ORIGINAL RESEARCH
published: 16 June 2022
doi: 10.3389/fenrg.2022.928919

An Ontology-Based Dynamic Attack Graph Generation Approach for the Internet of Vehicles

Shuning Hou[1], *Xiuzhen Chen*[2]*, *Jin Ma*[1], *Zhihong Zhou*[1] and *Haiyang Yu*[1]

[1]*Institute of Cyber Science and Technology, Shanghai Jiao Tong University, Shanghai, China,* [2]*Shanghai Municipal Key Lab of Integrated Management Technology for Information Security, Shanghai, China*

With the development of automobile intelligence, the security of the Internet of Vehicles has become a key factor that affects the development of intelligent vehicles. However, existing security risk analysis methods for the IoV either focus only on certain levels, such as the component level, or perform only a static analysis. This paper proposes a dynamic attack graph generation method for the IoV to identify and visually display the security risks caused by the associated vulnerabilities in an IoV system. First, using the actual architecture of the IoV, this paper shows how to model the security elements and their relationships in the IoV system and proposes a network security ontology model for this system. Second, it shows how to construct a reasoning rule base according to the causal relationship between the vulnerabilities using the Semantic Web Rule Language Finally, in view of the rapid change in the network topology of the IoV, a dynamic attack graph generation algorithm based on an ontology reasoning engine is proposed, which can effectively reduce the overhead caused by the changes in the attack graph. The effectiveness of the algorithm is demonstrated through an actual security event scenario and a constructed scenario. The experimental results show that the algorithm can dynamically and accurately display the network attack graph of the IoV. The proposed method is helpful in globally analyzing the threat caused by the combined exploitation of the vulnerabilities in an IoV system and risk management.

Keywords: internet of vehicles, dynamic attack graph, ontology, vulnerability, security risk

OPEN ACCESS

Edited by:
Linfeng Yang,
Guangxi University, China

Reviewed by:
Chen Zhang,
University of Shanghai for Science and Technology, China
Wei Lan,
Guangxi University, China

****Correspondence:***
Xiuzhen Chen
chenxz@sjtu.edu.cn

Specialty section:
This article was submitted to Smart Grids, a section of the journal Frontiers in Energy Research

Received: *26 April 2022*
Accepted: *16 May 2022*
Published: *16 June 2022*

Citation:
Hou S, Chen X, Ma J, Zhou Z and Yu H (2022) An Ontology-Based Dynamic Attack Graph Generation Approach for the Internet of Vehicles. Front. Energy Res. 10:928919. doi: 10.3389/fenrg.2022.928919

1 INTRODUCTION

Owing to the development of modern automobiles, people's daily travel is becoming increasingly more convenient and comfortable. In addition to efficiency and convenience, the rapid development of the technology of the Internet of Vehicles (IoV) has brought about a series of potential security threats, such as cloud supply chain issues, private data security, protocol cracking, and illegal data injection. For attackers, the attack surface of the IoV has become more extensive, and it is no longer limited to near-range physical attacks.

The 2020 Global Automotive Network Security Report (Upstream Security, 2020) released by Upstream Security reported an investigation of the security incidents happening in the automotive field since 2010 and concluded that the most common attack vectors use different entry points such as servers, mobile applications, and on-board diagnostic (OBD) ports. In addition, IOActive (Thuen, 2016), an Israeli automotive network information security company, scored the collected vehicle-related vulnerabilities on a one to five scale and found that approximately 72% of the vehicle-related

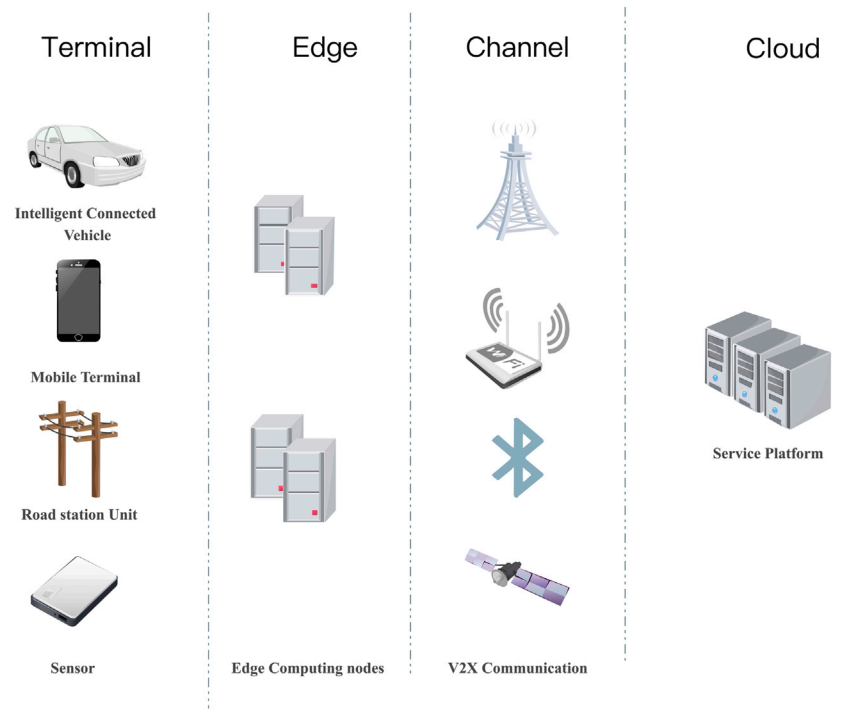

FIGURE 1 | The "cloud-channel-edge-terminal" architecture of the IoV.

vulnerabilities belonged to the medium- and low-risk categories. IOActive also pointed out that this did not necessarily mean that there was no significant risk. These single vulnerabilities may not be harmful; however, when multiple vulnerabilities are exploited in combination, the attack consequences can be immeasurable.

Taking the Tesla attack chain disclosed by Tencent Keen Lab as an example (Keen, Security, 2022), the researchers first considered the wireless network as the starting point of the attack. The vulnerability of the vehicle browser was used to execute arbitrary code in the browser. Furthermore, the kernel privilege escalation vulnerability was exploited to obtain the root privilege to extract information about the vehicle system. The researchers then bypassed the integrity check mechanism through the electronic control unit (ECU), modified and refreshed the firmware, and finally broke through the gateway to realize the transmission of any controller area network (CAN) message on the CAN bus. This is a complex attack path that exploits multiple vulnerabilities on various attack surfaces. The defense of an IoV system against an attack is insufficient if it only analyzes the vulnerabilities from a single level. It is necessary to comprehensively consider and analyze various attack surfaces to discover potential attack paths leading to the target under the "cloud-channel-edge-terminal" architecture of the IoV, as shown in **Figure 1**. Therefore, this study considered attack graph technology as an effective way to analyze the potential attack paths in an IoV system.

Attack graph technology (Lallie et al., 2020) can show potential attack paths and attack consequences through vertex and directed edge structures from the perspective of attackers in combination with specific network structure information. Most of the studies on attack graph are oriented toward ordinary enterprise network information systems. There are few studies on the attack graph of the IoV, a special information system, and there is also a lack of unified and standardized expression of IoV security knowledge. This paper proposes a dynamic attack graph generation approach based on ontology (McGuinness and Van Harmelen, 2004) for the IoV. A security ontology of the IoV is established, formulates a unified and standardized expression for security knowledge, which including the network topology, vulnerabilities and their relationships, and other security elements in the IoV, the proposed method can mine all potential attack paths in advance using the ontology inference engine HermiT (Glimm et al., 2014). The main contributions of this study are as follows:

1. This paper proposes an IoV network security ontology. The ontology models various security elements and their relationships in the IoV system, and then formulates a unified and standardized expression for IoV security knowledge.
2. This paper builds a knowledge base of the reasoning rules of the IoV. This paper analyzes different types of security vulnerabilities and corresponding attack methods, such as long-distance wireless attacks, short-range wireless attacks, and physical contact attacks, and describes specific attacks by Semantic Web Rule Language (SWRL) (Horrocks et al., 2004) rules.

3. This paper proposes a dynamic attack graph generation algorithm. The algorithm can update incrementally according to the change in network topology, which is more suitable for the IoV characterized by rapid changes in network topology. This can effectively demonstrate the vulnerability of the global IoV network and help with risk management and has lower computational complexity while updating attack graph.

The rest of this paper is organized as follows. **Section 2** discusses the existing related works. **Section 3** presents the components of the proposed model. **Section 4** shows how to construct different IoV network attack scenarios to verify the effectiveness of the algorithm. Finally, **Section 5** concludes the paper.

2 RELATED WORKS

The security threats of the IoV will not only cause economic losses to individuals and enterprises but also endanger personal safety and even national public security in severe cases. Therefore, several studies have been conducted to improve the security performance of the IoV. Existing studies on the security vulnerability analysis of the IoV are mainly divided into three levels: platform, network, and component levels (Li, 2019). They not only discover hidden vulnerabilities through penetration testing but also evaluate the risk globally using attack graphs and other methods such as attack trees and matrices. Detailed information on related works at each level is given as follows.

2.1 The Network Level

Studies on security at the network level mainly focus on the network communication security of the IoV, including identity authentication and privacy disclosure. Researchers have proposed group signature schemes (Shao et al., 2016), batch authentication schemes (Sutrala et al., 2020), and lightweight anonymous authentication schemes (Sadri and Rajabzadeh Asaar, 2020) to solve these problems.

2.2 The Platform Level

Studies on security at the platform level mainly aim at the security of the vehicle CAN bus and in-vehicle sensor network, which distinguishes normal behaviors from attack behaviors by extracting the vehicle characteristics (electrical, physical, and data packet characteristics) or by using a deep learning algorithm in the normal state (Choi et al., 2018; Song et al., 2020).

2.3 The Component Level

Studies on security at the component level focus on various vulnerabilities in the IoV system, analyze vulnerabilities and security risks based on the penetration testing results, and propose security protection suggestions or measures for the IoV network. Through the results of laboratory simulation experiments and actual road tests, Koscher et al. (Koscher et al., 2010) proved that attackers who gain control of key ECUs can evade the internal security features of intelligent networked vehicles and conduct malicious operations on them. In 2015, (Miller and Valasek, 2015), successfully hacked an in-vehicle communication system remotely by exploiting the vulnerability of the Uconnect system port in the Jeep vehicle. The final experimental results proved that attackers could exploit this vulnerability to control car braking and steering. In addition, they (Miller and Valasek, 2014) demonstrated the long-range attack surface of intelligent connected cars and analyzed the security of different manufacturers' car networks.

In general, the above studies on the security risks of the IoV mainly focus on the detection and mining of single vulnerabilities through simulation experiments, road experiments, and the use of fuzzing and feature extraction technologies, without considering the possibility of escalating the risks from the combined vulnerabilities at the same level or multiple levels.

In recent years, some researchers have looked into the application of graph technology in the field of IoV security. In the SAE J3601 "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems" (SAE J3061 Vehicle Cybersecurity Systems Engineering Committee, 2016) launched by the American Society of Automotive Engineers in 2016, the attack tree technology is proposed to model the risk of the vehicle system network and quantify the system risk from the perspective of a threat. Kong et al. (Kong et al., 2018) proposed a security risk assessment framework for smart cars, to identify and assess security risk. The framework is based on the Guidelines for the Management of IT Security, using attack tree to analyze and categorize the assets, threats, and vulnerabilities. However, due to characteristics of attack tree, an attack tree can only analyze one attacker purpose. when faced with multi-target attackers, attack tree analysis is more complex than attack graph.

Salfer and Eckert, (2018) proposed an automatic generation model, called security analyzer for exploitability risks (SAlfER), which can semi-automatically quantify the risk of a given attacker by his/her exploitation steps with both budget and cost with cycles. In addition, they proposed an algorithm for a random attack graph for the security evaluation of a vehicle network, which creates a path for each starting node, continuously expands, clones, completes the path, and writes the path with similar dependencies in the attack graph to generate the result. The model mainly aims at the design stage before the vehicle goes into production, refers to the vehicle development documents, comprehensively considers various possibilities, calculates the security risk, and makes key business decisions in terms of safety and sustainability. Ibrahim et al. (Ibrahim et al., 2020) used the Architecture Analysis and Design Language (AADL) to assess the risk of vehicle system security cases. The vehicle system design, connection, weakness, resources, potential attack examples, and their pre-conditions and post-conditions were modeled using the AADL. The generated final attack diagram is displayed graphically to help the system administrator select the best countermeasures. The above two references demonstrate the feasibility of applying attack graph technology to security studies on the IoV. However, they either focused only on in-vehicle systems or performed only a static analysis using vehicle development documents.

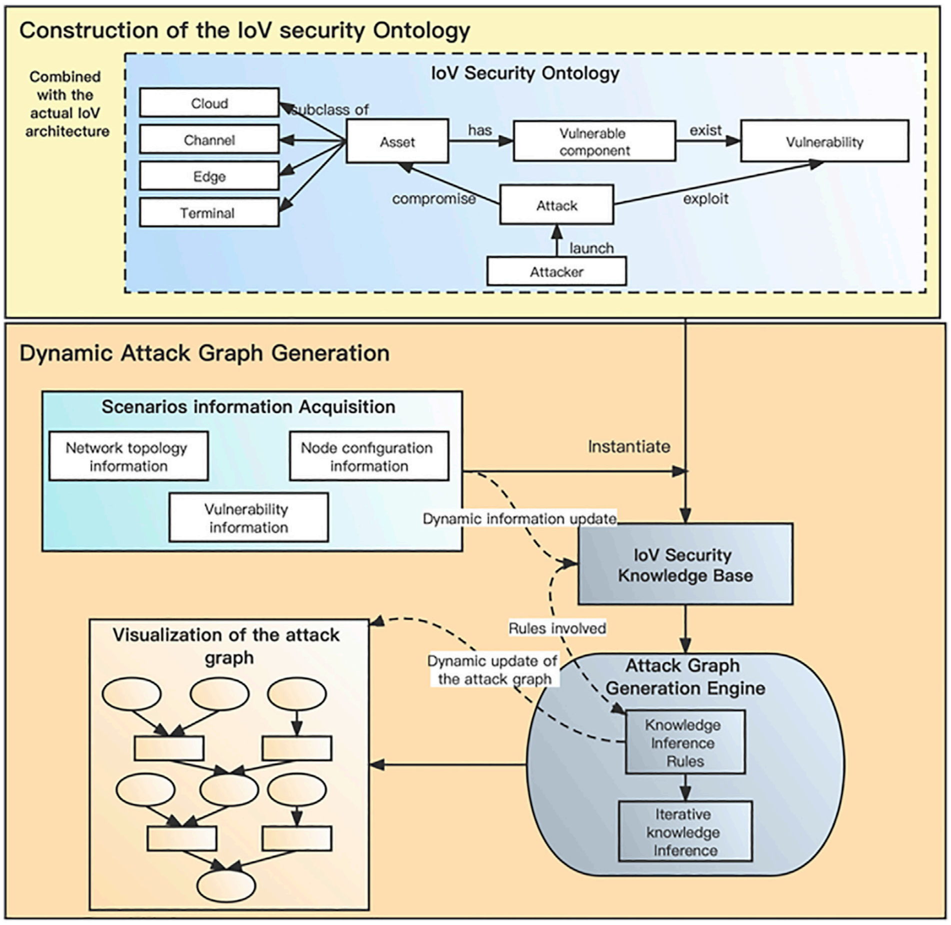
Construction of the IoV security Ontology
Combined with the actual IoV architecture
IoV Security Ontology
Cloud
Channel
Edge
Terminal
subclass of
Asset
has
Vulnerable component
exist
Vulnerability
compromise
Attack
exploit
launch
Attacker
Dynamic Attack Graph Generation
Scenarios information Acquisition
Network topology information
Node configuration information
Vulnerability information
Instantiate
Dynamic information update
IoV Security Knowledge Base
Rules involved
Visualization of the attack graph
Dynamic update of the attack graph
Attack Graph Generation Engine
Knowledge Inference Rules
Iterative knowledge Inference

FIGURE 2 | IoV attack graph generation model.

In this paper, ontology and attack graphs were combined and applied to the field of IoV security to formulate a unified and standardized expression of IoV security knowledge. In view of the current situation of multistep attacks using the combined vulnerabilities in the IoV, this study integrates the overall architecture of the cloud-channel-edge-terminal to generate a comprehensive attack graph, which is conducive to the comprehensive risk analysis of IoV systems and has far-reaching significance for the construction of active defense systems for the IoV.

3 ONTOLOGY-BASED ATTACK GRAPH GENERATION ALGORITHM

Owing to the lack of a unified and standardized expression for security knowledge in existing studies on attack graphs of the IoV, this paper proposes a security ontology for the IoV system and formalizes a normative definition of the security elements in this system that can better describe its network architecture and vehicle-related security vulnerabilities. The IoV attack graph generation model proposed in this paper is shown in **Figure 2**. It includes two modules: construction of an IoV security ontology and generation of a dynamic attack graph. The former models the security elements and their relationships with the IoV system. The latter instantiates the entities to construct a knowledge base, inputs them into the inference engine HermiT, and finally generates a complete attack graph through the graph generation engine. When the scenario information changes, the inference engine locates the corresponding rules and nodes to update the attack graph in a timely manner.

3.1 Construction of the IoV Security Ontology

Ontology (Guarino et al., 2009) is a method and theory used to describe the essence of things, which refers to the formal specification of shared concepts in the same field. Research on building a network security ontology model (Iannacone et al., 2015) can better describe the intelligence and information related to attacks. This paper proposes an IoV security ontology by analyzing the cloud-channel-edge-terminal architecture of the IoV and abstracting the related security elements and their relationships, including the assets, vulnerabilities, and attacks.

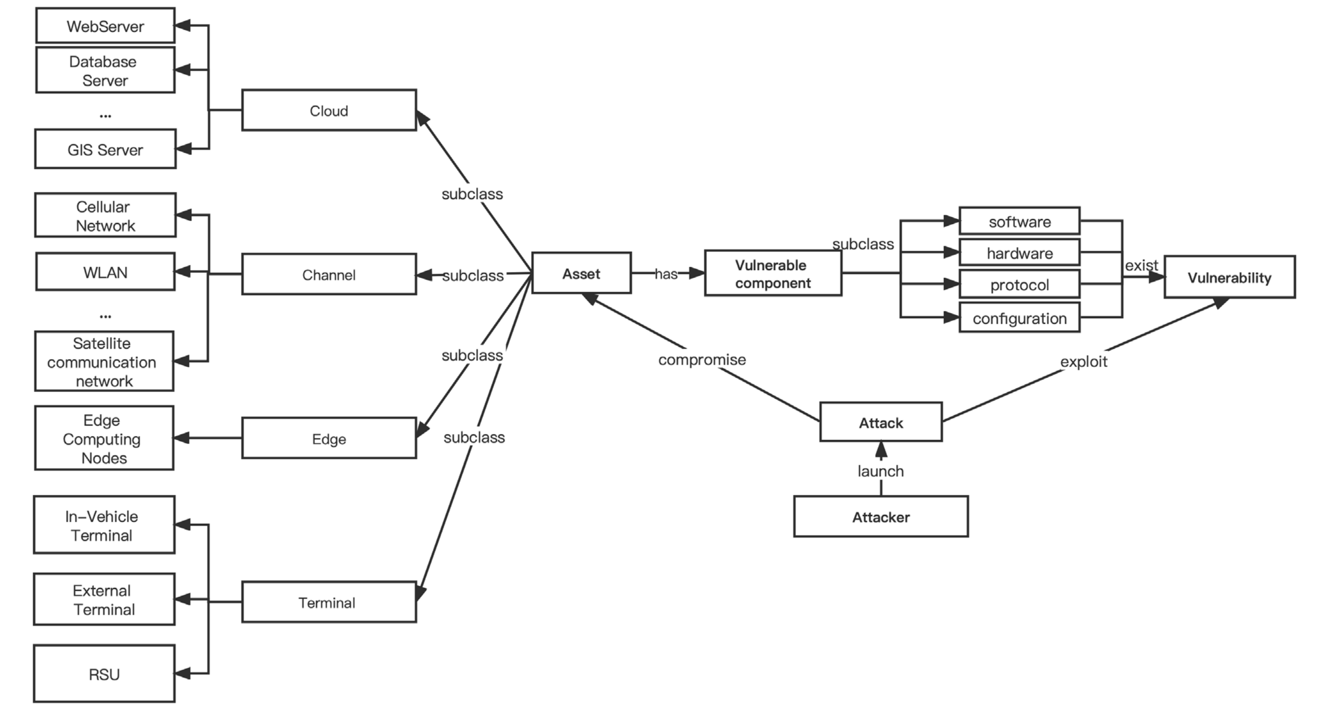

FIGURE 3 | IoV security ontology.

3.1.1 Definition of the IoV Security Ontology

This study combines the actual architecture of the IoV with existing security ontologies to build a specific security ontology for the IoV, as shown in **Figure 3**, which mainly includes five types of entities: assets, vulnerable components, vulnerabilities, attacks, and attackers. The entity classes are defined as follows:

Definition 1: The asset class. This class includes the equipment and sensitive data at all levels of the IoV system, i.e., a four-layer architecture of the cloud-channel-edge-terminal. The cloud layer corresponds to the application layer of the IoV, and the main equipment consists of various data collection and processing and cloud service support servers, including database, web, and travel navigation data processing servers. The channel layer corresponds to the network layer, including the cellular network 2G/3G/4G, WLAN, and the satellite communication network. The edge layer includes many edge computing nodes, which generally refer to the roadside equipment. The terminal layer corresponds to the perception layer, which is responsible for collecting vehicle location information and traffic information around the vehicles and for perceiving the environment and its state during driving. It primarily includes RFID readers and various communication terminals, such as vehicle terminals, mobile applications, and sensors. In addition, because ICVs have more than one attack surface, the asset entities at the terminal layer are further subdivided into in-vehicle terminals, external terminals, etc.

Definition 2: The vulnerable component class. This class shows the location of the vulnerability in the attack target assets. It is divided into subclasses such as software, hardware, programs, and services. These are all common types of components in the network assets of the IoV.

Definition 3: The vulnerability class. This class shows the technical drawbacks that can be exploited by attackers to launch attacks. Vulnerabilities are divided into software, hardware, and protocol vulnerabilities, according to the location where the vulnerability is hosted.

Definition 4: The attack class. The main subclasses include long-range wireless, short-range wireless, and physical access attacks. Among them, the long-distance wireless attacks are mainly carried out through Wi-Fi, cellular networks (3/4/5G), cloud platforms, GPS, etc. The short-range wireless attacks are mainly carried out through Bluetooth, keyless entrance systems, dedicated short-range communications, etc. The physical access attacks are performed through actual physical contact by the attackers.

Definition 5: The attacker class. According to the identity of the attacker, the attacker class is divided into internal and external attackers.

3.1.2 Relationship Model Between Entities

The relationship between entity classes is reflected by the entity properties, which are divided into datatype and object properties (McGuinness and Van Harmelen, 2004). Datatype properties are mainly properties of a single individual class, whereas object properties are properties that define the relationships between different individual classes.

- Datatype properties

The datatype properties of the asset class entities include the asset name, asset level, asset importance level, and asset motion status, which are only limited to vehicle entities. The value of the motion status is either still or moving.

The datatype properties of the vulnerable component class entities include the component name, component version, component function, and component-related information importance level.

The datatype properties of the vulnerability entities include the vulnerability ID, vulnerability description information, functions affected by the vulnerabilities, Common Vulnerability Scoring System score, patch information, and utilization probability.

The datatype properties of the attack entities include their Common Attack Pattern Enumeration and Classification ID, attack preconditions, and attack postconditions, i.e., attack benefits. Attack preconditions are prerequisite permissions required to implement this type of attack and are mainly composed of two parts: specific permissions and location. That is, the preconditions describe the specific type of software/hardware the attack needs to obtain the corresponding permissions. The attack postconditions are similar to the attack preconditions. The postconditions describe the expected benefits after the attack is launched and consist of specific permissions and locations.

The datatype properties of the attacker class include the attacker's location (distinguishing between long-distance and close-range attackers), capabilities, and permissions.

- Object properties

According to the relationship between the entity classes above, the object property defined in the IoV security ontology is as follows:

access property: This is a symmetric property, which indicates that both instances can access each other.

$$access\,(Asset, Asset)$$

compromise property: This property indicates that an attacker can successfully compromise an asset instance.

$$compromise\,(Attack, Asset)$$

hasComponent property: This property indicates that an asset instance has a containment relationship with a component instance.

$$hasComponent\,(Asset, Component)$$

exist property: This property represents the existence of a vulnerability in a component instance.

$$exist\,(Component, Vulnerability)$$

exploit property: This property indicates that an attacker needs to exploit the vulnerability of an instance to an attack.

$$exploit\,(Attack, Vulnerability)$$

launch property: This property indicates that an attacker needs to use a certain type of attack to launch an attack behavior.

$$launch\,(Attacker, Attack)$$

3.2 Attack Graph Generation Method for the IoV

Attack graph generation is divided into two stages: the initial attack graph generation and the dynamic attack graph update. The generation of attack graphs is based on the IoV security knowledge base, which is constructed by the SWRL rules of the security elements in IoV attack scenarios. Through inference engine HermiT, initial attack graph generates. When the network topology, vulnerability information, and other data change dynamically, the inference engine locates the corresponding rules and nodes to update the attack graph in a timely manner.

3.2.1 Inference Rules Knowledge Base

The SWRL (Horrocks et al., 2004) is a language that presents rules in a semantic manner and can be used in attack graph generation algorithms to infer the process of exploiting vulnerabilities to invade assets. An SWRL rule includes the body (inference precondition) and the head (inference result). The body part points out all the preconditions required for inference, including specific instances and the relationships between instances. And the head provides the inference results that can be obtained under the rule.?in rules represents this station is a variable. Entities can substate them.

The reasoning rules in this paper are divided into three types according to their different functions:

1. Vulnerability Existence Inference Rules

Vulnerability existence inference rules can infer whether an asset has a vulnerability based on its category and current version. An example of a vulnerability existence inference rule is provided below.

$$\begin{aligned}&Component\,(?comp)\hat{}hasversion\,(?comp, ?x)\\&\hat{}Vulnerability\,(?vul)\\&\hat{}hasupdateversion\,(?vul, ?y)\\&\hat{}swrlb\colon lessThan\,(?x, ?y)\\&\rightarrow exist\,(?comp, ?vul)\end{aligned}$$

2. Vulnerability Exploitability Inference Rules

Vulnerability exploitability inference rules can construct a single-step attack path and determine whether the attacker's resources and attack capabilities in the current state can attack the specified assets. An example of a vulnerability exploitability inference rule is provided below.

$$\begin{aligned}&Asset(?asset)\hat{}Component(?comp)\\&\hat{}Vulnerability(?vul)\hat{}Attacker(?attacker)\\&\hat{}Attack(?attack)\\&\hat{}hasComponent(?asset,?comp)\\&\hat{}exist(?comp,?vul)\\&\hat{}exploit(?attack,?vul)\\&\hat{}launch(?attacker,?attack)\\&\hat{}assess(?asset,?attacker)\\&\rightarrow compromise(?attacker,?asset)\\&\hat{}attackbenefit(?attacker,?asset)\end{aligned}$$

This rule means that when an asset instance has a vulnerability group and there is a vulnerable instance in the vulnerable component, there is a certain attack method that can exploit the vulnerability. If the attacker knows how to use this attack method and can successfully access the asset, it can be inferred that the asset can be compromised by the attacker and attacker gains the attack benefits.

Take attacker launches the command injection attack by exploit the vulnerability V1 in CAN bus to vehicle as an example, the corresponding rules rule are as follows:

$$\begin{aligned}&Asset(vehicle)\hat{}Component(CANbus)\\&\hat{}Vulnerability(V1)\hat{}Attacker(attacker)\\&\hat{}Attack(commandInjection)\\&\hat{}hasComponent(vehicle,CANbus)\\&\hat{}exist(CANbus,V1)\\&\hat{}exploit(commandInjection,V1)\\&\hat{}launch(attacker,commandInjection)\\&\hat{}assess(vehicle,attacker)\\&\rightarrow compromise(attacker,CANbus)\\&\hat{}commandInjection(attacker,\ CANbus)\end{aligned}$$

The rule means if the permissions obtained by the attacker meet all the preconditions above, the attacker can launch the command injection attack successfully and compromise the CAN bus on this vehicle.

3. Network Connectivity Inference Rules

The network connectivity inference rules can infer the attacker's access privileges according to the reachability between the resources and the assets compromised by the attacker. An example of a network connectivity inference rule is presented below.

$$\begin{aligned}&compromise(?asset1,?attacker)\\&\hat{}assess(?asset1,?asset2)\\&\rightarrow assess(?attacker,?asset2)\end{aligned}$$

This rule means that when the asset of instance 1 is compromised by an attacker instance and instance 1 can assess the asset of instance 2, it can be inferred that the attacker can access the asset of instance 2.

3.2.2 Initial Attack Graph Generation Algorithm

In the initial state, all the security elements in IoV attack scenarios and their relationships are instantiated. The inference engine then mines the causal relationships among the scattered vulnerabilities according to the inference rules, determines the potential attack paths of the attacker, and finally synthesizes all possible attack paths to generate a complete attack graph for the output.

Definition 6: Assume an attack graph *G=<C, V, E>*, where *C* represents the condition set (including all initial conditions, preconditions, and postconditions), *V* represents the set of vulnerabilities, and *E* represents the edge set.

The initial attack graph generation algorithm is mainly based on the breadth-first traversal (BFS) algorithm. The algorithm takes the security ontology instance of the target network, the attack scenario, and the attacker's target (optional) as inputs and generates an attack graph G in the form of a tuple. The algorithm adopts the method of forward chaining, starting from the initial conditions, based on the BFS to obtain additional properties by continuously searching for vulnerable hosts in the network. To generate an attack graph, line two to five in theAlgorithm 1 first obtains the initial object properties of the attacker CA and assets A through function Properties (). The second step (line 7–8) is to call inference engine by function onto. reasoner (), and obtain inferential facts, then locate the specific inference rule set F. The third step is to find the corresponding vulnerability nodes and pre-post-condition nodes for each rule of F and to construct the node set and edge set of the attack graph (line 12–25). The resources owned by the attacker are then updated until the inference ruleset is traversed (line 26). Then, the second step is repeated until the attacker's goal is reached and the algorithm terminates (loop starts on line 6). If the attacker does not specify a specific goal, it is assumed that the attacker's goal is to obtain all the resources that can be compromised in the system as much as possible, and the algorithm terminates when no new inference facts are generated.

3.2.3 Dynamic Attack Graph Update Algorithm

Owing to the rapid movement of vehicle nodes, the network topology update frequency is higher than that of traditional enterprise information systems. Therefore, this paper summarizes three situations that can cause changes in the IoV attack graph. These three situations are described below.

Network topology changes: Vehicles leaving/entering the communication area will cause network topology changes. This type of situation causes a change in the vehicle terminal node Nv. When a vehicle leaves or enters the communication area, the node should be deleted or added.

Network connectivity changes: Changes in firewall rules or when a moving vehicle communicates with different roadside base stations/cloud servers can cause changes in network connectivity. Changes in network connectivity are mapped to changes in attack scenarios, which are changes in object properties between asset nodes, and the properties of related nodes should be added or deleted.

Node information changes: The change in the motion state of a vehicle and in the node-related vulnerability information belongs to the change in the node information. When a vehicle is stationary, physical contact is possible; therefore, physical contact attacks are possible. When the motion state of a

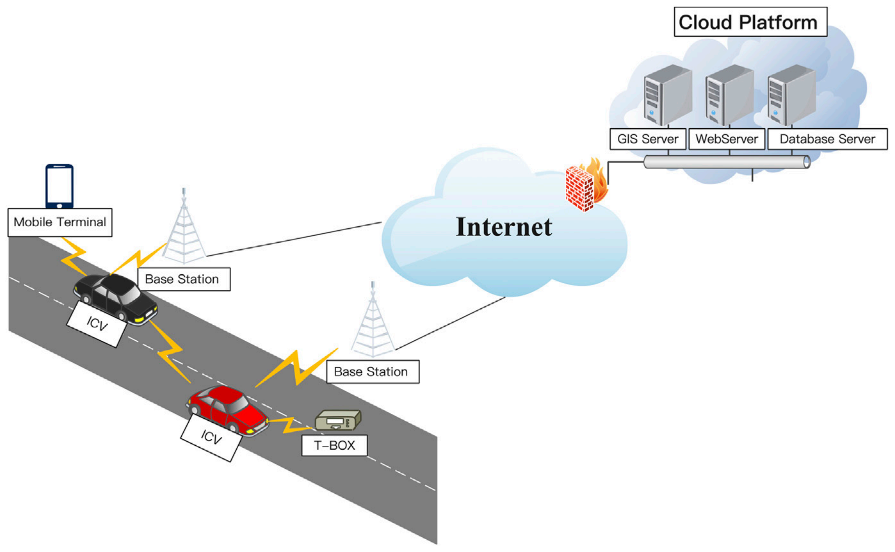

FIGURE 4 | Typical attack scenario.

vehicle changes, the connectivity relationship between the vulnerable nodes is related to physical contact changes, and the related security properties of the node should be added or deleted.

Algorithm 1 Attack graph generation algorithm

```
Input: $IoVOnto$ , $attackScene$ ,$Goal$
output: $G$
1:  $Instantiate(IoVOnto, attackScene)$
2:  $A, P, CA \leftarrow \emptyset$
3:  $A \leftarrow onto.asset()$
4:  $P \leftarrow onto.Properties(A)$
5:  $CA \leftarrow onto.Properties(attacker)$
6:  while $Goal$ not in $CA$ do
7:      $onto.reasoner()$
8:      $F \leftarrow reasoner.inferredFacts()$
9:      if $F == \emptyset$ then
10:         break
11:     else
12:         $FP \leftarrow onto.Properties(F.attack)$
13:         $P \leftarrow onto.Properties(F)$
14:         for $fp \in FP$ do
15:             $r \leftarrow getRuleOfAttack(fp)$
16:             $v_i \leftarrow r.vul()$
17:             $s_{pre} \leftarrow body(r)$
18:             $s_{post} \leftarrow head(r)$
19:             if $s_{pre} \in P$ then
20:                 $c_i \leftarrow s_{pre}$
21:                 $C \leftarrow C \cup c_i \cup c_{i+1}$
22:                 $V \leftarrow V \cup v_i$
23:                 $E \leftarrow E \cup (c_i, v_i) \cup (v_i, c_{i+1})$
24:             end if
25:         end for
26:         $CA \leftarrow onto.Properties(attacker)$
27:     end if
28: end while
29: checkLoop(C,V,E)
30: $G \leftarrow < C, V, E >$
```

Algorithm 2 Algorithm of Adding Security Properties

```
Input: $IoVOnto$,$changes$,$Goal$,$G$
output: $G'$
1:  $CA \leftarrow onto.Properties(attacker)$
2:  $p_{add} \leftarrow \emptyset$
3:  $n_{add} \leftarrow \emptyset$
4:  for $change \in changes$ do
5:      if $isvul(change)$ then
6:          $V \leftarrow V \cup change$
7:          $IoVOnto.add(change)$
8:          $p_{add} \leftarrow p_{add} \cup onto.Properties(change)$
9:      else
10:         $p_{add} \leftarrow p_{add} \cup change$
11:     end if
12: end for
13: $IoVOnto.add(p_{add})$
14: $line\ 6-30\ in\ algorithm1$
```

Algorithm of Adding Security Properties: When a node or security properties are added, first, the corresponding node should be instantiated to build the datatype properties associated with Nv and the object properties with the newly added vulnerability node Nv (line 1–13 in Algorithm 2). Then, the algorithm calls the inference engine to determine the new inference facts, finds the reasoning rules related to the reasoning fact, adds nodes and edges to the attack graph, and adds the attack consequences to the scene for further reasoning until no new reasoning facts are generated, which is the same diagram in Algorithm 1.

Algorithm of Deleting Security Properties: In the case of deleting a node or security properties, the algorithm first obtains all the properties of Nv, finds all sets of the vulnerable nodes V that Nv is related to and the condition nodes related to Nv, deletes all vulnerable nodes in V and the edges related to the vulnerable nodes in the attack graph, and deletes the related condition nodes and the edges connected to them at the same time.

Algorithm 3 Algorithm of Deleting Security Properties

```
Input: $IoVOnto$ ,$changes$,$Goal$ ,$G$
output: $G'$
1:  $CA \leftarrow onto.Properties(attacker)$
2:  $p_{delete} \leftarrow \emptyset$
3:  $n_{delete} \leftarrow \emptyset$
4:  for $change \in changes$ do
5:      if $isvul(change)$ then
6:          $V \leftarrow V \backslash change$
7:          $p_{delete} \leftarrow p_{delete} \cup onto.Properties(change)$
8:      else
9:          $p_{delete} \leftarrow p_{delete} \cup change$
10:     end if
11: end for
12: $delete \leftarrow p_{delete}$
13: $R \leftarrow getSWRLRule(IoVOnto)$
14: while $delete! = \emptyset$ do
15:     for $n \in delete$ do
16:         for $r \in R$ do
17:             if $n \in body(r)$ then
18:                 $c_i \leftarrow body(r)$
19:                 $v_i \leftarrow r.vul()$
20:                 $C \leftarrow C \backslash c_i$
21:                 $E \leftarrow E \backslash (v_i, c_i)$
22:                 $c_{i+1} \leftarrow head(r)$
23:                 $C \leftarrow C \backslash c_{i+1}$
24:                 $E \leftarrow E \backslash (v_i, c_{i+1})$
25:                 $temp \leftarrow temp \cup head(r)$
26:             end if
27:         end for
28:     end for
29:     $delete \leftarrow temp$
30: end while
31: $G' \leftarrow < C, V, E >$
```

4 EXPERIMENT RESULTS AND ANALYSIS

4.1 Experiment on a Typical Attack Scenario

4.1.1 Construction of a Typical Attack Scenario

According to the White Paper on Security Penetration of Intelligent and Connected Vehicles (China Software Tesing Center, 2020), the types of attacks in typical attack scenarios can be divided into three categories according to different attack surfaces: long-distance wireless attacks, short-range wireless attacks, and physical contact attacks. Therefore, when constructing the attack scenario, considering these three attack methods, we selected ten vulnerabilities related to the IoV system and constructed the attack scenario shown in **Figure 4**. Among them, V1 and V2 are vulnerabilities in cloud servers, which are mainly related to long-distance wireless attacks, and the consequences include sensitive information collection and privilege escalation. V3 is a low-version browser vulnerability on in-Vehicle infotainment (IVI) system. V4 is an unverified vulnerability in a Wi-Fi connection, which is related to short-range wireless attacks. A short-range attacker can gain access to a vehicle from this vulnerability. V10 is a physical contact vulnerability exposed outside of the vehicle. An attacker can access the internal bus of a vehicle through physical contact with the OBD interface and implement command injection attacks on the vehicle. The vulnerability information is presented in **Table 1**.

4.1.2 Attack Graph Generation and Analysis

The attack graph generated by the experiment at T_0 is shown in **Figure 5**. There are four attack paths in total.

Path 1:Through the cloud vulnerability V1, attacker can launch attack $Info\_Collection(attacker, Vehicle1)$, and collect private information of vehicle 1. Based on that, attacker conducts social engineering attack $exploit(SocialEngineeringAttack, CVE-2015-5065)$, and successfully gains the account information of vehicle 1 $hasAccount(attacker, Vehicle1)$. The attacker obtains the invading interface and local user privilege of invading vehicle 1. Finally, by exploiting vulnerability V7 on ECU Gateway, attackers can finally achieve arbitrary code execution attacks $execArbitraCode(attacker, Vehicle1)$. Vehicle 1 will receive arbitrary codes that cause denial of service attacks or other malicious operations possibly.

Path 2 is a short-range wireless attack. Owing to the lower version of the installed car browser, a permission vulnerability V3 in IVI was exploited by attacker $gainUserPrivi(attacker, Vehicle1.IVI)$. The attacker obtains the local user privilege of vehicle 1. Then attacker discovers a shell vulnerability on the system, escalates the shell privilege to obtain the root permission of the IVI system

TABLE 1 | Vulnerability information.

Vulnerability	CVE ID	Vulnerability Component	Attack Consequence
V1	CVE-2015-1761	SQLServer	Information Collection
V2	CVE-2013-5065	WebServer	Privilege Escalation
V3	CVE-2009-1725	In-vehicle Browser	Permission Acquisition
V4	CVE-2018-11476	Wi-Fi	Unauthenticated Access
V5	CVE-2018-11477	Wi-Fi	Information Collection
V6	CVE-2018-11478	OBD Dongle	Command Injection
V7	CVE-2013-6282	ARM Linux	Privilege Escalation
V8	CVE-2016-9337	ECU Gateway	Arbitrary Code Execution
V9	CVE-2018-9311	T-Box	Initial Access
V10	CVE-2018-9322	OBD Interface	Verify Bypass

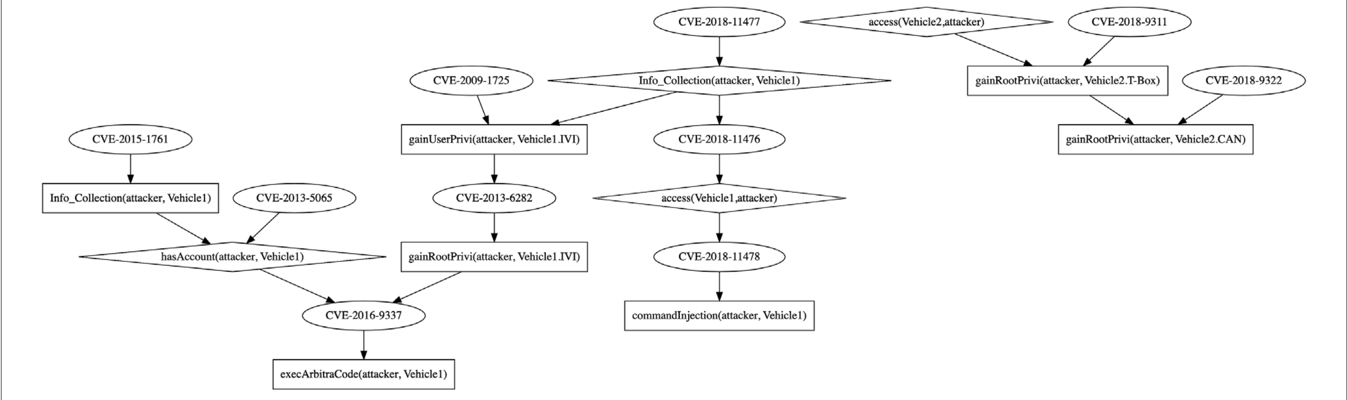
CVE-2018-11477
access(Vehicle2,attacker)
CVE-2018-9311
CVE-2009-1725
Info_Collection(attacker, Vehicle1)
gainRootPrivi(attacker, Vehicle2.T-Box)
CVE-2018-9322
CVE-2015-1761
gainUserPrivi(attacker, Vehicle1.IVI)
CVE-2018-11476
gainRootPrivi(attacker, Vehicle2.CAN)
Info_Collection(attacker, Vehicle1)
CVE-2013-5065
CVE-2013-6282
access(Vehicle1,attacker)
hasAccount(attacker, Vehicle1)
gainRootPrivi(attacker, Vehicle1.IVI)
CVE-2018-11478
CVE-2016-9337
commandInjection(attacker, Vehicle1)
execArbitraCode(attacker, Vehicle1)

FIGURE 5 | Attack graph at time T_0.

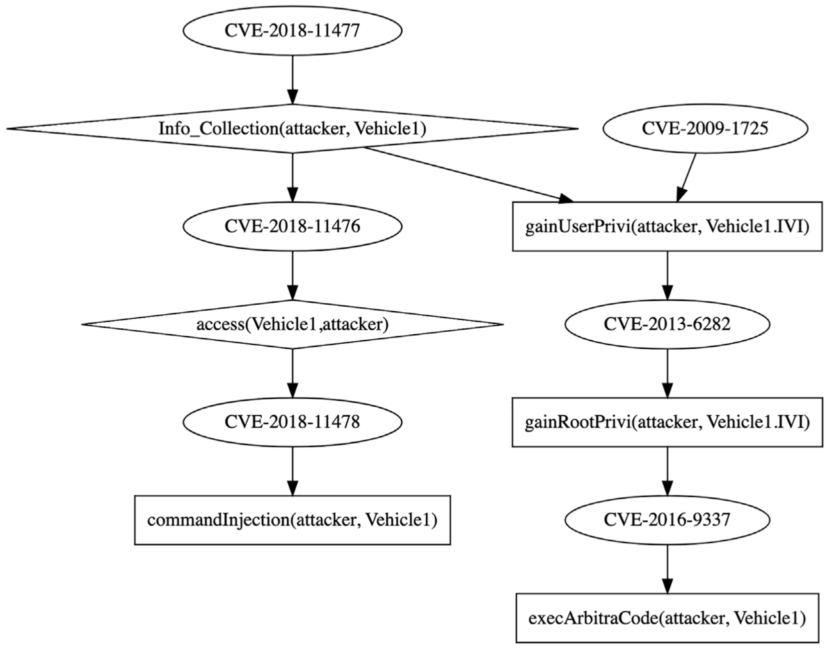
CVE-2018-11477
Info_Collection(attacker, Vehicle1)
CVE-2009-1725
CVE-2018-11476
gainUserPrivi(attacker, Vehicle1.IVI)
access(Vehicle1,attacker)
CVE-2013-6282
CVE-2018-11478
gainRootPrivi(attacker, Vehicle1.IVI)
commandInjection(attacker, Vehicle1)
CVE-2016-9337
execArbitraCode(attacker, Vehicle1)

FIGURE 6 | Attack graph at time T_1.

$gainRootPrivi(attacker, Vehicle1.IVI)$, and finally passes the ECU gateway vulnerability. Command injection attacks can thus be exploited $execArbitraCode(attacker, Vehicle1)$ .Vehicle 1 will receive arbitrary codes that cause denial of service attacks or other malicious operations possibly.

Path 3 is a physical contact attack. Here, the attacker successfully collects sensitive vehicle information $Info\_Collection(attacker, Vehicle1)$ and accesses vehicle 1 through an unauthenticated Wi-Fi connection $access(Vehicle1, attacker)$, and then obtains command injection permissions through a vulnerability in the OBD dongle $commandInjection(attacker, Vehicle1)$. Vehicle 1 will receive and execute malicious commands possibly.

Path 4 is a combination of short-range wireless and physical contact attacks. Here, the attacker uses a pseudo base station and a signal amplifier to conduct a man-in-the-middle attack between vehicle 2 and the telematics service provider (TSP) and then uses the telematics communication box (T-Box) vulnerability V9 to implant a backdoor $gainUserPrivi(attacker, Vehicle2.T-Box)$. At the same time, the attacker obtains the access rights of the vehicle 2 bus by using the vulnerability V10 bypassed by the OBD interface verification and finally obtains the control rights

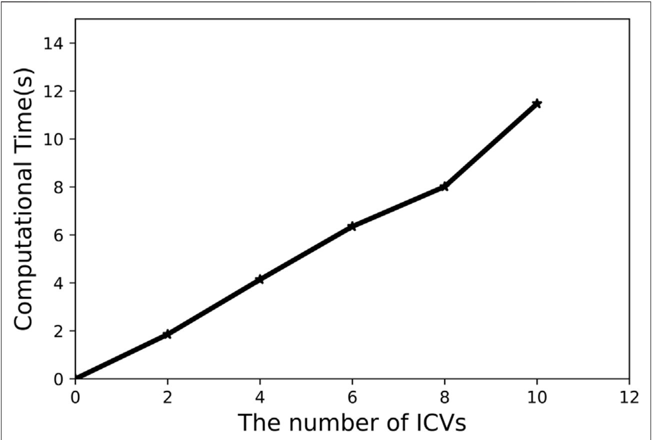

FIGURE 7 | Computational time of Algorithm.

of the T-Box and the vehicle CAN bus $gainRootPrivi(attacker, Vehicle2.CAN)$. The attacker gained top administrative privileges on the vehicle 2.

Consider the following situation: the cloud webserver has updated at T_1 and the vulnerability V2 is remedied. At the same time, the motion state of vehicle 2 changes from static to moving, and the V9 vulnerability exploitation condition is not established. In this case, the V2 and V9 vulnerability nodes should be deleted. Moreover, inference rules whose preconditions involve changing the object properties should be determined, as well as the relevant condition nodes, and then the nodes and the connected edges should be deleted. The dynamic attack graph generated after the node information changes at T_1 is shown in **Figure 6**.

4.2 Algorithm Complexity Analysis

To demonstrate the scalability of algorithm, a series of experiments are conduct. The network configuration information of cloud platform is the same as typical attack scenario. And we simulated a standard vulnerable ICV configuration which has four exploitable vulnerabilities. We gradually increase the number of standard ICVs and obtain the computational time of the algorithm. **Figure 7** shows average computational time in each attack scenario. The experimental results show that with the increase of the number of vehicles, the computational time of attack graph generation does not increase exponentially. After the initial attack graph generation, the attack graph only updates the local attack graph rather than the overall attack graph generation.

4.3 Algorithm Complexity Analysis

To verify the scalability of the algorithm, this paper evaluates the efficiency of the algorithm by analyzing its time complexity. Given the number of nodes in graph generated N and the number of edges E, when the network information of the IoV changes, the algorithm only updates the corresponding instance relationships and properties in the ontology and makes local changes to the attack graph. The time complexity of this local change is $O\ (\Delta n+\Delta e)$. The traditional attack graph generation tool MulVAL (Ou, Govindavajhala and Appel, 2005) needs to reconstruct the overall attack graph, and the time complexity of this algorithm is $O\ (n + e)$. Obviously, $O\ (\Delta n+\Delta e) < O\ (n + e)$. Compared with regenerating the complete attack graph, the dynamic attack graph generation algorithm based on the local update proposed in this paper reduces the computational overhead, improves the timeliness of the attack graph, and can better adapt to the rapid change in the IoV topology.

5 CONCLUSION

This paper proposes an IoV network security ontology model in combination with the "cloud-channel-edge-terminal" and constructs an IoV reasoning rule knowledge base by SWRL rules. Both of them help with applying attack graph technology to the field of network security of the IoV and can describe attack scenarios in IoV system well. On this basis, this paper proposes a dynamic attack graph generation algorithm that can be updated incrementally according to the changes in the network topology, which is more suitable for IoV networks characterized by rapid changes in network topology. The algorithm can show the vulnerability of the global IoV network effectively and help carry out better risk management and has lower algorithm complexity while updating attack graph. Finally, the experiment results demonstrated the effectiveness and feasibility of the proposed algorithm.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

HS:Conceptualization, Methodology,Software, Writing—Original Draf CX:Writing—Review and Editing, Supervision MJ:Supervision, Resources ZZ:Supervision, Resources YH:Data Curation.

FUNDING

This study was supported by the Joint Funds of the National Natural Science Foundation of China (no. U2003206).

REFERENCES

China Software Tesing Center (2020). Intelligent Internet Connected Vehicle Safety Penetration White Paper. Available at: https://www.cstc.org.cn/info/1202/231181.htm.

Choi, W., Joo, K., Jo, H. J., Park, M. C., and Lee, D. H. (2018). VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion Detection System. *IEEE Trans.Inform.Forensic Secur.* 13 (8), 2114–2129. doi:10.1109/TIFS.2018.2812149

Glimm, B., Horrocks, I., Motik, B., Stoilos, G., and Wang, Z. (2014). HermiT: An OWL 2 Reasoner. *J. Autom. Reason.* 53 (3), 245–269. doi:10.1007/s10817-014-9305-1

Guarino, N., Oberle, D., and Staab, S. (2009). "What Is an Ontology?," in *Handbook on Ontologies* (Springer), 1–17. doi:10.1007/978-3-540-92673-3_0

Horrocks, I.,(2004). SWRL: A Semantic Web Rule Language Combining OWL and RuleML. *W3C Memb. Submiss.* 21 (79), 1–31.

Iannacone, M. (2015). "Developing an Ontology for Cyber Security Knowledge Graphs," in Proceedings of the 10th Annual Cyber and Information Security Research Conference, April 2015, 1–4. doi:10.1145/2746266.2746278

Ibrahim, M., Al-Hindawi, Q., Elhafiz, R., Alsheikh, A., and Alquq, O. (2020). Attack Graph Implementation and Visualization for Cyber Physical Systems. *Processes* 8 (1), 12. doi:10.3390/pr8010012

Keen, Security (2022). *Lab Blog* (No Date) *Keen Security Lab Blog*. Available at: http://keenlab.tencent.com/index.html (Accessed May 13, 2022).

Kong, H.-K., Hong, M. K., and Kim, T.-S. (2018). Security Risk Assessment Framework for Smart Car Using the Attack Tree Analysis. *J. Ambient. Intell. Hum. Comput.* 9 (3), 531–551. doi:10.1007/s12652-016-0442-8

Koscher, K. (2010). "Experimental Security Analysis of a Modern Automobile," in Proceedings of the IEEE symposium on security and privacy, Oakland, CA, USA, May 2010 (IEEE), 447–462. doi:10.1109/sp.2010.34

Lallie, H. S., Debattista, K., and Bal, J. (2020). A Review of Attack Graph and Attack Tree Visual Syntax in Cyber Security. *Comput. Sci. Rev.* 35, 100219. doi:10.1016/j.cosrev.2019.100219

Li, X. (2019). Survey of Internet of Vehicles Security. *J. Cyber Secur.* 4 (03), 17–33. doi:10.19363/J.cnki.cn10-1380/tn.2019.05.02

McGuinness, D. L., and Van Harmelen, F. (2004). OWL Web Ontology Language Overview. *W3C Recomm.* 10 (10), 2004.

Miller, C., and Valasek, C. (2014). *A Survey of Remote Automotive Attack Surfaces*. South Las Vegas, Nevada: black hat USA, 94.

Miller, C., and Valasek, C. (2015).Remote Exploitation of an Unaltered Passenger Vehicle. Black Hat USA, S 91.

Ou, X., Govindavajhala, S., and Appel, A. W. (2005). "MulVAL: A Logic-Based Network Security Analyzer," in *USENIX Security Symposium* (Baltimore, MD, 113–128.

Sadri, M. J., and Rajabzadeh Asaar, M. (2020). A Lightweight Anonymous Two-Factor Authentication Protocol for Wireless Sensor Networks in Internet of Vehicles. *Int. J. Commun. Syst.* 33 (14), e4511. doi:10.1002/dac.4511

SAE J3061 Vehicle Cybersecurity Systems Engineering Committee (2016). *Cybersecurity Guidebook for Cyber-Physical Vehicle Systems*. Warrendale, Pennsylvania: SAE International.

Salfer, M., and Eckert, C. (2018). "Attack Graph-Based Assessment of Exploitability Risks in Automotive On-Board Networks," in Proceedings of the 13th International Conference on Availability, Reliability and Security. ARES 2018: International Conference on Availability (Hamburg Germany: Reliability and SecurityACM), 1–10. doi:10.1145/3230833.3230851

Shao, J., Lin, X., Lu, R., and Zuo, C. (2016). A Threshold Anonymous Authentication Protocol for VANETs. *IEEE Trans. Veh. Technol.* 65 (3), 1711–1720. doi:10.1109/TVT.2015.2405853

Song, H. M., Woo, J., and Kim, H. K. (2020). In-vehicle Network Intrusion Detection Using Deep Convolutional Neural Network. *Veh. Commun.* 21, 100198. doi:10.1016/j.vehcom.2019.100198

Sutrala, A. K., Bagga, P., Das, A. K., Kumar, N., Rodrigues, J. J. P. C., and Lorenz, P. (2020). On the Design of Conditional Privacy Preserving Batch Verification-Based Authentication Scheme for Internet of Vehicles Deployment. *IEEE Trans. Veh. Technol.* 69 (5), 5535–5548. doi:10.1109/TVT.2020.2981934

Thuen, C. (2016). Commonalities in Vehicle Vulnerabilities. Retrieved from Infosecurity.

Upstream Security (2020). Global Automotive Cybersecurity Report | Upstream. Available at: https://upstream.auto/upstream-security-global-automotive-cybersecurity-report-2020/(Accessed May 13, 2022).

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.