ESCAR EU – SUMMARY

Tomas Olovsson, Chalmers

ESCAR EU took place in Brussels in November 2018 and contained several interesting talks. This is a summary of the talks. Please note that this is an interpretation of the talks and may differ from the presenters views. For a longer report and more information, feel free to contact me.

Access to full papers and lectures is possible after registration on the ESCAR home page.

- - -

A couple of talks focused on keyless access to vehicles and the (in)security of these cards. Typically, a vehicle equipped with keyless access cards (RFID cards) broadcast their ID periodically and a keycard in vicinity will respond and get a challenge from the vehicle to encrypt.

Lennert Wouters et.al. from KU Leuven (*Fast, Furious and Insecure: Passive Keyless Entry and Start Systems in Modern Supercars*) reverse engineered Tesla model S key cards and found a way to clone key cards. They discovered that it is possible to talk to any key card and give it a challenge which it will answer. They selected a fixed challenge (40 bit long, 0x636f736963) which they encrypted with all possible keys and the results were stored in a file for easy reverse-lookup. Then (somewhat simplified, exact details are omitted here) it is possible to talk to any key card, give it the challenge and look up what key it has. Card access and lookup takes just a second or two and cloning a key is now trivial. This method is not limited to Tesla vehicles but works with other brands using the same tehchniques.

In another talk, Tobias Hoppe (*Automotive anti-theft retrofit kits and their Safety and Security implications*) continued this approach by analyzing a third party device, a gadget similar to a key card doing a first level of authentication. When accepted by this device, power is enabled to the OEM ECU which performs the normal vehicle authentication. They described in detail how the protocol was analyzed and how it works. There were many interesting shortcomings, one was that number of challenges were limited to 16 and then they were reused. Consequently, replay attacks were trivial to perform, so also relay attacks when an attacker talks to a card and relays the communication to the vehicle using own equipment. Maybe worse was that if the communication between the new device and the vehicle failed, power was cut to the vehicle ECU after 60 seconds. May be ok but the ECU likely controls other functions in the vehicle as well, which could result in a safety problem if someone jams the communication when driving.

Other talks focused on machine learning and robustness. Machine learning algorithms and neural networks are used in many applications around autonomous vehicles. A problem with these is that the system's view of what it sees differ from what we see, and it is hard to know what decisions are actually

based upon and how robust the decisions are. One example brought up were traffic sign recognition systems, but similar algorithms are used in other applications such as traffic lane detection, object detection from sensors, etc. One way to analyze robustness which was described in detail, was to introduce “dots” or disturbances to the data to see how they behave. This way, *heat maps* can be created where it is possible to see, for example on a traffic sign, what areas are most sensitive to disturbances (compare with brain scans). With such techniques it would be possible to see what input data affect decisions and how robust the systems are to disturbances.

Another talk focused on the need to carefully select training data and to protect them from manipulation. An attacker may deliberately manipulate or poison the data to create backdoors or unwanted behavior in selected situations. Defense is hard, how do we know training data is biased in some direction? The result could be a system where an attacker knows about situations where incorrect decisions will be made by the system. Takeaway from this talk is that more research is needed here.

Other talks focused on how to obtain firmware from ECUs. In short, this is almost trivial regardless of how the ECU is built. One talk proposed fault injection methods as a shortcut to execute own code on ECUs instead of trying to find vulnerabilities in the existing code. By changing voltage, frequency or create power glitches, tests for proper authentication could easily be circumvented. This way, firmware and all memory contents could be retrieved. Takeaway here is that the method is also useful for own testing: use fault injection to make your code more robust.

Another talk from Tel Aviv University (*Woodpecker, a Software-only True Random Generator for the CAN Bus*) showed a method to obtain entropy for random number generators using CAN bus inter-arrival rates of messages. They have analyzed this source for randomness/entropy and found it as good as Linux systems. However, a question was raised that even if the inter-arrival rates may be random to an outside observer, all ECUs should be able to see these patterns and from this perspective, the numbers would not be as random. This would be a problem if we have compromised ECUs on the network.

Finally, a new mechanism for detecting falsified CAN messages was described (*TCAN: Authentication without Cryptography on a CAN Bus Based on Nodes Location on the Bus* by Eli Gavril: Technion, computer science dept., Israel). Their method is patent pending and uses timing measurements by a monitor (an ECU) installed on the CAN bus. It measures the timing of messages which depends on the distance between the monitor and the transmitting ECU, as well as an echo signal from a repeater at the other end of the network. The echo signal may use different voltages and will not be visible for normal ECUs.