Automated Discovery of Denial-of-Service Vulnerabilities in Connected Vehicle Protocols

Shengtuo Hu, University of Michigan; Qi Alfred Chen, UC Irvine; Jiachen Sun, Yiheng Feng, Z. Morley Mao, and Henry X. Liu, University of Michigan

https://www.usenix.org/conference/usenixsecurity21/presentation/hu-shengtuo

This paper is included in the Proceedings of the 30th USENIX Security Symposium.

August 11–13, 2021

978-1-939133-24-3

Open access to the Proceedings of the 30th USENIX Security Symposium is sponsored by USENIX.
Automated Discovery of Denial-of-Service Vulnerabilities in Connected Vehicle Protocols

Shengtuo Hu
University of Michigan
Qi Alfred Chen
UC Irvine
Jiachen Sun
University of Michigan
Yiheng Feng
University of Michigan
Z. Morley Mao
University of Michigan
Henry X. Liu
University of Michigan

Abstract

With the development of the emerging Connected Vehicle (CV) technology, vehicles can wirelessly communicate with traffic infrastructure and other vehicles to exchange safety and mobility information in real time. However, the integrated communication capability inevitably increases the attack surface of vehicles, which can be exploited to cause safety hazard on the road. Thus, it is highly desirable to systematically understand design-level flaws in the current CV network stack as well as in CV applications, and the corresponding security/safety consequences so that these flaws can be proactively discovered and addressed before large-scale deployment.

In this paper, we design CVAnalyzer, a system for discovering design-level flaws for availability violations of the CV network stack, as well as quantifying the corresponding security/safety consequences. To achieve this, CVAnalyzer combines the attack discovery capability of a general model checker and the quantitative threat assessment capability of a probabilistic model checker. Using CVAnalyzer, we successfully uncovered 4 new DoS (Denial-of-Service) vulnerabilities of the latest CV network protocols and 14 new DoS vulnerabilities of two CV platoon management protocols. Our quantification results show that these attacks can have as high as 99% success rates, and in the worst case can at least double the delay in packet processing, violating the latency requirement in CV communication. We implemented and validated all attacks in a real-world testbed, and also analyzed the fundamental causes to propose potential solutions. We have reported our findings in the CV network protocols to the IEEE 1609 Working Group, and the group has acknowledged the discovered vulnerabilities and plans to adopt our solutions.

1 Introduction

With the emerging Connected Vehicle (CV) technology [64], Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) wireless communication enables vehicles to exchange important safety and mobility information with other entities in real time. In September 2016, the U.S. Department of Transportation (USDOT) launched the CV Pilot Program in three sites, New York City, Wyoming, and Tampa, to spur early CV technology deployment and test CV safety applications in the real world. As of Fall 2018, the program has entered the third phase, which requires at least 18-month period for long-term operation and key performance measurements [66].

While CV technology can greatly benefit transportation mobility and safety, such dramatically increased connectivity inevitably increases the attack surface of both vehicles and the transportation infrastructure. For example, if the CV communication protocol stack is not sufficiently secure, attackers can directly cause safety hazard to human drivers on the road [1, 13, 45]. Thus, it is imperative to understand the potential security vulnerabilities in the CV network stack as early as possible so that they can be proactively addressed before large-scale deployment. To achieve this, it is necessary to start with a systematic study of potential design-level security flaws in the CV network stack, since both the discovery and defense solutions of such flaws can most generally affect the security of their corresponding implementation instances.

Existing work on the analysis of Vehicular Ad-Hoc Network (VANET) or CV security [1, 4, 10, 28, 29, 44, 55, 55, 73] generally suffer from three limitations:

(L1): they lack systematic approaches and rely on manual inspection to identify potential threats [1, 44, 55, 73], which is both insufficient and inefficient. It is also hard to automate the risk assessment of identified threats in these works. For example, Laurendeau et al. [44] use ETSI’s threat analysis methodology [24], which relies on human to qualitatively rank the risks of the threats. Similarly, Petit et al. [55] manually characterized threats in the automated vehicle (e.g., the cooperative automated vehicle with V2X communication), only annotated the qualitative risk.

(L2): The threats to the availability of the higher-layer protocols (i.e., IEEE 1609 protocols [32, 34] and CV applications), which can prevent legitimate protocol participants from accessing critical services in the network, are largely under explored [4, 12, 28, 54, 65]. Although USDOT and the protocol designers have already employed security mechanisms to protect the integrity and confidentiality of CV network com-
In the following text, without specific notations, “IEEE 1609.*” represents the latest version (e.g., “IEEE 1609.2” and “IEEE 1609.3” stands for “IEEE 1609.2-2016” and “IEEE 1609.3-2016” respectively).
layer from processing incoming packets, and 15 vulnerabilities (14 of 15 are new) in PMPs, which can block the communication among platoon members. Our quantification results show that their exploits can have as high as 99% success rates, and can double the delay in packet processing, which violates the latency requirement of CV communication.

- For these newly-discovered vulnerabilities, we have constructed practical exploits and validated them in a real-world testbed. We have also reported to and received confirmations for P2PCD attacks from IEEE 1609 Working Group [35]. Besides, our case studies demonstrate that P2PCD attacks can lead to traffic accidents, and PMP attacks can affect the speed stability of the victim vehicle. These results thus concretely demonstrate the effectiveness of CVAnalyzer.

- For the identified vulnerabilities, we discuss the fundamental reasons and propose effective mitigation solutions, including avoiding using truncated hash value (e.g., 3-byte hash value), mandating verification for P2PCD learning responses, and requiring P2PCD learning requests to be broadcast (§7). After our discussion with the IEEE 1609 Working Group [35], mitigation solutions against P2PCD attacks are planned to be integrated into the next version of IEEE 1609.2.

2 Technical Background

In this section, we introduce the necessary technical background about the CV network stack and the platoon management protocols (PMPs).

2.1 CV Technology & Network Stack

CV network provides connectivity in support of mobile and stationary CV applications, which offers users greater situational awareness of events, potential threats, and imminent hazards, with the goal of enhancing the safety, mobility, and convenience of everyday transportation [36]. In the CV network, there are two basic types of devices: (1) On-Board Unit (OBU) in a roaming vehicle and (2) stationary Road-Side Unit (RSU) along the road. Usually, the communication pattern of the CV network is individual messages that are broadcast without response [34].

IEEE 802.11p [37] and its extension IEEE 1609.4 [33] together define the basis of the CV network stack, in which IEEE 802.11p disables the authentication, association, and data confidentiality services at the MAC layer to minimize the message latency. Above them, IEEE 1609.3 [34] defines the WAVE Short Message Protocol (WSMP), which is optimized to minimize communication overhead. The Basic Safety Message (BSM, a.k.a., the beacon message) defined in SAE J2735 is used by a variety of applications, such as Forward Collision Warning (FCW), Cooperative Adaptive Cruise Control (CACC), to exchange safety data regarding vehicle state (e.g., location and speed). The transmission rate of BSM is typically set to 10 times per second [2, 3, 25].

Due to the safety-critical nature of CV applications, IEEE 1609.2 [32] specifies security mechanisms to provide confidentiality, authenticity, integrity, and non-repudiation. It introduces digital certificates to enable digital signature (ECDSA), with the support of a Public-Key Infrastructure (PKI) system called Security Credential Management System (SCMS) [12]. Also, SCMS supports the misbehavior detection and certificate revocation to prevent malicious vehicles from communicating with others, while the development of the misbehavior detection algorithms is still ongoing.

In particular, IEEE 1609.2 specifies a unique feature called Peer-to-Peer Certificate Distribution (P2PCD) that helps a CV device to learn unknown certificates. When a device receives a signed secured protocol data unit (SPDU), it will construct a certificate chain for the signing certificate within the SPDU. The certificate chain links the signing certificate to a known trust anchor, which usually refers to the root certificates shared by all CV devices, so the incoming SPDU can be trusted by the receiver. However, the CV device may be unable to construct such a certificate chain due to not recognizing the issuer of the signing certificate. In this case, the received SPDU is referred to as a trigger SPDU, and the CV device will attach P2PCD learning request field in the next outgoing SPDU to request peer devices to provide the necessary certificates to complete the chain. P2PCD learning responses, which contains requested certificates, will be sent back through WSMP by peer devices. Note that, a P2PCD learning response is sent as a protocol data unit (PDU) rather than an SPDU. That is, the P2PCD learning response itself does not carry the digital signature. The current IEEE 1609.2 does not mention the verification for the payload of the learning response (cf. IEEE 1609.2-2016, Clause 8.2.4.1 c)). Besides, the P2PCD example in IEEE 1609.2 (cf. 1609.2-2016 Clause D.4.3.6) only considers VerifyCertificate primitive as an optional step before AddCertificate primitive.

2.2 Platoon Management Protocol (PMP)

CVs form a platoon with minimal following distances to improve traffic density and fuel economy. The PMP is an essential component for platoon applications to control platoon maneuvers. Typically, vehicles in a platoon exchange speed, location, platoon ID, platoon depth by broadcasting beacon messages periodically. The platoon leader has a depth of 0, and it increases as we go farther. The leader acts as the coordinator and controls platoon decisions such as join/merge, split, leave, and dissolve. In this paper, We study two PMPs; since PLEXE [56] only specifies the join-at-tail maneuver that is the same as Join/Merge maneuver in VENTOS, we thus mainly follow the description of PMP in VENTOS [69].

Join/Merge Maneuver Two platoons, traveling in the same lane, can initiate a merge maneuver to form a bigger platoon. The leader of the rear platoon will send a MERGE_REQ to the front platoon leader, if it observes that the combined platoon size is no greater than the optimal platoon size by
inspecting the beacon message from front vehicles. Upon receiving a MERGE_ACCEPT from the front leader, the rear platoon leader will speed up to reduce the front spacing. Then, the rear leader sends CHANGE_PL to notify its followers to change the platoon leader to the front leader. Meanwhile, the rear leader switches to the follower role after sending a MERGE_DONE to the front platoon leader.

**Split Maneuver** To break the platoon into two smaller platoons, a platoon leader can either actively initiate this maneuver at a specific position, or passively trigger this maneuver when the platoon size exceeds the optimal platoon size. A platoon leader first sends a SPLIT_REQ to the splitting vehicle where the split should occur. After receiving a SPLIT_ACCEPT, the platoon leader sends a CHANGE_PL to make the splitting vehicle a potential leader. Besides, the platoon leader needs to inform followers behind the splitting vehicle, if any, to change their leader to the splitting vehicle. After that, the platoon leader sends a SPLIT_DONE to the splitting vehicle, which then switches to the leader role.

**Leave Maneuver** A platoon member may initiate a leave maneuver, when approaching the destination. For the leader leave, the leader will send a VOTE_LEADER to all followers to vote on the new platoon leader. The newly elected platoon leader needs to send a ELECTED_LEADER to the current leader. Then, the leader splits at the position of the elected leader by initiating the split maneuver, and thus hands over the leadership to the elected leader. For the follower leave, the follower will send a LEAVE_REQ to the leader and wait for a LEAVE_ACCEPT. The leader needs to split at both the succeeding vehicle, if any, of the follower, and the follower to make it a free agent, defined as a one-vehicle platoon. At this time, the follower can slow down. Once there exists enough space for the follower to change the lane, it will send a GAP_CREATED to the old leader and finally leave the platoon.

**Dissolve Maneuver** This maneuver is only initiated by the platoon leader, who broadcasts a DISSOLVE to all followers. Upon receiving all ACK messages, all platoon members act as free agents and are free to leave.

### 3 Threat Model

**CV communication capability.** In our work, we assume that the attacker can compromise OBUs on her own vehicles or others’ vehicles, which follows recent works on CV security [14, 15, 74]. This assumption is reasonable, as previous works [13, 39] have already shown that in-vehicle systems can be compromised physically or remotely. In this case, the attacker can send malicious packets to other vehicles through compromised CV devices. All malicious packets should comply with protocol specifications. Notably, the attacker is allowed to unicast malicious packets to a specific vehicle (cf. IEEE 1609.3, Subclause 5.5.1).

**Passive monitoring.** The attacker can passively eavesdrop and capture all network traffic in her wireless communication range under the promiscuous mode of the wireless adapters.

**Cryptography operations.** We assume that cryptography operations used in CV protocols (e.g., signing, verification, and hash) are secure. The attacker thus cannot forge digital signatures used for packet authentications but can use valid certificates installed in compromised vehicles to sign outgoing packets. However, the attacker can still (1) passively collect valid certificates by sniffing the CV network traffic, and (2) construct local certificates, which are not signed by trusted anchors.

### 4 Analysis Methodology

In this section, we first present how we construct each component in the model, including the adversary model and each protocol state machine. We then describe how we reduce the state space and document how we implement CVAnalyzer.

#### 4.1 Model Construction

As shown in Figure 2, our model, consisting of the environment and protocol state machines (\(P\)), is driven by network and timer events. In general, the environment manages packet/time events generated by protocol state machines. It delivers triggered events (e.g., packet reception, timeout) to protocol state machines.

**Adversary-controlled communication environment.** We follow the design in prior works [59] and define three sequential steps in a loop for the environment:

1. **Retrieve:** the environment picks one of many different packet/time events if such an event is available.
2. **Process:** the protocol state machine processes an event.
3. **PostProcess:** after processing a given event, the protocol state machine either sends a new packet, adds a new timer, cancels an existing timer, or does nothing. The environment needs to update its internal states and keeps track of newly added events.

Our threat model (§ 3) assumes that the attacker has communication and eavesdropping capabilities. Thus, we add one more step for the attacker to send and receive arbitrary packets:

4. **Attack:** the attacker is able to monitor all packets in the environment. If needed, she can inject arbitrary packets into the environment, which allows a protocol state machine to process all possible packet events.
To model the network, we construct the communication channel $C = \{c_{i,j}| i \in [1,n], i \neq j\}$, where $c_{i,j}$ is a FIFO queue from $P_i$ to $P_j$. In this case, the packet sending and reception are abstracted as enqueue and dequeue operations on $c_{i,j}$. Notably, we do not consider network factors for vulnerability discovery, such as network latency and packet loss, because the lossy and erroneous network weakens the attack’s capability and increases the complexity of the model. Placing the attacker in her best position can help us uncover all potential attacks. On the other hand, to model timers, we do not keep track of the absolute time but only care about the temporal ordering of events, which is a common practice in model checking distributed system [43]. For progress advancing, all timers will count down simultaneously if there are no active events that should be delivered to protocol state machines.

**Protocol state machine.** All protocol participants ($P_i, i \in [1,n]$) are identical; therefore, each of them can be represented as the same finite-state machine (FSM). Then, our model $M$ can be defined as a concurrent system $M = C[|i|e|[1,n]P_i$, including an adversary-controlled environment $C$ and $n$ isomorphic processes $P_i$, where $|\_|$ is commutative and associative.

In our analysis, we abstract the higher-layer protocols in the CV network stack: (1) the communication model defined in networking services and message sublayer, (2) security services, and (3) PMP described in [5, 56, 60, 69]. We follow their specifications or codebases to define packet and timer handlers, which update the internal states of $P_i$ while processing packets and timeouts delivered by the environment. Our model excludes the handler of certificate revocation in security services, because it relies on an external public key infrastructure (PKI) like SCMS [12] to revoke certificates, which is out of the scope of the network stack itself. We will discuss how SCMS affects identified vulnerabilities in §8.

For the security services, we first abstract away cryptographic constructs because we assume that the cryptography operations in CV protocols are secure. Then, we model both packet type and packet header data, as they are required by the internal security mechanisms. In CV network, each protocol participant will have a batch of unique end-entity certificates (a.k.a., signing certificates). To trigger all internal security mechanisms, for the certificate configuration, we assume that the issuer of each batch of signing certificates is different from each other and is attached with packets in transmission.

**Probabilities.** Network protocol involves many concurrent events (e.g., packet transmission), leading to concurrent transitions in state machines. While building probabilistic models, we develop a discrete-time Markov chain (DTMC) model that assigns uniform probabilities to concurrent state transitions, originating from the same state (§4.2).

**State reduction.** We now show how we abstract the model to reduce states through a concrete example. For ease of exposition, we rely on a simplified example (Figure 3) derived from N4 (§5.1.3). Our goal is to reduce unnecessary states to get an abstracted model. Also, we want to ensure that the counterexample found in the abstracted model is a valid counterexample in the original model.

$$h(x) = x \mod M$$

Figure 3: A simplified example derived from N4 ($N$: the total number of events; $M$: the total number of timers; $TIMEOUT$: the maximum value of the timeout).

In the example, we develop a simplified protocol, in which the model updates the timer according to the event (Line 9-13), in which the function $h(x)$ abstracts the hash truncation operation in P2PCD. Assuming that, without the attacker, the range of event is $[0,X − 1]$, where $X < M \leq N$. The attacker in the environment can trigger all possible events $[0,N − 1]$.

For a given event, if $h(event)$ equals to $i$, then $timer[i]$ will be initialized (Line 10). For other unmatched timers, a timer will (1) remain unchanged if $timer[i]$ is not initialized (None), (2) count down if it has been initialized, or (3) set as uninitialized if it expires. To capture the attacker’s behavior, for each Next step in Figure 3, we randomly select a value in EventRange as the next event (Line 8). Notably, events within $[X,N − 1]$ are triggered by the attacker and can lead to the initialization of all timers. Last, to find counterexamples, we specify a liveness property (Line 14-16) that all timers should eventually expire if it has been initialized.

Obviously, the state space of the model depends on $N$ and $M$, which can be arbitrarily large. For example, the timer index range in P2PCD would be $[0, 2^{24} − 1]$ (i.e., $M = 2^{24}$). The number of events $N$ can be $2^{256}$. Unfortunately, the model checker cannot handle such large state space.

By analyzing the model, we observe that we do not need to track all timer[i], as the protocol only updates a small set of timers when no attacker is presented. As stated before, without the attacker, the range of event is $[0,X − 1]$; the model thus only updates $timer[i]$, where $i \in [0,X − 1]$. Usually, the protocol instance does not care whether other timers can eventually expire. Therefore, apart from reducing TimerIndexRange to $[0,X − 1]$, we also derive a weakened property, $\hat{Prop}$:

$$\forall i \in [0,X − 1] : (timer[i] = TIMEOUT) \Rightarrow (timer[i] = 0)$$

Since $\neg\hat{Prop} \Rightarrow \neg Prop$, our decision ensures that if the identified counterexample violates $Prop$, it also violates $Prop$ and is a valid counterexample in the original model.

On the other hand, we observe that many events trigger the same update on timer. For example, both event = 0 and
event = M leads to the initialization of timer[0]. Thus, we
decide to keep a small set of EventRange. We first partition
EventRange into several equivalence classes:

EventRange = \{ i \in [0, M - 1] \}

where every event in EventRange triggers the same update
on timer[i]. For each equivalence class EventRange, we then
pick one value, EventRange = \{ i \}, so that we can trigger
all updates on timer. In this case, we reduce EventRange
to [0, M - 1]. However, among this range, only events in
\{ X, M - 1 \} is triggered by the attacker, meaning that the
attacker itself cannot trigger all updates on timer. We thus
e large [X, M - 1] to [X, 2M - 1] so that the attacker itself can
triggers the initialization of all timers. Finally, we derive a
small EventRange = [0, 2M - 1] and a mapping function:

f(x) = \begin{cases}
    x, & x \in [0, M - 1] \\
    x + i, & x \in [i, M + |i|] - [1, \frac{M}{2} - 1] \}
\end{cases} (i \in [0, M - 1])

Moreover, f is a surjective function; thereby, for every \( \hat{x} \in [0, 2M - 1] \), we can always find at least one \( x \in [0, N - 1] \)
such that \( \hat{x} = f(x) \). In another word, for every identified coun-
terexample in the abstract model, we can always find at least
one corresponding counterexample in the original model
by applying the inverse function \( f^{-1} \) on event.

By combining the aforementioned two strategies together,
we can successfully reduce the state space of the example
and ensure no wrong property violations. In particular, we
reduce TimerInRange and EventRange to [0, X - 1] and
[0, 2X - 1] respectively.

4.2 Model Checking

The goal of using the general model checker is for vulnera-
bility discovery. Given a model M and security properties,
once the model violates a property, the general MC will gen-
erate a counterexample, an execution trace leading to the
violation. Formally, a model can be defined as consisting in a
finite set of states S, initial states I \( \subseteq S \), the transition relation
T \( \subseteq S \times S \), and a labeling function from states to a finite set of
atomic propositions L : S \( \rightarrow 2^{AP} \) [17]. Table 1 summarizes the
high-level properties to analyze P2PCD and PMPs. For each
property, we first refine \( \phi \), to get a new property \( \phi_i \) such that
\( \phi_i \Rightarrow \phi \) and \( \neg \phi_i \Rightarrow \neg \phi \). For example, a refinement over \( \phi_1 \)
would be at least one CV device should eventually broadcast a
learning request after observing an unknown certificate.

Then, MC is used to find property violations. By analyzing
the counterexample, we can formulate the attack procedure
(\S 5) and analyze the fundamental reasons for identified at-
tacks, which is helpful for the mitigation design (\S 7). Last,
we patch the model to ensure that the general MC will not
generate the same type of violations later.

PMC aims at avoiding manual risk assessment and does
not discard identified vulnerabilities from the general MC.
It helps assess the severity of the exposed vulnerabilities
and thus allows the protocol designers to prioritize the so-

<table>
<thead>
<tr>
<th>ID</th>
<th>Availability properties</th>
</tr>
</thead>
</table>
| 1  | The application layer should be always able to con-
    sume valid incoming packets. |
| 2  | Refinement over \( \phi_1 \): All CV devices should eventu-
    ally learn unknown certificates. |
| 3  | Refinement over \( \phi_2 \): All platoon members should |
    eventually switch to idle state. |

\( \forall s \in S : \sum_{t \in T} (s, s) = 1 \). Since we assign uniform probabili-
ties to concurrent state transitions, for all reachable successor
states of \( s \) in \( Succ_t = \{ t \in T | (s, t) > 0 \} \), the transition proba-
bility between \( s \) and any \( t \) is \( \frac{1}{|Succ_t|} \). A transition matrix can
be derived from the transition probabilities. Thus, PMC can
calculate the likelihood of transitioning from initial states
to any target states. If we can formalize the states of the at-
tack success, PMC can help us generate the attack success
rate. Apart from the probability, PMC can also assign "time"
costs for state transitions, which can be used to quantify time-
related properties. In \S 5, we leverage PMC to quantify the
severity of non-deterministic attacks NI-4, which are defined
as attacks that may not always succeed per attempt. We ob-
serv e that, P2PCD attacks can succeed, only if malicious
packets are delivered to the victim vehicle exactly within the
attack time window. However, the attacker cannot precisely
infer the start and end of the time window, but only roughly
predict the start time. Thus, we use PMC to quantify their
severity based on the success rate and the time delay.

4.3 Implementation

Following the proposed approach, to instantiate CVAnalyzer,
we use TLC [75] as the general model checker due to its ex-
pressiveness of constructing the model, and pick PRISM [42]
as our probabilistic model checker. As the prior step of model
checking, we manually extract the abstract model of the IEEE
1609 protocol family [67] and PMPs [56, 69]. The abstract
model includes two (i.e., \( n = 2 \)) legitimate vehicles and one
malicious vehicle (i.e., the attacker). Then, we need to im-
plement concrete models in the modeling languages used by
TLC and PRISM. As the supported maneuvers of PLEXE
is a subset of VENTOS, we merge them together as one model.

The properties that we want to verify in this paper are shown
in Table 1 and Table 3, covering availability and quantitative
properties respectively.

5 Analysis Results

In this section, we describe 4 DoS attacks in P2PCD and 15
attacks in VENTOS [69] and PLEXE [56] in detail (Table 2).
Then, we analyze the security implications of identified at-
tacks, and quantify the success rate and the average time delay
in packet processing of those non-deterministic attacks.

5.1 P2PCD Vulnerabilities

In summary, CVAnalyzer finds 4 new DoS attacks that can
compromise the availability of CV network. All 4 vulnerabili-
ties come from P2PCD [32], which prevents victim vehicles
from learning unknown certificates (see Figure 4). Without
Table 2: Summary of attacks found in the CV protocols. (N: CV network protocol, P2PCD. A: CV application, PMP)

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Assumption</th>
<th>New?</th>
<th>Implications</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>Response Mute</td>
<td>Known response threshold, optional response verification, enough computing power</td>
<td>Yes</td>
<td>Stop the CV device from sending learning responses; result in traffic accidents (§ 6.2.1)</td>
</tr>
<tr>
<td>N2</td>
<td>Request Mute</td>
<td>Optional response verification, enough computing power, known MAC address</td>
<td>Yes</td>
<td>Stop the CV device from sending learning requests; result in traffic accidents (§ 6.2.1)</td>
</tr>
<tr>
<td>N3</td>
<td>Numb</td>
<td>Known MAC address</td>
<td>Yes</td>
<td>Stop the CV device from recording unknown certificates; result in traffic accidents (§ 6.2.1)</td>
</tr>
<tr>
<td>A1</td>
<td>(Prerequisites)</td>
<td>Available platoon space</td>
<td></td>
<td>Cause traffic collision [1], lead to A3-15</td>
</tr>
<tr>
<td>A2</td>
<td></td>
<td></td>
<td></td>
<td>Interfere the traffic flow stability, decrease efficiency and safety (§ 6.2.2)</td>
</tr>
<tr>
<td>A3</td>
<td></td>
<td>Centralized platoon coordination</td>
<td></td>
<td>Prevent platoon members from performing any maneuvers</td>
</tr>
<tr>
<td>A4</td>
<td></td>
<td>Inappropriate validity check</td>
<td>Yes</td>
<td>Lead to failures of the split maneuver and the leader/follower leave maneuver</td>
</tr>
</tbody>
</table>

knowing necessary certificates, the victim vehicles cannot verify incoming packets; the CV network stack thus cannot deliver data to the application layer. Besides, we discuss the fundamental reasons for these vulnerabilities. Also, we assess their security consequences.

In the following descriptions, two CV devices, Vehicle 1 (V1) and Vehicle 2 (V2), broadcast SPDUs every 100 ms. However, V2 cannot verify packets sent by V1 because V2 does not know the issuer cal of the signing certificate ee1 used by V1. V2 thus wants to learn the unknown certificate cal. For each attack presented below, V1 first sends a trigger SPDU to V2. In the normal case without the attacker, after receiving the trigger SPDU, V2 initializes P2PCD learning process and attaches learning request information in the next outgoing SPDU. V1 will construct and send the learning response after receiving the learning request.

5.1.1 Response Mute Attack

N1 can prevent a peer CV device from sending the learning response. This attack exploits the optional verification of learning responses and the throttling mechanism of P2PCD that limits the number of responses to a single request. The attacker intentionally interact with V1 by sending multiple malicious learning responses to ensure that the response counter of V1 exceeds the response threshold. As a consequence, V1 choose not to send the learning response, and V2 fails in learning the unknown certificate cal.

Assumptions. For successfully carrying out this attack, the attacker needs to know the exact value of the response threshold. For example, the response threshold of BSM is 3 [20]. We assume that V1 does not mandate the verification for incoming learning responses, which is consistent with the current protocol specification (§ 2.1). Also, we assume that the attacker has enough computing power to efficiently construct learning responses that can cause partial hash collision (e.g., low-order 3 bytes collision).

Figure 4: Four P2PCD attacks can break the whole pipeline of P2PCD learning process to prevent the CV device from learning/storing the unknown certificate.

Figure 5: N1: the attacker can stop V1 from sending learning responses to V2 by sending multiple malicious learning responses.

Attack steps. Figure 5 illustrates the attack steps in detail. V1 first sends a trigger SPDU to V2. Instead of immediately sending the learning request, V2 stores the HashedId8 value of the unknown certificate cal in a queue (cf. IEEE 1609.2 [32], Subclause D.4.2.1.1). V2 attaches the HashedId3 value of cal in the learning request field of its next outgoing SPDU. In P2PCD, HashedId8 and HashedId3 stands for the low-order 8-byte and 3-byte hash of a certificate respectively. After receiving the learning request, V1 starts to prepare a learning response. Based on the throttling mechanism, V1 initializes the response backoff timer and the response counter for the requested certificate.

However, the attacker can observe the trigger SPDU and the learning request, so she can determine that V2 wants to learn an unknown certificate from V1. The attacker thus deliberately constructs multiple learning responses, in which the HashedId3 value of the first certificate in the payload matches with the unknown certificate cal. The attacker then sends out these malicious packets to saturate V1’s response counter (i.e., making it no less than the response threshold). On receiving malicious learning responses, V1 wrongly updates its response counter (via AddCertificate primitive defined in IEEE 1609.2). When the response backoff timer expires, V1 checks whether the response counter is less than...
or equal to the response threshold. Obviously, based on the current status of the response counter, V1 decides to discard the response at this time.

Discussion. The reason for N1 can be attributed to the use of truncated hash. By design, the hash function should be resistant to collision attacks. However, the use of truncated hash value compromises the security provided by the hash function. For example, for HashedId3 used in CV network (i.e., three-byte hash), collision could be found in the brute-force number of $2^{24}$. Most importantly, the response counter uses HashedId3 as the identifier, which means that the attacker can manipulate the response counter if she constructs certificates leading to the partial hash collision. On the other hand, as introduced in §2.1, IEEE 1609.2 does not mandate the verification for the learning response. Thus, it is still possible that some poorly implemented CV protocols may not verify the incoming learning response but just store certificates in the payload. Even if the CV device mandates the verification, the attacker can collect certificates with the attacker-desired hash values offline (§7). Note that, since P2PCD learning responses do not carry digital signatures, the attacker does not need to possess a legitimate certificate to launch N1, making the attack much more stealthy.

5.1.2 Request Mute Attack

Both N2 and N3 can stop CV device from sending learning requests. Similar to N1, N2 exploits the hash issue. Readers can refer to §A for more details.

N3 exploits the unicast capability and injects a malicious SPDU with the same learning request field (i.e., the HashedId3 value of cal) as what V2 intends to send. As a result, V2 can observe the malicious learning request and decides not to send its own learning request. V2 hence fails in learning unknown certificate cal because V1 does not receive any learning requests.

Assumptions. To successfully launch this attack, the only requirement is that the attacker needs to know the MAC address of the victim vehicle V2. This is reasonable because the attacker can monitor all traffic in the network; it can thus observe V2’s MAC address from packets sent by V2.

Attack steps. As presented in Figure 6, V2 initializes P2PCD after receiving a trigger SPDU from V1. V2 stores the HashedId8 value of the unknown certificate cal in a queue. Meanwhile, since the attacker can observe the trigger SPDU, she constructs a malicious learning request, in which the learning request field m.lr equals to the HashedId3 value of the unknown certificate cal. In P2PCD, after receiving a learning request, V2 removes any matching HashedId8 entries in the queue. Therefore, V2 removes the entry of the unknown certificate h8(cal) in the queue, where h8 is a function to get the low-order eight-byte hash of the input. As the queue becomes empty, V2 decides not to attach the learning request information in the next outgoing SPDU. Consequently, V2 is unable to learn the correct unknown certificate.

Discussion. The fundamental reason for N3 is that once a vehicle observes an active P2PCD learning request, it will not send the learning request for the same unknown certificate. In the normal case, this mechanism is helpful to reduce the number of simultaneous learning requests in the fly. However, the attacker can unicast the learning request to the victim vehicle. Notably, the attacker should not send such learning request to the owner of the unknown certificate (i.e., V1 in Figure 6). This attack misleads the victim vehicle to believe that some other legitimate vehicles are requesting the same unknown certificate. The protocol designers do not consider the use of unicast in P2PCD, which makes the victim vehicle vulnerable to N3. On the other hand, N3 does not require the attacker to possess a legitimate certificate to sign the learning request but only uses self-generated certificates. As long as the digital signature of the learning request is valid, the vehicles will process the learning request field in the packet header. In this case, the signing certificate of the malicious learning request will be treated as an unknown certificate and will trigger another P2PCD learning process. Therefore, even if the certificates used by the attacker is revoked, the attacker can always generate new certificates for future use.

5.1.3 Numb Attack

First, like N3, this attack exploits the unicast capability and injects a malicious SPDU with the same learning request field (i.e., the HashedId3 value of cal) as what V2 intends to send. This causes the same consequence as N3, in which V2 chooses not to send the learning request and thus cannot learn the unknown certificate. Then, due to the request active timer (e.g., reqActiveTimer), V2 still thinks that there should be an active request in the fly. Therefore, while receiving the next trigger SPDU, V2 chooses not to add the HashedId8 value of the unknown certificate cal into the queue and keeps waiting for learning responses.

Attack steps. As described in Figure 6, this attack is similar to N3, but the attacker has different attack goal that it tries to prevent the victim vehicle V2 from recording unknown certificates. Since V1 broadcasts BSMs every 100 ms, V2 will receive a trigger SPDU again in a few milliseconds. At this time, V2 still cannot verify the incoming packet. However, because the request active timer has been initialized in the last communication round, and the timer is usually set to 250 ms [20], V2 believes that there is still an active learning request in the fly. Thus, V2 does not add anything into the queue, which means that it will not attach any learning request information in the next outgoing SPDU. V2 cannot recover from this malicious state until the request active timer expires.

Discussion. N4 has the same fundamental reasons as N3. The only difference is that the request active timer blocks the victim vehicle from recording unknown certificates in that the initial value (i.e., 250 ms) of the timer is around 3 times larger than the broadcast interval (i.e., 100 ms). Fortunately, P2PCD allows the user to configure the parameters for the
What is the success rate of the attack? What is the expected time delay of processing next SPDU? Quantitative properties

Before sending a learning request unknown cert.

We observe that, N1-4 can succeed, only if the attacker delivers the malicious packets to the victim vehicle exactly within the attack time window. However, one challenge for the attacker is that she cannot precisely determine the start or end of the attack time window but can only roughly estimate the time window. Thus, we are motivated to quantify the probability of successfully launching the attack by using the probabilistic model checker in CVAnalyzer.

Table 3: Quantitative properties used by CVAnalyzer to quantify the security consequences of N1-4

<table>
<thead>
<tr>
<th>ID</th>
<th>Quantitative properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>What is the success rate of the attack?</td>
</tr>
<tr>
<td>W2</td>
<td>What is the expected time delay of processing next SPDU?</td>
</tr>
</tbody>
</table>

5.1.4 Assessment

We observe that, N1-4 can succeed, only if the attacker delivers the malicious packets to the victim vehicle exactly within the attack time window. However, one challenge for the attacker is that she cannot precisely determine the start or end of the attack time window but can only roughly estimate the time window. Thus, we are motivated to quantify the probability of successfully launching the attack by using the probabilistic model checker in CVAnalyzer.

Table 4: Attack assessment results of N1-4.

<table>
<thead>
<tr>
<th>ID</th>
<th>Attack packet</th>
<th>Attack time window</th>
<th>Succ. Rate</th>
<th>Time delay (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>0-250 ms</td>
<td>99.47%</td>
<td>580 (280 + 300)</td>
<td></td>
</tr>
<tr>
<td>N2</td>
<td>≤ 100 ms</td>
<td>99.99%</td>
<td>370 (280 + 90)</td>
<td></td>
</tr>
<tr>
<td>N3-4</td>
<td>≤ 100 ms</td>
<td>99.99%</td>
<td>370 (280 + 90)</td>
<td></td>
</tr>
</tbody>
</table>

Table 4 summarizes the quantification results. Since N3 and N4 use the same type of packet to attack the victim vehicles, and the attack time window of them are the same, we merge these two attacks together and quantify the probability results based on the type of attack packet.

For N1, the success rate is 99.47%. We set the response threshold as 3 in our experiments. To successfully launch one attack, the attacker has at least send 4 malicious learning responses, while the rest attacks only need to send one malicious packet. This is why the success rate of N1 is slightly lower than other three attacks. For N2-4, the success rates are 99.99%. If V2 is able to send the learning request before receiving the malicious packet, the attacker will fail. However, this is unlikely to happen based on our results.

Figure 7: The success rate of N1-4 under packet loss.

To have a deeper understanding how the network factor will affect the success rate, we leverage packet loss to demonstrate the capability of PMC. Figure 7 show that the success rate of N1 decays much more than the other three attacks, because the attacker of N1 needs to successfully send at least 4 malicious packets to ensure success. As N2-4 target the same attack time window, they have the same success rate. For N1-4, the attacker should immediately launch the attack once the victim vehicle enters her communication range. Bai et al. [7] show that the packet loss rate (PLR) and the distance between two CV devices are positively correlated in real-world settings. In a freeway environment, the PLR is around 42% if two CV devices are 450m apart, in which 450m is the longest communication distance presented in their study. Thus, we highlight the success rate when the victim vehicle enters the attacker’s communication range, which is the worst case for the attacker (PLR: 42%). Although the packet loss decreases the attack success rate, it also affects the transmission of normal packets, leading to the loss of critical CV safety packets.

Besides, CV communication is time-sensitive [2, 3, 25], so we would like to know the time delay caused by one round of N1-4, which is defined as the time duration from waiting for the trigger SPDU to successfully processing an SPDU from other vehicles. By knowing this, we can infer how long the CV network will recover from the attack if the attacker terminates attacking.

Table 4 shows that three of them can at least double the time delay in packet processing. During the experiments, we notice that there still exists 280 ms time delay even if we disable the attacker, which is one-time delay introduced by P2PCD itself. For N2, the extra time delay introduced by the attacker is 90 ms, around one broadcast interval, because the malicious learning response cancels out the learning request process triggered by the SPDU from V1. V2 thus needs to wait for next SPDU from V1, which takes one more round of broadcast interval. For N1, N3, and N4, the extra time delay caused by the attack is about 300 ms. If the attacker stops attacking at some time, it takes around three broadcast intervals (i.e., 300 ms) for V2 to recover from DoS.

In N1, the extra time delay comes from the long processing time of P2PCD, due to the long time interval of the response backoff timer, with a random timeout value between 0 and 250 ms. As shown in Figure 5, the attacker sends malicious learning responses to V1 right after V1 initializing the response backoff timer. Since the attack occurs at a very late stage, all the time before the transmission of the learning request...
become useless. Also, a new P2PCD learning process to the unknown certificate ca1 will not be initialized again until both the response backoff timer of V1 and the request active timer of V2 expire. After that, V2 needs to initialize P2PCD again; thus, one round of N1 double the one-time delay of P2PCD. In N3 and N4, V2 is unable to process incoming trigger SPDUs until the request active timer expires. However, this timer is usually set to 250 ms, which largely increases the time delay.

5.2 PMP Vulnerabilities

CVAnalyzer identifies 15 attacks in the PMPs of VENTOS [69] and PLEXE [56] (see Table 2). Among identified vulnerabilities, A1-4 are not directly related to availability issues but are building blocks of other attacks. Although the PMPs analyzed in this paper are academic prototypes, our main contribution is the verification methodology, which can be generally applied to future PMP protocols. Our results demonstrate the necessity of such a systematic verification methodology: using manual efforts, a very recent work [1] can only uncover 1 vulnerability (A1). In contrast, using CVAnalyzer for the same PMP implementation, we are able to automatically uncover not only the same one but also 14 more (A1-15), which demonstrates both substantially improved efficiency and effectiveness.

In the following descriptions, V1 and V2 still stand for vehicles. V1 is a platoon leader, and V2 is usually a follower. Their relative positions differ case by case.

5.2.1 PMP Attack Prerequisites

A1 and A2 allow the attacker to become a valid platoon leader and follower. Abdo et al. [1] have demonstrated that A1 can lead to the traffic collision and slow down the emergency vehicle. Although they do not directly cause security or safety breaches, we list A1 and A2 alone because they are prerequisites of other attacks. As described in §2.2, a platoon leader will send a merge request to a front platoon, if the combined platoon size is no greater than the optimal platoon size. Thus, the attacker can claim herself as a front platoon to take over another platoon or initiate a merge maneuver to join a platoon, leading to the success of A1 or A2 respectively.

5.2.2 Split Trigger Attacks

Both A3 and A4 (see §A for details on A4) can trigger the split maneuver at any positions. Without sacrificing her own speed stability, in A3, the attacker can further lead to a high-rate of vehicles entering and exiting a platoon, which decreases efficiency and safety [5].

Attack steps. In A3, the attacker first merges with V1 as a malicious follower. Then, V2 sends a MERGE_REQ to V1 and join the platoon. At this time, the attacker intentionally sends a LEAVE_REQ with a wrong depth number of 2 to V1, in which the depth number indicates the splitting vehicle is V2. V1 thus wrongly initiates the split maneuver at the position of V2. After the split process, V2 receives beacon messages from the attacker and merges with the front platoon again, as described in §2.2. By repeatedly triggering merge and split maneuver of V2, the attacker downgrades the speed stability of V2.

Discussion. The reason for A3 is that the platoon leader does not verify whether the platoon depth in the LEAVE_REQ matches with the sender ID or not. Usually, if the sender ID is related to unique signing certificates [32], it is difficult for the attacker to falsify the identity. However, the design of PMP uses the depth information as the identity, which can be easily modified by the attacker. Thus, PMP opens a door for the attacker to trigger the leave maneuver, leading to a split maneuver at arbitrary positions.

5.2.3 PMP Block Attacks

This is the most common type of vulnerabilities (A5-14) in the current PMP design of both VENTOS and PLEXE, which misleads the victim vehicle to stay at a busy state. We only describe A7 here. Please refer to §A for more details on others.

Attack steps. In A7, the attacker first joins the platoon by launching A2 and aims at blocking the split maneuver. Usually, only the platoon leader can initiate the split maneuver, but the platoon follower cannot. However, the attacker can leverage A3 and A4 to mislead the platoon leader to send a SPLIT_REQ to any specified platoon members. In A7, the attacker receives a SPLIT_REQ from V1 but chooses not to reply with a SPLIT_ACCEPT. Thereby, the platoon leader will keep waiting for the split reply. At this time, if V2, which is ahead of the attacker, approaches the destination and wants to leave the platoon, the leader V1 will not be able to process the leave request or manage the split process to create space for V2. Without enough space at the front and rear of the vehicle, it is dangerous for V2 to directly change the lane.

Discussion. The fundamental reason for A5-14 is the lack of error recovery mechanism on communication failures. By design, the CV network stack does not provide reliable communication; it is the applications’ responsibility to handle communication failures [34]. Researchers have already discussed the impact of communication failures on the CACC controller [5, 47], but do not pay much attention to communication failures on PMP. Also, we observe that PMPs in both VENTOS and PLEXE do not consider “offline” platoon members; thus, they do not design any error recovery mechanisms to reset the vehicle’s state. Although we understand the PMPs of VENTOS and PLEXE are research prototypes, identified PMP block attacks still emphasize the importance of error recovery mechanisms in CV application design.

5.2.4 Inconsistency Attack

This attack aims at assigning a wrong depth number to a victim follower, which is inconsistent with the index in the platoon member list. The platoon depth is used in the split maneuver, so the inconsistent depth number can lead to failures of the split maneuver and the leader/follower leave maneuver.

Attack steps. In this attack, the attacker first joins V1’s platoon as a follower. Then, the attacker slows down to create
large gap (e.g., 100 m) between herself and V1. At this time, V2 change its lane and drives behind V1. V2 receives the beacon message from V1 and sends a ME A GE_REQ to V1. After merging with V1, V1 updates its local state by appending V2’s ID to the platoon member list, indicating the real platoon depth of V2 is 2. However, V2 only receives a beacon message with the depth of 0 from the front vehicle V1; V2 thus wrongly sets its platoon depth to 1. At this time, the attacker sends a LEAVE_REQ to V1. Since, V1 thinks that the attacker is a middle follower, and V2 is behind the attacker, it sends a SPLIT_REQ to V2 to create rear space for the attacker. In VENTOS, we observe that CHANGE_PL does not present the absolute depth but carries the relative change of depth information, because it is convenient for the platoon leader to send all followers one CHANGE_PL rather than multiple different CHANGE_PL. During the split maneuver, V2 receives a CHANGE_PL from V1 with the depth change of −2. While updating the depth information locally, PMP of V2 throws an error for the invalid new depth: 1 − 2 = −1, which may compromise the availability of PMP, as well as terminates the split maneuver.

Discussion. The reason for A15 can be attributed to the inconsistent platoon view on the platoon leader and follower. When joining a platoon, the vehicle relies on the depth information in the beacon message from the front vehicle to set its own depth number, while the platoon leader simply appends a new member to the platoon member list without checking the relative location information. If the front vehicle is a benign last follower, no inconsistency will appear; otherwise, any CHANGE_PL from the leader to the victim vehicle will lead to a wrong new depth number. However, the attacker can either create a large gap for the victim vehicle (A11), or can send a beacon message with a wrong depth number if the attacker is the last follower.

6 Evaluation

In this section, we conduct extensive experiments and answer the following three research questions:

- **RQ1**: Are identified vulnerabilities practical in a real-world setting?
- **RQ2**: What are the security/safety impact of identified vulnerabilities?
- **RQ3**: What is the runtime performance of CVAnalyzer?

6.1 RQ1: Practicality of Identified Attacks

We implement and validate all attacks from both P2PCD and PMP, detected by CVAnalyzer, in a real-world testbed, which thus concretely demonstrates the effectiveness of CVAnalyzer. Interestingly, we also find some poor implementation details in real-world CV devices that actually make our attacks easier.

6.1.1 Testbed Setup and Tool Preparation

As shown in Figure 8, we set up a CV network using three Cohda OBUs [19] in our lab. Among these three OBUs, denoted as OBU 1, 2, and 3 respectively, OBU 1 and 2 are used as victim CV devices, and OBU 3 is used as the attack device.

To control the experiments, we connect a laptop with three OBUs via Ethernet connections.

![Figure 8: Testbed setup for attack validation.](image)

The Cohda OBU that we use in our experiments is an ARM embedded device running Ubuntu 16.04. It implements the latest version of the CV network stack, which conforms with IEEE 802.11p [37, 38], IEEE 1609-2016 [32–34], and SAE J2735-2016 [20]. Notably, the implementation of IEEE 1609.2, called Aerolink, is developed by OnBoard Security [52] and closed source.

**Victim OBU setup.** To implement the CV communication model, both victim OBUs run a simple program that periodically broadcasts a correctly-signed SPDU. This broadcast-based communication also allows the attacker to observe all network traffic. For P2PCD, we place random data in the SPDU. For PMP, the SPDU stands for the beacon message, which contains the platoon ID and depth. Besides, both OBUs run PMP programs that are extracted from the source codes of VENTOS [69] and PLEXE [56].

For two different protocols (i.e., P2PCD, PMP), we assign different roles to OBU 1 and 2. In P2PCD attacks (i.e., N1–4), following the same assumption in §5.1, OBU 2 cannot verify packets sent by OBU 1 due to a missing certificate, so OBU 2 wants to initialize P2PCD to learn the unknown certificate from OBU 1. In PMP attacks (i.e., A1–15), by default, OBU 1 and 2 belong to the same platoon. OBU 1 and 2 are the platoon leader and the platoon follower respectively.

**Tool preparation.** To launch the attacks, we need to prepare tools that allow us to (1) parse and construct arbitrary packets and certificates, and (2) sign and verify CV network packets correctly. For (1), we use as n1c [71] to extract C data structures used by CV network services from ASN.1 modules in protocol specifications, and port platoon message types from the source codes of VENTOS and PLEXE. For (2), we follow IEEE 1609.2 to implement the signing and verification functionalities. We start from ECDSA APIs provided by OpenSSL 1.1.0j [53]. The elliptic curve and the hash function that we use with ECDSA is NIST P-256 and SHA-256, respectively. We cross-validate the correctness of our tools using APIs of Cohda CV network stack. The Cohda CV network stack can process packets and certificates generated by our tool without throwing any errors.

**Certificate configurations.** As N1–4 require triggering P2PCD, we need to configure the pre-installed certificates in both victim OBUs to ensure that OBU 2 cannot construct a certificate chain while verifying packets sent by OBU 1. Both OBU 1 and 2 can correctly verify packets from OBU 3
We add both ca1 and ca2 to the local certificate database of OBU 1, but only add ca2 to the database of OBU 2. To generate end-entity certificates for signing packets, we utilize ieeeAcfGenerator in Cohda SDK to issue two batches of certificates: batch1 for OBU 1 and batch2 for OBU 2. Each batch is an Aerolink-specific file and contains 20 end-entities certificates. Besides, we use ca2 to issue another end-entity certificate ee3 for the attacker so that OBU 1 & 2 can construct a valid certificate chain for packets sent by the attacker.

Apart from generating these normal certificates, we also need to construct certificates that can cause hash collisions. In N1 and N2, the first certificate in the malicious learning response should match with the low-order 3-byte and 8-byte hash value of the unknown certificate respectively. We therefore use our certificate generator to construct two CA certificates: ca1-h3 and ca1-h8, which can lead to 3-byte and 8-byte hash collision with ca1.

**Attack programs.** Following the attack processes in §5, we implement different attack programs. For each attack program, we set the start condition and the fail condition. The attack programs will stop only if the fail conditions are satisfied; otherwise, they will keep running. For P2PCD attacks, the attack fails if she observes any learning response from OBU 1. For example, the attack program for N1 will send malicious learning responses after observing a learning request sent by OBU 2 (i.e., Vehicle 2 in Figure 5). If it observes a learning response sent by OBU 1, the program will stop, which means that the attack fails. For PMP attacks, the attack fails if the victim platoon member can still finish the merge, split, leave, or dissolve maneuver.

### 6.1.2 Validation Results

In the real-world experiments, we find that all attacks from P2PCD and PMP are successfully validated. Interestingly, we further find that some implementation details in Aerolink can actually make P2PCD attacks, N1 and N2, even easier and even block the CV communication indefinitely.

First, we observe that N1 and N2 can indefinitely block the P2PCD learning process. Based on our model-checking findings in §5.1, once the adversary stops sending malicious learning responses, the victim devices should eventually be able to recover from DoS. However, in our real-world experiments, we find that even after the attack program terminates, OBU 2 still cannot learn the correct unknown certificate from OBU 1. After analyzing the execution log, we find that OBU 1 keeps sending the fake certificate (i.e., ca1-h3), while OBU 2 sends learning requests for the unknown certificate ca1 to OBU 1. By design, a CV device responds to an incoming learning request only if the learning request field matches with a signing certificate which is recently used by that device. With the help of a binary disassembler called Hopper [11], we find that Aerolink actually does not check whether the certificate used for a learning response is indeed a recently used certificate. For example, in N1, OBU 1 stores the fake certificate (i.e., ca1-h3) carried by the malicious learning response from the attacker. Thus, during the preparation of the future learning response, OBU 1 has two candidates, ca1 and ca1-h3, as they have the same low-order three-byte hash. When receiving learning requests, OBU 1 always picks ca1-h3 and sends it to OBU 2, which thus permanently prevents OBU 2 to learn the correct certificate.

Second, to launch N1, we find that the attacker only needs to send 3 malicious learning responses instead of 4. Before running real-world experiments, we first measure the response threshold in Aerolink, and find that the threshold set in Aerolink is actually 2 instead of 3 in the protocol specifications. This finding is also confirmed using Hopper. In this case, the attacker only needs to send 3 malicious responses to succeed. Although this may not be a big improvement for the attacker, it still uncovers an implementation choice in Aerolink that is unexpectedly favorable to the attacker.

Third, we find that N2 only requires 3-byte hash collision rather 8-byte hash collision, which largely lowers the bar of launching N2. In P2PCD, by design, a CV device records an unknown certificate by adding the identity of that certificate (i.e., an 8-byte hash value) into a queue. If the 8-byte hash of a certificate in an incoming learning response matches with any entries in the queue, that entry will be removed. To launch N2, the attacker has to intentionally cause the 8-byte hash collision to let the victim CV device wrongly remove an entry in the queue. However, according to our binary analysis through Hopper, we find that Aerolink actually uses a 3-byte hash of the unknown certificate to record its status. Therefore, in our real-world experiments, we use ca1-h3 in N2, and the results further validates this finding. Later in §7, we will show why this small truncated hash (e.g., 3-byte hash) is not secure enough. Although the protocol specification does not clearly state how to record unknown certificates, Annex D in IEEE 1609.2 [32] gives an example of P2PCD implementation that uses the 8-byte hash as the identity to record the unknown certificate. Also, while recording the unknown certificate, the most complete identity about the unknown certificate is the 8-byte hash value. A CV network implementation should always use complete information rather than truncated information.

### 6.2 RQ2: Attack Impact

The following two case studies demonstrate the impact of identified attacks: (1) P2PCD attacks can lead to traffic accidents, which eliminates the benefits of V2V safety applications (e.g., Forward Collision Warning (FCW)); (2) PMP attacks can affect the speed stability of the victim vehicle.

**Simulator setup.** To evaluate the impact of identified attacks, we use a simulator, VENTOS (VEhicular NeTwork Open Simulator) [69], so that we can demonstrate the driv-
Table 5: Vehicle parameters in the rear-end collision scenario.

<table>
<thead>
<tr>
<th>Vehicles</th>
<th>Initial Speed</th>
<th>Max. Speed</th>
<th>Max. Decel.</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leader (V1)</td>
<td>30 m/s</td>
<td>30 m/s</td>
<td>5 m/s²</td>
<td>10 m</td>
</tr>
<tr>
<td>Follower (V2)</td>
<td>20 m/s</td>
<td>30 m/s</td>
<td>2 m/s²</td>
<td>5 m</td>
</tr>
</tbody>
</table>

6.2.1 Safety Impact

By design, the CV safety application promises to increase personal safety [63]. However, our experiment results show that P2PCD attacks can fully eliminate the benefits of CV applications (e.g., Forward Collision Warning (FCW)), violating the original goal of CV applications.

Rear-end collision scenario w/ FCW. We first set up a rear-end collision scenario and demonstrate that vehicles with Forward Collision Warning (FCW), a V2V safety application, can avoid the accident (Figure 9a). The rear-end collision scenario includes a leading vehicle (V1) and a following vehicle (V2) with the initial parameters in Table 5. FCW alerts the driver in order to help avoid the severity of crashes into the rear end of other vehicles on the road [63]. We follow the FCW’s design in Cohda SDK to actively monitor the distance between two vehicles. Once the distance is smaller than the safe distance, FCW will warn the driver. As FCW does not directly control the vehicle, after receiving FCW warnings, we ask the simulated vehicle to maintain a safe speed. Notably, we leverage Krauss car-following model [40], which is collision-free, to calculate the safe distance and safe speed.

During the simulation, both vehicles drive in the same lane. By exchanging BSMs, they can monitor each other’s speed, position, and acceleration. The initial distance between two vehicles is 30 m, which is smaller than the safe distance at that time, thus triggering FCW. After starting the simulation for 10 s, V1 suddenly stops at the maximum deceleration (i.e., 5 m/s²). Figure 9a shows that, before 10 s, V2 keeps increasing the distance to the leading vehicle due to the FCW. Therefore, after the leading vehicle suddenly decelerates, V2 has enough space to slow down safely.

Vehicles w/ FCW under attacks. Then, we place an attacker on the roadside who follows § 5.1 to launch P2PCD attacks and aims at causing traffic accidents, leading to a rear-end collision shown in Figure 9b. At the beginning of the simulation, both vehicles launch P2PCD to exchange certificates so that they can verify and process following BSMs. However, P2PCD attacks prevent them from learning certificates, meaning that they cannot process any BSMs from the peer vehicle. During the simulation, we observe that FCW is never triggered, so V2 accelerates to the maximum speed and follow V1. At 10 s, V1 starts decelerating at the maximum deceleration (i.e., 5 m/s²). Since two vehicles are too close to each other (i.e., 54 m), and the maximum deceleration of the V2 is 2 m/s², V2 eventually collides into the rear end of V1.

6.2.2 Traffic Efficiency Impact

By design, CACC aims at increase traffic throughput and improve traffic flow stability [46, 57, 72]. However, A3 and A4 can interfere with the traffic flow stability, even without sacrificing her own speed stability, which violates the design goals of CACC. We place V1, the attacker, and V2 sequentially in the same lane and follow the attack steps of A3 to run the simulation for 100 seconds. Figure 10 presents the speed profiles of V2, the victim platoon. In the normal case (Figure 10a), all vehicles will eventually reach a stable speed of 20 m/s; after launching the attack starting around time 27 seconds, we increase the standard deviation of V2’s speed by 43%, further disturbing the following traffic.

6.3 RQ3: Performance of CVAnalyzer

Table 6 presents the runtime performance of CVAnalyzer. We run CVAnalyzer on a server with four 2.60GHz (8-core) CPUs and 128G memory. CVAnalyzer first explores all reachable states and then verifies given properties. Notably, without applying the state reduction, these two model checking tasks will take too long to explore reachable states. The results highlight the importance and effectiveness of state reduction.
Table 6: Runtime statistics of CVAnalyzer.

<table>
<thead>
<tr>
<th>Attacks</th>
<th>Distinct States</th>
<th>Model Checking Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2PCD</td>
<td>2209351</td>
<td>16s</td>
</tr>
<tr>
<td>PMP</td>
<td>142133161</td>
<td>1h 35min</td>
</tr>
</tbody>
</table>

7 Defense Proposals

Based on the discussions from previous sections, we propose defense solutions at the protocol design level:

1. Mandate verification for all learning responses;
2. Increase the truncated hash size for the issuer field in the certificate and the learning request field in SPDU;
3. Disallow unicast learning requests;
4. Bind the sender identity with the CV certificate;
5. Track platoon configuration data locally or remotely;
6. Design and integrate an error recovery mechanism.

Defense against N1 and N2. Solution 1 and 2 are proposed for N1 and N2. Solution 1 by nature prevents N1 and N2 with local certificates. However, such solution can be evaded if attackers are still able to collect legitimately signed certificates with the attacker-desired hash values by sniffing CV network traffic. As estimated in Table 7, as long as the attacker can collect over 12000 different certificates, she can almost guarantee (>98% probability) that she can always have a certificate ready for triggering a 3-byte hash collision, which thus allow her to still launch N1 and N2 in real time. Collecting this many different certificates is completely realistic, considering that such collection process can be done offline. In addition, the collection process can also be greatly accelerated since the attacker can actively broadcast learning requests to trigger surrounding vehicles to return certificates with desired hash values, and also can place multiple attack devices in different locations to parallelize the collection process.

Table 7: Number of hash values needed for hash values of n-bits to cause a hash collision probability at p.

<table>
<thead>
<tr>
<th>Number of hash values (k)</th>
<th>Prob. of hash collision (p)</th>
<th>Number of bits of the hash value (n)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.5</td>
<td>24 64 80 256 512</td>
</tr>
<tr>
<td></td>
<td>0.99</td>
<td>4823 5.00 1.29 4.01 4.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12431 1.30 3.34 1.03 3.51</td>
</tr>
</tbody>
</table>

Solution 2 aims at increasing the difficulty of causing a hash collision, the key enabler for N1 and N2. As shown in Table 7, it will be much more difficult for the attacker to compute or to gather proper malicious learning responses. However, this will increase the DSRC packet size and thus may decrease the network performance, e.g., increasing network latency. We have reached out to the protocol developer, and confirmed that it is indeed a design choice to reduce the DSRC packet size. Thus, when applying Solution 2, the new size of the truncated hash type needs to be carefully chosen to balance such trade-off between security and protocol performance.

From our discussion above, neither solution 1 or 2 can fully eliminate the attack possibilities for N1 and N2. Thus, to maximize the chance of preventing the attack in practice, the best choice would be using them jointly.

Defense against N3 and N4. Solution 3 is proposed for N3 and N4, which thwarts both attacks by making it impossible to unicast the malicious learning request to block the P2PCD process. However, the down side is that this may break designed usage of unicast-based learning request. For example, as specified in IEEE 1609.0-2019 [36], CV applications will decide whether to use either unicast or broadcast, while receiving advertised services. Systematically understanding this trade off requires surveying and quantifying the demands of unicast-based learning requests at the CV application level, which we leave as future work.

Defense against A3. Solution 4 can prevent the attacker from triggering the split maneuver at arbitrary positions, but cannot stop her splitting succeeding platoon members. The certificate defined in IEEE 1609.2 [12, 32] provides a unique identity for each CV device. Safety-critical CV applications like PMP should always use unique and secure identities (e.g., certificates) rather than using self-defined identity (e.g., depth number), which is easily spoofed by the attacker. However, the attacker can still send a LEAVE_REQ to split at the succeeding vehicle and herself, which is a designed follower-leave behavior. The attacker can then join back to the platoon and launch the attack repeatedly. To completely address A3, we may require the assistant of misbehavior detection [12]. For example, a vehicle that keeps leaving and joining a platoon is highly suspicious. Designing an effective misbehavior detection requires comprehensively characterizing malicious behaviors, which we leave as future work.

Defense against A4 and A15. Solution 5 aims at eliminating wrong and inconsistent platoon information caused by A4 and A15. In centralized PMP, a platoon leader is responsible for passing platoon configuration data to the new leader, when it leaves the platoon. The new leader can only accept the information from the old leader because it does not store any platoon configuration data. The design goal of the centralized PMP is to improve coordination efficiency and to enhance privacy because followers dynamically enter and exit the platoon [5]. However, the centralized design sacrifices the security, as a malicious leader can provide wrong platoon configuration data. To address A4 and A15, on one hand, the platoon members can maintain a local copy of platoon configurations. On the other hand, RSUs can also provide services to remotely assist platoon members for tracking platoon configurations and guarding PMP commands [1]. As RSUs are often deployed and managed by trustworthy authorities, platoon members can rely on the infrastructures to correct wrong or inconsistent information.

Defense against A5-14. Solution 6 is straightforward and proposed for all PMP block attacks. As we mentioned before, CV applications should design their own error recovery mechanisms. With the error recovery mechanism, PMP should be able to recover from continuous packet loss. For example, PMP can define the retransmission and timeout threshold to avoid hanging at specific states. Apart from the classic solu-
tion to communication failures, it’s worth noting that PMP should also adjust the intra-platoon spacing between the “off-line” member and the trailing platoon members accordingly to avoid traffic collision. If necessary, the platoon leader can dissolve the platoon and falls back to ACC mode.

8 Related Work

CV security analysis. Since the idea of VANET (i.e., the original idea of CV) has been around for more than ten years, many researches have already studied general threats to CV network [4, 28, 29, 44, 55, 57]. However, as discussed in §1, existing works generally suffer from three limitations: (1) rely on manual inspection to identify potential threats [44, 57], as opposed to automatic discovery in our work, (2) focus on security properties such as integrity, confidentiality, and privacy, as opposed to availability in our work, and (3) focus on prior generations of protocols or are conducted before the standardization of IEEE 1609 [4, 28, 44, 57], as opposed to the latest version studied in our work.

Model checking security protocols. Model checking is a mature formal verification technique for finite state concurrent systems, and has been applied to several complex network protocols [22, 26, 30, 48, 50]. These works aim at exposing vulnerabilities in network protocols but does not consider quantitative assessments. CVAnalyzer can finish the attack discovery and the quantitative threat assessment without touching implementation details. Therefore, CVAnalyzer can be used by the protocol designer to evaluate the correctness of the protocol and also understand the severity of identified attacks, which can further guide the design of mitigation solutions. Also, this can largely minimize the cost to fix vulnerabilities, as all problems can be solved at the early stage.

Secure membership management. For a wireless ad-hoc network, a secure membership management system is necessary. Usually, network nodes form a peer group to share data with each other [49, 58, 76], and a group leader or other trusted entity is responsible for membership management. Wagner et al. [70] designed a decentralized blockchain-based system membership management for the platoon, in which each platoon member maintains a local copy of the blockchain, storing platoon information; however, it has scalability issues.

Due to the high mobility, the latest CV network does not form different communication groups, but adopts the digital signature (ECDSA), with the support of a PKI system, SCMS [12], to secure the communication. Any CV devices with valid certificates can broadcast data to others. To manage membership, the recently deployed SCMS [12] introduces misbehavior detection to identify malicious or malfunctioning members and then revoke their certificates. For our attacks, the certificate revocation in existing SCMS cannot prevent P2PCD attacks but can mitigate PMP attacks. The learning response in N1 and N2 does not require any signing certificates, so the certificate revocation cannot prevent the attacker from launching these two attacks. In N3 and N4, the attacker can always generate new syntax-valid certificates for the learning request (i.e., an SPDU). Since the vehicle cannot distinguish the self-generated certificates with unknown certificates, the learning request field will still be processed. Unless the vehicle can always connect to the PKI (through RSU) to check the validity of unknown certificates, it is impossible to prevent the attacker from using self-generated certificates in the current CV network stack. Unfortunately, communication with the infrastructure may not always be present due to the deployment difficulties. We admit that if the PKI supports the online certificate status check, with the infrastructure coverage increase, the impact of P2PCD attacks will be diminished.

9 Conclusion

In this paper, we present CVAnalyzer that harnesses the attack discovery capability of the general model checker and the quantitative threat assessment of the probabilistic model checker to automate the analysis. CVAnalyzer successfully detects 4 new DoS attacks in P2PCD and 15 attacks in PMP, also, we construct practical exploits and validate them in a real-world testbed. We have reported 4 P2PCD attacks to IEEE 1609 Working Group [35] and received confirmations. Also, we discuss the fundamental reasons for these vulnerabilities and propose effective mitigation solutions.

Future work. In the future, we would like to extend CVAnalyzer to verify more security properties, such as unlinkability. Though we only inspect the availability property in this paper, CVAnalyzer is actually general and can be extended to improve the verification capabilities. On the other hand, CVAnalyzer can be also extended to support other protocols in the context of CV (e.g., SCMS [12]). Also, we would like to improve the usability of CVAnalyzer. For example, we can introduce an intermediate representation for the model that can be automatically converted into the modeling language used by different model checkers. Therefore, we do not need to write the model twice for two different model checkers.

Acknowledgments

We would like to thank Yulong Cao, David Ke Hong, Yuru Shao, and the anonymous reviewers for providing valuable feedback on our work. This research was supported in part by an award from Mcity at University of Michigan, and by the National Science Foundation under grant CNS-1930041, CNS-1526455, CNS-1850533 and CNS-1929771.

References


[33] IEEE 1609 WG. Ieee standard for wireless access in vehicular environments (wave) – multi-channel operation. IEEE Std 1609.4-2016 (Revision of IEEE Std 1609.4-2010), 2016.

[34] IEEE 1609 WG. Ieee standard for wireless access in vehicular environments (wave) – networking services. IEEE Std 1609.3-2016 (Revision of IEEE Std 1609.3-2010), 2016.


[38] IEEE 802.11 WG. Ieee standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007), 2012.


A Attack Summary

N2 Request Mute Attack: This attack injects a malicious learning response with the same HashedId8 value of ca1. Thus, V2 chooses to remove the matching entry with the HashedId8 value of ca1. V2 fails in sending a learning request because V2 wrongly thinks she has learned the unknown certificate but not.

Figure 11: N2: the attacker can stop V2 from sending learning requests to V1 by sending a malicious learning requests.

Assumptions. Similar to N1, we assume that V1 does not mandate the verification for incoming learning responses. Also, we assume that the attacker has enough computing power to efficiently construct a learning response that can cause partial hash collision (e.g., low-order 8 bytes collision).

Attack steps. As shown in Figure 11, V2 initializes P2PCD after receiving a trigger SPDU from V1. V2 stores the HashedId8 value of the unknown certificate ca1 in a queue. Meanwhile, since the attacker can observe the trigger SPDU, she constructs a malicious learning response, in which the HashedId8 value of the first certificate in the payload matches with the unknown certificate ca1. As defined in P2PCD, after receiving a learning response, V2 extracts all certificates in the learning response and stores them via AddCertificate. At this time, V2 wrongly thinks that it has successfully learned the unknown certificate but actually not. Thus, V2 removes the entry of the unknown certificate h8(ca1) in the queue, where h8 is a function to get the low-order eight-byte hash of the input. As the queue becomes empty, V2 decides not to attach the learning request in the next outgoing SPDU. Consequently, V2 is unable to learn the correct unknown certificate.

Discussion. Similar to N1, N2 is also caused by the use of truncated hash, and the attacker does not need to possess a legitimate certificate. In IEEE 1609.2, the issuer field in a certificate is a HashedId8 value. Therefore, on receiving the trigger SPDU, the vehicle can only store the truncated hash value in the queue. This opens a door for the partial hash collision attack. Although HashedId8 is larger than HashedId3 and makes the attacker harder to find a hash collision, a resourceful attacker (e.g., nation-states, terrorists) can always have enough computing power to efficiently find the hash collision. The attacker can even prepare these malicious learning responses...
responses in an offline way. On the other hand, due to the
possible implementation of the learning response, it is still possible
that some poorly implemented CV protocols may not verify
the incoming learning response but just store them.

**A4 Split Trigger Attack**

**Assumptions.** We assume that the leader keeps the configurations
(e.g., platoon size, members) hidden from followers [5].

**Attack steps.** A4 requires the attacker to be the leader of the
victim platoon, which is consist of V1 and V2 sequentially.
After becoming the leader, the attacker immediately sends a
\texttt{SPLIT\_REQ} to V1. At the last step of the split maneuver,
the attacker sends a \texttt{SPLIT\_DONE} to V1, which contains necessary
platoon configuration data. Notably, the attacker can control
the optimal platoon size in \texttt{SPLIT\_DONE} and sets it to 1. Since
V1, as a follower, does not store any platoon configurations,
it can only trust the attacker. However, the platoon size exceeds
the optimal platoon size; V1 thus initiates the split maneuver.
Most importantly, A4 leads to a chain reaction that will pass
the wrong configuration to the last member in the victim
platoon (i.e., V2 in this case). Moreover, V1 and V2 will not
able to merge into other platoons or accept any incoming
merge requests, because there is no available space.

**A5, A6 Merge Disruption Attack:** The attacker initiates a
merge maneuver but does not faithfully complete the whole
procedure, so the victim platoon leader V1 is trapped at the
busy state and cannot switch back to the idle state. Therefore,
V1 cannot process any incoming messages.

**Attack steps.** In A5, the attacker first sends a \texttt{MERGE\_REQ}
to V1. Since there exists available space in the victim platoon,
V1 will accept the request and send a \texttt{MERGE\_ACCEPT} to the
attacker. In the normal case, V1 will wait for a \texttt{MERGE\_DONE}
from the merge request initiator. However, the attacker
chooses not to send a \texttt{MERGE\_DONE}; thus, V1 will keep waiting.

In A6, the attacker first utilizes A2 to join the victim platoon.
If V1 initiates a merge maneuver to join a front platoon and
receives a \texttt{MERGE\_ACCEPT}, V1 will inform all the followers,
including the attacker, to change their platoon leader by
sending \texttt{CHANGE\_PL} to them. The attacker can either passively
wait for the happening of the merge maneuver or intentionally
trigger the merge maneuver of V1 by conducting the platoon
takeover attack (A1). As a malicious follower of V1, after
receiving a \texttt{CHANGE\_PL} from V1, the attacker chooses not to
reply with an \texttt{ACK}. According to the merge FSM in [5], V1 will keep sending \texttt{CHANGE\_PL} to the attacker if V1 does not receive the corresponding \texttt{ACK}.

**A8-9 Split Disruption Attack:** A8 and A9 have the same
goal and consequence as A5 and A6, but have different attack
targets. They focus on vulnerabilities of the split maneuver.

**Attack steps.** In A8 and A9, the attacker first joins the
platoon, which is consist of V1 (leader) and V2 sequentially,
by launching A2 and acts as a malicious follower. In A8, V8 sends
a \texttt{SPLIT\_REQ} to the attacker. After accepting the request,
the attacker does not respond to the following \texttt{CHANGE\_PL} sent
by V1. Therefore, V1 will not be able to switch back to the
idle state. Differently, in A9, V1 sends a \texttt{SPLIT\_REQ} to V2,
the splitting vehicle. After V2 accepting the split request and
acknowledging \texttt{CHANGE\_PL}, V1 needs to inform the follower
behind the attacker to change the platoon leader. The attacker
can remain silent, keeping both V1 and V2 at the busy state.

**A10 Follower Block Attack:** This attack is the immediate
consequence of A1 and is more powerful than A5-9, because
this attack can block all vehicles in the victim platoon rather
than one or two of them. All members in the victim platoon
will be unable to respond any incoming platoon messages.

**Attack steps.** The attacker first takes over the victim platoon.
Then, she sends \texttt{SPLIT\_REQ} to all her followers (i.e.,
V1 and V2). V1 and V2 accept the split request and reply with \texttt{SPLIT\_ACCEPT}.
Following the protocol, the attacker sends \texttt{CHANGE\_PL} to V1 and V2.
After that, the attacker can drive away or keep silence; all followers thus will never receive
\texttt{SPLIT\_DONE} from the attacker and keep sending \texttt{ACK}.

**A11 Gap Attack:** The basic idea of this idea is to prevent
the vehicle from “creating” enough space in the front of
the splitting vehicle during the leader/follower leave maneuver.

**Attack steps.** The attacker is the last follower in the victim
platoon and initiates a follower leave maneuver. V1 approves
the leave request sent by the attacker. Then, the attacker faithfully respond to \texttt{SPLIT\_REQ} and \texttt{CHANGE\_PL} from V1.
To make the attacker a free agent, V1 sends a \texttt{SPLIT\_DONE}
to the attacker. Before the completion of the leave maneuver,
V1 has to guarantee that there exists enough space at the front
of the attacker to perform lane change. If the attacker does not
send a \texttt{GAP\_CREATED}, V1 will keep busy as it wrongly
thinks the leave maneuver is still on-going.

**A12, A13 Leave Disruption Attack:** A12 and A13 exploit
timers in the leader leave maneuver and the follower leave
maneuver respectively.

**Attack steps.** In A12, when the leader wants to leave the
platoon, its followers have to elect a new leader. The elected
leader then sends a \texttt{ELECTED\_LEADER} to the old leader who
then hands over the leadership to the elected leader, by initiating
the leader leave maneuver and safely leave the platoon.
However, if the attacker is one of the followers (A2) and becomes
the elected leader, she can choose not to respond. As
well, A10 can be used to mislead all followers to a busy state
in advance, so no followers can send \texttt{ELECTED\_LEADER} to the
leader, blocking the leader leave maneuver.

In A13, a follower wants to leave the platoon and sends a
\texttt{LEAVE\_REQ} to the leader; if no response is received from
the leader, the follower is unable to finish the follower leave
maneuver. The attacker can place herself at the position of
the leader through A1, and keep silent. On the other hand,
the attacker can utilize A5-9 to prevent the benign leader
from communicating with other followers. Thus, the victim
follower cannot finish the follower leave maneuver.

**A14 Dissolve Disruption Attack:** To make a follower unavailable, the attacker can either use A10 to block all followers
or join the victim platoon as a silent follower through A2.