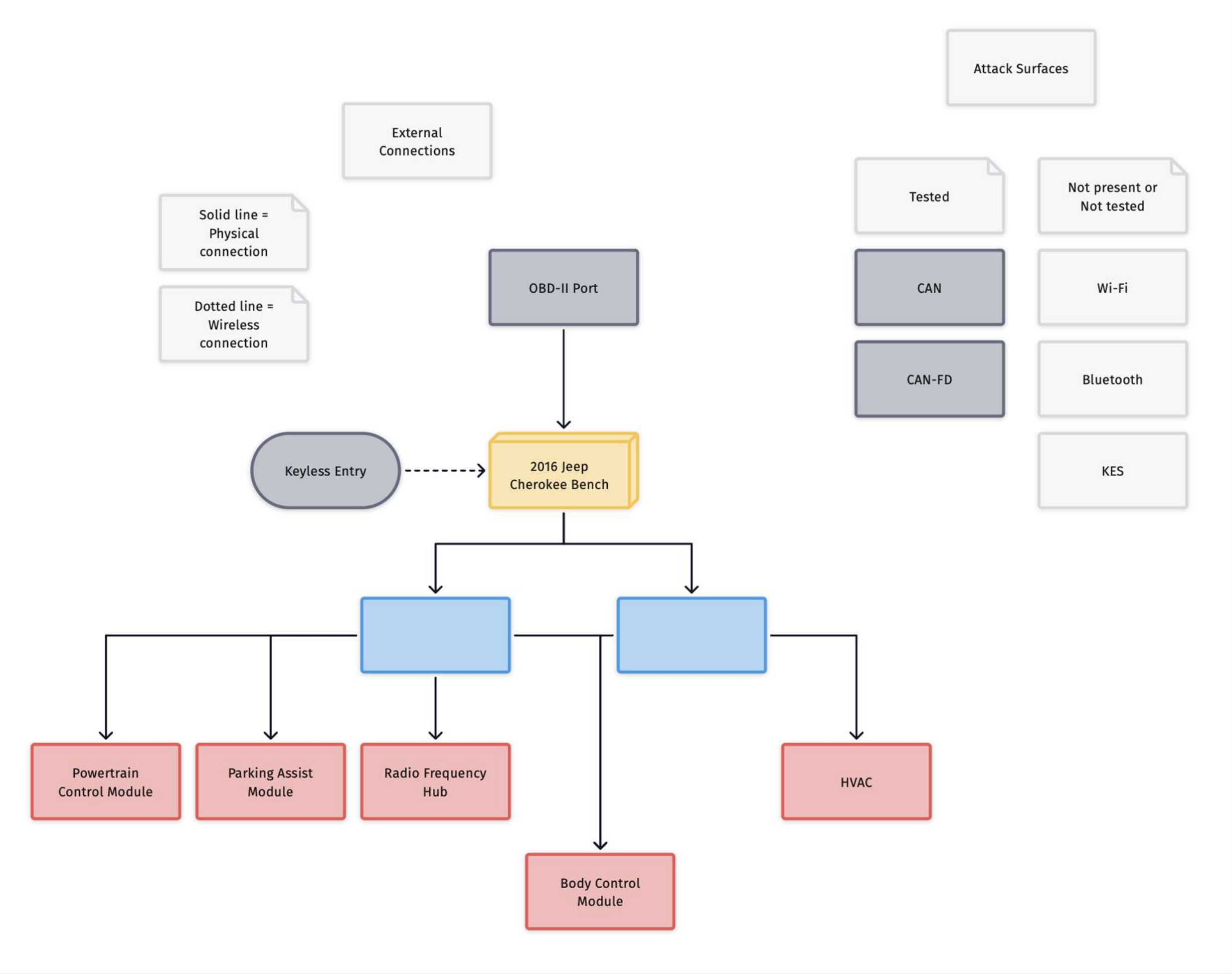


```
62|ms | ID  | data ...      < can0 # l=20 h=100 t=500 slots=14 >
00010 | 1F8 | C2 42 00 48 41 D1 84 7D .B.HA..}
00010 | 1FC | 00 00 3A 22 30 00 82 DF ..:"0...
00019 | 1FE | 00 00 8C 0F 5C          ....\
00010 | 200 | FF F4 01 D1 3A 20 02 2D ....: .-
00019 | 202 | 00 00 F0 00 00 7A 0B BF .....z..
00099 | 4D0 | 00 00 D2 00 00 00 00 00 ........
00199 | 4D8 | 25 C0 BD 00 00 00 00 27 %......'
00099 | 4EC | 00 31 43 34 50 4A 4C 43 .1C4PJLC
```

If we had more time with the project, we would've practiced more with software such as CaringCarinbou, Kayak, and VehicleSpy, as we were unable to test these programs due to our limited knowledge of how they work. We also attempted to decode the Arbitration IDs by looking for a DBC file, but were not able to determine where these messages were coming from. Learning this information would aid in progressing the exploitation of the ECUs.

Attack Surfaces

In the early stages of a penetration testing investigation, the team must first discover possible avenues to attack the modules inside the test bench. These means of accessing the internal system are known as Attack Surfaces. In the context of this vehicle penetration test, we would use these surfaces to exploit the ECU modules responsible for various functions in the vehicle, listed below. During our initial investigation, the team was able to find multiple ECUs connected to the module using the web interface of the WiTech 2.0 tool provided. We also included possible exploits that could occur if an attacker gained access to these modules.



- PAM (Parking Assist Module)
    - The Parking Assist Module (PAM) is used by the vehicle to facilitate features such as lane detection, automatic parking, brake warning/auto brake, and steering.
    - **Possible Exploits:**
        - Once an attacker has access to the PAM inside a vehicle, they could then be able to control the entire vehicle and cause it to drive without any user interaction required. This presents an obvious safety hazard, as vehicles can be dangerous without complete control by the driver. If a customer were to be attacked in this way, they may be more likely to be in an accident causing injury or death.
- PCM (Powertrain Control Module)
    - The Powertrain Control Module (PCM) is the ECU responsible for ensuring the engine runs smoothly, managing aspects such as gear control, speed control, fuel injection sensors, O2 sensors, etc. It has over 100 functions of control for the vehicle, and if any of those fail, the PCM will send a check engine light to the dashboard.
    - **Possible Exploits:**
        - If the PCM is accessed, attackers may be able to control various functions of the vehicle's engine mechanisms such as shifting and fuel consumption. It may be possible to block the car from shifting or trick the vehicle into thinking it's in an entirely different gear altogether.
        - An attacker may be able to reduce fuel mileage, or even cause a fire, by injecting too much fuel into the cylinders, causing the engine to work harder than it needs to.
        - The engine's power may also be vulnerable to attackers through the PCM, causing the vehicle to slow down or speed up depending on the attacker's goal.
- BCM (Body Control Module)
    - The Body Control Module (BCM), in a basic sense, is the central computer in the vehicle that moderates communications between other ECUs in the system. For CAN messages, it works as a router to determine where to send packets on the network. Because the ECUs are intended for specific purposes, the BCM is a vital component that issues commands for the ECUs to execute. It also controls the physical power features in the vehicle, including power locks, power windows, windshield wipers, headlights, etc.

    - **Possible Exploits:**
        - Access to the BCM could allow an attacker to try and distract the driver by causing malfunction in various powered devices in the vehicle. For example, if an attacker wanted to reduce the visibility of the road from the driver, they would only have to activate the windshield wiper fluid to render the windshield unusable.
        - An attacker may be able to remotely unlock the vehicle and allow a thief to enter.
- HVAC (Heating Ventilation and Air Conditioning)
    - The HVAC module is exactly what the name implies. It controls the interior air quality for driver/passenger comfort, including cabin temperature, fan speed, circulation, and defrost. Newer models of vehicles may feature heated/cooled seats, which would also be handled by the HVAC unit.
    - **Possible Exploits:**
        - Once the HVAC module is accessed, an attacker may be able to disable certain functions of the vehicle's climate system. For example, if that attacker wanted to raise the temperature inside to an unreasonably high amount, the passengers may be endangered and become severely dehydrated.
        - The attacker may also control aspects of the vehicle such as window defrost, which may cause the driver to lose visibility during a cold and rainy day.
        - If the heated seats are controlled remotely, an attacker may be able to burn the passengers by exploiting and modifying the heating element inside the seat.
- RFH (Radio Frequency Hub)
    - Throughout a vehicle's network, certain radio frequencies must be generated for features to work properly. The Radio Frequency Hub (RFH) generates and receives these frequencies. For instance, on most vehicles built after 2013, keyless start is a standard feature that uses an NFC field to ensure a serialized chip is inside the vehicle. If this chip (located inside the key), is not present, the vehicle will not start.
    - **Possible Exploits:**
        - Attackers may spoof the key's NFC frequencies to trick the vehicle's immobilizer into starting the vehicle without the proper serialized key present.
        - Lane departure and lane assist systems use IR frequencies generated by the RFH. Attackers could diable these features to stop working entirely, or send a false flag to the system in order to cause the vehicle to stop prematurely.

Issues

During this process our group ran into many issues, some that were due to mistakes we've made and some that were caused by technical errors. Mainly, our inexperience in automobile hacking and tools for CAN analysis limited our ability to achieve progress quickly. Due to this inexperience, it took a long amount of time to understand much of these tools' capabilities, as well as how we can use them to speed up our workflow. YouTube videos and tutorials were a good source of help to use for gaining familiarity with VehicleSpy. Despite our inexperience, we learned a lot of new information that we didn't have upon commencement of this project, which is included in previous sections. We had to find a lot of the answers for ourselves, which helped the group learn and understand why automobile hacking works the way it does, as well as improve our skills on how to solve problems as they occur. We also struggled to decode the CAN messages to find out where they were coming from and their purpose. We attempted to look for the database (DBC) file, which would help us decode the CAN messages and correlate the arbitration IDs to the ECUs, but these are highly confidential and are not available to the public. Our team would find bits and pieces of other DBC files, but none that would match the arbitration IDs for the CAN messages. Because we had to use TeamViewer to remote control the pc connected to the bench, we had an instance where we were connected to the wrong ValueCan and did not realize until we came into the facility the next week. This was a setback to some of our progress, but ultimately taught us a lesson about checking to make sure our hardware is working properly. Another issue we ran into that went along with having the wrong Valuecan plugged in was other groups touching our equipment in the cage, this made it difficult to trust that we are actually using the right things when we would remote in.. Overall, the problems that we had during this project caused our group to strengthen our troubleshooting skills along with being able to work together to make conclusions on what went wrong/right.

Feedback

Throughout the semester, multiple setbacks were responsible for delaying our progress. While we recognize that some of these setbacks were due to inexperience and mistakes on our part, multiple issues cropped up through little fault of our own. Per Bryan's request, we've provided feedback for Bryan and Dr. Tout so that future capstone projects can improve and run more efficiently. This section will also detail what we as a group could have done better.

During the first half of the semester, our group did not feel we were briefed in a way that clearly communicated what was expected of us. We believe there was some misunderstanding of our experience level in the automotive security world and in the early days of our project, we were unsure of how to proceed with establishing communications with Bryan. It appears we expected to be guided through the entire process but were instead left to our own devices, making it difficult to come up with a plan to see this project through. We recognize that some of this lies on us, as we could have been more active and vocal in any questions we had. However, for future semesters, we feel it would be important to remember that this is a learning process for the students and most of the time, they have not a clue where to start or where to go. Establishing clear guidance for the students would help in preventing this from being repeated. We were also slowed down by the time it took to receive feedback on multiple reports that we had turned in on time. Bryan has been very apologetic about the return time, and we understand that his business comes before our group, but we were not left with enough time to edit our work based on the feedback provided. Instead, we had to move onto trying to make progress on the project without any idea if we were conducting our workflow properly. Bryan has been very helpful in answering any questions we've had while trying not to spoil anything or give the answers away. We appreciated the manner in which Bryan answered our questions and encouraged us to find the answers ourselves with a little guidance. As for the academic side of the project, we felt that Dr. Tout was unable to guide us on our project due to the differences between all of the groups' goals. Our instruction was fairly broad and did not seem to apply to many of the concepts required for Bryan's expectations. Finally, we felt our handling of the project could have been better. Because we are all college students that have busy schedules, most of us were exhausted and stressed about the other things we needed to accomplish and resultantly neglected to put all of our attention onto the project. In order to prevent this, we could have worked harder on managing our time to ensure our deliverables were of professional and detailed quality. For future projects, this is a lesson we will all cherish and take with us for the improvement of our professional lives.

Conclusion

Our group's initial plan was to go through each phase of the penetration testing methodology provided by Block Harbor Cybersecurity, but we were unsuccessful in testing past the passive reconnaissance stage due to time limitations and the issues listed in the previous section. While the outcome did not reach our expectations, we were able to gain an understanding of the vehicle and learn more than we thought possible. Our project was inspired by the 2014 Jeep penetration test performed by Charlie Miller and Chris Valesek, in which they used the head unit to execute commands and exploit the vehicle. We all enjoyed learning about the penetration testing process and the mechanisms that allow modern vehicles to function. We were able to create a threat model through careful research, enumerate the attack surfaces, list all ECUs connected, learn how CAN works, and think like a professional penetration tester.