**System Proposal**

1. **Executive Summary**

Cars have had more technological advancements done to them in the last 15 years than all the years before since they began being mass produced. This mass growth in the car industry is not an isolated event. Across all industries, we are creating new technologies faster than we are able to master the old technologies. Even industries that have been notorious for being slow to adapting to new technologies are now changing quicker than ever before, including the automotive industry. The implementation of these new technologies is having a positive impact on people’s daily lives. Cars are becoming more fuel efficient and environmentally safe, from a manufacturing standpoint as well as an emissions standpoint. Cars are also getting new features that have increased the safety of the driver of the car as well as anyone else around.. This includes during collisions and also with features such as lane departure detection, better headlights, distance locking cruise control, and other similar features. However, all of these technologies are being implemented without thinking about the repercussions. With new technologies comes new ways for dangerous people to be able to hack in these connected devices. It goes without saying that there a huge safety concern if someone were to be able to hack into a vehicle, especially if it were moving. This is where our project comes into focus.

We are tasked with designing a control system that is an adequate analogy for a modern control system in today’s cars. For our project, this comes in the form of a CAN bus system comprised up of arduino minis and a beaglebone black. A CAN bus network was created because that is the industry standard for the communication used internally in almost every vehicle on the market. Our network is broken up into multiple nodes, with each arduino handling different aspects of a vehicle’s internal network, such as speedometer, temperature sensors, and breaking. The beaglebone black will be used to act as the vehicle’s ECU, which does the interpolation of the information being sent on the CAN bus. Then once our system is designed, we must think of different ways to hack into the system and compromise the signals being sent and received. Once again, this will be done with a different beaglebone black. We will connect our beaglebone black directly to our CAN bus network and perform different attacks such as DDOS attack and man-on-the-middle attack. Once we have identified multiple security risks, we then need to devise and document cheap and effective ways to harden the system from these attacks. We will be constructing a duplicate version of the system that is our CAN bus, the arduinos and beaglebone black aforementioned. Once this network has been duplicated and has been tested to confirm that it is running the same way as the original network, we will harden it against the malicious attacks mentioned above. The creation of a second system is done for demonstration purposes so we have a way of showing our clients what hacks can be performed on a standard, unprotected CAN bus and then show what a hardened CAN bus network looks like and how well it stands up to the same hacks as the other system.

1. **Project Deliverables**

The client, Rolls-Royce, wants the team to research and evaluate growing cyber-security concerns, and a demonstration that can be presented as an introduction for the client's engineering team to these cyber-security concerns. This demonstration and accompanying materials will take the form of:

1. all design and technical documentation
2. a fully functioning, closed system, demonstration that includes but is not limited to
   1. Three attacks models based on baud rate manipulation, id replication, and denial of service
   2. Three defence models based on state prediction, redundant networks, and per-node correction.
3. a user manual for the demonstration system
4. a white paper documenting our findings and resulting recommended hardening steps

The demonstration system will show an example control system, based on an automotive control system, operating under its normal conditions, and then walk the audience through an exploit and show the reaction of the system. A few assumptions are being made that the system is on and started, the hack begins a nominal amount of time after the system has been running, and that the attack is on the physical layer of the CAN Bus. The system should be able to show, through basic hardening steps, how the exploit can be prevented or made more difficult.

1. **Stakeholder Model**

|  |  |  |
| --- | --- | --- |
| **Stakeholder Name** | **Level of Importance** | **Description** |
| Rolls-Royce | Highest | The company supporting the research |
| Supervisor | Moderate | The faculty member overseeing the project |
| Manufacturer | Moderate | Company manufacturing and assembling vehicles |
| Clients | Highest | Customers buying RR products |
| Consumers | Lowest | Customers buying Client products |

*Figure 1 – A Table of stakeholders and their descriptions*

|  |  |
| --- | --- |
| **Feature Name** | **Feature Definition** |
| Safety | Research directly aids in safety and protection of all stakeholders |
| Security | Research directly aids in security by blocking or interrupting third party attacks |
| Affordability | Solution is low cost as to not lower margins |
| COTS | Solution is readily available and made with consumer-off-the-shelf components |
| Maintainable | Solution is easy repaired and replaced by technicians |
| Manufacturability | Solution is easily installed and uses existing technologies to create |

*Figure 2 – Table of features and their descriptions*

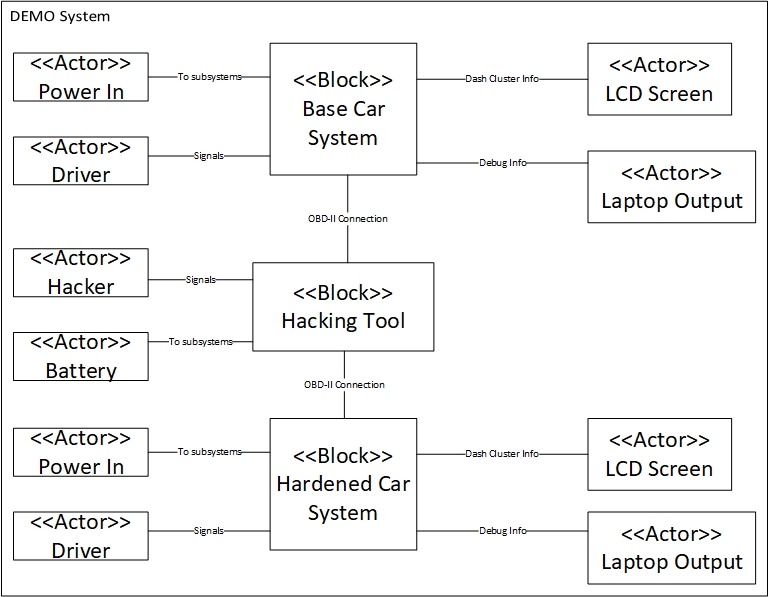
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Stakeholders** | | | | |
| **Feature** | **ID** | **Attribute/Metric** | **RR** | **Supervisor** | **Manufacturer** | **Clients** | **Consumers** |
| Safety | A1 | User Inputs | 1 | 1 | 0 | 1 | 1 |
|  | A2 | ECU Management | 1 | 1 | 0 | 1 | 0 |
|  | A3 | Safety Control Systems | 2 | 2 | 1 | 2 | 0 |
| Security | B1 | Attack Bypass Measure | 1 | 1 | 1 | 1 | 1 |
|  | B2 | Attack Interrupt Measure | 1 | 1 | 1 | 1 | 1 |
| COTS | C1 | Built with Arduino | 1 | 0 | 0 | 0 | 0 |
|  | C2 | Built with Beagle Bone | 1 | 0 | 0 | 0 | 0 |
|  | C3 | OBDII Compatible | 1 | 0 | 1 | 1 | 1 |
| Manufacturability | D1 | Easily Installed | 0 | 0 | 1 | 1 | 0 |
|  |  | Totals | 9 | 6 | 5 | 8 | 4 |

*Figure 3 – Examples of which stakeholders care about what features and how we found their level of importance*

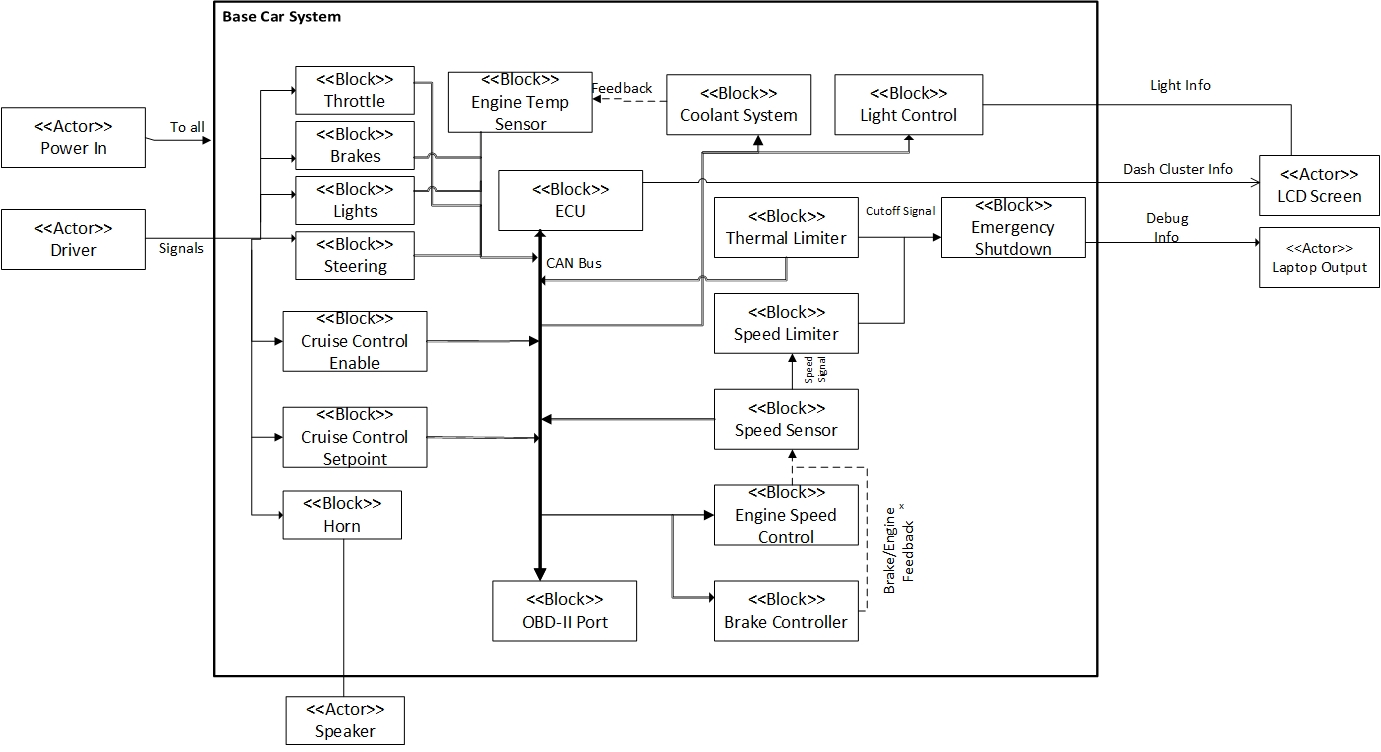
|  |  |
| --- | --- |
| Levels of Importance | Rating Scales |
| No Importance | 0 |
| Some Importance | 1 |
| High Importance | 2 |

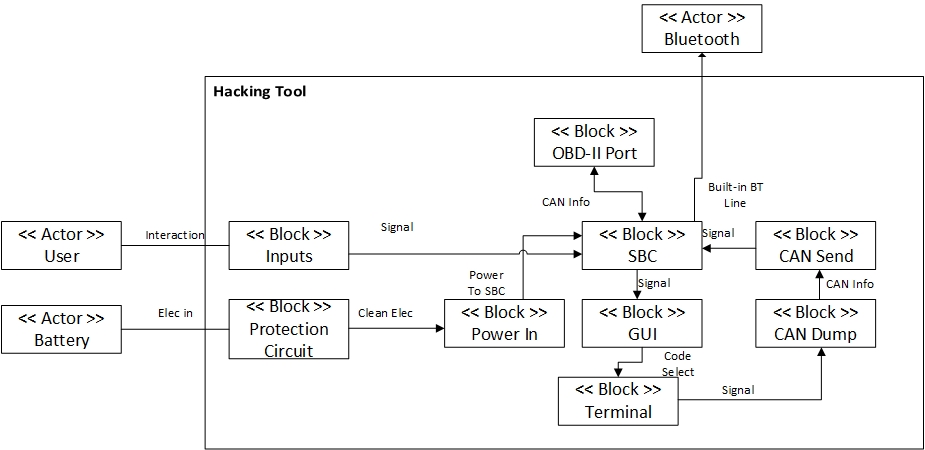
*Figure 4- Rating Scales of the level of importance*

1. ***Functional Architecture Model***



*Figure 5 – Functional Architecture for the Demo System*



*Figure 6 - Functional Architecture of the Base Car System*

*Figure 7 - Functional Architecture of the Hacking Tool*

|  |  |  |
| --- | --- | --- |
| **Actor Name** | **Description** | **I/Os** |
| Driver | The person using the system | User Button Press  User Wheel Turn  Dash Cluster  Speaker |
| User | The person attempting to attack the system | “Debug” messages |
| Power In | The power source for the car system | Power |
| Battery | The power source for the hacking tool | Power |
| OBD-II Port | Diagnostic connection between the car system and the hacking system | CAN Bus |
| Bluetooth | Built in service of the hacking tool that can be used to send information to an external source | Bluetooth |
| LCD Screen | Used to act as a cars Dash Cluster and display information concerning speed, brakes, lights, and emergency notifications | Dash Cluster Info |

*Figure 8 – Definitions of the Actors in the System along with their respective I/Os*

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Related I/Os** | **Related Features** |
| Ignition Switch | Takes a user input [Press] and starts the system [Send signal] | Press  Send signal | Safety  Security  Maintainable  COTS |
| Steering Wheel | Takes a user input [Turn] and sends the position data to the ECU | Turn  Send position | Safety  Security  COTS |
| Gas Pedal | Takes a user input [Press] and send the actuation signal to the ECU | Press  Send signal | Safety  Security  Manufacturability  COTS |
| Brake Pedal | Takes the user input [Press] and send the actuation signal to the ECU | Press  Send signal | Safety  Security  Manufacturability  COTS |
| Cruise Control | Takes the user input [Press] and send the toggle signal to the ECU | Press  Send signal | Safety  Security  Maintainable  COTS |
| Light Toggle | Takes the user input [Press] and send the toggle signals to the ECU | Press  Send Signal | Safety  Maintainable  COTS |
| ECU | Takes input signals and directs them to the Throttle control, Light control, Brake control, and Engine | Receive signals  Send signals | Safety  Security  Maintainable  COTS |
| Throttle Control | Takes the Throttle position input and converts it to a percentage for the throttle and for the dash cluster | Receive signal  Send throttle percent  Send cluster info | Safety  Maintainable  COTS |
| Light Control | Takes the light state signal and sends it to the headlights, HUD lights, cabin lights, or turn signals | Receive signal  Toggle light | Safety  Maintainable  COTS |
| Brake Control | Takes the braking force signal and sets the brake pressure | Receive signal  Send brake pressure  Send cluster info | Safety  Maintainable  COTS |
| Engine | Takes the engine signals and sets the engine motion | Receive signal  Output motion | Safety  Maintainable  COTS |
| Throttle | Takes the throttle percent and adjust the mechanism accordingly | Receive percentage  Set throttle line flow | Safety  Maintainable  COTS |
| Headlight | Takes the light state and toggles the headlights on/off depending on if the signal has changed | Receive signal  Toggle state | Safety  Maintainable  COTS |
| HUD light | Takes the light state and toggles the HUD light on/off depending on if the signal has changed | Receive signal  Toggle state | Safety  Maintainable  COTS |
| Turn signals | Takes the light state and toggles the turn signals on/off depending on if the signal has changed. The lights will then flick on/off on a set interval if the state is on. | Receive signal  Toggle state | Safety  Maintainable  COTS |
| Cabin Light | Takes the light state and toggles the cabin lights on/off depending on if the signal has changed. | Receive signal  Toggle state | Safety  Maintainable  COTS |
| Brakes | Takes the brake pressure signal and applies the necessary force to meet the signal | Receive signal  Set brake pressure | Safety  Maintainable  COTS |
| Dash Cluster | Takes in cluster info and displays it on the dash LCD | Receive signals | Safety  Maintainable  COTS |

*Figure 9– Table of Functions from the Functional Architecture*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Interaction | Description | Involved Actors | Related I/Os | Related Features |
| Steering Wheel rotates | Steering wheel rotates and sends electrical signal to a controller that then turns the wheels | User  Power Supply | User input  ECU to wheels  Steering Wheel to ECU | Safety  Security |
| Gas Pedal being pressed/released | Gas Pedal is either pressed to add more fuel to the engine, or released to stop the engine from receiving fuel | User  Power Supply | User input  Pedal to ECU  ECU to throttle control  Feedback to Dashboard cluster | Safety  Security  Manufacturability |
| Brake Pedal being pressed/released | Brake pedal is either pressed to enable the braking of the car, or is released to allow the car to start moving forward | User  Power Supply | User input  Pedal to ECU  ECU to brake control  Feedback to Dashboard cluster | Safety  Security  Maintainable |
| Headlight Toggle being turned on  \*this interaction applies for all lights on/in the vehicle | The user turns the lights of the car on or off by turning the knob | User  Power Supply | User input  Toggle setting to ECU  ECU to lights  Feedback to Dashboard cluster | Safety  Security  Maintainable |
| Ignition Switch being pressed | Pressing the ignition switch sends signal to ECU to start the vehicle motor | User  Power Supply | User input  Ignition switch to ECU  ECU to Engine  Feedback to Dashboard cluster | Safety  Security  Maintainable |
| Power Source supplies electrical power | The power source sends usable power to all auxiliary systems within the vehicle | Power Supply | Power Supply to the entire system | Safety  Security  Maintainable  Affordable |
| Cruise Control adjusts throttle | The cruise control determines how much fuel the throttle is supplying to the engine and therefore controls the speed of the vehicle | User  Power Supply | User input  Cruise control signal to ECU  ECU signal to the throttle  Feedback to Dashboard cluster | Safety  Security |
| Horn bellows loud sound | The user can press on the horn to activate a loud sound | User  Power Supply | User input  Button in center of steering wheel to the horn  Feedback to Dashboard cluster | Safety  Security |

*Figure 10 – Interactions between Actors and I/O*

1. **Validation of Models and Final Signatures**

Client:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Supervisor: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team Member 1: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team Member 2: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team Member 3: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team Member 4: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_