An Extensible CAN Bus Reverse-Engineering Methodology

Christopher Hoder
Grayson Zulauf
Ted Sumers

Thayer School of Engineering
Dartmouth College
Hanover, NH 03755

March 5, 2013

Contents

1 Introduction

Siege requested an extensible reverse-engineering methodology for cars' CAN buses as a deliverable for the project. To be successful, a hacker needs a large library of information about the target car, detailing the operation and protocols for each ECU, as well as their respective interactions. This information, however, is vehicle-specific, and lacks value when applied to exploiting the vulnerabilities of other vehicles. Instead, Siege requested the *methodology* behind obtaining the information, which can then be used to build knowledge and eventually hack any vehicle utilizing the CAN protocol. This methodology was developed in tandem with our software package and refined through experimentation.

At the highest level, the methodology follows a familiar process for system characterization. Any system can be characterized by its response to input signals, and a thorough understanding of our system (i.e. the electrical network present on our experimental car) would allow the a) prediction of the packets outputted in response to a physical input and b) prediction of the physical response ("output") in response to inputted packets. Our methodology first requires the characterization of the unperturbed, background state, before developing the transfer function to predict output responses based on various inputs, as shown in Figure 1. This method allows the user to quickly move from the black-box state of a car to an impactful hack.

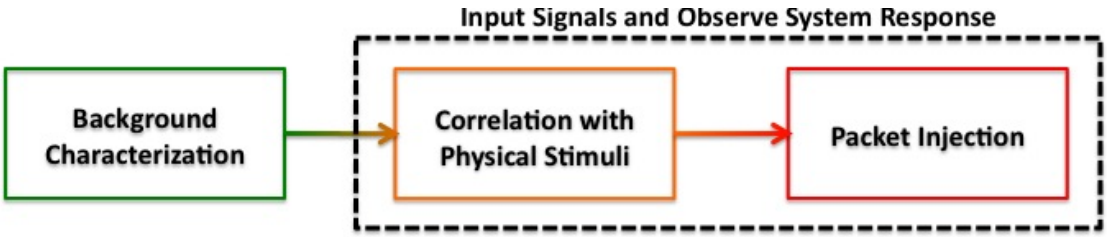

Figure 1: High Level Methodology Flowchart for hacking an automobile's CAN bus. The three main components to reverse-engineering the system are outlined and the appropriate experiments are addressed inturn

This packet includes a rigorous outline of each general experiment necessary to hack the CAN bus on a modern automobile, with the format of the experiments drawn from Montgomery's *Design and Analysis of Experiments, 7th Edition.* Each experiment includes detail of the necessary inputs and expected outputs, as well as a methodology of how to conduct the experiment for both hack expediency and completeness. Our team followed

the methodology, starting with a black-box car, to implement a successful hack on a 2004 Ford Taurus in under 4 hours.

2 Background Analysis Experiments

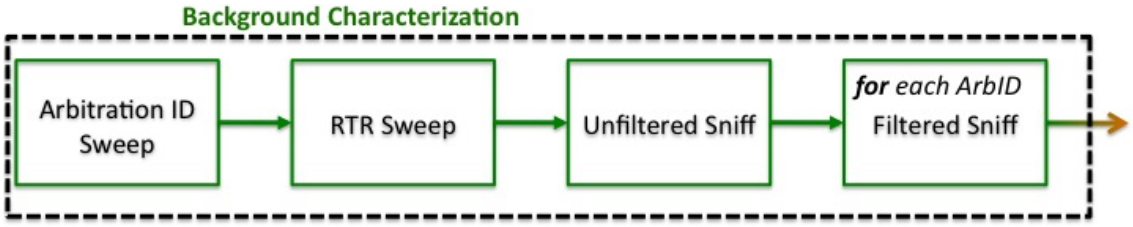

Figure 2: The four experiments required to characterize the background state of the car. The ArbID and RTR Sweeps ensure that the user captures all ECUs on the network, while the sniffs characterize the background packet transmissions and frequencies.

2.1 Arbitration ID Sweep

Objective: Characterize the ArbIDs that are transmitting on the network during steady-state operation.

Control Variables: Car is on in park, all adjustable components are turned off; doors are shut, seatbelts are clicked in; parking brake is up; brakes are released.

Response Variables

1. Observed data packets on CAN bus

Methodology: Sweep across all Arbitration IDs, using `Sweep Std IDs` function included in the GUI. This experiment will filter on each Arbitration ID across the range specified by the user, filtering on each for the amount of seconds inputted in Time.

Repeat for: None.

Inputs:

1. Seconds per ArbID
2. ArbID range

Outputs:

1. Catalogue of ArbIDs that transmit during steady-state operation

Analysis Questions: What ArbIDs are communicating when the car is in this state? How active is the bus with no external stimuli introduced?

2.2 Remote Transmission Request Sweep

Objective: Characterize the ArbIDs that are on the network but not transmitting during normal operation by sending Remote Transmission Requests to all possible ArbID values.

Control Variables: Car is on in park, all adjustable components are turned off; doors are shut, seatbelts are clicked in; parking brake is up; brakes are released.

Response Variables

1. Observed data packets on CAN bus

Methodology: Sweep across all Arbitration IDs and request a response from each, using `RTR sweep response` in the GUI. This experiment will send a remote transmission request (RTR) to each ArbID in the inputted range, and then listen for a response from that ArbID for the user-specified time.

Repeat for: None.

Inputs:

1. Seconds per ArbID
2. ArbID range

Outputs:

1. Catalogue of ArbIDs on the network but do not transmit during steady-state operation

Analysis Questions: What ArbIDs do not transmit during normal operation? How many ArbIDs are on the network in total?

2.3 Unfiltered Passive Listening

Objective: : Characterize the common ArbIDs and their "steady-state" data bytes. Characterize the holistic network operation in the steady-state.

Control Variables: Car is on in park, all adjustable components are turned off; doors are shut, seatbelts are clicked in; parking brake is up; brakes are released.

Response Variables

1. Observed data packets on CAN bus

Methodology: `Sniff` for 120 seconds with no physical stimuli introduced to the car and no set filters. Use this to characterize the unfiltered background steady state of the car in three separate modes: with the car on, with the ignition on but the engine off, and with the car off. Especially note IDs discovered in prior experiments that do not appear here.

Repeat for: None.

Inputs:

1. Seconds

Outputs:

1. Updated ArbID catalogue, in all three car modes.
2. Background data packets for each ArbID

Analysis Questions: What is the percent idle time? What ArbIDs are communicating in each state? For each ArbID, what data packets can we expect to see? How active is the bus with no intentional stimuli introduced?

2.4 Filtered Passive Listening

Objective: : Characterize the common ArbIDs and their "steady-state" data bytes.

Control Variables: Car is on in park, all adjustable components are turned off; doors are shut, seatbelts are clicked in; parking brake is up; brakes are released.

Response Variables

1. Observed data packets on CAN bus

Methodology: For each Arbitration ID discovered above, filter and `sniff` for 15 seconds with no physical stimuli introduced to the car, in each of the three modes described above. Use these packets to characterize the background behavior for each ID, graphically (plot each data byte versus input time) and with the range of values for each data byte.

Repeat for: Each ArbID, car engine off, battery on.

Inputs:

1. Seconds
2. ArbID Catalogue

Outputs:

1. Updated ArbID catalogue, in all three car modes.
2. Background data packets for each ArbID

Analysis Questions: What is the percent idle time? What ArbIDs are communicating in each state? For each ArbID, what data packets can we expect to see? How active is the bus with no intentional stimuli introduced? What are the background packets for each ArbID?

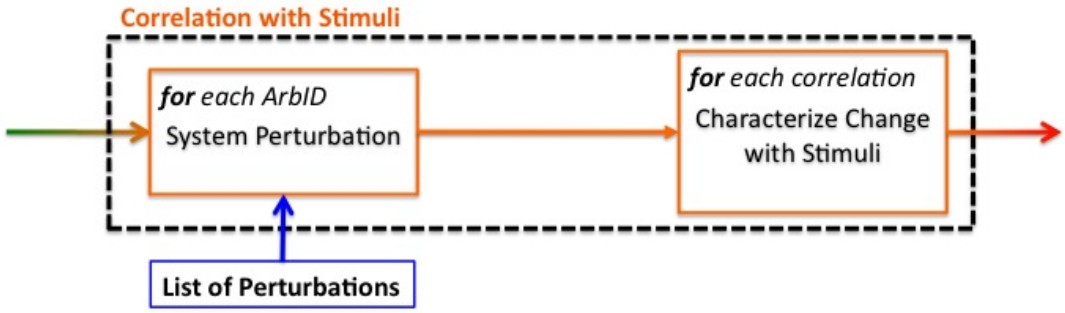

Figure 3: The two experiments used to predict packets from a physical stimulus. All perturbations will be run for each Arbitration ID on the network, and then each ArbID-Perturbation combination must be characterized. The list of perturbations is developed from determining a list of components or actions that are believed to be communicated on the electrical system

3 Correlation with Stimuli Experiments

3.1 System Perturbation

Objective: : Observe CAN bus packet response to injection of physical stimuli to the system (i.e. revving the engine). This will allow us to begin correlating ArbIDs and their data payloads to the change in state of physical components of the car.

Control Variables: Car is on in park, all adjustable components are turned off (except for the component under test); doors are shut, seatbelts are clicked in; parking brake is up; brakes are released.

Response Variables

1. Observed data packets on CAN bus

Methodology: Do a complete system perturbation run to see if each ID is correlated with physical stimuli. It is recommended to order a wiring diagram of the car, and document the modules connected to the CAN bus. Through this documentation, a systematic order should be defined for all of the components on the bus, so that for each Arbitration ID, the same physical components are altered, in a set order. If a change is observed in reaction

to any physical stimuli (via observation in the GUI, in the SQL Database, or graphically), repeat the fiddling until the exact physical stimuli have been identified.

Repeat for: All Arbitration IDs

Inputs:

1. ArbID Catalogue
2. List of system perturbations to be introduced

Outputs:

1. For each ID, a list of the physical stimuli that cause an observable change in the packets transmitted or the frequency at which they appear.

Analysis Questions: 1. Were there any new arbIDs? Do any specific arbIDs always follow each other? Is there a response delay? Is there a refresh rate of the packet where the same packet is repeated on the bus during component stimulation? Does only a particular data byte change? How many bytes change? What is their range? Are the bytes discrete or continuous?

3.2 Characterize Change with Stimuli

Objective: : From the prior experiment, further refine our understanding of those ArbIDs that do respond to stimuli, to the point that we may predict the relationship between a physical stimulus and the exact packets that appear on the bus.

Control Variables: Car is on in park, all adjustable components are turned off (except for the component under test); doors are shut, seatbelts are clicked in; parking brake is up; brakes are released.

Response Variables

1. Observed data packets on CAN bus

Methodology: For each observed correlation and each ArbID, `sniff` and filter for only the respective ID and only change the stimulus recorded above. Use these experiments to characterize the change from the background state for each ID, and ideally for each data byte in the respective ID. Note how different states/value of the physical stimulus change the values transmitted by the ID (e.g. consider when the wipers are on and off, but also look at the intermediate values).

Repeat for: All Arbitration IDs, all perturbations.

Inputs:

1. Background characterization for each ID
2. List of physical stimuli that caused a change in each ArbID

Outputs:

1. Deviation from background characterization for each ID and physical stimulus pair

Analysis Questions: 1. Were there any new arbIDs? Do any specific arbIDs always follow each other? Is there a response delay? Is there a refresh rate of the packet where the same packet is repeated on the bus during component stimulation? Does only a particular data byte change? How many bytes change? What is their range? Are the bytes discrete or continuous?

4 Packet Injection Experiments

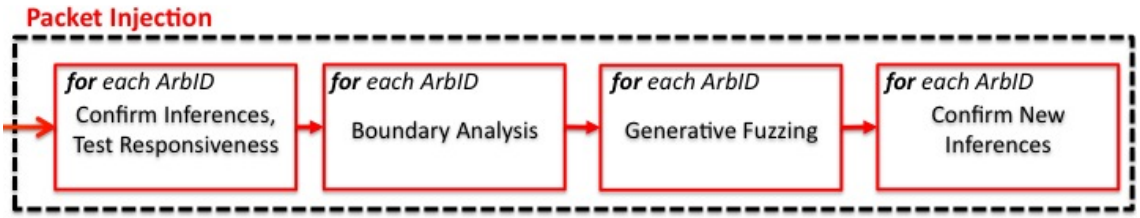

Figure 4: The four experiments used in the packet injection portion of the experimental methodology. These experiments outline a methodology for testing hypotheses about component correlations with network communication as well as fuzzing experiments to discover new ones.

4.1 Confirm Inferences, Test Responsiveness

Objective: Gain confirmation of our inferences about what data is being transmitted by a given ArbID by injecting the ArbID-packet combinations discovered in the last experiment into the CAN bus and monitoring the responses.

Control Variables: Car is on in park, all adjustable components are turned off; doors are shut, seatbelts are clicked in; parking brake is up; brakes are released.

Response Variables

1. Observed data packets on CAN bus
2. Physical response from component

Methodology: Confirm inferences from *Correlate Stimuli*, Section 3, by injecting packets and observing physical stimuli. In this stage, we should be able to predict the physical response to certain packets. For some ArbIDs, however, the predicted physical stimulus will not occur this does not necessarily imply our inference from *Correlate Stimuli* was incorrect, and instead likely implies that this ArbID is listen only or state-dependent.

Repeat for: All Arbitration IDs-Packet combinations discovered in *Correlate Stimuli*.

Inputs:

1. Physical stimulus and packet correlation for each ArbID

Outputs:

1. ArbIDs and stimuli that confirm inferences
2. ArbIDs and stimuli that change our inferences
3. ArbIDs and stimuli that do not respond to packet injection

Analysis Questions: Does physical response match expected response based on prior experimentation? If it is different, how so? Is the component listen-only? Was the same component affected? Does CAN traffic change after injection? Does this change match previously observed CAN traffic?

4.2 Boundary Analysis

Objective: Systems have a tendency to fail on boundaries; thus, we will test the boundaries of observed data bytes for a given arbID. There are 4 boundary values of importance: just before and just beyond the expected range of values, and the absolute minimum and maximum possible data values (0d00 and 0d255, respectively).

Control Variables: Car is on in park, all adjustable components are turned off; doors are shut, seatbelts are clicked in; parking is break up; brakes are released.

Response Variables

1. Observed data packets on CAN bus
2. Physical response from component.

Methodology: Perform boundary analysis on all ArbIDs, by injecting 0d00, 0d255, and the "one-beyond" values into the relevant data bytes using the `Write` feature in the GUI. Observe any physical stimuli in response to these injections. If physical stimuli are observed, go back to *Correlate Stimuli*, Section 3, to further refine your understanding of these particular packet and physical response combinations.

Repeat for: All Arbitration IDs on the bus, all data bytes on each ArbID.

Inputs:

1. ArbIDs catalogue

Outputs:

1. ArbID and boundary value combinations that have observable responses

Analysis Questions: Are there any abnormal behaviors? Component failures? Error lights? Any new packets on bus? Does traffic increase? How can we change the packet to induce a different response? How many bits do you need to change? Are the bit changes Boolean? Or linear? How sensitive is the component to a change in bit? How many bits need to be changed to generate a noticeable value shift?

4.3 Generative Fuzzing

Objective: Explore the protocol structure for signals not yet discovered by randomly creating correctly formatted data packets.

Control Variables: Car is on in park, all adjustable components are turned off; doors are shut, seatbelts are clicked in; parking brake up; brakes are released.

Response Variables

1. Observed data packets on CAN bus (percent traffic, actual packets).
2. Physical response from component.

Methodology: For each ID, use generative fuzzing (`Experiments->Generative Fuzzing`) to test for further physical stimuli. Place in a time that is equal to or faster than the ID frequency characterized in Experiment 4. Use the full range of 0d00 to 0d255 as a default for the values to fuzz over, but refine this range based on observed correlations or to test only certain data byte values. Note any new physical responses observed, the time at which they occurred, and save the associated fuzzing file with the injected packets.

Repeat for: All Arbitration IDs on the bus.

Inputs:

1. ArbIDs catalogue
2. Known correlations
3. Fuzzing ranges

Outputs:

1. ArbIDs with responses to fuzzing
2. Timestamps of physical responses
3. File with Injection Packets

Analysis Questions: Do we cause any unexpected behavior? Do we have an ArbID correlated to the component affected? Did it respond to the input itself? If not, have we seen the response received before?

4.4 Confirm New Inferences

Objective: Generate and confirm the new inferences predicted during *Boundary Analysis*, Section 4.2, and *Generative Fuzzing*, Section 4.3.

Control Variables: Car is on in park, all adjustable components are turned off; doors are shut, seatbelts are clicked in; parking brake up; brakes are released.

Response Variables

1. Observed data packets on CAN bus (percent traffic, actual packets).
2. Physical response from component.

Methodology: Generate and confirm new inferences from fuzzing reduce packets from successful fuzzing until exact data bytes and packet combinations are identified. Re-inject the saved fuzzing file (`Write->Inject from file`), reducing the number of packets included in the file until the packet(s) causing the physical response has/have been identified. Return to *Correlate Stimuli*, Section 3, to confirm these inferences, by predicting the response to appropriate physical stimuli.

Repeat for: All new stimuli observed in response to fuzzing and boundary analysis.

Inputs:

1. ArbIDs catalogue
2. Known correlations
3. Fuzzing ranges
4. Fuzzing packet files
5. New correlations

Outputs:

1. ArbID-packet combinations for new physical responses

Analysis Questions: Which packet or combination of packets is causing the new response? Was the fuzzing only affecting one ID? If not, what combination of IDs is necessary to generate the physical response? Can this be repeated by reducing the number of packets in the inject-from file? Does this occur every time fuzzed packets are injected, or only sometimes?