Evaluating V2V Security on an SDR Testbed

Geoff Twardokus
Department of Computing Security
Rochester Institute of Technology
Rochester, NY, USA
gdt5762@rit.edu

Hanif Rahbari
Department of Computing Security
Rochester Institute of Technology
Rochester, NY, USA
hanif.rahbari@rit.edu

***Abstract*—We showcase the capabilities of *V2Verifier*, a new open-source software-defined radio (SDR) testbed for vehicle-to-vehicle (V2V) communications security, to expose the strengths and vulnerabilities of current V2V security systems based on the IEEE 1609.2 standard. *V2Verifier* supports both major V2V technologies and facilitates a broad range of experimentation with upper- and lower-layer attacks using a combination of SDRs and commercial V2V on-board units (OBUs). We demonstrate two separate attacks (jamming and replay) against Dedicated Short Range Communication (DSRC) and Cellular Vehicle-to-Everything (C-V2X) technologies, experimentally quantifying the threat posed by these types of attacks. We also use *V2Verifier*'s open-source implementation to show how the 1609.2 standard can effectively mitigate certain types of attacks (e.g., message replay), facilitating further research into the security of V2V.**

I. Introduction

Vehicle-to-vehicle (V2V) communication allows vehicles to communicate directly with each other, increasing vehicles' awareness of their surroundings and allowing non–line-of-sight (NLOS) coordination to avoid collisions. V2V promises to prevent up to $80\,\%$ of non–alcohol-related vehicle crashes in the U.S. every year [1], significantly reducing annual vehicular injuries and fatalities. Despite the standardization of security services for V2V (e.g., IEEE 1609.2 [2]), new security threats continue to emerge, in part because of the lack of sufficient evaluation of these standards. V2V security schemes (e.g., [3]) frequently rely on theoretical analysis or simulation for validation, neither of which is sufficient to prove real-world viability. Since roadway testing is only possible in specific locations [4], and can be quite expensive, configurable hardware testbeds are an affordable alternative to validate V2V systems in real wireless environments.

In this demo paper, we utilize our fully open-source software-defined radio (SDR) testbed for V2V security, *V2Verifier*[1], to demonstrate novel attacks and experimentally validate mitigation techniques. *V2Verifier* combines SDRs and a prominent type of commercial V2V on-board unit (OBU)[2] to emulate an IEEE 1609.2-compliant V2V system. Combined with modular implementations of the two prominent V2V protocol stacks for future expansions, this facilitates further research into V2V security. Featuring the first open-source implementation of the IEEE 1609.2 security standard for V2V [2] and its amendments, *V2Verifier* is uniquely capable of revealing the strengths and weaknesses of security schemes when they are deployed as a part of a real V2V protocol stack.

This research was supported by the National Security Agency under Grant Number H98230-19-1-0318. The United States Government is authorized to reproduce and distribute reprints notwithstanding any copyright notation herein. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Security Agency.

[1]A limited edition of *V2Verifier* is introduced briefly in [5] along with a preliminary version of only one of the attacks demonstrated here. A new version of that attack along with another one appear here for the first time.

[2]from *Cohda Wireless [6].*

II. Emulating Secure V2V Communications

Each V2V-equipped vehicle broadcasts a *basic safety message* (BSM) containing its GPS location, velocity, and direction of travel at least once every $100\,ms$ [1]. *V2Verifier* emulates this using Ettus USRP B210s—which have the advantage of running on a USB connection alone or portable power supplies (if a GPS antenna is activated) for mobility scenarios—to act as vehicles. With one USRP per emulated vehicle, there is no limit in *V2Verifier* on the number of emulated vehicles, making it a scalable solution for small or dense scenarios. On top of existing, open-source implementations of Physical (PHY) and MAC layers (e.g., in GNURadio [7]) of both major V2V technologies—Dedicated Short Range Communication (DSRC) based on IEEE 802.11p standard [8], and Cellular Vehicle-to-Everything (C-V2X) [9]—*V2Verifier* implements the IEEE 1609.2 security standard for V2V communication.

To provide message authentication and integrity protection, every BSM is digitally signed using the Elliptic Curve Digital Signature Algorithm [2] and a private key supported by a public-key V2V certificate [10]. Message signing and certificate management are the two key security mechanisms of 1609.2 and are therefore at the heart of *V2Verifier*. In open-source, we have implemented key and certificate management utilities to support these mechanisms; when commercial OBUs are used, the Aerolink [11] security suite from Qualcomm—a *de facto* industry standard for managing certificates on V2V equipment—is integrated with the testbed.

III. Validating V2V Attacks and Defenses

A. Jamming threats in V2V

We demonstrate how new jamming attacks under both LOS and NLOS conditions can have debilitating effects on V2V safety scenarios. Through experimental validation of a novel (reactive) jamming attack that acts upon the detection of its target's 1609.2 pseudonym in a BSM, we show the utility

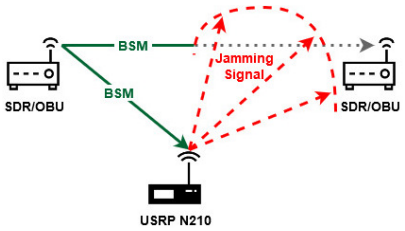

Fig. 1. Threat model for a reactive jamming scenario in *V2Verifier*.

of *V2Verifier* by demonstrating that such an attack is not effectively mitigated by the 1609.2 standard.

1) Reactive jamming attacks in V2V: With a USRP jammer, we show the low expense and ease for an attacker to execute a stealthy, debilitating denial-of-service attack against V2V safety and security measures. Fig. 1 shows our threat model. A USRP (or an OBU) is broadcasting 10 BSMs per second. The attacker, running a reactive jammer on a USRP N210, listens to the channel waiting for any transmission from the target, whose messages are distinguished by its 1609.2 privacy-preserving pseudonym [2]. When the target sends a BSM, the attacker broadcasts a brief ($< 500\,\mu s$) jamming signal (a short, bogus frame) to make it difficult for others to successfully receive that BSM. Due to PHY-layer error correction and/or 1609.2 signature verification, the receiving device will likely drop the packet due to the unrecoverable, jammer-induced corruption in the packet. Although Fig. 1 shows only two devices, the jamming signal may impact many nearby receivers (i.e., vehicles). Among other consequences, this attack can (as shown in [5]) disrupt the operation of misbehavior detection systems by increasing the rate of false positives.

2) Evaluating jamming attacks under adverse conditions: Using USRPs to emulate vehicles facilitates experimentation in a highly configurable environment. Experimental parameters (e.g., distance between devices, (N)LOS channel, transmitter power, mobility of devices) can be varied at will. Consider an attacker who is attempting to reactively jam BSMs from a specific vehicle (as above). The attack may come from different distances, and dynamic vehicular environments do not guarantee a line of sight. Fig. 2 shows how these factors affect the effectiveness of the jamming attack; in particular, it is clear that the attacker is significantly less successful in NLOS conditions with just a small increase in distance between the attacker and target. Nevertheless, a reactive jammer is able to catastrophically degrade safety communications in less than half a minute under all conditions. Further experiments could involve placing the USRPs farther apart, or even placing them in moving vehicles to evaluate realistic mobility scenarios.

B. Mitigation of Replay Attacks

Using *V2Verifier*, we can further demonstrate the (in)effectiveness of 1609.2 mechanisms for mitigating certain attacks (e.g., message replay) despite its vulnerability to other attacks exposed by *V2Verifier*, like jamming. In a replay attack, the attacker captures and re-transmits valid, signed BSMs in the hope that other vehicles will believe the re-transmissions, thereby causing chaos as vehicles are expected

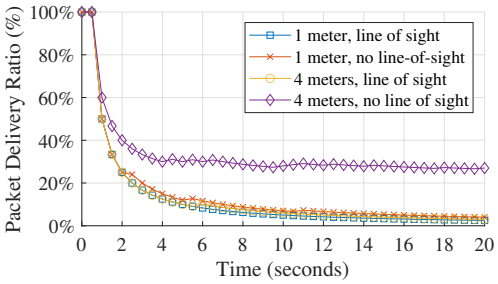

Fig. 2. Effectiveness of the reactive jamming attack under LOS and NLOS conditions. The jammer activated at $1\,s$, resulting in a brief $100\,\%$ PDR.

Fig. 3. A replay attack shown in *V2Verifier*. The translucent red vehicle behind the actual vehicle (depicted translucently to indicate a security alert) is displayed at the location claimed in a replayed message.

in locations where they have not actually been for some time. Fig. 3 shows *V2Verifier*'s representation of an ongoing replay attack against C-V2X OBUs, with a translucent vehicle shown at the location claimed in a replayed BSM. The translucence indicates successful mitigation by 1609.2 (in this case, through use of the mandatory expiration time check to determine message "freshness"). The *V2Verifier* display is useful for identifying vulnerabilities in more complex scenarios, as the changing appearance of vehicles can help visually determine when attacks are, or are not, successfully going undetected (and to what extent a mitigation technique is effective).

REFERENCES

[1] National Highway Traffic Safety Administration, "Technical report 11078-101414-v2a," 2014, accessed: Feb. 20, 2020. [Online]. Available: https://bit.ly/35EggyG

[2] *Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages*, IEEE Standard 1609.2-2016, 2016.

[3] N. Lyamin, D. Kleyko, Q. Delooz, and A. Vinel, "AI-based malicious network traffic detection in VANETs," *IEEE Netw.*, vol. 32, no. 6, pp. 15–21, Nov. 2018.

[4] U.S. Department of Transportation, "Interactive connected vehicle deployment map," Accessed: Jan. 11, 2021. [Online]. Available: http://bit.ly/38ySrL4

[5] G. Twardokus, J. Ponicki, S. Baker, P. Carenzo, H. Rahbari, and S. Mishra, "Targeted discreditation attack against trust management in connected vehicles," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Montreal, QC, Canada, Jun. 2021.

[6] Cohda Wireless, "MK6c EVK - Cohda Wireless," Accessed: Jan. 8, 2021. [Online]. Available: https://bit.ly/2TCgCQt

[7] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "An IEEE 802.11a/g/p OFDM receiver for GNU Radio," in *Proc. Second Workshop Softw. Radio Implementation Forum*, Hong Kong, China, 2013, pp. 9–16.

[8] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments*, IEEE Standard 802.11p, 2010.

[9] *Summary of Rel-14 Work Items*, 3rd Generation Partnership Project Technical Specification 21.914 V14.0.0, 2018.

[10] *Wireless Access in Vehicular Environments (WAVE)–Certificate Management Interfaces for End Entities*, IEEE Standard 1602.2.1-2020, 2020.

[11] Qualcomm, Inc., "Aerolink communications security," Accessed: Jan. 10, 2021. [Online]. Available: http://bit.ly/3qaLaap

DEMO PROPOSAL

Testbed Configuration

We demonstrate *V2Verifier* with varying combinations of USRPs and commercial OBUs. The complete set of experimental equipment is shown in Fig. 4: two Linux laptops, two USRP B210s, one USRP N210, and two Cohda Wireless MK6c OBUs. All USRPs are equipped with $5\,dBi$ antennas capable of operating on the 5.9 GHz V2V band; the Cohda devices each use dual $4\,dBi$ antennas designed for the same frequencies. Note that Fig. 4 shows the devices used, but not their placement relative to each other during experimentation.

In this demo, we demonstrate a reactive jamming attack and show how changes in distance, relative velocity, and the existence (or absence) of a line-of-sight (LOS) channel affect the effectiveness of jamming attacks. The devices emulating vehicles are initially placed approximately $2\,m$ apart, with the USRP representing the attacker being moved (on a rolling cart to varying distances from those devices). Experimentation with NLOS attack scenarios is shown by placing the attacker USRP on the other side of a building wall from the other devices.

We start by emulating a V2V environment where every vehicle broadcasts a BSM once every $100\,ms$. In *V2Verifier*, each USRP or OBU represents one vehicle and sends BSMs at this rate to mimic realistic operations. To generate BSMs, each device is provided with a map trace of GPS coordinates, obtained from a commercial traffic simulator, to use for generating BSMs. Each generated BSM contains the next coordinate point in a trace as its location, with velocity and heading calculated from the distance (over a $100\,ms$ interval) and angle between the current and previous coordinate set.

For each attack scenario, two or more devices (USRPs or OBUs) are run in this manner. Each device can also, optionally, be connected to an instance of the graphical user interface (GUI) depicited in Fig. 5. The GUI displays the location of the "vehicle" emulated by that device, along with the perceived locations of neighboring vehicles as reported in their most recent BSMs. Also, the GUI displays a scrolling report of security information about each incoming message. The results of each message's verification (including signature verification and freshness check) along with security-related packet statistics are displayed in real time as well. The statistics include important metrics, such as, the percentage of received packets that have been validly signed by other vehicles. An attacker, capable of several attacks including reactive jamming, is represented by a USRP N210 placed among the four "legitimate" devices.

Experimental Steps

1) We begin by showing the proper operation of our emulated V2V environment. The testbed, configured as described above, is run for a brief period to show vehicles moving on the *V2Verifier* GUI as BSMs are received and securely verified.
2) We initiate a reactive jamming attack against USRP B210s (or Cohda OBUs) using the USRP N210 as the attacker. The effect will be visually obvious as the number of neighboring vehicles shown (in red) on the *V2Verifier* GUI will suddenly drop from four to zero; also, the packet statistics shown on the GUI will indicate the effectiveness of the attack (e.g., by displaying a high packet error rate) as well as its impact on signature verification rate (used for misbehavior detection). This attack will be shown in multiple configurations with variable distance ($1-10\,m$) and variable relative velocity between the attacking USRP and the communicating devices, as well as under both LOS and NLOS conditions.
3) We demonstrate the effectiveness of 1609.2 security mechanisms to mitigate a packet replay attack. The *V2Verifier* GUI will begin displaying a number of translucent vehicles as replayed BSMs are received, indicating successful detection of the replay attack.

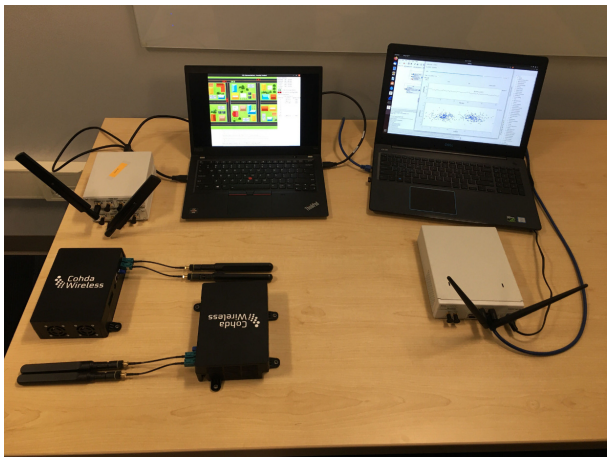

Fig. 4. Testbed configuration for the demo.

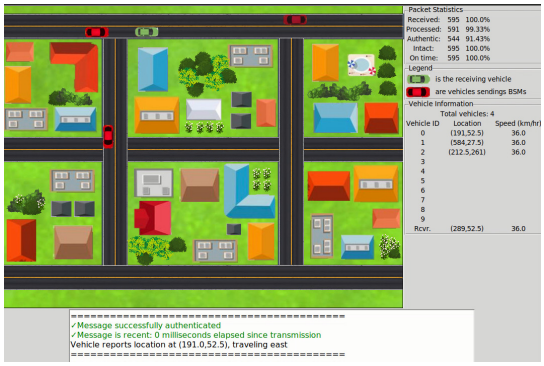

Fig. 5. Vehicles are displayed in red at the locations indicated in their BSMs, while the location of the receiver is displayed in green. Information about the vehicles' speed and heading is shown in the panel on the right of the display.

Public Materials

The *V2Verifier* source code used for this demonstration will be made available on our project's GitHub repository[3].

Reproduciblity–See the link below for a simple example.

[3]https://github.com/twardokus/v2verifier