Heavy Vehicle Security Testbed

Aniruddha Malkar, Dr. Indrakshi Ray, Database Management Systems, Colorado State University

***Abstract*— With scaling in technology, the ECU count inside the vehicles are increasing. These ECUs requires complex communication between them to securely perform some complex operations. To enable such type of intra vehicle communication CAN protocol was proposed. The integration of CAN network connecting ECUs with the other wireless, internet and standard interfaces the vehicle became extremely vulnerable to attacks. Recent studies and experiments have shown that such attacks are fishable. To provide security from such attacks the researcher must understand the working of control systems in vehicle, that needs deep reverse engineering to understand their working and signals flows. To obtain the detailed analysis of such cyber physical system the researchers need to obtain the complex interaction between hardware components and control software. Therefore, the software simulator has limited results for the real time input. The researchers spend their maximum time, money and energy to build an experiment and tools to study & capture the vehicle behavior. There is a high demand for the open source accurate real time testbed with capturing tool so that researchers can spend their time on actual research that build the experiment. There is a steep barrier for the open source testbed because of the cyber physical nature of the vehicle, by reveling some sensitive information of manufacturer and vulnerabilities can lead to high damage to vehicle vendors and user. The heavy vehicles carry most of the good in United States by attacking such vehicle attacker can obtain huge revenue. The heavy vehicle has huge mass though their internal controlling structure is same as other vehicle classes. Such high mass can not only endanger the passenger's lives but also peoples surrounding the vehicle.**

This paper present Heavy vehicle testbed, which is an open source real time physical testbed that provide the Deter like feature with high security to manufacturer and the testbed user through attribute-based access control and customized VM support. This testbed opens the doors for the researchers who want to study, experiment and propose solutions for it with keeping the cyber physical nature of the vehicles.

I. Introduction

The vehicles are considered as heavy vehicles when their gross vehicle mass (GVM) and aggregation trailer mass (ATM) is more than 4.5 tons as defined by Heavy Vehicle National Low. Around 8,156,769 heavy vehicles are manufactured in United States. There are more than 115 million heavy vehicles are register in United States. Around 8 decades, the mass production of automobiles mostly remained static; with a gasoline combustion engine with four wheels and the driver interfaces such as steering wheel, accelerator, gearshift & breaks. But from the last 2 decades the underlying control system have changed drastically. They are no longer an assembly of mechanical systems; but these mechanical systems are exclusively monitored and controlled by tens of digital computers through intra vehicular networks. With the technology scaling the heavy vehicles can support more and more features in them. With increment in features, more and more ECUs are getting added into heavy vehicles with thousands of lines of code. An ECU is an application specific microprocessor/ microcontroller (ASIP/ASIC) that takes the input from the sensors, computes the data and controls the actuators output. These application specific embedded systems are resource constraint devices that has limited security, processing, storage, networking, power supply and ports capabilities. The ECUs manages the driver display, emission controls, engine timing, breaking, steering, safety critical operation and more.

These ECUs require communication between them to make complex decision & to perform critical operations. A usual heavy vehicle uses around 80 to 120 ECUs to control vehicle parameters. Connecting the ECUs with Mesh networking topology will create a mess inside the heavy vehicle due to limited space and structure design. Therefore, to overcome from such problems most of the manufacturer uses the CAN bus network to connect ECUs. The Control Area Network bus is an elegant vehicle bus standard designed and developed to enable the communication in between microprocessors / devices applications without host computer. CAN is a message protocol developed using multiplexing the electric wiring.

The CAN bus protocol was designed to carry the just ECU communication traffic therefore CAN protocol has limited bandwidth that supports low data rate requirements of ECUs. The automobile not only provides features, control and safety operations but also it enables maintenance through OBD port and telematics unit by the reading the error code using testing tool and reprogramming the ECUs which reduces maintenance cost. Over the time the vehicle manufacturing companies have added information systems interface such as music/ video players with USB, DISC, iPod interface & Standard monitoring, debugging and control interface such as OBD, short range wireless interface such as Bluetooth, remote keyless entry, tire pressure, RFID car keys, CICAS-V,WIFI & hotspot etc. & long range wireless interface such as Global positioning system (GPS), Satellite radio, Digital radio, radio data system, traffic message channels & telematics unit (crash reporting, diagnostics, anti-theft, hands free data driving directions/ weather) in the vehicle network. In United States, the federal has mandated the On-Board Diagnostics (OBD-II) port that has direct access to the vehicular network. The CAN Network was designed and developed before interfacing it with the wireless interface, information system interface and OBD interface. Therefore, the CAN Network don't have provision for the risk added by the interfaces. The vulnerabilities of CAN network & ECUs are exploited through such added interfaces.

In the year 2015 Chrysler had recalled around the 1.5 million vehicles because of the security vulnerabilities exploited in the vehicle. The vulnerability allows remote code execution inside the vehicle through vulnerabilities in telematics unit via cellular network. To patch such vulnerabilities vehicle manufacturer

where working on Over-The-Air (OTA) updates. This system has authorization to modify/ update the firmware of ECUs, telematics units, information systems, navigation systems and information system apps updates. The telematics unit connects the automobile network and subsystems with the remote command center through wide area cellular network. The companies like General motors have modified the telematics unit (OnStar) that it provides features like automatic crash response, remote diagnostics and reporting stolen vehicle over a long-range link.

The Over-The-Air Update system have endangered the vehicle by allowing the manufacturer to update the firmware remotely, such communication and communicating system can be hacked & by performing integrity britches the attackers can push their own customized firmware/ patches into ECUs. Such continuous remote connection not only provides attackers monitoring and controlling capabilities of the vehicle but also provides mass vehicle capturing capabilities as updates are sent to all the vehicles of same model and manufacturer. Moreover, the automobile companies have proposed car as a platform model for third party development. The Hughes telematics have developed an "app store" for in vehicle applications. The Ford motors have proposed to synchronize the telematics unit with the third-party application platforms. With the third-party integration with the telematics unit further endangers the vehicle security. The newly proposed vehicle to vehicle and vehicle to infrastructure communication further widens the attack surface.

The automotive system has become a cyber-physical system. The research on automotive cyber physical system is challenging due to absence of open source precise model & high potential barrier to enter into system. The simulation software of vehicular networks provides limited/least scope researchers to create, describe, analyze & exploit different experiments in vehicular network in the simulation software. For comprehensive study the researchers need to extract the iterations between the hardware components and controlling software. The heavy vehicle are costly & poses high risk on testing the live vehicle. Also researchers waste their most of the time in building the testbed and building tool to analyze the vehicle. Moreover, the automobile manufacturers build manufacture specific propitiatory application layer software on top of the standard protocol that extend the standard protocol. Where they are not standardized, well documented and out for public this leads researchers to build own test-bed if they want to research on any model. Furthermore, they are often confined by restrictive agreement by the automobile manufactures which prevent researchers to release their testbed [4].

Apart from weight the heavy vehicles are differentiated from small to medium vehicles by manufacturing process. The parts of small to medium size vehicles are manufactured in different places and assembled at manufacturer site but the heavy vehicles parts are manufactured and assembled at one place.

Most of the passenger vehicles follows the common set of standards SAE J1939 specified by SAE International to develop their proprietary specification used for decision making. Such commercial vehicles are more vulnerable to attacks that aims uniquely used SAE J1939 protocol stack. Most of the trades of goods are made through the heavy vehicles. Stealing such goods through remote attacks can lead to huge economic loss. Moreover, the mass of heavy vehicles is much higher than the small to medium size vehicles, remote attacks on such heavy mass vehicles will not only endangers the driver's life but also peoples surrounding it & can lead to mass destruction. Therefore the heavy vehicles are high value targets. Therefore, we must pay serious attention to such problems. The small to medium heavy vehicles uses same vehicle electronics architecture as heavy vehicles although the small and heavy vehicles uses the Flex-Ray and LIN bus, but the main hub remains the CAN network. Therefore, the basic networking architecture of the heavy, medium and small vehicles is same. Moreover, the commercial vehicles poses almost same set of attack vectors as the passenger vehicles, such attacks can be performed using similar toolset. Therefore, we are considering the all class of vehicle testbed in our paper to compare our testbed comprehensively with other vehicle testbeds.

The massive success of Emu-lab / Deter-lab has changed the way of looking towards testbed. The Emu-lab is first ever electronic testbed that has series of computing nodes connected with various network topologies. The researcher who want to perform an experiment ask for the experiment setup and perform the experiment on nodes. The testbed prevents such tests to come out. Such high configurable real time and physical testbed provides accurate outcome of experiment than the simulators. But such testbed doesn't have provision to test/ experiment the heavy vehicle / automotive security problems [5] [6].

This paper presents an open source cyber physical testbed a vulnerability testing platform that facilitates the researchers to perform automotive experiment and reverse engineering with deter-lab like capabilities on a real automotive hardware. It exploits the J1939 protocol and ECU based attack. It is a physical testbed not a simulator or a prototype. To enable the Deter like facilities in the testbed is hard to achieve & has its own problem because automotive industry is more restrictive due to cyber physical nature of automotive system that can lead to safety critical attacks and mass destructions. This paper aims to cover all the novel problems occurred while building the heavy vehicle testbed and their solutions. Due to the similar networking architecture the paper includes all the class of vehicle testbed created by the researchers. The paper also aims to compare the Heavy vehicle testbed with other vehicles security testbeds.

II. PAPER ROADMAP

1. Background
2. Related Work
3. Shortcomings of CAN protocol
4. SAE J1939 description
5. Heavy Vehicle Testbed
   a. Overall design of testbed
   b. Design of remote server
   c. Packet transmission from testbed to VM and Vice versa.
   d. Some critical problems while building testbed
6. Future work
7. Conclusion

III. Background

In the year 1970 the first in vehicle digital controllers came into automobile market because of gorging prices of gasoline & California Clean Air Act which mandated the reduction in pollutant from automobiles. The digital controller was made to regulate engine operations named as Engine control unit (ECUs). Automobile manufacturer were using the ECUs to increase the fuel efficiency & reduce released pollutants. The designed ECUs were measuring the oxygen exhaust fumes & was proportionally adjusting the fuel to oxygen mixture before combustion that leaded to drastic improvement in efficiency. Due to the huge success of combustion control ECUs, the digital controllers got popular in automobile market and since then the ECUs were used to support advance features in automobiles. The automobile manufacturer used ECUs for diagnostics, throttle, transmission, breaks, climate and light controls, external lights, entertainment systems and so on, that leaded to more generalized form of term ECU to Electronic Control Unit. In last few decades the ECU count has increased drastically and reached to 50-120 ECUs with thousands of lines of code.

Most features required complicated interaction between ECUs. For example, Electronic Stability Control (ESC) System captures individual wheel speed, data from accelerometers, throttle position & steering angle. In response of that it regulates the engine torque and wheel speed to increase the traction in the skid condition where vehicle fails to comply with steering angle. In such condition if breaks are applied then the ESC must co-ordinate with Anti-Lock Braking system (ABS), such a way that it applies breaks in parts to reduce the throttle and regulate the steering angle to prevent rolling over of the vehicle. The feature such as Active Cruz control, auto breaking (in absence of user input), pre-crash features, lane assistant system, auto parking & steer by wire features requires further complex communication between the ECUs. The manufacturer proprietary network undergoes many complicated situations such as time to market pressure, documentation overhead, communication overhead between the plants & limited scaling facilities problems. To overcome such problems automobile industry have standardized the in-vehicle networking protocols such as Controller Area Network (CAN), Flex-Ray, for heavy vehicles Penj1939 & some software technologies shared among the automobile manufactures and vendors. Moreover, the distributed nature of the automotive sensor has forced the automobile manufactures to use the standardized protocols rather than developing their own full stack protocols.

Normally, the automobiles consist of multiple buses that covers the various groups of components. The different group of components has different requirement in terms of bandwidth and complexity (e.g. telemetry, sensors). Thought it perceives like the CAN buses are physically isolated but in practice they are not isolated, the CAN buses are bridged together to support complex functionality and operations. The integration of conventional system inside the vehicle (information system and door lock system etc.) with the safety critical systems (breaking, stirring control, throttling control, crash detection) through bridging more endangers the vehicle operations.

In mid 90s the automobile manufacturer began using more powerful ECUs inside the vehicle created Unix like environment & such environment were adding more and more peripheral devices into it like Global positioning system, reach-back which connects the vehicle with manufacturer through back-haul cellular links. The GMs OnStar was the most popular environment among them. The OnStar environment facilitated the remote diagnostic of vehicle. Moreover, the OnStar ECUs continuously capture the crash sensors data and places an emergency calls in crash conditions that not only connect the passenger with the emergency personnel but also provides GPS location of the vehicle to emergency personnel. In 2009 the upgraded OnStar system enabled the remote car unlocking feature (that enabled the OnStar personnel to remotely unlock the doors of automobile) and remote stop feature that stops fuel to engine through remote control in case of stolen vehicle. To enable such functionality the OnStar unit connected all the vehicle safety critical systems to Internet and Verizon cellular network; this trend is then followed by the other automobile manufacturers.

Previously, researchers had firm thought that the external attacks are impossible and to perform the attack on ECUs and CAN network physical access to the vehicle is necessary. Such attacks are criticized as critics demonstrated that the attacker with physical access can hamper security by non-computerized attacks as well by physical damaging the vehicle components.

[1] This paper renewed the thinking of researchers by demonstrating the attacks on vehicle can be performed without any physical access. The paper to explores, experiment and analyze the external attack vector & severity of the attacks that attacker can launch [1]. The paper categorized the study into five parts. The first part Automotive threat model differentiate the technical capabilities (what attackers knows about vehicle and ability to analyze the system with malicious input producing the various outputs) and operational capabilities (describes the requirements to successfully deliver malicious input to the access vector).The threat model was classified into three types interfacing systems: indirect physical access, short range wireless access & long range wireless access. The indirect physical access comprehensively analyzed the attack vector inside the car that doesn't have any wireless communication interfaces in the car such as OBD, Entertainment: Disc, USB, iPod attack vectors. The short-range wireless attacks comprehensively analyzed the attack vector inside the car that has short range (less than 100 m) wireless communication interfaces (attacker needs to be inside wireless area range) in the car such as Bluetooth, Remote key-less entry, Tire pressure, RFID car keys & Emerging short-range channels (Wi-Fi, hots-pots in cars). The long-range wireless attacks comprehensively analyzed the attack vector inside the car that has long range wireless communication interfaces in the car (more dangerous as vehicle can be controlled from any location) such as Broadcast channel, Addressable channels, Stepping back attack vectors. In vulnerability analysis the

researchers described the investigation performed on each attack vector and procured the practically exploitable vulnerabilities for each attack vector that hampers the vehicle security without physical access. In the third part remote exploit control uncovers the attack surface that has two-way communication facilitates the remote monitoring and remote control. This section analyses the capabilities of TPMS, Bluetooth, FM RDS and Cellular channels and type of attack & their capabilities. In the fourth section threat assessment, it explores the vulnerabilities such that the attacker can leverage the external interface to compromise the control. The fifth section synthesis compares the characteristics of attacks took place on vehicle and experimental attacks. With this sturdy approach the researchers have demonstrated that the external attacks are possible and needs a serious attention.

IV. Related work

1. The authors of Truck Hacking: An Experimental Analysis of the SAE J1939 Standard created their own experiment setup to testify and verify the vulnerabilities of SAE J1939 protocol and preform attacks on it.

2. The authors of Comprehensive Experimental Analyses of Automotive Attack Surfaces created their own experimental setup described below:

A. Experiment information:

The researchers have used mid-range sedan models in their experiments. The setup includes the manufacturer standard devices used to diagnose and reprogram ECUs also the setup includes the telematics unit. The researchers have analyzed the messages and signals used to send cars CAN bus to control key component, also they have analyzed the messages and signals used to send cars CAN bus to control key component. Then they inserted malicious code into ECUs such that the code will be persistent in the vehicular network and hamper functionality of other ECUs. The vehicular network is implemented with multiple buses each has subset of ECUs. Such buses are connected with bridge ECUs. Modifying such components by vulnerability exploitation or flushing the attacker can amplify the attack. Such attack can hamper working of several control ECUs. To analyze the attack vectors and to demonstrate the researchers obtained complete control over vehicle the researcher embedded such control and bridge ECUs in their setup. To reverse engineer the firmware, I/O code and data flow the researchers used diagnose and reprogramming manufacturer standard device, also the researchers obtain CAN packet from the CAN bus. Using firmware, the researcher performed 3 operations: row code analysis, situ observation and interactive debugging with control input. At first, they identified the microprocessor type and used industrial standard IDA pro Dissembler to map control flow and identify potential vulnerabilities. Debugging and logging feature facilitates the simple reverse engineering. Situ observation provides the normal operation of ECU and possible vulnerabilities in normal used operation and frequently used code path. At the end the ECUs where tested in controlled environment where inputs are regulated, and outputs are monitored. By using interactive debugger to test memory and analyze the vulnerable code. The researchers have created their own debugger and provided control input and output interface through UART that function as native debugging I/O. The media player and telematics unit are connected to UART that exposes the internal messages through probing into I/O. The ECUs are reprogrammed using manufacturer standard ECU programming tools. For the telematics the researcher used their own driver to export the shell to UNIX OS connected to OBD port for active capture and control.

B. Platform / Setup information:

The Telematics unit is connected to ECUs through the UART which captures the packets between the ECU and telematics unit. The ECUs are first connected to manufacturer specific debugging tool which exploits the vulnerability and launches attacks such as insertion of packets, flush memory, reprogram the firmware. The researchers created debugger to exploits the telematics unit's vulnerability and communication vulnerabilities between the ECUs and Telematics unit. Finally, the comprehensive attack was launched. The attack was combination of vulnerabilities of telematic unit and ECUs/ CAN network. The attacks on ECU/ CAN network was launched through telematics unit vulnerabilities that alter functionality of ECUs/ CAN network. The telematics vulnerabilities were discovered to get access of vehicle remotely & ECU/ CAN attacks exploited to hamper the functionality of the vehicle.

The attacks were launched to Telematics unit through the customized driver which exploits the ECU and its network. The CAN packets were monitored through probe inserted into CAN bus.

Disadvantages of Setup:

The researchers have spent most of the time, energy and money in creating the experiment testbed & drivers to study and analyze the functioning of ECUs, CAN networks and Telematics unit rather than actually working on the exploitation of external attacks.

The implemented experimental setup was costly and took time to repair the parts in case of failure.

The experimented setup was just providing access to one vehicle, if the researchers want to test their attack on another make and model then they need to repeat the process again for the models.

3. The authors of OCTANE: An Extensible Open Source Car Security Testbed created their own open source testbed prototype in Simulink with flowing details:

Octane open source testbed allowed researchers an open source software to reverse engineer and perform experiment on vehicular networks. Octane is a Cyber physical test-bed consist of software packages and hardware framework. The software packages facilitate monitoring and transmission of CAN messages which enabled the diagnostics and debugging. Moreover, the test-bed facilitates automated security testing and reply ECU testing. The packet monitoring software consists of customized packet identification (e.g. identify door lock packet, identify ECU reprogramming request packet) and customized packet transmission (e.g., the ECU re-programming request packet actually initiates the ECU reprogramming process, the identified lights-on packet actually turns on the

lights), this naming is done in XML document format, such decoding facilitates the researchers to analyze the packet functionality and aids reverse engineering process. The software consists of GUI layer and business logic layer (processing and threat layer), presentation layer & hardware middle layer (HMI). The hardware middle layer uses API to connect with hardware layer. The hardware layer is API that hardware device manufacture provided or a code for a hardware. The adapter consists of advance bus control interface software package that controls operation on automotive networks which enables users to choose & configure operation on different hardware controllers. The receive filter is used to highlight and filter the bulk of packets coming from the testbed. The XML filter identifies the known packets. The bit priority filter is used to review the QoS of CAN networks. The Test present provides bus monitor interface. Transmitter select provides selective packets to users. Packet response enables reverse engineering of the vehicle is the IFTHEN feature that is included in the bus monitor GUI. XML Automation provides description to the given packet and the functionality packet. Custom transmit feature allows researchers to select the car type, model series and type of function they want to perform. Transmit function provides transmit interface. Hardware consists of lab network, automobile networks, protecting components, proprietary software.

Disadvantages of Testbed:

The test- bed and users are insecure as the software packages can easily be bypassed and sensitive information about the vendors and users can be revealed to the attacker.

The test-bed lack in configurability as the test need to be done for models only, the ECUs can't be joined according to requirement of researchers in ad-hoc fashion.

It is a prototype not an actual testbed.

Also, the research on this topic was abandoned by George Mason University in 2013.

4. The author of Towards a Testbed for Automotive Cybersecurity created their own testbed, with flowing details:

The test-bed is focused on hardware-in loop (HIL) environment. This environment configured in such a way that the hardware model under tested interfaced with the interfacing simulators such as inputs and output simulators to analyze the behavior of model comprehensively. The research is mainly focused on OBD ports because the CAN bus is directly connected to OBD ports and don't have any encryption and access control and the OBD ports are connected to dongles which has remote access. The researchers have used Vector Informatik GmbH commercially available, real time CAN bus simulator. The CAN simulator supports CAN traffic monitoring, capturing and analysis. The CAN simulator is connected in with hardware in loop with ECUs to analyze the behavior of ECUs and CAN bus. The CAN simulator is configured with descriptive database that facilitates the virtual model of vehicle. Such type of database is difficult to obtain therefore the researchers have used the CANoe vector

Simulator to obtain accuracy of vehicle model. The OBD port simulator is connected with Bluetooth dongle to perform the remote attacks. The CAN messages are injected through the Bluetooth interface. The test-bed is configured in fowling way: At first the Bluetooth is connected to OBD simulator which provides interface to user to insert malicious CAN packets. The OBD simulator is connect with CAN simulator (which accuracy is verified for a model), the CAN simulator simulates the CAN network and enables monitoring, capturing and analysis of CAN packets and responses. The CAN simulator is connected with ECU simulator which simulates the behavior of ECU and provides the response to ECU. The ECU simulator and CAN simulator is connected to Vehicle simulator which describes the behavior of entire vehicle by taking inputs from ECUs and CAN. The test-bed is verified with headlight flickering attack.

Disadvantage:

It is a simulator won't capture the behavior of CAN network and ECU entirely and accurately.

Testbed uses just OBD as external attack vector.

5. The authors of the A testbed for Security analysis of modern vehicle system created their own open source testbed, with flowing details:

The paper provides detailed report of the flexible real-time simulating test-bed and experiment preformed on the test-bed. The test-bed writes a log file for each packet and displays CAN messages in hex format, it also allows user to inject malicious packets. The testbed uses hardware in loop technology to test the ECUs, which is commonly avoided in test-beds. The ECU of the test-bed is modeled in LAB-VIEW software by taking the specification from the manufacturer and verifying the modeled ECU with the manufacturer. The test-bed aim to build a real time simulator that integrates of CAN bus simulated system with emulated information systems. The test-bed is constructed using 5 different units. The ECU unit simulates the ECU units used in vehicles. The simulated ECU listen on the CAN bus and filters out the packets using unique identifier and accept the appropriate packets for the particular ECU. The simulated ECU also broadcast the message using unique arbitration ID. The ECU data is flowing from & into the VCU and CAN gateway through simulated CAN bus. The Vehicle Control Unit (VCU) model is used to simulate the dynamic of vehicle. The module navigates the current position of vehicle and velocity messages broadcast on CAN bus from ECU. The VCU provides supervisory UI to exhibit the virtual vehicle and realistic driving context. The CAN gateway is used to provide two main functionalities, the module simulates in vehicle CAN to Ethernet gateway and facilitate data capturing gateway to capture the data flowing from/to CAN bus and Ethernet. The CAN gateway resides in both the CAN bus as well as Ethernet, to convert the CAN packets into UDP packets and send on Ethernet and convert the UDP packet into CAN and send on CAN bus. It receives every packet on CAN bus and re-transmit it to Ethernet. The CAN gateway exchanges data with inform Information Unit which facilitates a channel to analyze the remote CAN bus attacks. The main purpose of this module is to reside above the CAN and Ethernet networks. The information units preform some processing on these packets and display the results in remote in vehicle telemetry display the information includes speed, direction, location, fuel gauge and battery. The information unit receives all the messages sent from gateway and re-transmit the telemetry data and UI system. All the attacks and experiments are launched from information Unit. The Information Unit exchanges the data using UDP packets with

in vehicle telemetry display. The telemetry unit is a separate unit designed outside of the information unit which displays emulated infotainment system in vehicle. The researchers have experimented swerve to an arbitrary direction to drive the vehicle out of the road, swerve back and forth to roll over the vehicle, accelerate suddenly to crash the vehicle to the object in front, decelerate suddenly to crash the vehicle to the object behind, Denial of Service (DoS) attack and a combination of above actions attacks.

Disadvantages of the testbed:

The researcher of the paper admits that the physical test-bed provides the accurate results but due to flexibility issues they are channeling their research to the software based test-bed.

The researchers have created prototype of the test-bed in LAB VIEW.

6. The authors of the Experimental Security Analysis of a Modern Automobile created their own testbed, with flowing details:

The experimental setup is based on the two 2009 automobiles with same make and model. The researchers have purchased two vehicles to differentially test and validate the results.

The researchers have physically extracted the hardware from vehicle to analyze it into the lab. The vehicle uses CAN protocol to interact with the components and ECUs. The CAN protocol was used by the researchers to access and inspect the individual components. The researchers have used CAN to USB converter to extract the packet and they have used an oscilloscope for the deep inspection of packet and signals flowing inside the CAN bus.

The researcher then conducted experiments on vehicle in stationary conditions. The researchers connected laptop with the standard OBD II port. They used CAN to USB converter to interact with the high-speed CAN network & used ATMEL AT90CAN128 development board with custom firmware to interact with the Low speed CAN network. The researchers also conducted an experiment on running vehicle by interfacing the 802.11 module with CAN bus that was connected to laptop in chasing car. The researchers have created their own customized tool called as CARSHARK that not only captures & analyze the CAN packets, but it also used to inject the packets in CAN network.

The researcher first extracted the manufacturer specific extension to the CAN bus protocol. Then the researchers have performed testing based on commercially available CAN sniffer connected with manufacturer specific diagnostics service tools. Such configuration does not allow capturing of ECU memory, loading custom code into ECU insertion of malicious packets into CAN network. To overcome this disadvantage the researchers have created their own tool that can read ECU memory, modify the ECU firmware and insert malicious packets in the CAN network.

For testing researcher need to create their own tool to monitor and insert packet in CAN network.

Disadvantages of the Setup:

Expensive research needs to buy two cars, oscilloscope to study the CAN network.

Risky experiment performed on road

Just provides detailed idea about same vehicle make and manufacture.

Researchers spent most of the time in developing and assembling the experiment setup and tool to extract the information and perform attacks.

As we are continuously observing in past related work either the researchers spending most of the time in developing and assembling the experiment setup and tool or satisfying their self with the inaccurate and unprecise results produced by the simulator and prototype. Most time and money were consumed in build the experiment and tools rather than actual experiment. The Heavy vehicle testbed overcomes the disadvantage of such experimental platforms and testbeds with unique open source design.

V. SAE J1939 DESCRIPTION

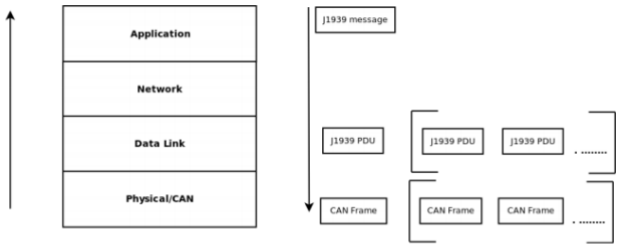


Figure 1: SAE J1939 Protocol Stack

The SAE J1939 protocol operates on top of the CAN protocol. It uses physical communication standard of CAN protocol. As shown in the above figure the J1939 protocol stack is developed and organized with respect to seven layer of OSI networking protocol stack. ECUs formulates the J1939 messages at application layer and transmits the packet in form of sequence of bits at the CAN the physical layer once they are combined into fixed size of Protocol data Units. The J1939 PDU composed of 29 bits identification field and 64 bit data field. The J1939 messages can separately identified using parameter group number (PGN). The information needed to build PGN is integrated into identifier field. The vehicle parameter are included into data field.

Once the message is received by ECU it first obtains the PNG from identifier field. After that according to SAE standards it obtains set of parameters identifiers called as suspect parameter numbers allocated to each PGN. Set of attributes such as starting position, length, resolution, offset and name are assigned to each SPN that are used to interpret the content of data field.

VI. SHORTCOMINGS OF CAN PROTOCOL

CAN bus protocol:

There are various networking protocols proposed by the researchers to connect ECUs and the components. In 2008 the US government standardized the CAN protocol (ISO 11898) mandated the implementation of CAN protocol inside the vehicle that made the CAN protocol dominant protocol in automobile industry. The CAN is a data link layer protocol that uses the publish-and-subscribe communications model rather

than the conventional addressing model. It consists of CAN ID header that specifies the packet type.

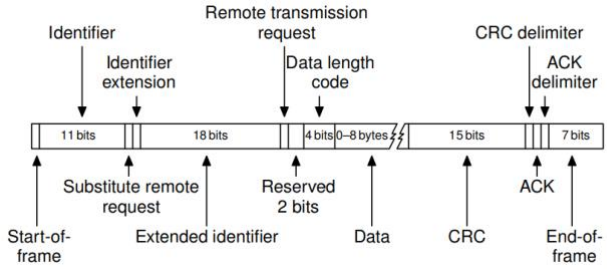


Figure 2: CAN protocol

Each packet in CAN network is physically and logically broadcasted to all the nodes, where the node decides to the process the packet or not. Along with technology the CAN network is updated according to the communication needs. Some connections require higher data rate (ECU to ECU connection, ECU to Telematics unit connection) than other connection (sensor to ECU connections). Such industrial demands are supported using the different signaling and modified wiring. The gateway bridges are used to integrate multiples CAN buses connecting group of devices.

Security challenges:

Broadcasting nature:

The CAN protocol broadcast the CAN packets logically as well as physically to all the nodes. The malicious components can intercept such CAN packets and can insert a malicious packet inside the CAN network which causes confidentiality and integrity britches.

Susceptibility to DoS:

The CAN protocol is highly vulnerable to denial of service and flooding attacks. CAN protocol uses the priority-based arbitration scheme which enables nodes to set dominant state on bus for indefinite time causes all other nodes to back-off because most of the ECUs are logically designed to avoid the accidental breaking of network connections.

Absent of Authenticator field and source identifier field:

CAN packet don't have any provision for the Authenticator field and source identifier field, that enables the any component in CAN network can send packets to any other component anonymously/ indistinguishably. In such design the malicious/ compromised component can control most of the components on CAN network until and unless the components themselves provides the defense.

Fragile Access control:

The CAN network has challenges and response scheme to enhance the security of certain operations like re-flashing the ECU memory and testing. But such scheme has fixed challenging seeds and fixed keys stored in ECUs. Moreover, the encryption algorithms are available in public. The attacker can easily extract the fixed seed and key by snipping the packet from CAN network and reverse engineering such packet using the algorithms available in public. In most of the vehicles, the CAN network uses the hardcoded challenges and a hardcoded response common to all similar units. Exploiting such vulnerabilities will endanger the other ECUs as well.

Furthermore, in the study they have shown that the even such protocol are less broadly implemented, the result of challenge and responses are never used anywhere. Therefore, the attacker can re-flash the ECUs without completing the challenge response scheme. They can load a malicious script into telematics unit without any authentication [8].

According to standards, when ECU detect an unsafe operation ideally it should reject "disable CAN communication" message. But in practice such messages are not rejected by the ECUs. The attacker can take advantage of such messages and disable communication of an ECU in critical conditions.

Moreover, the standards also state that the ECU must rejects the firmware updates and re-flashing event in moving conditions. The researchers have discovered that the engine control module and transmission control module can be set on re-flashing mode in driving conditions, that in turn switch the engine off while driving.

VII. HEAVY VEHICLE SECURITY TESTBED

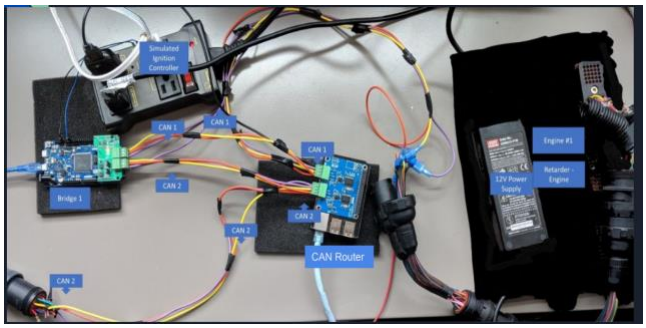


Figure 3: The Heavy Vehicle Security testbed

The Heavy Vehicle testbed facilitates the users with deter like configurability & manageability with very high security. It also allows researchers to implement their own script and allow users to save the script in the database. Such features reduce the researchers' efforts to implement the scripts again and again. These saved scrips are secured with attributed based access control where each saved attribute of the user has session tag which is matched with the credentials' session tags and then the decisions are made to grant or revoke the access. Moreover, the users can implement the ready-made scripts using the vulnerability database. The generalized block diagram of testbed is given below:

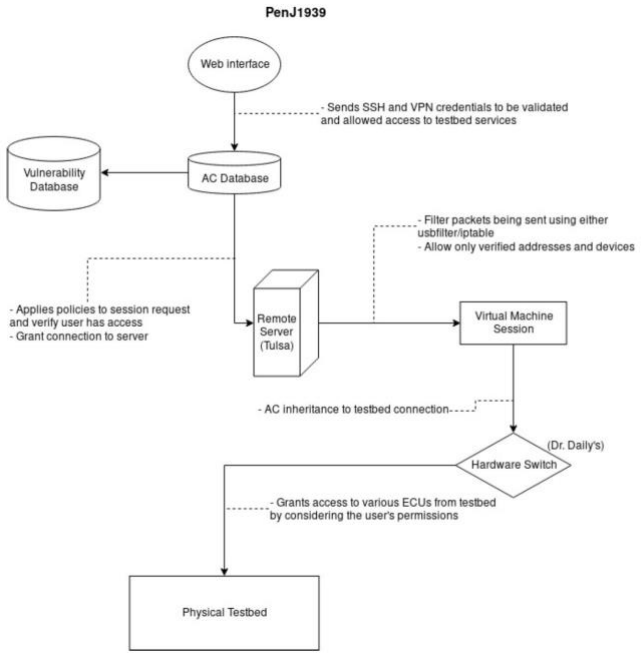


Figure 4: Block diagram 1 of the testbed

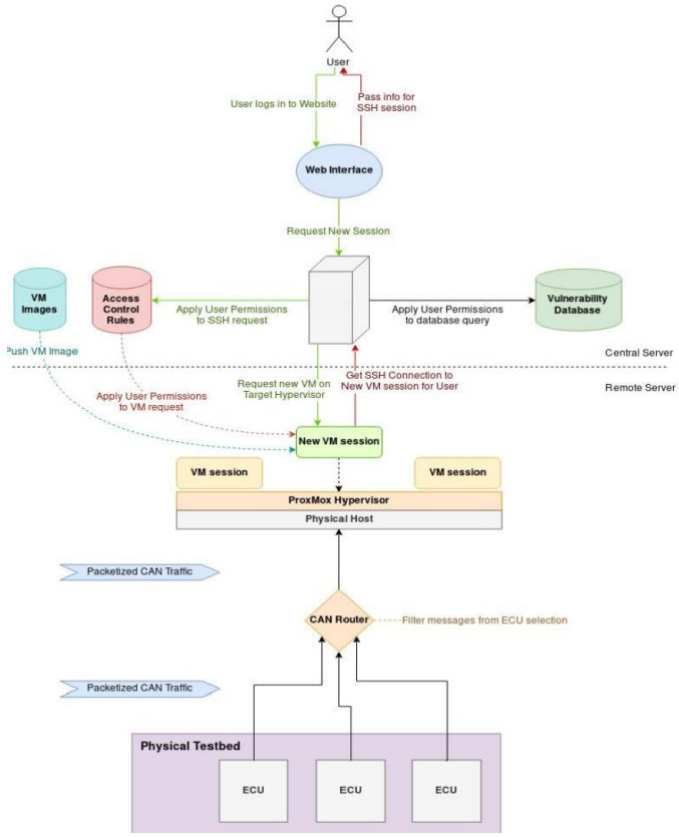


Figure 5: Block diagram 2 of the testbed

I. Operation:

When testbed users' requests for the remote access of the testbed. The access database verifies the users saved credentials with the entered credentials & the decision is made to grant or refuse the access to the testbed. On the grant the users get SSH links on their email. This feature facilitates additional authentication as users not only have to login in into website but also to their emails to access the session SSH key. The vulnerability database stores the vulnerability configurations. When the users request for the readymade vulnerabilities scripts the access control verifies the user's attributes with the scripts attribute & make decision. On a grant these scripts are mounted on physical devices and networks. These vulnerabilities scripts are converted into set of policies that configures the testbed & its environment (ip tables/ usb filter) for the user session. These policies are sent in terms of packets. These policies configure & prepare the virtual machine by embedding dynamic iptables rules and USB filtering rules into it. The iptables rules and USB filtering rules varies with the configuration, vendor make and model. Therefore, the iptables rules & usb filter rules are configured dynamically into OS for the user session. The OS only accept the filter packets from verified addresses and devices inside the testbed. The USB/IP filter are built inside the OS and are configured dynamically. The deployed a Virtual Machine with filter OS which is connected to physical testbed through the Hardware networking Switch. The SSH link allows users to enter into Virtual Machine with filter OS. These virtual machines allow the users to access the physical test-bed & perform experiment on it. The configuration and setup are enforced by remote server in the inspection of access control. The features like USB/IP filtered OS and access to testbed through SSH into VMs (no physical access) hikes and ensures the integrity and confidentiality of the testbed. With such features the testbed, manufacturer and users are protected as not only the packets are not leaked outside of the VM but also outside packets, policies and scripts are inspected before implemented into testbed.

II. Overall Design of Testbed

a) Web-interface: -

Web interface facilitates interaction between users and test-bed. It provides documentation of the testbed to the user and allows user to sign in or sign up into the testbed. When user sign up into the testbed, the user data get stored into the user's database. On sign in the access control server checks the user information in users' database & accordingly it allocates Virtual Machine and testbed resources to the user. The ssh token link is sent to the user's email id registered while signing up in the web. The toke link is time limit that expires after a specific duration. This feature ensures only authorize user can receive the SSH link.

b) AC database:

AC database consist of access control server and user database. The user's database store the users' information and their scripts that they want to implement. The access control server ensures user, service and vender privacy, confidentiality and integrity (this section will be covered in detailed later).

c) Vulnerability database:

The vulnerability database consists of scripts of various attacks that can be implemented on a vehicle. The scripts are stored in the form of ready-made templets & libraries. Such that calling for a particular attack provides scripts for the attack. The testbed facilitates user to study the script of an attack or mount it on testbed. These saved scrips in vulnerability database are secured with attributed based access control where each saved

attribute of the user has session tag which is matched with the credentials' session tags and then the decisions are made to grant or revoke the access.

d) Remote server:

The remote server consists of a bare-metal hypervisor. The hypervisor stores pool of VM. In accordance with the request from the access control, the remote server allocates VM from the VM pool. It provides ssh link, host IP, hostname & password to the access control server which is then sent to users Email ID. This remote-control server is physically attached to the remote testbed through the testbed networking switch. The remote server leverages VM to communicate with the physical testbed implemented for the user. Depending on the users request the remote-control server install the IP rules inside the OS.

e) Hardware Switch: -

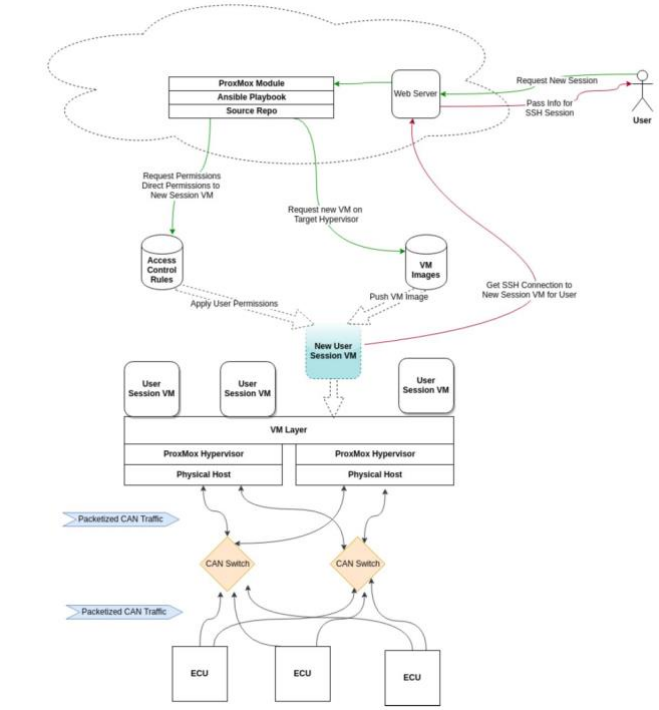
The hardware switch connects the hypervisor with the physical testbed. The request from the remote server and access control server set of ECUs and subnetworks are allocated to the user. The hardware switch connects the allocated ECUs and sub networks and places them into single VLAN network. The VLAN configuration into hardware switch is done by the access control and remote server. The VLAN ensures the isolation between users' data, packet flowing to and from the user & allocated ECU's and subnetwork to the users.

f) Physical Testbed: -

The physical testbed consists of ECU's used in heavy vehicle connected through CAN bus. This ECU subnetwork is connected to hardware switch. Each remote site has its own testbed setup depending the make and model of heavy vehicle. On the request from user for a particular make and model the testbed allocates the remote server and testbed that contain the make & model requested by the user.

III. Design of Remote server

Designing such complex testbed is crucial and each component in the testbed has its own prospective of looking towards testbed. This section provides information about the testbed from the Remote server prospective.



The Deter-lab project provides the access to testbed through ssh link of a node situated inside the testbed. Such functionality increases the usability, testability, configurability and security of the testbed. To enable such functionality in heavy vehicle testbed is challenging. We can't provide a direct access to the node inside the heavy vehicle testbed, as the nodes are ECUs that are inefficient resource to get access. For the comprehensive study the users must able to access the CAN network that shows the status of ECUs and subnetworks. The attacks in automobiles are carried through different way than the computer node attacks. The researchers in heavy vehicles testbed must have access to CAN network to monitor and verify the attack and its causes. To build an application specific cyber-physical that has completely different functionality than a generic cybersecurity electronic testbed requires different level of efforts. To enable such functionality we require a computing node inside the testbed which can communicate with the physical testbed and user can access it remotely. Such functionality is supported by virtualization environment. The virtualized environment support creation and deployment of VMs. The virtualized environment is located inside the testbed configured in remote server. Accessing the Virtual machine that is located inside the testbed enables Deter like functional with keeping the uniqueness of the testbed.

The remote servers are located at manufacturers' place where the manufacturers build the physical testbed with the same ECUs and networks used in a model. The physical testbed and remote server are connected with each other through networking switch. The remote server is configured with the hypervisor that stores pool of VM. After authenticating the user, the user's request is forwarded to access control through the

web server, the ansible script in remote server starts a virtual machine session by deploying the customized OS. For each new session new virtual machine will be created for all the users. The SSH key of these VMs are forwarded to user's email ID. The Virtual Machine are running on top of hypervisor, where hypervisor manages the resources of the VMs. The access control rules are implemented into the virtual machine while creating. The remote server pushes the IP table rules and USB filter rules while installing the OS. In this way the testbed can configure the set of IP table rules and USB filter rules dynamically into the session VM depending on the user requirements for each session.

Web-server: -

Web interface facilitates interaction between users and test-bed. It provides documentation of the testbed to the user and allows user to sign in or sign up into the testbed. When user sign up into the testbed, the user data get stored into the user's database. On sign up the access control server checks the user information in user's database & accordingly it allocates Virtual Machine and testbed resources to the user. The ssh token link is sent to the user's email id registered while signing up in the web. The toke link has time limit which expires after some duration. This feature ensures only authorize user can receive the ssh link. The web server forwards the user request to the ProxMox hypervisor, which spins the VM with the help of ansible script. Once the VM is allocated to the user the VM data such as VM IP, VM name is provided to the user through the ssh link to the user Email.

Remote server:

The remote server consists of a bare-metal hypervisor that stores pool of VMs. In accordance with access control request the hypervisor creates VM from the VM pool using ansible script and provides ssh link and information such as host IP, hostname etc. This remote-control server is physically attached to the remote testbed through the testbed networking switch. The remote server leverages VM to communicate with the physical testbed implemented for the user.

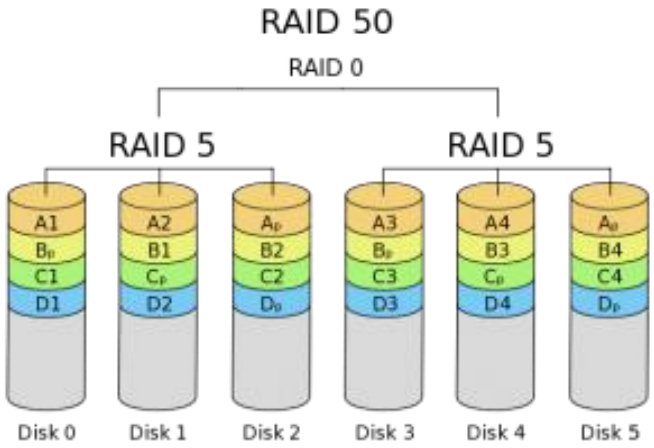
Proxmox virtual environment: Proxmox is an open source server virtualization environment with bare-metal hypervisor. The Proxmox is a Debian based distro with refined Ubuntu LTS kernel. The Proxmox virtual environment facilitates deployment & management of containers and Virtual Machines. The environment includes the web and command line interfaces. The Proxmox enables high available clusters, by distributing the cluster across multiple servers. The Proxmox also supports the live migration where the virtual machine can be moved from one physical host to another without any down time.

In this project we have used the Proxmox virtualization environment to enable the virtualization. We have implemented the Proxmox on HP server machine. Using the Proxmox web interface we have uploaded the guest Ubuntu OS. We have installed OpenSSH package into Ubuntu. This configuration enables the users from outer network to able to get access to virtual machine using SSH protocol. Using the IP address, host name and password we are able to get SSH into the VM from the outer network.

Fault tolerant server:

The testbed has web-based access that enable the Deter like availability to the users. To facilitate such availability the testbed must provide fault tolerant services. In the testbed the remote server is the single point to failure. There is a high need to provide fault tolerance to the hypervisor so that the user can have high availability and continuous access. Such functionality can be provided through RAID system. Redundancy Array of Independent Disk is the data storage virtualization technology that aggregate the several disk drive components into on or multiple logical units to facilitate data redundancy, performance enhancement or both. The data is distributed across the disks in various ways defined by the RAID levels which provide different balance of reliability, availability, performance and capacity.

In the testbed for the remote server we have used RAID 50. The RAID 50 provides significant balance between the storage performance, storage capacity & data integrity. The RAID 50 strips the data across multiple sets of RAID 5 as shown in the diagram below.



Such configuration reduces failures in the remote server and enhances the availability of the remote server and testbed.

Ansible Playbook: The Ansible playbook is a Ansible configuration, deployment and orchestration language. The Ansible scripts repent the policies that users want to enforce into the remote machine or set up a general IT process. The Ansible playbooks are used to deploy a remote machine and manage it configuration. We have created Ansible playbook scrip that to starts a new virtual machine on web server request, installs a customized Ubuntu OS into the virtual machine and configure the dynamic IP table and USB filter rules into the VM. The source repo is used enable source hosting.

Hardware Switch: -

The hardware switch connects the hypervisor with the physical testbed. The request from the remote server and access control server set of ECUs and subnetworks are allocated to the user. The hardware switch convenes allocated ECUs and sub networks and place them into single VLAN network. The VLAN configuration into hardware switch is done by the access control and remote server. The VLAN ensures the isolation between user data, packet flowing to and from the user & allocated ECU's and subnetwork to the users.

Physical Testbed: -

The physical testbed consists of ECU's used in heavy vehicle connected through CAN bus. This ECU subnetwork is connected to hardware switch. The CAN switch are used in CAN network leverages ECUs to communicate with each other.

IV. Packet flow:

The automobiles use CAN protocol to transfer messages between ECUs and components. The CAN protocol is designed to transfer messages in vehicular network. For reverse engineering and examining the attacks on the testbed and its response, user must know such CAN packets flowing through CAN network. To enable such functionality the CAN packet must transferred from physical testbed to users' virtual machine to provide monitor & analysis functionality to the user. Also, the testbed should support packet flow from users' virtual machine to physical testbed to allow testbed users to insert malicious / test CAN packets and verify the results of such CAN packets.

To enable reverse engineering of ECUs and CAN network the Heavy vehicle security testbed provides probing into CAN network. This CAN probing allows the users to monitor and inspect the CAN packets. Also, the CAN probing enables the users to insert the malicious CAN packets into testbed's CAN network and record the response of ECUs and CAN network in real time. The probing is done through hardware networking switch. The networking switch first creates a VLAN of ECUs and CAN subnetworks that are requested by the users. Such VLAN replicates the CAN network used in Heavy vehicles. The networking switch is configured to tunnels all the traffic flowing in the VLAN to the user's virtual machine and vice versa.

The main problem of using this approach is that, the CAN protocol is created and developed to transfer messages inside the vehicle where the hardware switch is configured transfer the message to outside of the vehicle network. To connect entities in the computing environment we need a computing networking protocol. Therefore, we can't use CAN packet as it is to tunnel it into the user VM as CAN protocol don't use IP protocol for connection. The CAN packet contains the CAN ID of a intended component broadcasted over the CAN network. To connect the testbed VLAN with Virtual machine over a network we need to provide IP addresses for identification and correct transfer of packet between the hosts, as CAN protocol fails to transfer packet in computing environment & the users VM environment is not set to decode the CAN packets, we can't use CAN packets as it is to tunnel it to VM.

In such cases we need a packet converting tool that converts the CAN packet into TCP and vice versa to transfer the CAN packet over internet standards to the users VM. The TCP is a reliable protocol guarantees the delivery of packet to the destination. The Transmission control protocol is a connection-oriented protocol that establishes the maintains the connection until the application program finishes transferring of messages. The TCP facilitates communication services at middle level in between the application program and the internet protocol. It enables the Host to Host connectivity at transport layer of the OSI model. At transport layer of OSI model the TCP manages handshaking, flow control, packet lost detection and retransmission, packet out of order detection and rearrangement, duplicate packet detection, reporting the failure of transmission if the data is not delivered even after some attempts, error correction, acknowledgements of arrived packets and so on. Once all the data is received and rearranged into the sequence then the data is passed to the application layer. We are using the TCP protocol in our conversion tool because the TCP provides the reliable, ordered and error checked transmission. Such features of TCP provide the advantage to our testbed. The CAN packet observed must be in order, if the sequence is not match then the testbed users can't comprehensively study and inspect the attack or reverse engineer the setup, also missing some packet can lead to wrong interpretations. Such problem in our testbed is taken cared by the TCP protocol with its reliable, ordered and error free connection. We have created a tool in Python 3 high level language that convert the CAN packets into the TCP packet filled with CAN data. This section converting tool is connected to the sending program of the socket that send the data of CAN network to VM in TCP form. To enable users to insert the malicious packet into the CAN network we have created the conversion tool section that convert the TCP packets from VM to CAN packet such that the CAN malicious packets can in interfaced with the setup provided through TCP packets from the users VM.

The TCP provides the abstraction of network connection to the application layer via network socket interface that connects the host with each other. The network sockets are internal sending endpoints for sending and receiving in the hosts. They represent the endpoint of networking software. The testbed requires bidirectional communication to monitor and interact with testbed CAN network. We can't create a socket with continuous flow of TCP packet in a bidirectional way that enables real time monitoring and interacting capabilities. Therefore, to enable the bidirectional communication we need to create two socket programs for continuous sending and continuous receiving of CAN data for bidirectional communication. These programs are installed on both the ends: the VM end and the switch end. These programs are wasting the resources of the host by running individually, which can lead to dropping of CAN packets. To avoid such problem, we took help of multiprocessing library of Python 3. We integrated the sending program and receiving program into a single program. These two programs are mounted on each core of the processor which makes efficient use of multiple processing architecture. The sending and receiving program run at same time on each core of the host processor enables the Realtime continuous packet exchange along with the efficient use of resources. Such programing increases the speed of TCP sockets that makes it faster than the CAN protocol which ensures no CAN packets are dropped.

The code for the Virtual machine and networking switch is given below:

Code for PI/ bigle bone testbed switch:

```
#piprocessing
import socket
import sys
import multiprocessing as mp


#sending masg from Pi
def pisend(port):

    # create socket server for sending

    try:
        serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # prevent socket.error:[Errno 98] Address already in use
        serversocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

        # setup host
        hostname = socket.gethostname()
        IP = socket.gethostbyname(hostname)

        print('\nServer address is: %s' %(str(IP)))

        # setup bind
        serversocket.bind((IP,port))

        # setup maximum connections, moore than the nummbe need to wait
        serversocket.listen(5)
    except socket.error as msg:
        print(msg)
        sys.exit(1)

    print('Waiting for connection')

    # start connecting to VM

    while 1:
        # accept connection from VM
        pisend, addr = serversocket.accept()


        print('Accept new connection from {0}\nWaiting for typing\n' .format(IP))

        # start sending msg to VM

        while 1:
            #send_msg = input('please input work: ')+ '\r\n'
            send_msg = 'P1'+'\r\n'
            pisend.send(send_msg.encode())
            print(pisend.recv(1024).decode())
            #send exit to stop connection
            if send_msg == 'exit'+'\r\n':
                recv_msg = pisend.recv(1024)
                print(recv_msg.decode())
                break

        pisend.close()

#receiveing msg from VM
def pirecv(port):
    # create socket server for receiveing
    try:
        serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # prevent socket.error:[Errno 98] Address already in use
        serversocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

        # setup host
        hostname = socket.gethostname()
        IP = socket.gethostbyname(hostname)

        print('\nServer address is: %s' %(str(IP)))

        # setup bind
        serversocket.bind((IP,port))

        # setup maximum connections, moore than the nummbe need to wait
        serversocket.listen(5)
    except socket.error as msg:
        print(msg)
        sys.exit(1)

    print('Waiting for connection')

    while 1:
        # accept connection from VM
        pirecv, addr = serversocket.accept()


        print('Accept new connection from {0}\nWaiting for typing\n' .format(addr))
        #clientsocket.send('Welcome to the server!\n'.encode())

        # start receive msg from VM
        while 1:
            recv_msg = pirecv.recv(1024).decode()
            print('{0} client sends data is {1}'.format(addr,recv_msg))
            if recv_msg != 'exit'+'\r\n':
                recv_msg = ' '
                send_msg = 'Data is received.'
                pirecv.send(send_msg.encode())
                print('Waiting for typing')
            # If receive exit. stop the connection
            elif recv_msg == 'exit'+'\r\n' or not recv_msg:
                print('{0} connection closes.' .format(addr))
                end_msg = 'Connection is closed'
                pirecv.send(end_msg.encode())
```

```
            break
        pirecv.close()

if __name__ == '__main__':
    q = mp.Queue()
    p1 = mp.Process(target=pisend, args=(1024,))
    p2 = mp.Process(target=pirecv, args=(2048,))
    p1.start()
    p2.start()
    p1.join()
p2.join()
```

Code for the Virtual Machine:

```
#vmprocessing
import socket
import sys
import multiprocessing as mp

#receiving msg from Pi
def vmrecv(port):

    # create socket client
    try:
        vmrecv = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        # setup host
        hostname = socket.gethostname()
        IP = socket.gethostbyname(hostname)
        # request connection to Pi
        vmrecv.connect((IP, port))

    except socket.error as error_msg:
        print(error_msg)
        sys.exit(1)

        # start receiving msg from Pi
    while 1:
        recv_msg = vmrecv.recv(1024).decode()
        print('{0} client sends data is {1}'.format(IP,recv_msg))
        #Check if receive msg
        if recv_msg != 'exit'+'\r\n':
            recv_msg = ' '
            send_msg = 'Data is received.'
            vmrecv.send(send_msg.encode())
            print('Waiting for typing')
        #If receive exit, stop connection
        elif recv_msg == 'exit'+'\r\n' or not recv_msg:
            print('{0} connection closes.' .format(IP))
            end_msg = 'Connection is closed'
            vmrecv.send(end_msg.encode())
            break
    vmrecv.close()

#sending msg to Pi
def vmsend(port):
    # create socket client

    try:
        vmsend = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        # setup host
        hostname = socket.gethostname()
        IP = socket.gethostbyname(hostname)
        # request connection to Pi
        vmsend.connect((IP, port))

    except socket.error as error_msg:
        print(error_msg)
        sys.exit(1)

    # start sending msg to Pi
    while 1:
        #send_msg = input('please input work: ')+ '\r\n'
        send_msg = 'P2'+'\r\n'
        vmsend.send(send_msg.encode())
        print(vmsend.recv(1024).decode())
        #If send exit, stop connection
        if send_msg == 'exit'+'\r\n':
            recv_msg = vmsend.recv(1024)
            print(recv_msg.decode())
            break

    vmsend.close()

if __name__ == '__main__':
    p1 = mp.Process(target=vmrecv, args=(1024,))
    p2 = mp.Process(target=vmsend, args=(2048,))
    p1.start()
    p2.start()
    p1.join()
p2.join()
```

V. Design of Access control server

The Access control server looks into the user policies and grant or revokes the permission on allocated VM.

Attribute based access control

XCAML

Why you need a access control

Why it is important

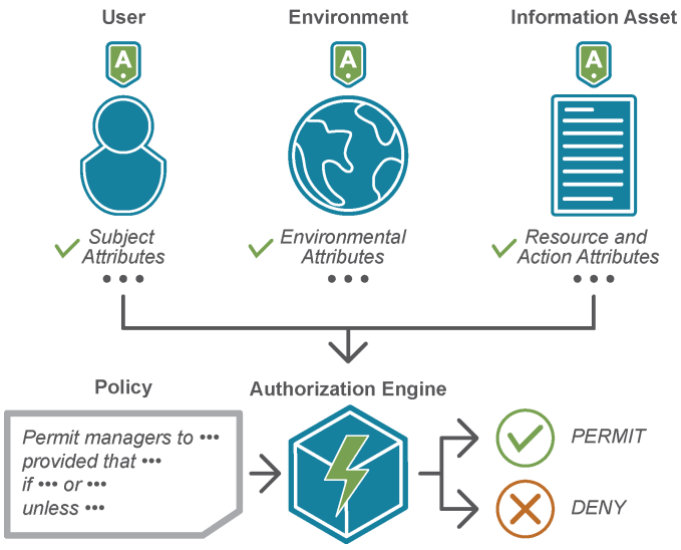
To make an application specific real time web based cyber physical testbed is crucial. There are many modules and elements that require security from outsiders. The security is not all about the encryption and decryption, but it is an infrastructure. To enable security for the testbed we have created our own infrastructure. The main element in the security is access control. The testbed facilitates the implementations of security attacks in testbed, so the testbed must be immune to

such attacks. The testbed must protect the user data and scripts in order to maintain the user's privacy. The testbed must protect vehicle manufacturer from reveling their defects in design in order to prevent manufacturer and its user from getting exploited & to prevent their security. The testbed must protect its services in order to maintain its correct functionality. Such extensive security can only be provided through the access control. The Role based access control and Discretionary access control fails to provide such extensive security to specific attributes of elements used in testbed.

To enable such security, we will be using attribute-based access control to monitor and control the data access.

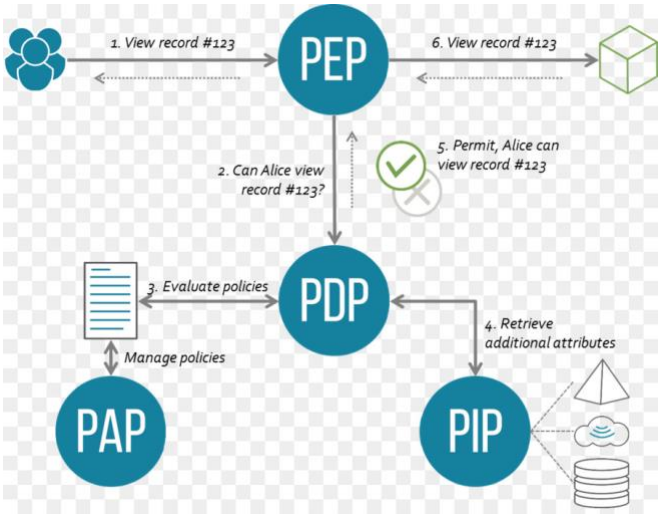
The attribute-based access control is a policy-based access control that grants or revokes the access to the users through policies that integrates the multiple attributes. The policy can use multiple types attributes including users' attributes, resource attributes, object attributes and environmental attributes. The ABAC uses the Boolean logic along with IF and THEN statement to identify the accessing entities, their requested resources and action associated with it. The ABAC is considered as a next generation authorization because of its dynamic, context-aware & risk intelligent access control.

On the request from the user the ABAC checks the subject attributes, environmental attributes and resource and action attributes, based on their declaration the authorization engine makes a decision to permit or deny the access.



The architecture of ABAC consists of following entities:

The Policy Enforcement Point protects the resources and data that the owner wants to protect. The PEP investigates the users request and creates an authorization request that is sent to PDP. The policy Decision Point is the brain of ABAC architecture. PDP evaluates the incoming request with the polices configured with the entities. The PDP returns the Permit/ Deny decision. The PDP also uses PIP for the missing meta data. The Policy information point (PIP) connect the PDP with external sources of attributes.



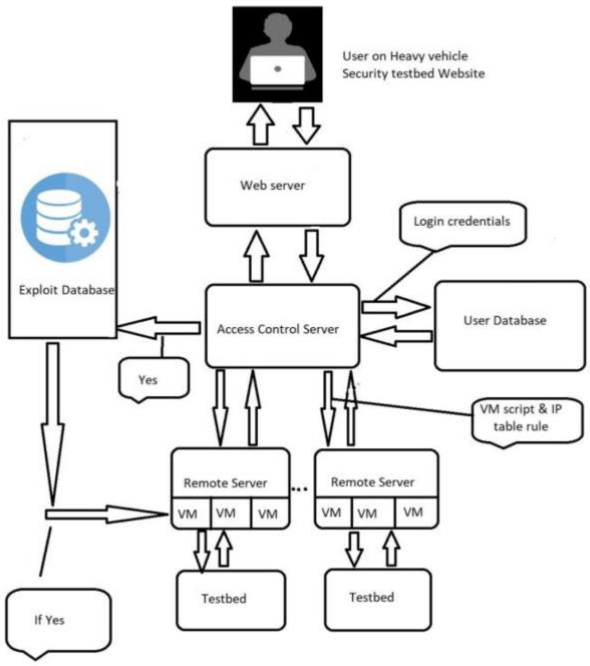
In the heavy vehicle security we are using XACML to implement the attribute based access control. Extensible Access control Markup language is a fine-grained attribute-based access control policy language, access control request / response language, architecture and processing model. The policies describe the access control requirements that contains standard extension point to define a function, data types and logic. The request and response language allow users to form query that describes certain actions allowed or not. The XACML provides four types of response: allow, Deny, Intermediate and Not applicable. The user make request to access of the protected resources. The PEP forms the request by users' attributes, and other information related to resources & sends it to the PDP. The PDP fetches the policies from the PAP and PIP pertaining to the resources, users & environmental requirements, where these policies or policy sets are made up of users, environmental and resource attributes. After deep inspection PDP makes a decision to allow or deny the request. The XACML allocates the security tag to each of the attribute used in the policies and quires. Also, the XACML separates the user request point, decision point and policy defining points.

Top level constructs the policies and policy sets. A policy or policy set is defined at the top of all the policies and policy sets. The policy set is a container that stores the policies or policy sets, or references of external policies found in remote locations. The policy describes the single access control policy made up of set of rules. Every policy or policy set is provided with the dedicated policy / policy set root XML tag. The XACML contain Target that is a simplified version of the condition for the subject. Once the policy/ policy set is found the PDP evaluates rules with respect to subject condition and makes the decision. The rules are set using attributes, where the attributes are characteristic of subject, resource, Action and Environment. From the attributes submitted to the PEP through the query is stored in a file called as Bag. The PDP applies matching function to this user generated Bag that compares the attributes used in policies with the attributes stored in Bag and make a decision.

By enforcing access control, the testbed wants to protect 3 elements from confidentiality, Integrity and arability britches:

users, services, vendors. At users end the testbed access control protect user's personal information, their intellectual properties contain scripts and exploits. On service prospective the access control protects unauthorize access of user's information such as website login, access generic information, users' restrictions & VM configuration file. On testbed service access control protects, vulnerability database, attack groups, vulnerability exploit scrips on VM & message comments. vendors information and bus traffic. The access control also protects the uptime of webpage, admin portal, cross VM and cross portal attacks. The access control protects the central server to send faulty information to the remote server. Central server contains the access control which enforces setting and rules on both web and remote server. On vendors end the access control server prevents ECU's restricted message & prevents firmware from getting modified through dump files. The access control sever establishes the IP tables rules on VMs.

3



The web server store information about the web. When user initiates the communication and login into the testbed then the web server forwards the credentials to access control server. The access control server verifies the user information with user database and fetches its configuration and generous information. If the user get access, then the access control server notifies the remote server to start a VM and configure user specific IP tables inside the VM. Then the ssh link is sent to the user Email. When the user login in ssh and query the exploit, the access control server accesses the exploit server scripts and mounts it on the Virtual machine. After that, the access control server continuously monitors the communication between the user and testbed and restrict the unauthorize packet sent from testbed to VM, to prevent vendors getting VM to testbed, to prevent the testbed getting modified or getting exploited. When the user completes the work the access control saves the user data in the users' database and clears the VM.

VIII. Future Scope

In the future we are planning to add telematics unit and other interfaces into the testbed to provide comprehensive analysis of external attack vectors and their consequences along with the packet interface to CAN bus. By facilitating such functionality, the testbed users can use external attack vector to get into the vehicle network and the CAN interface provides deep analysis of CAN packet flowing inside the CAN bus. Moreover, we are also planning to add the firewall & other proposed technologies inside the vehicle so that the researchers can study proposed technologies comprehensively.

Apart from such physical devices we are also planning to provide a simulation support such that the testbed user can define their ECU functionality or security module functionality basis on which the simulating module will be created & implemented in CAN network.

Moreover, we are planning to interface this testbed with PENJ1939 framework protocol for easy exploitation of vehicle vulnerability.

Furthermore, we are planning to add a virtual vehicle simulator such a way that it will provide information about real time consequences on vehicle dynamics to the user. This feature will reduce the risk of performing experiments on running vehicles.

IX. Conclusion

The automobiles became a cyber-physical system due integration of in vehicle control systems with the external network interface. All the vehicle class has similar kind of tends in their technologies. The heavy vehicles are especially dangerous due to good caring capacity and huge mass that can cause the huge damage to society. The studies and experiments have shown that the vehicles are vulnerable to attacks and the attacks from external vectors are feasible. To prevent such attacks the researcher must know the working of the elements used in heavy vehicle which requires reverse engineering of vehicle's control system. The absence of open source real time accurate physical testbed makes the jobs of researchers even harder. The software simulations limited in terms of real time nature of vehicular system and the accuracy of the result produced. The cyber physical nature of vehicle enables steep barrier for a open source testbed as the attacker can study and discover the vehicle models vulnerabilities and types of attacker & can completely destroy the company & its users. The heavy vehicle testbed is an open source testbed that support the Deter like configurability while maintaining the vendors security with attribute base access control scheme.

References

[1] Stephen Checkoway, Damon McCoy, Brian Kantor,Danny Anderson, Hovav Shacham, and Stefan Savage, "Comprehensive Experimental Analyses of Automotive Attack Surfaces", www.autosec.org/pubs/cars-usenixsec2011.pdf.
[2] Wikipedia.com
[3] https://www.youtube.com/watch?v=bHfOziIwXic
[4] Parnian Najafi Borazjani, Christopher E. Everett, Damon McCoy, "OCTANE: An Extensible Open Source Car Security Testbed", https://pdfs.semanticscholar.org/f61e/4312185bb7aa7b0795db6533dc9d3c989300.pdf

[5] https://www.isi.deterlab.net/index.php3
[6] https://www.emulab.net/portal/frontpage.php
[7] Daniel S. Fowler, Madeline Cheah, Siraj Ahmed Shaikh, Jeremy Bryans, "Towards a Testbed for Automotive Cybersecurity", 2017 IEEE International Conference on Software Testing, Verification and Validation (ICST).
[8] Karl Koscher , Alexei Czeskis , Franziska Roesner , Shwetak Patel , Tadayoshi Kohno , Stephen Check, "Experimental Security Analysis of a Modern Automobile", 2010 IEEE Symposium on Security and Privacy.
[9] http://blogs.windriver.com/hermeling/2009/03/what-is-a-baremetal-hypervisor.html
[10] https://www.statista.com/
[11] https://docs.ansible.com/ansible/2.6/user_guide/playbooks_intro.html
[12] https://en.wikipedia.org/wiki/Proxmox_Virtual_Environment
[13] https://en.wikipedia.org/wiki/RAID
[14] https://docs.ansible.com/ansible/2.7/user_guide/playbooks.html
[15] http://sourcerepo.com/
[16] https://searchnetworking.techtarget.com/definition/TCP
[17] https://en.wikipedia.org/wiki/Attribute-based_access_control
[18] https://www.axiomatics.com/blog/the-benefits-of-fine-grained-dynamic-authorization-an-introduction-to-attribute-based-access-control/
[19] https://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html
[20] Subhojeet Mukherjee , Noah Cain , Jacob Walker , David White , Indrajit Ray & Indrakshi Ray ,"PenJ1939: An Interactive Framework for Design and Dissemination of Exploits for Commercial Vehicles" , https://dl.acm.org/citation.cfm?id=3138844
[21] https://www.ixxat.com/technologies/all4can/sae-j1939-technology
[22] Yelizaveta Burakova, Bill Hass, Leif Millar, and Andr´e Weimerskirch,"Truck Hacking: An Experimental Analysis of the SAE J1939Standard",https://www.usenix.org/system/files/conference/woot16/woot16-paper-burakova.pdf
[23] Xi Zheng, Hongxu Chen, Lei Pan, Rick Di Pietro, Lynn Batten, "A testbed for Security analysis of modern vehicle system" , https://ieeexplore.ieee.org/document/8029560/
[24]
[25]