

# 上海交通大学

SHANGHAI JIAO TONG UNIVERSITY

## 学士学位论文

BACHELOR'S THESIS



论文题目： V2X 车联网攻击图生成技术

学生姓名： 刘浩文  
学生学号： 517021911065  
专业： 信息安全  
指导教师： 马进  
学院(系)： 电子信息与电气工程学院



## 上海交通大学 学位论文原创性声明

本人郑重声明：所呈交的学位论文《V2X 车联网攻击图生成技术》，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

刘浩文

日期：2021年06月03日



# 上海交通大学

## 学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密，在\_\_年解密后适用本授权书。

本学位论文属于

不保密.

(请在以上方框内打“√”)

学位论文作者签名:  指导教师签名: 

日期:2021年06月03日

日期:2021年06月03日



# V2X 车联网攻击图生成技术

## 摘 要

近年来，车联网（Internet of Vehicles, IoV）技术飞速发展，车辆到万物（Vehicle-to-Everything, V2X）的通信正逐渐成为现实。然而，相比于传统网络，车联网的组成与结构更为复杂，脆弱性更多，攻击面更大。而车联网的动态性、实时性、复杂性和规模大等特点又使得人们难以对车联网整体进行有效的安全风险分析。因此，我们需要一种车联网安全风险分析与评估技术，来感知车联网实时的安全状态，辅助管理人员做出安全决策，高效地维护车联网安全。

本文提出了一种 V2X 车联网攻击图生成与分析方案，来对车联网进行实时、有效的安全风险分析与评估。本文首先构建了车联网安全本体模型来标准化描述车联网中的实体及实体间的复杂关系，并基于该安全本体模型以及现有的车联网安全知识库构建了车联网的攻击图生成规则集。然后，基于车联网边缘计算技术和蜂窝 V2X 车联网通信架构，设计了分布式的攻击图生成与分析方案，来低延迟地生成与分析车联网的实时攻击图，对车联网进行实时量化风险评估。最后，实现了车联网量化风险评估的原型系统，并构建了攻击场景测试用例，对该原型系统进行了正确性、有效性和实时性的测试。实验结果表明，原型系统可以低延迟地正确生成车联网全局贝叶斯攻击图，展示车联网的完整攻击路径，并且给出车联网局部和整体的量化风险值，可有效地辅助车联网的安全管理工作。

**关键词：** 攻击图，量化风险评估，蜂窝 V2X 车联网，边缘计算，车联网安全



# ATTACK GRAPH GENERATION TECHNIQUE FOR V2X INTERNET OF VEHICLES

## ABSTRACT

Recent years have witnessed the rapid development of Internet of Vehicles (IoV) technology, and Vehicle-to-Everything (V2X) communication is gradually becoming a reality. However, compared with traditional networks, the composition and structure of the IoV are much more complex and vulnerable. What's worse, the dynamicity, timeliness, complexity, and large-scale of the IoV make it difficult for people to analyze the overall security risk of the IoV effectively. Therefore, there is an urgent need for an IoV security risk analysis and assessment technique to obtain real-time IoV security status, assist network administrator to make security decisions, and efficiently maintain IoV security.

This paper proposes a V2X IoV attack graph generation and analysis scheme for real-time and effective security risk analysis and assessment of the IoV. Firstly, we construct an IoV security ontology model to standardize the description of entities and the complex relationship between entities in the IoV. Then, based on the IoV security ontology model and the IoV security knowledge database, we construct an IoV attack graph generation rule set. After that, based on the IoV MEC (Multi-access Edge Computing) and C-V2X (Cellular-V2X) communication architecture, we design a distributed attack graph generation and analysis scheme to generate and analyze the real-time attack graph of the IoV with low latency, to quantitatively assess the risk in the IoV. Finally, we implement the prototype system based on the scheme, and build attack scenario test cases to test the prototype system's correctness, effectiveness and real-time. Experimental results show that the prototype system can generate global Bayesian attack graphs of the IoV correctly with low latency, present the complete attack paths in the IoV, and provide local and global quantitative risk value to assist the security management of the IoV.

**Key words:** attack graph, quantitative risk assessment, C-V2X, MEC, IoV security



## 目 录

|                                   |    |
|-----------------------------------|----|
| 第一章 绪论 .....                      | 1  |
| 1.1 研究背景与意义 .....                 | 1  |
| 1.2 国内外研究现状 .....                 | 1  |
| 1.2.1 车联网安全研究 .....               | 1  |
| 1.2.2 攻击图技术研究 .....               | 2  |
| 1.3 本文主要研究内容 .....                | 2  |
| 1.3.1 方案设计 .....                  | 3  |
| 1.3.2 原型系统实现与测试 .....             | 4  |
| 1.4 本文的组织结构 .....                 | 4  |
| 第二章 车联网安全本体模型 .....               | 7  |
| 2.1 车联网理论基础 .....                 | 7  |
| 2.1.1 车联网的组成与架构 .....             | 7  |
| 2.1.2 车联网的特点 .....                | 8  |
| 2.2 车联网的信息安全风险与需求 .....           | 9  |
| 2.2.1 车联网面临的信息安全风险 .....          | 9  |
| 2.2.2 车联网的信息安全需求 .....            | 9  |
| 2.3 车联网安全本体建模 .....               | 10 |
| 2.3.1 基于本体的车联网安全要素及其关系建模 .....    | 10 |
| 2.3.2 车联网安全本体模型的攻击场景用例 .....      | 12 |
| 2.4 本章小结 .....                    | 17 |
| 第三章 车联网实时攻击图生成与分析的方案设计 .....      | 19 |
| 3.1 系统模型 .....                    | 19 |
| 3.1.1 方案相关技术概述 .....              | 19 |
| 3.1.2 车联网信息安全分析与评估面临的挑战 .....     | 21 |
| 3.1.3 车联网攻击图生成与分析方案系统模型 .....     | 21 |
| 3.2 车联网中的攻击图生成规则集构建 .....         | 22 |
| 3.2.1 云层的攻击图生成规则集 .....           | 24 |
| 3.2.2 管层的攻击图生成规则集 .....           | 25 |
| 3.2.3 端层的攻击图生成规则集 .....           | 25 |
| 3.3 车联网的实时安全信息收集方案 .....          | 27 |
| 3.3.1 生成攻击图所需的输入信息 .....          | 27 |
| 3.3.2 端层实体的安全信息收集 .....           | 28 |
| 3.3.3 蜂窝服务区的安全信息收集 .....          | 28 |
| 3.3.4 云平台的安全信息收集 .....            | 29 |
| 3.4 基于 MEC 的车联网实时攻击图生成与分析方案 ..... | 29 |
| 3.4.1 云平台的局部攻击图生成与分析 .....        | 30 |

|                                     |    |
|-------------------------------------|----|
| 3.4.2 蜂窝服务区的局部攻击图生成与分析 .....        | 30 |
| 3.4.3 车联网实时量化风险评估结果的生成 .....        | 30 |
| 3.5 基于贝叶斯攻击图的量化风险评估方案 .....         | 31 |
| 3.5.1 攻击图的预处理 .....                 | 32 |
| 3.5.2 贝叶斯攻击图的计算 .....               | 32 |
| 3.5.3 量化风险评估模型 .....                | 33 |
| 3.6 本章小结 .....                      | 34 |
| <br>第四章 车联网实时量化风险评估原型系统的实现与测试 ..... | 35 |
| 4.1 原型系统实现 .....                    | 35 |
| 4.1.1 攻击图生成工具 MulVAL 简介 .....       | 35 |
| 4.1.2 车联网攻击图的 Datalog 规则集构建 .....   | 36 |
| 4.1.3 车联网实时攻击图生成与分析的实现 .....        | 39 |
| 4.1.4 基于贝叶斯攻击图的量化风险评估方案的实现 .....    | 41 |
| 4.2 原型系统测试 .....                    | 46 |
| 4.2.1 攻击测试场景构建 .....                | 47 |
| 4.2.2 静态车联网攻击图生成与分析测试 .....         | 48 |
| 4.2.3 动态车联网实时攻击图生成与分析测试 .....       | 58 |
| 4.2.4 车联网攻击图应用示例 .....              | 61 |
| 4.2.5 车联网实时量化风险评估性能测试 .....         | 63 |
| 4.3 本章小结 .....                      | 64 |
| <br>第五章 总结与展望 .....                 | 65 |
| 5.1 本文主要内容总结 .....                  | 65 |
| 5.2 局限性及展望 .....                    | 65 |
| <br>附录 A 攻击图节点完整信息 .....            | 67 |
| <br>参考文献 .....                      | 71 |
| <br>谢 辞 .....                       | 75 |

## 插图索引

|                                     |    |
|-------------------------------------|----|
| 图 1-1 主要研究内容框图 .....                | 3  |
| 图 1-2 论文结构图 .....                   | 5  |
| 图 2-1 V2X 车联网的“云”、“管”、“端”三层架构 ..... | 7  |
| 图 2-2 通用网络安全本体模型 .....              | 10 |
| 图 2-3 V2X 车联网安全本体模型 .....           | 11 |
| 图 2-4 V2X 车联网攻击测试场景 .....           | 14 |
| 图 3-1 基于 MEC 的智能交通系统 .....          | 20 |
| 图 3-2 V2X 车联网攻击图生成与分析的系统模型 .....    | 22 |
| 图 3-3 V2X 车联网攻击图生成与分析流程 .....       | 23 |
| 图 3-4 生成攻击图所需的输入信息 .....            | 27 |
| 图 4-1 一个 MulVAL 攻击图的例子 .....        | 35 |
| 图 4-2 原型系统进程间关系 .....               | 40 |
| 图 4-3 贝叶斯攻击图计算流程 .....              | 42 |
| 图 4-4 两种攻击图环路 .....                 | 43 |
| 图 4-5 LEAF 型节点 .....                | 45 |
| 图 4-6 AND 型节点 .....                 | 45 |
| 图 4-7 OR 型节点 .....                  | 45 |
| 图 4-8 直接使用 MulVAL 生成的车联网全局攻击图 ..... | 49 |
| 图 4-9 云平台局部贝叶斯攻击图 .....             | 51 |
| 图 4-10 服务区 1 局部贝叶斯攻击图 .....         | 52 |
| 图 4-11 服务区 2 局部贝叶斯攻击图 .....         | 53 |
| 图 4-12 服务区 3 局部贝叶斯攻击图 .....         | 54 |
| 图 4-13 全局贝叶斯攻击图（右半部分） .....         | 56 |
| 图 4-14 全局贝叶斯攻击图（左半部分） .....         | 56 |
| 图 4-15 拓扑变化后的全局贝叶斯攻击图（变化部分） .....   | 59 |
| 图 4-16 漏洞变化后的全局贝叶斯攻击图（变化部分） .....   | 60 |
| 图 4-17 车辆状态变化后的全局贝叶斯攻击图（变化部分） ..... | 61 |
| 图 4-18 修复 SQL 服务漏洞后的全局贝叶斯攻击图 .....  | 62 |
| 图 4-19 修复蓝牙漏洞后的全局贝叶斯攻击图 .....       | 63 |
| 图 4-20 车联网量化风险评估延迟 .....            | 64 |



## 表格索引

|                            |    |
|----------------------------|----|
| 表 2-1 V2X 车联网的五种通信方式 ..... | 8  |
| 表 2-2 V2X 车联网应用 .....      | 9  |
| 表 2-3 车内网的安全风险 .....       | 10 |
| 表 2-4 本体类间关系 .....         | 12 |
| 表 2-5 攻击场景信息 .....         | 13 |
| 表 4-1 安全风险等级评估参考表 .....    | 46 |
| 表 4-2 攻击测试场景安全信息 .....     | 47 |
| 表 4-3 漏洞具体信息 .....         | 49 |
| 表 A-1 攻击图节点完整信息 .....      | 67 |



## 算法索引

算法 3-1 全局贝叶斯攻击图局部拼接更新算法 ..... 31

算法 4-1 基于 DFS 的最深原子攻击消环算法 ..... 44



## 第一章 绪论

### 1.1 研究背景与意义

现代智能交通系统 (Intelligent Transport System, ITS) 非常依赖于车辆到万物 (Vehicle-to-Everything, V2X) 车联网通信系统，该系统构成了 ITS 体系结构不可或缺的基础部分。V2X 车联网的主要目标是增强道路安全和改善交通管理。其定义了车辆与行人、其他车辆、路基设施、云平台和互联网之间的信息交换<sup>[1-3]</sup>。当前研究展示了 V2X 车联网支持各种高级应用的潜力，例如远程车辆诊断，交叉路口避碰，车载互联网访问和协作式碰撞警告等。通过增强行人、车辆和交通基础设施之间的协作，V2X 车联网可以实现更安全的交通系统。2015 年，西门子在德国的 A9 高速公路上实施了首个全动态系统。结果表明，事故减少了 35%，道路上受伤的人数减少了 31%<sup>[4]</sup>。目前，我国已经开始在一些城市推广应用 V2X 车联网。2019 年我国车联网的规模约为 574 亿元<sup>[5]</sup>。

然而，随着 V2X 车联网技术的高速发展与广泛应用，其所面临的安全风险也逐渐显现<sup>[6-8]</sup>。与传统网络相比，V2X 车联网的应用环境更加特殊，网络结构与成分更加复杂，面临的安全威胁也更加突出和复杂。V2X 车联网不仅会受到传统互联网中恶意攻击的威胁，还会面临车联网中特有的安全风险。车联网终端如果被攻击者攻陷，会造成隐私信息被泄漏、资产被盗窃、甚至车辆被控制等重大安全问题。并且，由于 V2X 广泛用于车辆通信的多种应用中，包括交通安全应用和信息娱乐应用，这些应用对 V2X 提出了各种要求，使 V2X 车联网安全研究面临诸多挑战。

由于车联网所面临的复杂网络威胁和安全要求，传统的网络安全分析技术难以直接适用于车联网，从而使得车联网对恶意攻击更加难以预测、预防与防御。现有针对车联网安全威胁的措施多是对基础设施的改良<sup>[9-15]</sup>，而缺少对车联网整体的安全分析与风险感知技术。因此，本文尝试将传统网络安全中用于网络安全分析的攻击图技术迁移至车联网的安全防护领域，使用攻击图技术对 V2X 车联网进行网络脆弱性分析。

攻击图技术已经是一种相对成熟的网络安全分析与评估技术，应用于传统网络，如企业网络的脆弱性分析和安全评估中<sup>[16-23]</sup>。然而，由于 V2X 车联网相对于传统网络的特殊性，如高速移动的车辆导致的快速而剧烈的网络拓扑变化，以及异质网通信导致的复杂网络结构，使得攻击图技术不能直接应用于车联网安全分析中。所以，为了将攻击图技术迁移应用至车联网安全领域，首先需要结合车联网网络环境，分析车联网的脆弱点产生原因，总结出车联网环境的攻击图生成规则；其次，需要结合车联网网络环境的特点，探索一种实时、高效、实用的攻击图生成与分析方法与基于攻击图的安全评估与建议体系。

### 1.2 国内外研究现状

#### 1.2.1 车联网安全研究

在车联网为人们提供便捷的同时，其面临的风险也在时刻威胁人们的安全。360 网络攻防实验室在 2015 年通过伪造钥匙的原始射频信号控制发动机的电子控制单元 (Electronic Control Unit, ECU)，成功入侵了特斯拉汽车<sup>[24]</sup>。同是 2015 年，美国的 Charlie Miller 通过远程指令入侵了 Uconnect 车载系统，利用远程命令致其翻车<sup>[25]</sup>。腾讯科恩实验室在车联网终

端漏洞挖掘中也有不少成果。2018 年，腾讯科恩实验室发现多款 BMW 汽车（2012 至 2018 年生产的汽车）上的 Head Unit HU\_NBT 组件存在多个安全漏洞，研究人员指出，攻击者可以利用这些漏洞，在目标车辆行驶时向车辆的 CAN (Controller Area Network, CAN) 总线和发动机控制单元注入恶意信息，从而控制或干扰车辆的运行<sup>[26]</sup>。2020 年 11 月 23 日，来自 Belgian University KU Leuven 的安全研究者 Lennert Wouters 通过组合利用特斯拉 Model X 汽车上的几个验证绕过漏洞，只使用了约 300 美元的设备，在 90 秒之内通过蓝牙解锁了特斯拉汽车，并在进入汽车后的一分钟之内取得了汽车的控制权<sup>[27]</sup>。

面对车联网中严重的安全威胁，国内外研究者也提出了非常多的安全措施来保护车联网安全。然而，现有针对车联网安全威胁的措施多是对基础设施的改良，如提出更安全的车联网架构<sup>[9, 11, 28]</sup>、增强通信协议的安全性<sup>[12-13, 15]</sup>、加固车联网软件与固件的安全防护，以及入侵检测和异常检测<sup>[14, 29]</sup>，而缺少对车联网整体的安全分析与风险感知技术。现有的车联网安全分析与评估相关研究主要集中在通过仿真测试及实际道路实验挖掘系统中的攻击面，并定性分析漏洞带来的风险，如 Miller 和 Valasek 在 2014 年对不同厂商汽车进行渗透测试后，总结了智能网联汽车存在的远程攻击面，并分析了不同厂商汽车网络的安全性<sup>[30]</sup>；Kelarestaghi 等人在 2019 年基于实际的智能网联汽车攻击案例提出了利用 NIST 安全风险评估方法来定性确定攻击者利用车载网络安全漏洞会造成的不利影响<sup>[31]</sup>。现有的车联网安全分析评估技术缺乏对车联网系统整体的实时安全态势感知，以及对安全风险的定量分析。

### 1.2.2 攻击图技术研究

攻击图模型由 Philips 和 Swiler 于 1998 年提出<sup>[32]</sup>，已经是一种相对成熟的传统网络脆弱性分析技术。早期关于攻击图技术的研究都致力于研究攻击图的自动生成，如 2005 年 Ou 等人实现了基于 Datalog 受限一阶谓词逻辑的攻击图自动生成原型系统 MulVAL<sup>[22, 33-34]</sup>。2010 年前后关于攻击图的研究主要转向了两个方向，其一是提高大规模网络的攻击图生成和分析效率，其二是将攻击图应用于安全风险评估与安全告警。2015 年，Kaynar 和 Sivrikaya 设计了一种分布式的攻击图并行生成算法，达到了提高生成效率的效果<sup>[19]</sup>。2015 年，王秀娟等人提出了攻击环路消除算法和攻击路径贝叶斯概率计算方法。他们通过这两种算法将属性攻击图转化为贝叶斯攻击图，建立了贝叶斯属性攻击图模型<sup>[20]</sup>。2016 年，Ahmad 等人将 MulVAL 和入侵检测系统 SNORT 结合，从而实现攻击过程中的实时防御<sup>[21]</sup>。

除了对攻击图生成与分析技术的研究，近年来还有一些研究致力于将攻击图技术应用于特种网络中，如物联网和工控网。常昊等人在 2018 年将攻击图与贝叶斯理论结合，提出一种基于贝叶斯攻击图的工业控制系统信息安全动态风险评估模型<sup>[17]</sup>。Che 等人于 2020 年提出了基于物联网网络攻击图的关键节点评估算法，通过节点之间的利用关系以及节点攻陷之后的影响两方面的指标，结合改进的 PageRank 算法来计算节点风险值<sup>[18]</sup>。

虽然攻击图技术已经发展得比较成熟，也已有不少研究将攻击图技术应用于新型特种网络中，但关于将攻击图技术应用于车联网中的研究却很少。这一方面是因为车联网是近年来发展起来的新技术；另一方面，车联网本身的特性，如快速变化的拓扑、复杂的网络结构、庞大的设备数量与广泛的攻击面，使得攻击图技术难以直接应用到车联网上。

## 1.3 本文主要研究内容

正如前文所述，由于车联网本身的特性，攻击图技术难以直接应用到车联网上。我们按照传统网络安全态势感知的四个步骤（数据采集、态势理解、态势评估、态势预测<sup>[35]</sup>），拟



解决以下四个主要问题，从而解决将攻击图技术应用到车联网安全风险评估上的难题：

1. 如何获取车联网快速变化的实时网络信息，如网络拓扑、车联网终端的配置与漏洞？
2. 如何设计车联网中特有的攻击图生成规则，如对车联网的特殊漏洞的利用，或车联网中特殊的攻击方法？
3. 如何低延迟地生成车联网实时攻击图？
4. 如何使用攻击图低延迟地量化分析车联网安全风险？

针对这四个问题，本文提出了以下三个研究目标：

1. 设计适用于车联网的实时安全信息收集方案。
2. 应用本体论对车联网系统中的实体及复杂关系进行建模，形式化描述车联网中的攻击方法，总结出车联网的攻击图生成规则。
3. 设计实时攻击图生成与分析方案，构建合理的车联网量化风险评估模型，实现实时攻击图生成与分析原型系统。

如图 1-1 所示，本文的主要研究内容如下，主要分为车联网攻击图生成与分析的方案设计和车联网量化风险评估原型系统的实现与测试两部分。

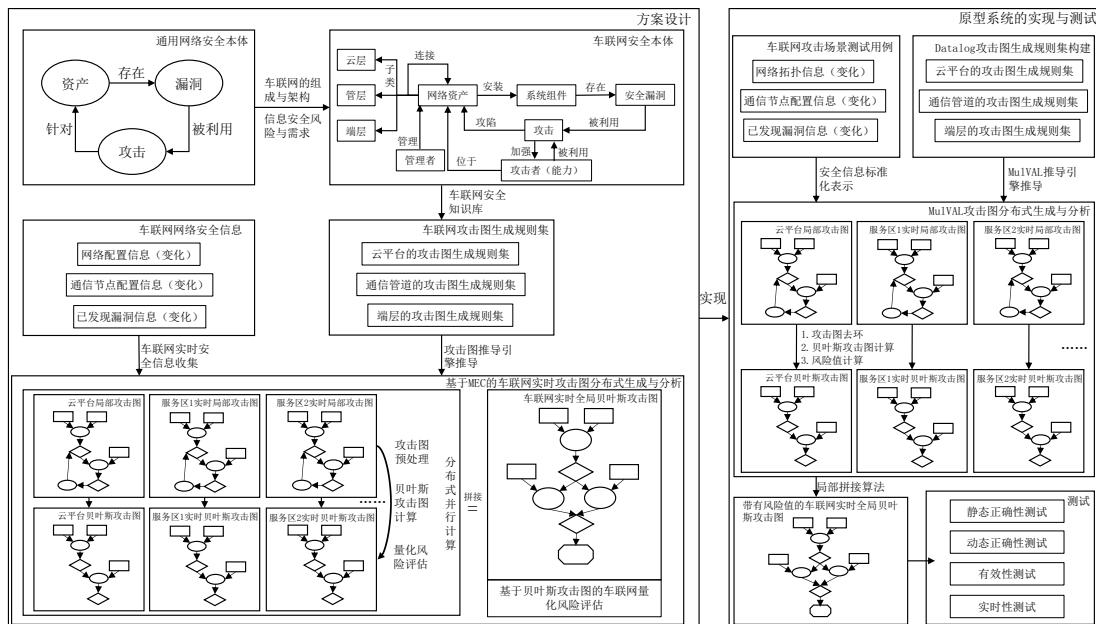


图 1-1 本文主要研究内容

Figure 1-1 Research contents

### 1.3.1 方案设计

在方案设计部分，本文首先概述了车联网的组成与系统架构，并分析了车联网的信息安全风险与需求，然后依据通用网络安全本体模型，提出了车联网安全本体模型。之后，本文设计了典型的攻击场景用例，验证了车联网安全本体模型的正确性和有效性。我们能使用该本体模型形式化地描述车联网中的漏洞利用和攻击方法。

生成攻击图时，需要先向攻击图推导引擎中输入网络中的安全信息，然后使用推导引擎依据攻击图生成规则进行攻击路径推导。本文使用车联网安全本体模型形式化描述车联网安全知识库中的漏洞利用和攻击方法，构建了原始的车联网攻击图生成规则集。

车联网中的网络安全信息主要有变化的网络配置信息、通信节点配置信息和漏洞信息。本文设计了一种车联网实时安全信息收集方案来对上述实时安全信息进行收集。

为了低延迟地生成和分析车联网实时攻击图，本文结合蜂窝 V2X 车联网通信架构的特点，创新性地提出了基于边缘计算（Multi-access Edge Computing, MEC）的实时攻击图分布式生成与分析方案。即，各蜂窝服务区的 MEC 服务器独立并行生成车联网实时局部攻击图，并通过基于贝叶斯攻击图的量化风险评估方案对攻击图进行分析，得到局部贝叶斯攻击图；云平台生成自身的局部攻击图，也使用基于贝叶斯攻击图的量化风险评估方案将攻击图转化为局部贝叶斯攻击图。云平台再将各服务区的局部贝叶斯攻击图与云平台的局部贝叶斯攻击图通过拼接算法生成实时全局贝叶斯攻击图，对车联网整体进行实时量化风险评估。

基于贝叶斯攻击图的量化风险评估方案分为三步。首先进行攻击图的预处理，去除攻击图中的环路。然后，通过漏洞评分系统计算原子攻击（即攻击图中的单步攻击）的成功概率，再进行攻击路径的贝叶斯概率计算，得到贝叶斯攻击图。最后，基于本文提出的量化风险评估模型，利用贝叶斯攻击图计算出车联网局部和整体的量化风险值。

### 1.3.2 原型系统实现与测试

基于上述的方案设计，本文实现了车联网实时量化风险评估的原型系统。

在攻击图生成部分，我们使用攻击图生成工具 MulVAL 的推导引擎。由于 MulVAL 使用 Datalog 受限一阶谓词逻辑进行推导，本文首先将原始的攻击图生成规则集用 Datalog 语言重新表达。

原型系统使用 Python 实现，主要分为多个服务区局部贝叶斯攻击图生成进程，一个云平台局部贝叶斯攻击图生成进程，和一个全局贝叶斯攻击图拼接进程。服务区局部贝叶斯攻击图生成进程监听特定端口的 TCP 连接，该端口收到数据，代表该服务区的安全信息发生了变化。该进程收到实时安全信息后，调用 MulVAL 推导引擎，使用 Datalog 车联网攻击图生成规则集进行攻击图推导，生成服务区实时局部攻击图；再使用基于贝叶斯攻击图的量化风险评估方案，将服务区局部攻击图转化为局部贝叶斯攻击图；最后，进程将该局部贝叶斯攻击图提交给全局贝叶斯攻击图拼接进程的对应端口。云平台局部贝叶斯攻击图生成进程采用同样的方法生成并提交云平台局部贝叶斯攻击图。全局贝叶斯攻击图拼接进程通过监听 TCP 连接，接收局部贝叶斯攻击图。该进程通过局部拼接更新算法将所有局部贝叶斯攻击图拼接为实时全局贝叶斯攻击图，得到车联网局部和整体的量化风险值。

本文最后还对该原型系统进行了测试。本文构建了攻击场景测试用例，测试场景中包含车联网云、管、端的多种漏洞与攻击方法，且其网络拓扑、漏洞信息、车辆状态动态变化。之后，使用原型系统对该车联网攻击场景进行实时量化风险评估，对贝叶斯攻击图生成结果和量化风险评估结果做了详细的分析，验证了原型系统的正确性与有效性。最后，对原型系统进行量化风险评估的性能进行了测试，发现其时间延迟满足车联网对于实时性的要求。

## 1.4 本文的组织结构

本文各章节的编排如图 1-2 所示。第一章为绪论，介绍了车联网的应用前景与严峻的安全形势，由此得出了本研究的意义。之后本章分别介绍了车联网安全研究和攻击图技术研究的现状，并由此引出本文的主要研究内容。最后本章简要介绍了文章的组织结构。

第二章基于车联网相关理论基础和车联网的信息安全风险与需求进行了车联网的安全本体建模。并构建了基础的车联网攻击场景，测试了车联网安全本体模型的正确性和有效

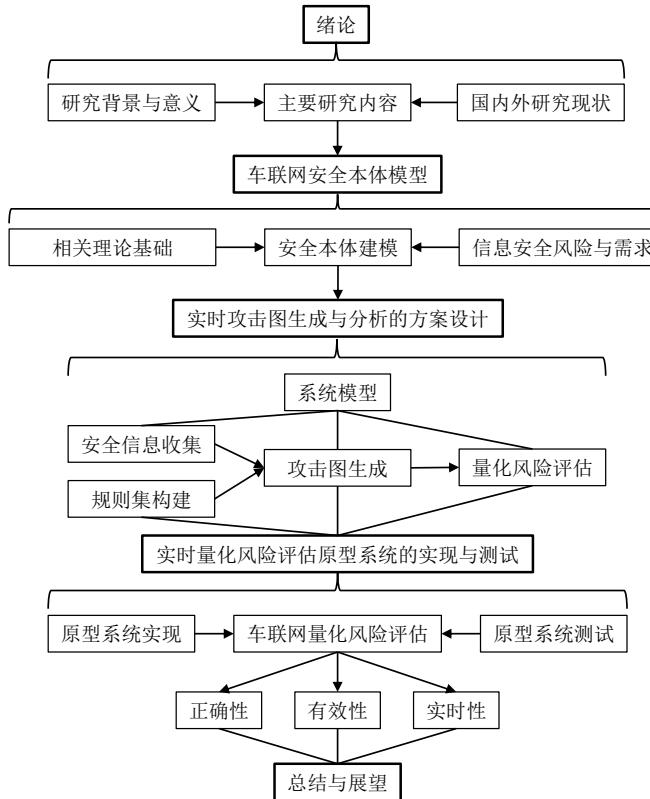


图 1-2 论文结构图

Figure 1-2 Thesis structure

性。

第三章基于车联网边缘计算与蜂窝 V2X 车联网通信架构，提出了车联网实时攻击图生成与分析的系统模型，包括安全信息收集、攻击图生成、量化风险评估三个部分，并分别详细说明了三个部分的方案设计。此外，本章还基于车联网安全本体模型进行了车联网中原始的攻击图生成规则集的构建。

第四章基于 MulVAL 推导引擎，实现了车联网实时量化风险评估的原型系统，并构建攻击场景测试用例对原型系统进行了正确性、有效性和实时性的测试。原型系统的实现包括原始规则集的 Datalog 实例化，以及基于 MEC 的攻击图生成与分析方案的实现。原型系统的测试包含了攻击测试场景的构建、静态与动态正确性测试、有效性测试和实时性测试。

第五章对本文的工作进行总结和展望，回顾了主要研究工作，讨论了本研究的创新点与局限性，并对未来的研究进行了展望。





## 第二章 车联网安全本体模型

本章首先介绍了车联网的组成架构与特点，基于此分析了车联网的安全风险与需求，然后在通用网络安全本体模型的基础上构建了V2X车联网安全本体模型，并设计了攻击场景用例对该模型进行了测试。

### 2.1 车联网理论基础

#### 2.1.1 车联网的组成与架构

V2X车联网将通信、传感器和数据处理等技术应用于行人、车辆和路基设施，以提高交通系统的效率和安全性。如图2-1所示，V2X车联网的系统架构一般为“云”、“管”、“端”三层架构。以下将依托该架构来介绍V2X车联网的组成。

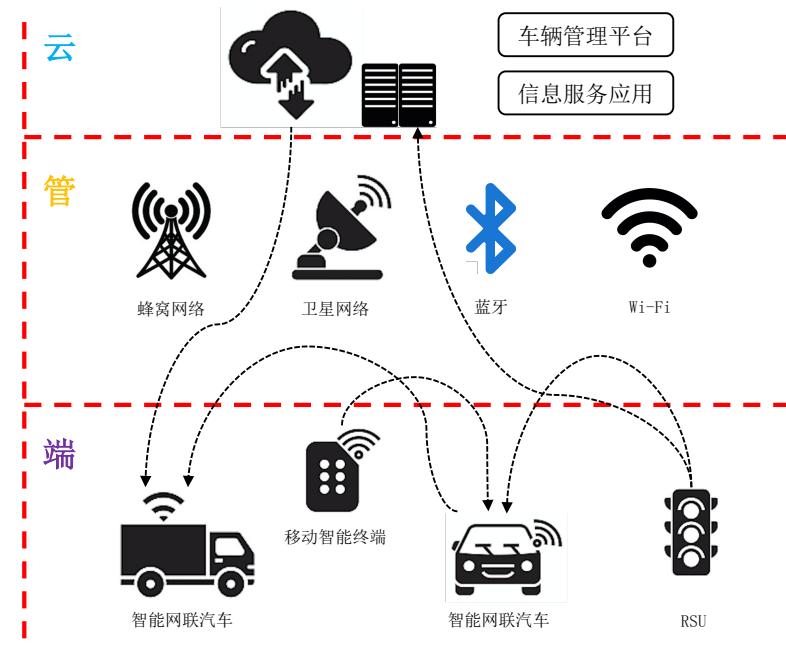


图 2-1 V2X 车联网的“云”、“管”、“端”三层架构

Figure 2-1 The Cloud-Channel-Device three-layer architecture of V2X IoV

**云层。** 云层即车联网服务云平台。它提供车联网汽车管理和交通信息内容服务<sup>[24]</sup>。车辆可以通过T-BOX和云服务平台交互。云平台是车联网数据汇聚、数据处理与远程管控的核心<sup>[36]</sup>。广义上的云层还包括互联网。

**管层。** 管层即V2X车联网通信层，提供了所连接实体的通信平台。它给予了道路实体向其他远距离或近距离道路实体发送信息的能力，例如其当前速度，位置和方向。之后，V2X车联网使用这些信息做出决策。道路实体间通信的类型取决于建立连接的实体。V2X车联网中使用了短程直连通信（Dedicated Short Range Communication, DSRC）、车联网无线连接

(Wireless Access in Vehicular Environments, WAVE)、蜂窝车联网 (Cellular-V2X, C-V2X)、蓝牙、Wi-Fi 等多种通信方式。V2X 车联网支持五种通信类型，如表 2-1 所示。

表 2-1 V2X 车联网的五种通信方式  
Table 2-1 Five communication types in V2X IoV

| 通信方式                               | 通信对象        | 通信技术           |
|------------------------------------|-------------|----------------|
| 车-云通信 (Vehicle-to-Network, V2N)    | 云平台、互联网     | 蜂窝网络           |
| 车-车通信 (Vehicle-to-Vehicle, V2V)    | 附近车辆        | 蜂窝网络、DSRC      |
| 车-路通信 (Vehicle-to-Road, V2R)       | 路基设施        | RFID、蜂窝网络、DSRC |
| 车-人通信 (Vehicle-to-Pedestrian, V2P) | 智能移动终端      | 蜂窝网络、Wi-Fi、蓝牙  |
| 车内通信 (Intra-vehicle communication) | 车内 ECU、操作系统 | CAN 总线         |

**端层。** 端层包括了智能网联汽车、道路基础设施、路侧单元 (Road-Side Unit, RSU)、移动智能终端等 V2X 车联网终端。移动智能终端以智能手机等终端设备为主，用于实现人与智能网联汽车、车联网云平台等的交互<sup>[36]</sup>。移动智能终端通过车联网 App 完成对车辆的控制，如开锁、启动车辆等功能。智能网联汽车由车内总线、T-BOX (Telematics BOX)、车载诊断接口 (On-Board Diagnostic, OBD)、各电子控制单元 (Electronic Control Unit, ECU)、以及车载综合信息系统 (In-Vehicle Infotainment, IVI) 等组成。

### 2.1.2 车联网的特点

由于 V2X 车联网是一种特殊的物联网系统，较之传统网络，V2X 车联网有其鲜明的特点，这些特点也对车联网的安全研究造成了非常大的影响，对 V2X 车联网安全技术提出了独特的要求。本节总结了 V2X 车联网的特点如下：

- **动态变化的网络拓扑：**由于 V2X 车联网是一种典型的网络-物理系统 (cyber-physics system)，其终端的物理特性，即移动性，如车辆的高速行驶、智能移动终端的移动，使 V2X 车联网的网络拓扑不断地动态变化。
- **网络规模大：**车联网是一个巨大的网络。即便是单个 RSU 或蜂窝服务区，也可能覆盖几十个甚至上百个通信节点。
- **异质网络通信：**V2X 车联网的异质网络通信有两层含义。其一是 V2X 车联网中包含多种通信方式和通信类型。如小节 2.1.1 所述，V2X 车联网使用了 DSRC、WAVE、C-V2X、蓝牙、Wi-Fi 等多种通信技术，包含 V2N、V2V、V2R、V2P、车内通信等五种通信场景。其二是现在的车联网在全球并没有一个统一的通信标准，甚至在同一个地区，不同的汽车厂商也使用不同的通信技术或协议。
- **网络复杂性：**V2X 车联网的复杂性包括通信场景的复杂性和车联网应用的复杂性。通信场景的复杂性由车联网的异质网络通信和网络规模大的特点造成，其通信技术多、通信协议种类多，而通信节点的数量与种类更是极为众多，导致 V2X 车联网通信场景具有很高的复杂性。而车联网应用的复杂性产生于车联网应用层的业务的复杂多样，如表 2-2 所示，车联网提供道路安全、交通管控、车载娱乐等一系列复杂的应用。

表 2–2 V2X 车联网应用  
Table 2–2 V2X IoV applications

| 应用类型   | 应用功能         | 通信方式                | 延迟  | 丢包率 |
|--------|--------------|---------------------|-----|-----|
| 道路安全应用 | 防撞、路标提示、事故避让 | DSRC、WAVE、Wi-Fi、蜂窝网 | 非常低 | 非常低 |
| 交通管控应用 | 交通决策、道路监控等   | DSRC, WAVE, 蜂窝网     | 低   | 低   |
| 车载娱乐功能 | 音乐、停车、预约等    | DSRC, WAVE, 蜂窝网     | 中等  | 低   |

## 2.2 车联网的信息安全风险与需求

随着 V2X 车联网的高速发展与广泛应用，其所面临的安全风险也逐渐显现。与传统网络相比，V2X 车联网的应用环境更加特殊，网络结构与成分更加复杂，攻击向量更为广泛，面临的安全风险也更加突出和复杂。

### 2.2.1 车联网面临的信息安全风险

根据小节 2.1.1 所述的车联网“云”、“管”、“端”三层架构，我们可以将车联网面临的信息安全风险分为云平台的信息安全风险、网络通信的信息安全风险和车联网终端的信息安全风险。

**云平台的信息安全风险。** 作为数据中心和服务中心，车联网云平台内部网络架构一般为传统网络架构，其既存在传统的网络配置和管理漏洞的威胁，也会受到传统的操作系统和软件漏洞的威胁。所以云平台容易遭受传统的网络攻击，并将传统的网络安全问题引入车联网中。

**网络通信的信息安全风险。** 如小节 2.1.1 所述，V2X 车联网使用了 DSRC、WAVE、蜂窝移动网、蓝牙、Wi-Fi 等多种无线通信技术。V2X 车联网继承了这些无线通信方式具有的安全风险，如身份验证绕过、通信数据泄露、通信数据被篡改或阻断、以及网络入侵等问题。攻击者可以通过利用车联网无线通信的漏洞侵犯用户的隐私，甚至危害驾驶安全。

**车联网终端的信息安全风险。** 由于 V2X 车联网的端层的终端数量与种类众多，其所受的安全威胁也最为复杂，攻击面也最广。其一是车联网移动智能终端所受到的安全威胁，主要体现为车联网 APP 的安全风险。车联网 APP 是车联网用户通过智能移动终端控制车辆的方式之一。而它们因为使用较广，而且容易得到，而备受攻击者青睐。另外，攻击者可能向移动终端系统的 App 植入恶意代码，并将其作为跳板，渗透进车辆内部。攻击者可以基于此窃取隐私信息甚至控制汽车。其二是智能网联汽车面临的安全威胁，主要体现为车内网的安全风险。如小节 2.1.1 所述，智能网联汽车由车内总线、T-BOX、OBD、ECU、以及 IVI 等组成，因此智能网联汽车平台的安全风险主要分布在这些组件上。本小节总结车内网的安全风险如表 2–3 所示。

### 2.2.2 车联网的信息安全需求

依据上一小节分析的车联网面临的信息安全风险，本小节总结了 V2X 车联网的信息安全需求如下：

表 2-3 车内网的安全风险  
Table 2-3 Intra-vehicle security risks

| 组件     | 风险             | 后果             |
|--------|----------------|----------------|
| CAN 总线 | 缺乏加密和访问控制机制    | 伪造指令           |
|        | 缺乏认证及消息校验机制    | 消息伪造、拒绝服务、重放   |
| T-BOX  | 固件被逆向分析        | 窃听和伪造指令        |
|        | 通信数据被窃取        | 用户隐私信息泄露       |
| OBD 接口 | OBD 接口被监听      | 攻击者得到 ECU 控制命令 |
|        | 接入意外外接设备       | 注入恶意代码         |
|        | 缺少鉴权与认证机制      | 无法识别恶意报文       |
| ECU    | 芯片设计漏洞         | 攻击者绕过认证或鉴权     |
|        | 固件应用程序漏洞       | 执行恶意代码         |
|        | ECU 更新程序缺乏校验机制 | CAN 总线被注入命令    |
| IVI    | 软件升级的漏洞        | 下载指定的恶意程序      |
|        | 硬件接口被拆解        | 获取系统内信息        |
| 车载 OS  | 传统操作系统漏洞       | 内核提权、缓冲区溢出等    |
|        | 操作系统组件及应漏洞     | 系统遭到入侵         |

- **验证**: 确保接收者能够从真正的发送者那里接收消息。
- **信息机密性**: 确保未经授权的访问不会泄露信息内容。
- **消息完整性**: 传输消息时未更改或替换传输的消息。
- **可用性**: 即使发生故障，服务和协议也应保持功能正常。
- **访问控制**: 为各种网络实体授予对特定服务的访问权限。访问控制授权节点在网络中执行允许的操作，例如某节点可以执行的网络协议。
- **隐私和匿名**: V2X 通信应提供保护网络用户隐私的功能，不得透露用户的隐私信息。

## 2.3 车联网安全本体建模

### 2.3.1 基于本体的车联网安全要素及其关系建模

在计算机科学领域中，本体被用于刻画信息对象来进行领域知识共享和应用<sup>[37]</sup>。其使用形式化的术语，形式化表达某一领域内的概念以及概念之间的相互联系。本体模型已经被用在网络安全模型研究。通用网络安全本体模型如图 2-2 所示。

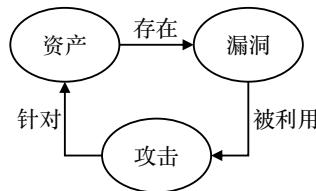


图 2-2 通用网络安全本体模型

Figure 2-2 General Network Security Ontology Model



结合本章前几节所介绍的车联网的“云”、“管”、“端”三层架构及其组成，以及车联网的安全风险与需求，本节使用 Protégé 和 OWL 语言构建车联网安全本体模型，如图 2-3 所示。其中的安全本体类及具体语义如下。

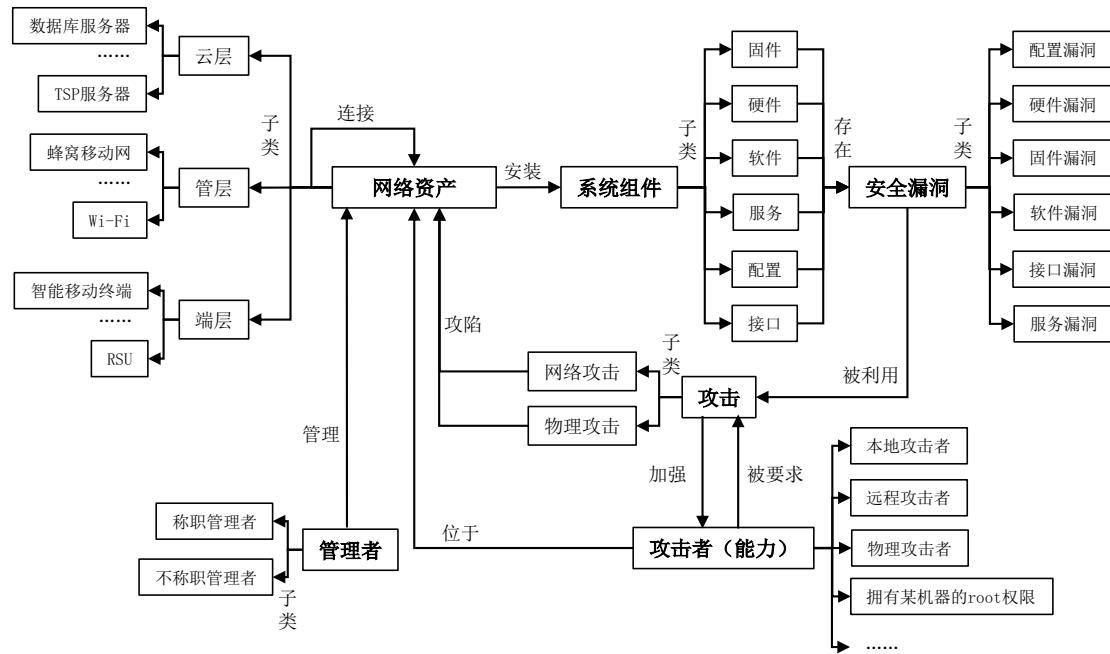


图 2-3 V2X 车联网安全本体模型

Figure 2-3 V2X IoV Security Ontology Model

- **网络资产 Asset:** 包含车联网中云、管、端各层次的网络资产。分为云层 Cloud\_Layer、管层 Channel\_Layer、端层子类 Device\_Layer。云层子类又分为数据库服务器 DBserver、Web 服务器 Webserver、与工作站 Workstation 等子类，它们是云平台网络中常见的服务器设备；管层子类又分为蜂窝网络通信 Cellular、DSRC 通信、LTE-V2X 通信、与 WLAN 通信等子类，它们是车联网中的通信方式；端层子类又分为智能移动终端 Mobileterminal、RSU、与车辆 Vehicle 等子类，它们是车联网的终端通信节点。
- **管理者 Principal:** 指车联网中的网络资产的所有者，即在网络资产中有账号的用户实体。可分为称职管理者类 CompetentPrincipal 与不称职管理者类 IncompetentPrincipal。称职管理者具有高度的安全意识，严格遵守网络安全操作规范和安全规则，且不会轻易被社会工程学攻击攻陷；而不称职者安全意识不高，经常有违反网络安全操作规范和安全规则的行为，如在其管理的所有机器中使用同一个密码，随意打开网页，随意插入设备，容易被社会工程学攻击攻陷。
- **系统组件 Component:** 一个车联网信息系统中具有的组件。分为固件 Firmware、硬件 Hardware、程序 Program、与服务 Service 等子类，它们都是车联网网络资产中常见的组件类型。
- **安全漏洞 Vulnerability:** 漏洞本体类包含有远程提权漏洞类 RemotePrivEscalation、本地提权漏洞类 LocalPrivEscalation、客户端远程提权漏洞类 RemoteClientPrivEsca、物理近程信息泄露漏洞类 PhyShortInforLeak、物理本地验证绕过漏洞类 PhyLocalAttack 等子类。这些漏洞子类分别描述的是常见的车联网漏洞利用的结果。值得注意的是，这里的安全漏洞类仅指狭义的、位于软件或硬件上的安全漏洞，并不包括配置漏洞、

管理漏洞等人为造成的网络漏洞，但是这些漏洞仍可以通过本文的车联网安全本体模型由多个本体类和类间关系表达出来。

- **攻击 Attack**: 攻击者进行攻击的方式。分为网络攻击 CyberAttack 与物理攻击 PhysAttack 两个子类。其中 CyberAttack 包含网络远程攻击 CyberRemoteAttack、网络本地攻击 CyberLocalAttack、网络远程客户端利用攻击 CyberRemoteClientAttack 等子类，表示几种常见的车联网网络攻击的方式。而 PhysAttack 则包含物理近程攻击 PhyShortAttack、物理本地攻击 PhyLocalAttack 等子类，表示几种常见的车联网物理攻击方式。
- **攻击者（能力） Capability**: 攻击者在攻击过程的各阶段具有的能力。包含物理攻击者 PhysAttacker、远程攻击者 RemoteAttacker、本地攻击者 LocalAttacker、与拥有 root 权限的攻击者 RootAttacker 等子类。

本体类间的关系有九类，如表 2-4 所示。

表 2-4 本体类间关系  
Table 2-4 Properties between ontology classes

| 关系名             | 关联本体类                     | 语义                |
|-----------------|---------------------------|-------------------|
| exploitedBy 被利用 | Vulnerability -> Attack   | 完成某攻击需要某漏洞        |
| requiredBy 被要求  | Capability -> Attack      | 完成某攻击要求攻击者具有某能力   |
| compromise 攻陷   | Attack -> Asset           | 某攻击成功将攻陷某网络资产     |
| installed 安装    | Asset -> Component        | 某网络资产中安装有某组件      |
| vulExists 存在漏洞  | Component-> Vulnerability | 某组件中存在某漏洞         |
| connectWith 连接  | Asset -> Asset            | 某网络资产和某网络资产可以网络连接 |
| deepen 加强       | Attack -> Capability      | 某攻击将加深某攻击者的能力     |
| hasAccount 管理   | Principal -> Asset        | 某管理者实体管理某网络资产     |
| locateIn 位于     | Capability -> Asset       | 某攻击者在某网络资产上有某种能力  |

### 2.3.2 车联网安全本体模型的攻击场景用例

本小节构造攻击场景用例来验证车联网安全本体模型的正确性与可用性。该攻击场景用例为一个典型的横跨“云”、“管”、“端”三层的车联网系统。在该车联网系统内，有一个在互联网远端的网络远程攻击者 A1，和一个在道路上的物理近程攻击者 A2。云平台中有一台 Web 服务器 Webserver，和一台数据库服务器 DBserver，端层有两辆智能网联汽车 V1 和 V2，两个智能移动终端 M1 和 M2。其中 Webserver 运行 HTTP 服务，可由外部或内部访问者访问；DBserver 运行 SQL 服务，可由外部或内部访问者访问；Webserver 和 DBserver 由同一个网络管理员实体 Manager 管理，该网络管理员实体在两台机器上都具有用户权限，且该管理员缺少安全意识，是不称职者。V1 和 V2 均存在使用车载操作系统的浏览器访问云平台 Webserver 网页的行为；M1 和 M2 分别可以通过 APP 用无线连接控制 V1 和 V2，且车主存在使用 M1 和 M2 访问 Webserver 以及用 APP 控制 V1 和 V2 的行为。车联网系统内的管理者、网络资产、系统组件、安全漏洞及它们之间的关系如表 2-5 所示。

根据构造的车联网系统，我们可以很容易地分析出几条攻击路径，如图 2-4 所示。

1. 互联网远程攻击者通过互联网，利用 DBserver 的 SQL 服务程序中的远程提权漏洞实例 CVE-2015-1761 获得 SQL 程序的当前权限，并能以该权限执行任意代码，从而

表 2-5 攻击场景信息

Table 2-5 Security information of attack test scenario

| 管理者     | 网络资产      | 系统组件       | 安全漏洞           | 漏洞利用结果   |
|---------|-----------|------------|----------------|----------|
| Manager | Webserver | Web 服务     | CVE-2019-0841  | 本地提权     |
| Manager | DBserver  | SQL 服务     | CVE-2015-1761  | 远程提权     |
| V1owner | V1        | Chrome 浏览器 | CVE-2021-21220 | 远程客户端提权  |
| V1owner | V1        | Android 系统 | CVE-2016-6728  | 本地提权     |
| V1owner | Wi-Fi     | Wi-Fi 接口   | CVE-2018-11477 | 物理近程信息泄露 |
| V2owner | V2        | OBD 接口     | CVE-2020-29440 | 物理本地验证绕过 |
| V2owner | V2        | 钥匙链        | CVE-2020-29438 | 物理近程信息泄露 |

可以诱导缺少安全意识的网络管理员 Manager 给出其在 Webserver 上的用户密码信息。因此，攻击者通过从 DBserver 访问 Webserver，也获得了 Webserver 上的用户权限，并能以该权限执行任意代码。此时攻击者再利用 Webserver 的 HTTP 服务程序的本地提权漏洞实例 CVE-2019-0841 进行提权，获得 Webserver 的 root 权限，此时攻击者在 Webserver 上能以 root 权限执行任意代码。V1 通过车载操作系统的 Chrome 浏览器访问 Webserver 的 HTTP 服务时，攻击者可通过 V1 车载操作系统的 Chrome 浏览器的远程客户端提权漏洞实例 CVE-2021-21220 获得 V1 车载操作系统的用户权限，进一步通过 V1 的本地提权的漏洞实例 CVE-2016-6728 获得 V1 车载操作系统的 root 权限，完全控制车辆。

2. 物理近程攻击者通过 V2 的钥匙链的物理近程信息泄露漏洞实例 CVE-2020-29438 解析获得 V2 的开锁钥匙，打开车门进入 V2 车内，成为物理本地攻击者。再通过 V2 的 OBD 接口的物理本地固件验证绕过漏洞实例 CVE-2020-29440，向 CAN 总线注入命令，从而控制车辆。该攻击路线也是 2020 年 11 月 23 日 Lennert Wouters 通过组合利用特斯拉 Model X 汽车上的几个信息泄露和验证绕过漏洞控制特斯拉汽车的攻击路线。
3. 物理近程攻击者利用 M1 和 V1 的 Wi-Fi 连接中的物理近程信息泄漏漏洞实例 CVE-2018-11477 获得 V1 的开锁钥匙，打开车门进入 V1 车内，成为物理本地攻击者。

上述攻击场景中的三条攻击路径可以用本章构建的 V2X 车联网安全本体模型描述如下：

### 1. 攻击路径一

#### 第一步攻击：远程提权

初始已知安全本体与关系：

语义：系统中存在安装 SQL 服务的 DBserver 服务器，该 SQL 服务含有远程提权漏洞实例 CVE-2015-1761。有一远程攻击者 A1 在 Internet 上，Internet 可以通过网络连接到该 DBserver 服务器。

```
Server(DBserver)^Service(SQL)^ installed(DBserver,SQL)^RemotePrivEscalation(CVE-2015-1761)^vulExists(SQL,CVE-2015-1761)^RemoteAttacker(A1)^locateIn(A1,Internet)^connectWith(Internet,DBserver)
```

进行攻击：

语义：攻击者 A1 利用漏洞 CVE-2015-1761 进行网络远程攻击 Remoteprivescalation。

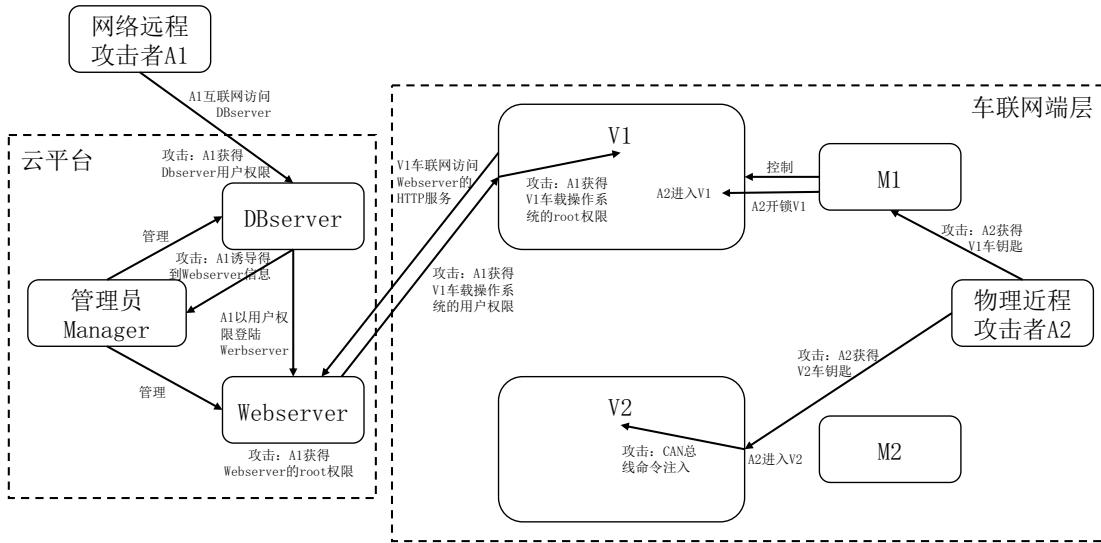


图 2-4 V2X 车联网攻击测试场景

Figure 2-4 V2X IoV Attack Scene

```
CyberRemoteAttack(Remoteprivescalation)^requiredBy(A1,Remoteprivescalation)^exploitedBy(CVE-2015-1761,Remoteprivescalation)
```

攻击结果：

语义：攻击者 A1 攻陷了服务器 DBserver，且加强了攻击者 A1 的能力，使攻击者 A1 成为在服务器 DBserver 上的本地攻击者，即得到了服务器的用户权限。

```
compromise(Remoteprivescalation,DBserver)^deepen(Remoteprivescalation,A1)^LocalAttacker(A1)^locateIn(A1,DBserver)
```

## 第二步攻击：网络管理漏洞利用

初始已知安全本体与关系：

语义：系统中存在某网络资产 DBserver 和网络资产 Webserver，网络资产 DBserver 和 Webserver 均由一个不称职的管理者 Manager 管理。有一本地攻击者 A1 在网络资产 DBserver 上，网络资产 DBserver 和 Webserver 可通过网络连接。

```
Asset(DBserver)^Asset(Webserver)^IncompetentPrincipal(Manager)^hasAccount(Manager,DBserver)^hasAccount(Manager,Webserver)^LocalAttacker(A1)^locateIn(A1,DBserver)^connectWith(DBserver,Webserver)
```

进行攻击：

语义：攻击者 A1 进行社会工程学攻击 cheat，诱导 Manager 给出他所管理的机器的账号和密码。

```
CyberSocialEngAttack(cheat)^requiredBy(A1,cheat)
```

攻击结果：

语义：攻击者 A1 攻陷了网络资产 Webserver，且加强了攻击者的能力，使攻击者 A1 成为在网络资产 Webserver 上的本地攻击者。

```
compromise(cheat,Webserver)^deepen(cheat,A1)^LocalAttacker(A1)^locateIn(A1,Webserver)
```

## 第三步攻击：本地提权

初始已知安全本体与关系：

语义：系统中存在安装了服务 HTTP 的服务器 Webserver，该服务 HTTP 含有本地提权漏洞 CVE-2019-0841。有一本地攻击者 A1 在 Webserver 上。

```
Server(Webserver)^Service(HTTP)^installed(Webserver,HTTP)^LocalPrivEscalation(CVE-2019-0841)^vulExists(HTTP,CVE-2019-0841)^LocalAttacker(A1)^locateIn(A1,Webserver)
```

进行攻击：

语义：攻击者 A1 利用漏洞 CVE-2019-0841 进行网络本地攻击 localprivup 进行内核提权。

```
CyberLocalAttack(localprivup)^requiredBy(A1,localprivup)^exploitedBy(CVE-2019-0841,localprivup)
```

攻击结果：

语义：攻击者 A1 攻陷了 Webserver，且加强了攻击者 A1 的能力，使攻击者 A1 成为在 Webserver 上的 root 权限攻击者。

```
compromise(localprivup,Webserver)^deepen(localprivup,A1)^RootAttacker(A1)^locateIn(A1,Webserver)
```

#### 第四步攻击：客户端远程提权

初始已知安全本体与关系：

语义：系统中存在某服务器 Webserver 和网络资产 V1，Webserver 安装有服务 HTTP，V1 安装有客户端程序 Chrome，该客户端有远程客户端利用漏洞 CVE-2021-21220。有一本地攻击者 A1 在 Webserver 上，V1 和 Webserver 可通过网络连接。

```
Server(Webserver)^Service(HTTP)^installed(Webserver,HTTP)^Asset(V1)^ClientSoftware(Chrome)^installed^(V1,Chrome)^RemoteClientPrivEsca(CVE-2021-21220)^vulExists(Chrome,CVE-2021-21220)^LocalAttacker(A1)^locateIn(A1,Webserver)^connectWith(V1,Webserver)
```

进行攻击：

语义：攻击者 A1 利用漏洞 CVE-2021-21220 进行客户端远程提权攻击 Remoteprivup。

```
CyberRemoteClientAttack(Remoteprivup)^requiredBy(A1,Remoteprivup)^exploitedBy(CVE-2021-21220,Remoteprivup)
```

攻击结果：

语义：攻击者 A1 攻陷了 V1，且加强了攻击者 A1 的能力，使攻击者 A1 成为在 V1 上的本地攻击者。

```
compromise(Remoteprivup,V1)^deepen(Remoteprivup,A1)^LocalAttacker(A1)^locateIn(A1,V1)
```

#### 第五步攻击：本地提权

初始已知安全本体与关系：

语义：系统中存在安装组件 Android 的网络资产 V1，该组件 Android 含有本地提权漏洞 CVE-2016-6728。有一本地攻击者 A1 在 V1 上。

```
Asset(V1)^Component(Android)^installed(V1,Android)^LocalPrivEscalation(CVE-2016-6728)^vulExists(Android,CVE-2016-6728)^LocalAttacker(A1)^locateIn(A1,V1)
```

进行攻击：

语义：攻击者 A1 利用漏洞 CVE-2016-6728 进行网络本地攻击 localprivup 进行内核提权。

```
CyberLocalAttack(localprivup)^requiredBy(A1,localprivup)^exploitedBy(CVE-2016-6728,localprivup)
```

攻击结果：

语义：攻击者 A1 攻陷了 V1，且加强了攻击者 A1 的能力，使攻击者 A1 成为在 V1 上的 root 权限攻击者，控制了 V1。

```
compromise(localprivup,V1)^deepen(localprivup,A1)^RootAttacker(A1)^locateIn(A1,V1)
```

## 2. 攻击路径二

### 第一步攻击：车钥匙提取

初始已知安全本体与关系：

语义：系统中存在智能网联汽车 V2，该汽车安装有电子钥匙链 TeslaXKeyFob，该电子钥匙链 TeslaXKeyFob 存在物理近程信息泄露漏洞 CVE-2020-29438。有一物理攻击者 A2 在 Road 上。

```
Vehicle(V2)^KeyFobs(TeslaXKeyFob)^installed(V2,TeslaXKeyFob)^PhyShortInfoLeak(CVE-2020-29438)^vulExists(TeslaXKeyFob,CVE-2020-29438)^PhyShortAttacker(A2)^locateIn(A2,Road)
```

进行攻击：

语义：攻击者 A2 利用漏洞 CVE-2020-29438 进行物理近程密钥解析攻击 Extractkey。

```
PhyShortAttack(Extractkey)^requiredBy(A2,Extractkey)^exploitedBy(CVE-2020-29438,Extractkey)
```

攻击结果：

语义：攻击者 A2 攻陷了汽车 V2，且加强了攻击者的能力，使攻击者 A2 成为在汽车 V2 的物理本地攻击者（开锁进入了车内）。

```
compromise(Extractkey,V2)^deepen(Extractkey,A2)^PhyLocalAttacker(A2)^locateIn(A2,V2)
```

### 第二步攻击：CAN 总线命令注入

初始已知安全本体与关系：

语义：系统中存在智能网联汽车 V2，该汽车安装有 OBD 接口 OBD\_v2，该 OBD\_v2 存在物理本地验证绕过漏洞 CVE-2020-29440。有一本地物理攻击者在 V2 内。

```
Vehicle(V2)^OBD(OBD_v2)^installed(V2,OBD_v2)^PhyLocalVeriBypass(CVE-2020-29440)^vulExists(OBD_v2,CVE-2020-29440)^PhyLocalAttacker(A2)^locateIn(A2,V2)
```

进行攻击：

攻击者 A2 利用漏洞 CVE-2020-29440 进行物理本地验证绕过攻击 CMDinject。

```
PhyLocalAttack(CMDinject)^requiredBy(A2,CMDinject)^exploitedBy(CVE-2020-29440,CMDinject)
```

攻击结果：

语义：攻击者 A2 攻陷了汽车 V2，且加强了攻击者的能力，使攻击者 A2 控制了汽车 V2 的 CAN 总线。

```
compromise(CMDinject,V2)^deepen(CMDinject,A2)^PhyCANController(A2)^locateIn(A2,V2)
```

## 3. 攻击路径三

### 第一步攻击：通信管道信息泄漏

初始已知安全本体与关系：

语义：系统中存在通信管道 Wifi 和智能网联汽车 V1，通信管道 Wifi 安装有组件 Interface，该组件存在物理近程通信信息泄露漏洞实例 CVE-2018-11477。Wifi 与 V1 连接。有一物理近程攻击者 A2 在 Road 上。

```
Channel(Wifi)^Component(Interface)^installed(Bluetooth,Daemons)^Vehicle(V1)^connectWith
(Wifi,V1)^PhyShortInfoLeak(CVE-2018-11477)^vulExists(Interface,CVE-2020-0022)^
PhyShortAttacker(A2)^locateIn(A2,Road)
```

进行攻击：

语义：攻击者 A2 利用漏洞 CVE-2018-11477 进行物理近程通信信息泄露攻击 Leakpassword。

```
PhyShortAttack(Leakpassword)^requiredBy(A2,Leakpassword)^exploitedBy(CVE-2018-11477,
Leakpassword)
```

攻击结果：

语义：攻击者 A2 攻陷了汽车 V1，且加强了攻击者的能力，使攻击者 A2 成为在汽车 V1 的物理本地攻击者（开锁进入了 V1 车内）。

```
compromise(Leakpassword,V1)^deepen(Leakpassword,A2)^PhyLocalAttacker(A2)^locateIn(A2,V1
)
```

可见，V2X 车联网安全本体模型可以正确描述车联网攻击测试场景的攻击链。通过上述用例可以验证，本文提出的基于本体的车联网网络安全模型可以用于形式化描述车联网系统潜在的攻击路径和方法，这也为之后构建车联网攻击图生成规则集打下了基础。

## 2.4 本章小结

本章首先介绍了车联网相关理论基础，通过分析 V2X 车联网的“云”、“管”、“端”三层架构概述了车联网的组成，并结合车联网的应用总结了车联网的四个特点。之后，本章按照“云”、“管”、“端”三层架构分析了车联网面临的安全风险，并总结了车联网中普遍的安全需求。最后，基于车联网的安全风险与需求，在通用网络安全本体模型的基础上构建了 V2X 车联网安全本体模型，通过该模型我们可以形式化地描述车联网中的实体及实体间的复杂关系。为了验证该安全本体模型的正确性和有效性，本章构造了一个典型的跨“云”、“管”、“端”三层的攻击场景测试用例；使用车联网安全本体模型正确且完整地形式化描述了该攻击场景用例中的三条攻击路径，证明了本章构建的车联网安全本体模型可以用于描述车联网系统潜在的攻击路径和方法。



## 第三章 车联网实时攻击图生成与分析的方案设计

本章首先提出了车联网实时攻击图生成与分析方案的系统模型，再详细阐述车联网实时攻击图生成与分析的方案设计，分为以下四个部分：车联网攻击图生成规则集构建、车联网实时安全信息收集、车联网实时攻击图生成与分析、基于贝叶斯攻击图的量化风险评估方案。

### 3.1 系统模型

#### 3.1.1 方案相关技术概述

##### (1) 蜂窝 V2X 通信架构

近年来，被学术界和工业界普遍接受的 V2X 车联网无线通信技术是基于蜂窝移动网络技术的车载通信（Cellular-V2X, C-V2X）。C-V2X 由第三代合作伙伴项目定义，其包含了两种通信技术与场景。一种是车、人、路之间的短距离直接通信（Device-to-Device communication, D2D），采用 PC5 接口，独立于蜂窝网络；另一种是集中式的蜂窝通信（Cellular-based communication），采用 Uu 接口，可实现长距离的可靠通信。

##### (2) 车联网边缘计算

欧洲电信标准协会（ESTI）内的 MEC 行业规范组定义了多接入边缘计算（Multi-access Edge Computing, MEC）服务器（也称为 MEC 平台）的参考框架。MEC 采用的是分布式计算模型。如图 3-1 所示，在网络边缘部署 MEC 服务平台，可以为车辆提供低延迟的服务。MEC 服务平台通过接收和分析来自车辆和路基设施的消息，能在很短的端到端延迟内发送障碍避让等时延敏感消息。而车辆可以根据收到的消息立即做出反应，使自动驾驶成为可能<sup>[38]</sup>。

##### (3) 攻击图技术概述

攻击图是一种基于模型的网络安全评估技术（Attack Modelling Techniques, AMTs）<sup>[39]</sup>。攻击者渗透网络的过程可以用一条攻击路径来表示，即由初始攻击者节点开始到目标节点的连续攻击行为。而攻击图可视化地展示了网络中所有可能被利用的路径。攻击图一般可分为属性攻击图和状态攻击图。属性攻击图生成效率相对更高，且非常直观，更适合应用于大规模网络。所以本文默认使用属性攻击图进行车联网安全分析与风险评估。

基本的攻击图生成一般由三步组成，分为安全信息标准化表示、攻击图生成推导、攻击图可视化。在实际应用攻击图技术进行网络安全分析时，进行攻击图生成前需要进行安全信息的收集，即网络安全态势感知的“数据采集”步骤；在进行攻击图生成后还需要进行对攻击图的分析，即网络安全态势感知的“态势评估”和“态势预测”步骤；而攻击图生成则是网络安全态势感知的“态势理解”步骤。

1. **安全信息收集：**在进行攻击图的生成前需要先进行安全信息的收集，依据收集到的安全信息进行对应时刻的网络安全状态的分析与评估。需要收集的安全信息有已发现的漏洞信息（如软件漏洞、硬件漏洞、系统漏洞等）、网络配置（如网络拓扑、网

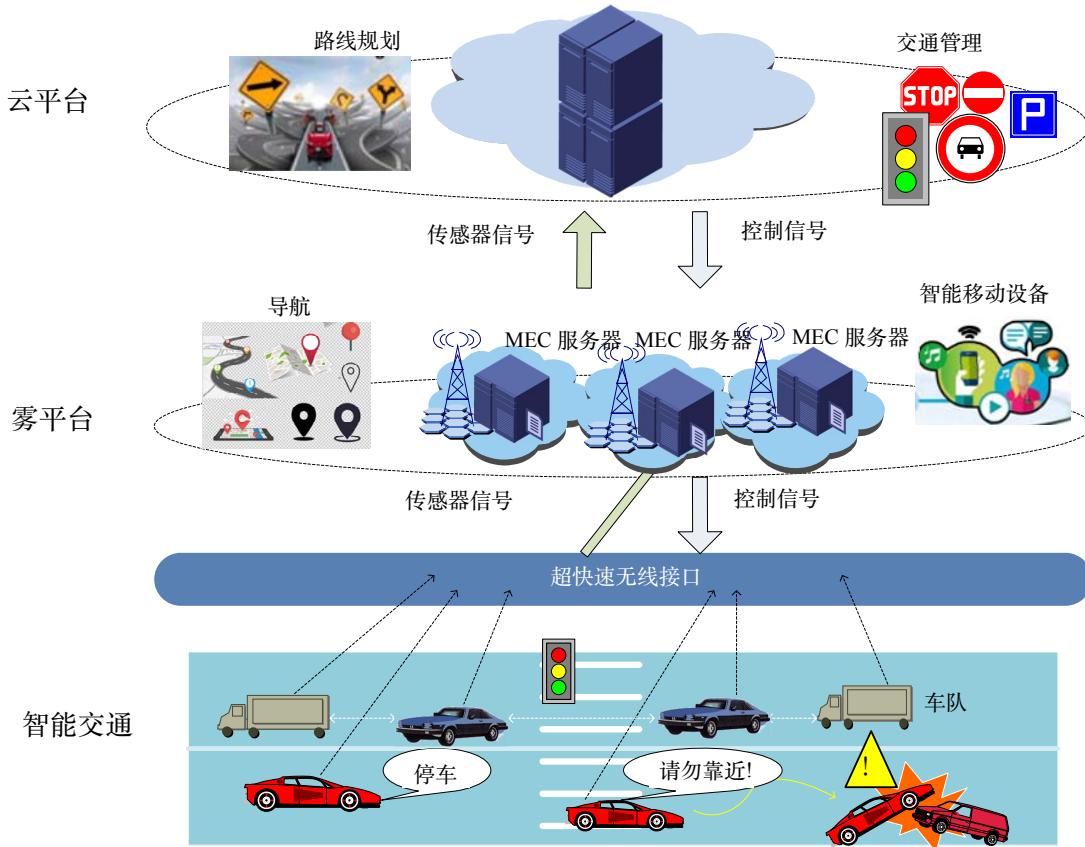


图 3-1 基于 MEC 的智能交通系统<sup>[38]</sup>

Figure 3-1 The MEC based intelligent transportation system

络服务、网络结构等)、机器配置(运行的程序、运行的服务、开启的端口等)等信息。

- 安全信息标准化表示:** 安全信息标准化表示指将安全信息按统一的标准格式进行表示,如企业网络攻击图生成工具 MulVAL 就将安全信息用 Datalog 语言的事实语句进行标准化表达,然后作为攻击图生成的输入。而运用本体模型进行攻击图生成则是先将安全信息用本体模型进行标准化表达后输入到攻击图生成流程中。
- 攻击图生成推导:** 攻击图生成推导指使用一定的推导引擎,基于已有的攻击图生成规则集,以输入的安全信息作为起点,推导出以设定的攻击目标为终点的所有可能的攻击路径。规则和推导是攻击图理论的核心。攻击图的规则系统就是由许许多多的规则组成,而攻击图的推导引擎依据规则系统自动完成攻击路径的推导。
- 攻击图可视化:** 攻击图的一大优点就是其优秀的直观性。攻击图直接把所有攻击路径展示出来,非常便于网络管理人员进行理解与分析。而攻击图生成推导得到的攻击图结果一般为标准化描述的路径,需要将其可视化为图像形式。目前不少研究使用 Graphviz 工具进行攻击图可视化。
- 攻击图的分析:** 原始攻击图一般很复杂,尤其是大型网络的攻击图。其节点众多,攻击路径众多,且常包含环路。这抵消了攻击图所具有的直观性,非常不利于阅读和理解,所以需要对原始攻击图进行分析。常见的初步分析有攻击路径的拆分、攻击路径的去环、以及攻击路径的筛选。其他的深入分析,包括对攻击图的量化分析,可

以完成网络的量化风险评估。

### 3.1.2 车联网信息安全分析与评估面临的挑战

车联网信息安全分析与评估技术面临的挑战来源于车联网的特点。结合小节 2.1.2，车联网安全分析与评估技术面临的主要挑战如下：

- **动态变化的网络拓扑**：通常，车辆以高速行驶，从而连接时间短，网络拓扑变化快。因此，使安全功能与受高速车辆影响的通信质量相适应，就成为一项艰巨的任务。特别对于网络安全态势感知来说，网络拓扑动态变化对其提出了实时性的要求。若安全分析耗时长，其结果对车联网来说也失去了意义。
- **网络规模大**：车联网巨大的网络规模对安全分析评估技术提出了很高的性能要求。由于实时性的要求，所以即使是对大规模的车联网进行安全分析评估，其耗时也不能过长，否则其结果就无法有效地衡量当前车联网的安全状态。
- **异质网络通信**：车联网异质网络通信的特点对安全分析评估技术提出了有效性的要求，即要求技术所基于的信息安全知识库足够完善，能够涵盖不同通信技术、通信协议、通信方法的安全知识，从而有效对车联网整体进行信息安全分析与评估。
- **网络复杂性**：车联网的网络复杂性在网络规模大的挑战之上提出了更高的性能要求。同时，由于车联网应用的复杂性，也对安全分析评估技术的有效性提出了更高的要求。
- **车联网自身的实时性要求**：如表 2-2 所示，车联网自身某些部分，如道路安全和交通管控等应用，对实时性要求非常高。而车载娱乐等应用对实时性要求并不高。车联网信息安全分析与评估技术应该结合车联网特点，把对车联网实时性的影响降到最低。

这些挑战使得车联网信息安全分析与评估变得困难，从而使得恶意攻击更加难以预测、预防与防御。而网络信息安全分析与风险评估又是维护网络安全必不可少的一部分。因此，本研究将基于 C-V2X 通信架构与车联网边缘计算，从车联网信息安全分析与评估技术面临的挑战出发，将传统网络安全中应用于网络安全分析的攻击图技术迁移至车联网的安全防护领域，使用攻击图技术对 V2X 车联网进行网络脆弱性分析，对车联网的实时安全状态进行量化风险评估，辅助车联网的安全管理工作。

### 3.1.3 车联网攻击图生成与分析方案系统模型

V2X 车联网实时攻击图生成与分析系统的系统模型如图 3-2 所示。该系统模型基于车联网的“云”、“管”、“端”三层架构，覆盖需要进行网络安全分析与评估的网络区域，包含三层架构中的所有安全实体。图 3-2 中表示出了云层实体，两个蜂窝服务区 C1 与 C2 及其中的端层实体，以及端层实体之间的通信连接。其中描述了一辆运动中的智能网联汽车正在从蜂窝服务区 C2 进入蜂窝服务区 C1。为了简化系统模型，我们假设只有同一个服务区内的端层通信节点才可以进行 D2D 通信，而不同服务区的端层通信节点由于距离较远，只能进行蜂窝通信。

车联网端层实体，如智能网联汽车、RSU、移动智能终端等，接收安全厂商的漏洞信息发布，或自主进行漏洞扫描，维护端层实体自身的安全信息数据库。

MEC 服务器与蜂窝基站配对（MEC 服务区和蜂窝服务区重合），利用安全信息收集方案维护服务区内的安全信息数据库，生成服务区的实时局部攻击图，并对该局部攻击图进行分析，生成局部贝叶斯攻击图并上传至云平台。

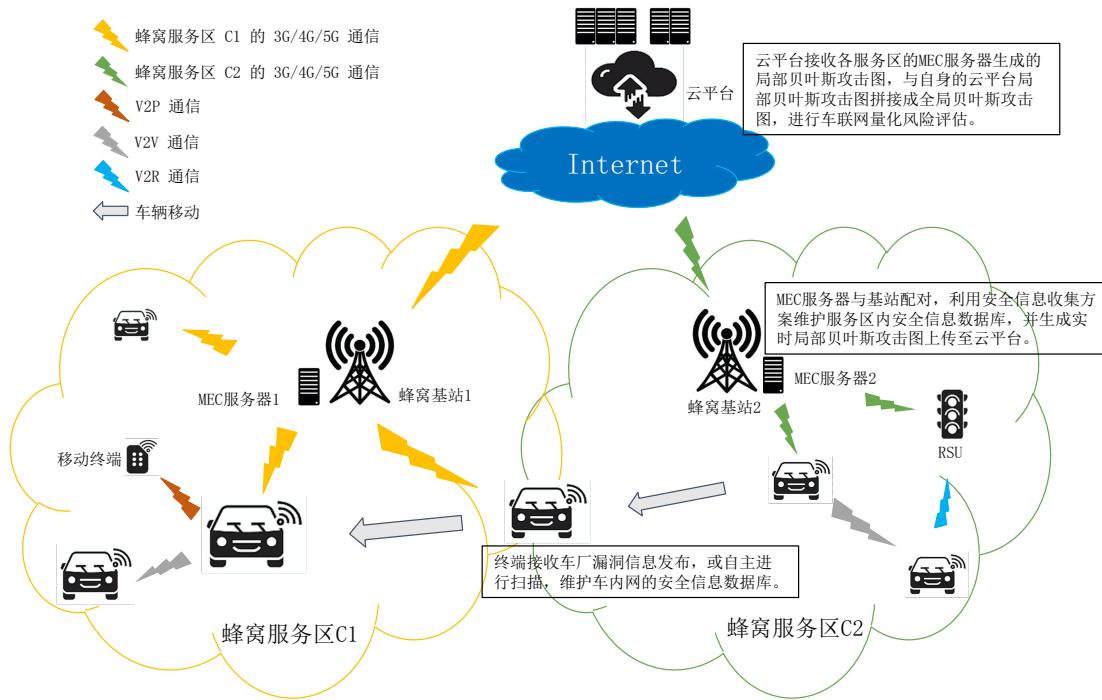


图 3-2 V2X 车联网攻击图生成与分析的系统模型

Figure 3-2 System model of attack graph generation and analysis in V2X IoV

云平台接收系统内各服务区 MEC 服务器生成的局部贝叶斯攻击图，与云平台的局部贝叶斯攻击图拼接，生成全局贝叶斯攻击图，对整个车联网的安全状态进行分析与量化风险评估。

攻击图生成与分析方案的流程分为安全信息收集、基于 MEC 的攻击图生成与分析、和全局贝叶斯攻击图的拼接三步。其中基于 MEC 的攻击图生成与分析又分为局部攻击图的生成和基于贝叶斯攻击图的量化风险评估（局部贝叶斯攻击图的生成）两部分。具体流程如图 3-3 所示。

本章提出的车联网实时攻击图生成与分析方案可以解决车联网安全分析与评估面临的主要挑战。

- 使用 MEC 方法分布式生成与分析各服务区的局部攻击图，再在云平台拼接生成量化风险评估结果，将计算成本分摊到了各 MEC 服务器，可以低延时地生成车联网全局贝叶斯攻击图，解决了车联网网络规模大的挑战，也满足了**动态变化的网络拓扑**对车联网安全分析与评估技术的时效性的要求。
- 基于车联网安全本体模型，构建普遍性强的车联网攻击图生成规则集，解决了**异质网络通信和网络复杂性**造成的挑战。
- 车联网实时攻击图生成与分析的计算和通信主要在 MEC 服务器和云平台间进行，不会对车联网终端造成负担，满足了**车联网自身的实时性要求**。

## 3.2 车联网中的攻击图生成规则集构建

如小节 3.1.1 所述，规则与推导是攻击图技术的核心。攻击图的生成是由推导引擎根据攻击图生成规则集进行推导完成的。然而，作为一种新型网络，V2X 车联网面临的安全威

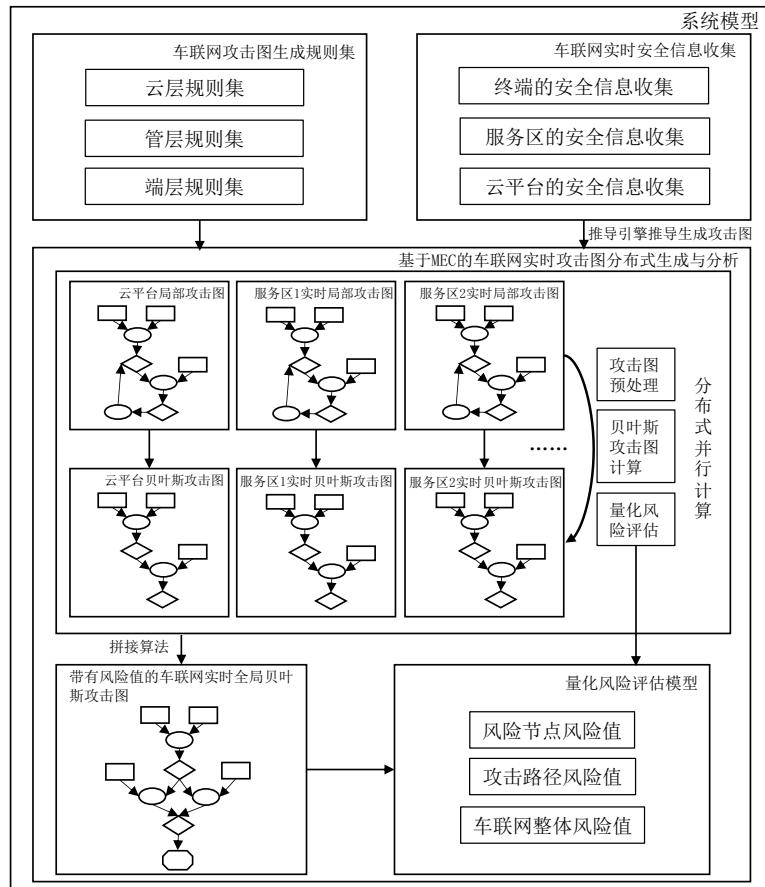


图 3-3 V2X 车联网攻击图生成与分析流程

Figure 3-3 Process of attack graph generation and analysis in V2X IoV

胁与传统网络相比更加突出和复杂。因此，之前的研究中所使用的传统网络的攻击图生成规则集已经难以适用于 V2X 车联网。需要构建新的车联网攻击图生成规则集来进行正确的攻击图的推导生成。

攻击图生成规则即为单步攻击的形式化描述，所以攻击图生成规则其实是车联网安全本体模型中“攻击”实体以及“被利用”、“被要求”、“攻陷”、“加强”关系的形式化描述。小节 2.3.2 中已经验证，本文构建的 V2X 车联网安全本体模型可以正确地形式化描述车联网系统中的攻击路径和方法。所以，通过使用车联网安全本体模型形式化描述车联网安全知识库中的攻击方法和漏洞利用方法，本节分别对云层、管层、端层的攻击图生成规则集进行构建。我们将在小节 4.1.2 将本节的原始规则集转化为 Datalog 语言描述的攻击图生成规则集。

本文构建攻击图生成规则集所基于的车联网安全知识库为车联网安全研究组收集和维护的车联网漏洞数据库<sup>[40]</sup>，包含 200 多条典型的车联网漏洞利用信息。然而，虽然攻击种类丰富多样，但攻击图规则更注重的是简洁和普遍性，因此本文的攻击图生成规则集不关注攻击的细节，而是关注攻击的共性和攻击的结果。

### 3.2.1 云层的攻击图生成规则集

车联网云平台的攻击图生成规则集实际上就是传统网络的攻击图生成规则集。如小节 2.2.1 所述，车联网云平台作为数据中心和服务中心，其内部网络架构一般为传统网络架构。所以云平台容易遭受传统的网络攻击。车联网云平台中通信节点所具有的漏洞一般为服务器或客户端上的提权漏洞、文件服务器配置漏洞、网络管理漏洞、被安装恶意木马程序、隐私泄露漏洞等。本小节利用车联网安全本体模型描述了车联网云平台常见漏洞的利用规则和攻击规则如下。

- 远程提权漏洞利用规则。典型漏洞：CVE-2015-1761。

```
Server(?asset)^Service(?service)^installed(?asset,?service)^RemotePrivEscalation(?vuln)
    ^vulExists(?service,?vuln)^RemoteAttacker(?attacker)^locateIn(?attacker,?anywhere)^
    connectWith(?anywhere,?asset)^CyberRemoteAttack(?attack)^requiredBy(?attacker,?
    attack)^exploitedBy(?vuln,?attack)^compromise(?attack,?asset)^deepen(?attack,?
    attacker)->LocalAttacker(?attacker)^locateIn(?attacker,?asset)
```

规则语义：系统中存在安装某服务的服务器，该服务含有某远程提权漏洞。有一远程攻击者在某个位置，该位置可以通过网络连接到该服务器。该攻击者利用该漏洞进行网络远程攻击，达到的效果是攻陷了服务器，且加强了攻击者的能力。攻击的结果是使攻击者成为在服务器上的本地攻击者，即得到了服务器的用户权限。

- 本地提权漏洞利用规则。典型漏洞：CVE-2021-3156。

```
Asset(?asset)^Component(?component)^installed(?asset,?component)^LocalPrivEscalation(?vuln)
    ^vulExists(?component,?vuln)^LocalAttacker(?attacker)^locateIn(?attacker,?asset)^
    ^CyberLocalAttack(?attack)^requiredBy(?attacker,?attack)^exploitedBy(?vuln,?attack)^
    ^compromise(?attack,?asset)^deepen(?attack,?attacker)->RootAttacker(?attacker)^
    locateIn(?attacker,?asset)
```

规则语义：系统中存在安装某组件的某网络资产，该组件含有某本地提权漏洞。有一本地攻击者在该网络资产上。该攻击者利用该漏洞进行网络本地攻击，达到的效果是攻陷了服务器，且加强了攻击者的能力。攻击的结果是使攻击者成为在服务器上的 root 权限攻击者。

- 网络管理漏洞利用规则。

```
Asset(?asset1)^Asset(?asset2)^IncompetentPrincipal(?principal)^hasAccount(?principal,?
    asset1)^hasAccount(?principal,?asset2)^LocalAttacker(?attacker)^locateIn(?attacker,?asset1)^
    connectWith(?asset1,?asset2)^CyberSocialEngAttack(?attack)^requiredBy(?attacker,?attack)^
    compromise(?attack,?asset2)^deepen(?attack,?attacker)->
    LocalAttacker(?attacker)^locateIn(?attacker,?asset2)
```

规则语义：系统中存在某网络资产 1 和网络资产 2，网络资产 1 和 2 均由某不称职的管理者管理。有一本地攻击者在网络资产 1 上，网络资产 1 和 2 可通过网络连接。该攻击者进行社会工程学攻击，达到的效果是攻陷了网络资产 2，且加强了攻击者的能力。攻击的结果是使攻击者成为在网络资产 2 上的本地攻击者。

- 客户端远程提权漏洞利用规则。典型漏洞：CVE-2021-21220。

```
Server(?asset1)^Service(?service)^installed(?asset1,?service)^Asset(?asset2)^
    ClientSoftware(?browser)^installed(?asset2,?browser)^RemoteClientPrivEsca(?vuln)^
    vulExists(?browser,?vuln)^LocalAttacker(?attacker)^locateIn(?attacker,?asset1)^
    connectWith(?asset1,?asset2)^CyberRemoteClientAttack(?attack)^requiredBy(?attacker,?attack)^
    exploitedBy(?vuln,?attack)^compromise(?attack,?asset2)^deepen(?attack,?attacker)->
    LocalAttacker(?attacker)^locateIn(?attacker,?asset2)
```

规则语义：系统中存在某网络资产 1 和网络资产 2，网络资产 1 安装有某服务，网络资产 2 安装有某客户端程序，该程序有远程客户端利用漏洞。有一本地攻击者在网络资产 1 上，网络资产 1 和 2 可通过网络连接。该攻击者利用该漏洞进行客户端远程提权攻击，达到的效果是攻陷了网络资产 2，且加强了攻击者的能力。攻击的结果是使攻击者成为在网络资产 2 上的本地攻击者。

### 3.2.2 管层的攻击图生成规则集

V2X 车联网存在多种无线通信技术。由于这些无线通信方式本身也存在网络安全风险，因此 V2X 车联网也继承了这些无线通信方式具有的安全风险，如身份验证绕过、通信数据泄露和网络入侵等问题。本节利用车联网安全本体模型描述了车联网中常见的关于通信管道的漏洞的利用规则如下。

- 通信信息泄露漏洞利用规则。典型漏洞：CVE-2020-15509。

```
Channel(?asset1)^Component(?component)^installed(?asset1,?component)^Vehicle(?asset2)^
connectWith(?asset1,?asset2)^PhyShortInfoLeak(?vuln)^vulExists(?component,?vuln)^
PhyShortAttacker(?attacker)^locateIn(?attacker,?physics)^PhyShortAttack(?attack)^
requiredBy(?attacker,?attack)^exploitedBy(?vuln,?attack)^compromise(?attack,?asset1)^
deepen(?attack,?attacker)->PhyLocalAttacker(?attacker)^locateIn(?attacker,?asset1)
```

规则语义：系统中存在某通信管道，该通信管道安装有某种组件，该组件存在通信信息泄露漏洞漏洞。该通信管道与某智能网联汽车连接。有一物理近程攻击者在道路上。该攻击者利用该漏洞进行物理近程通信信息泄露攻击，达到的效果是攻陷了该汽车，且加强了攻击者的能力。攻击的结果是使攻击者成为在该汽车的物理本地攻击者（开锁进入了车内）。

- 通信验证绕过漏洞利用规则。典型漏洞：CVE-2019-9503。

```
Channel(?asset1)^Component(?component)^installed(?asset1,?component)^Vehicle(?asset2)^
connectWith(?asset1,?asset2)^PhyShortVeriBypass(?vuln)^vulExists(?component,?vuln)^
PhyShortAttacker(?attacker)^locateIn(?attacker,?physics)^PhyShortAttack(?attack)^
requiredBy(?attacker,?attack)^exploitedBy(?vuln,?attack)^compromise(?attack,?asset2)^
deepen(?attack,?attacker)->LocalAttacker(?attacker)^locateIn(?attacker,?asset2)
```

规则语义：系统中存在某通信管道，该通信管道安装有某种组件，该组件存在通信验证绕过漏洞。该通信管道与某智能网联汽车连接。有一物理近程攻击者在道路上。该攻击者利用该漏洞进行物理近程通信验证绕过攻击，达到的效果是攻陷了该汽车，且加强了攻击者的能力。攻击的结果是使攻击者成为在该汽车的网络本地攻击者（绕过验证，获得了执行任意代码的能力）。

### 3.2.3 端层的攻击图生成规则集

车联网终端所受到的安全威胁主要是移动智能终端和智能网联汽车所受到的安全威胁。由于智能网联汽车的车载操作系统与移动智能终端均可以访问互联网，所以车联网终端会面临传统的网络安全风险。而车内网的组件复杂、通信环境复杂，所以也会受到车联网独有的威胁。传统网络安全威胁的规则集即小节 3.2.1 所构建的云平台规则集，本小节不再赘述。本小节构建车联网端层实体所面临的独有威胁的攻击图生成规则。本节利用车联网安全本体模型描述了车联网端层实体常见漏洞的利用规则以及常见的攻击规则如下。

- 车联网物理近程信息泄露漏洞利用规则。典型漏洞：CVE-2020-29438。

```
Vehicle(?asset)^Component(?component)^installed(?asset,?component)^PhyShortInfoLeak(?vuln)^vulExists(?component,?vuln)^PhyShortAttacker(?attacker)^locateIn(?attacker,?physics)^PhyShortAttack(?attack)^requiredBy(?attacker,?attack)^exploitedBy(?vuln,?attack)^compromise(?attack,?asset)^deepen(?attack,?attacker)->PhyLocalAttacker(?attacker)^locateIn(?attacker,?asset)
```

规则语义：系统中存在某智能网联汽车，该汽车安装有某组件，该组件存在物理近程信息泄露漏洞。有一物理攻击者在道路上。该攻击者利用该漏洞进行物理近程密钥解析攻击，达到的效果是攻陷了该汽车，且加强了攻击者的能力。攻击的结果是使攻击者成为在该汽车的物理本地攻击者（开锁进入了车内）。

- 车联网物理近程验证绕过漏洞利用规则。典型漏洞：CVE-2018-11478。

```
Vehicle(?asset)^Component(?component)^installed(?asset,?component)^PhyShortVeriBypass(?vuln)^vulExists(?component,?vuln)^PhyAttacker(?attacker)^locateIn(?attacker,?asset)^PhyShortAttack(?attack)^requiredBy(?attacker,?attack)^exploitedBy(?vuln,?attack)^compromise(?attack,?asset)^deepen(?attack,?attacker)->PhyController(?attacker)^locateIn(?attacker,?asset)
```

规则语义：系统中存在某智能网联汽车，该汽车安装有某组件，该组件存在物理近程验证绕过漏洞。有一物理攻击者（在道路上或在车内）。该攻击者利用该漏洞进行物理近程验证绕过攻击（发送伪造的消息），达到的效果是攻陷了该汽车，且加强了攻击者的能力。攻击的结果是使攻击者控制了该汽车的 CAN 总线。

- 车联网物理本地验证绕过漏洞利用规则。典型漏洞：CVE-2020-29440。

```
Vehicle(?asset)^Component(?component)^installed(?asset,?component)^PhyLocalVeriBypass(?vuln)^vulExists(?component,?vuln)^PhyLocalAttacker(?attacker)^locateIn(?attacker,?asset)^PhyLocalAttack(?attack)^requiredBy(?attacker,?attack)^exploitedBy(?vuln,?attack)^compromise(?attack,?asset)^deepen(?attack,?attacker)->PhyController(?attacker)^locateIn(?attacker,?asset)
```

规则语义：系统中存在某智能网联汽车，该汽车安装有某组件，该组件存在物理本地验证绕过漏洞。有一本地物理攻击者（在该车内）。该攻击者利用该漏洞进行物理本地验证绕过攻击（物理接入的方式），达到的效果是攻陷了该汽车，且加强了攻击者的能力。攻击的结果是使攻击者控制了该汽车的 CAN 总线。

- 车联网网络本地验证绕过漏洞利用规则。典型漏洞：CVE-2020-8539。

```
Vehicle(?asset)^Component(?component)^installed(?asset,?component)^LocalVeriBypass(?vuln)^vulExists(?component,?vuln)^LocalAttacker(?attacker)^locateIn(?attacker,?asset)^LocalAttack(?attack)^requiredBy(?attacker,?attack)^exploitedBy(?vuln,?attack)^compromise(?attack,?asset)^deepen(?attack,?attacker)->PhyController(?attacker)^locateIn(?attacker,?asset)
```

规则语义：系统中存在某智能网联汽车，该汽车安装有某组件，该组件存在本地验证绕过漏洞。有一本地攻击者（可以某一权限在车载操作系统内执行任意代码），该攻击者利用该漏洞进行本地验证绕过攻击，达到的效果是攻陷了该汽车，且加强了攻击者的能力。攻击的结果是使攻击者控制了该汽车的 CAN 总线。

- 移动智能终端远程/近程车控 APP 验证绕过漏洞利用规则。典型漏洞：蜚语安全<sup>[41]</sup>。

```
Vehicle(?asset1)^Mobile(?asset2)^APP(?app)^installed(?asset2,?app)^connectWith(?asset1,?asset2)^LocalVeriBypass(?vuln)^vulExists(?app,?vuln)^LocalAttacker(?attacker)^locateIn(?attacker,?asset2)^PhyAttack(?attack)^requiredBy(?attacker,?attack)^exploitedBy(?vuln,?attack)^compromise(?attack,?asset1)^deepen(?attack,?attacker)->PhyController(?attacker)^locateIn(?attacker,?asset1)
```

规则语义：系统中存在某智能网联汽车和某移动智能终端，该移动终端安装有一远程/近程车控 APP，该 APP 存在验证绕过漏洞。智能网联汽车和移动智能终端能够连接。有一本地攻击者在该移动终端上（具有在该移动终端上执行任意代码的能力）。该攻击者利用该漏洞进行远程/近程验证绕过攻击，达到的效果是攻陷了该汽车，且加强了攻击者的能力。攻击的结果是使攻击者控制了该汽车。

- 移动智能终端直接控制利用规则。

```

Vehicle(?asset1)^Mobile(?asset2)^connectWith(?asset1,?asset2)^RootAttacker(?attacker) ^
    locateIn(?attacker,?asset2)^Control(?attack)^requiredBy(?attacker,?attack) ^
    compromise(?attack,?asset1)^deepen(?attack,?attacker)->PhyController(?attacker) ^
    locateIn(?attacker,?asset1)
  
```

规则语义：系统中存在某智能网联汽车和某移动智能终端。智能网联汽车和移动智能终端能够连接。有一 root 攻击者在移动智能终端上。该攻击者可以直接控制该汽车。

### 3.3 车联网的实时安全信息收集方案

如小节 3.1.1 所述，攻击图生成是以网络中的安全信息为起点，推导出以设定的攻击目标为终点的所有可能的攻击路径的过程。所以，在进行攻击图生成前，需要先进行安全信息的收集，将收集到的安全信息作为攻击图生成的输入。然而，由于 V2X 车联网的特点，原本在传统网络中不太重要的时效性成了车联网安全信息的重要性质之一。所以，车联网的实时安全信息收集也成了一个重要的问题。本节基于 C-V2X 通信架构和 MEC，提出了车联网云、管、端协同的实时安全信息收集方案。由于本文的重点在于车联网攻击图的生成和量化风险评估，因此该实时安全信息收集方案将留待以后的研究来实现验证。

#### 3.3.1 生成攻击图所需的输入信息

如小节 3.1.1 所述，攻击图生成所需输入的车联网本身的安全信息有网络配置信息、机器配置信息、漏洞信息等。而除此之外，还需要输入车联网攻击者的信息，以及车联网攻击图生成规则集。这五种信息也正是组成车联网安全本体模型的六种安全实体与九种关系的信息，对应关系如图 3-4 所示。

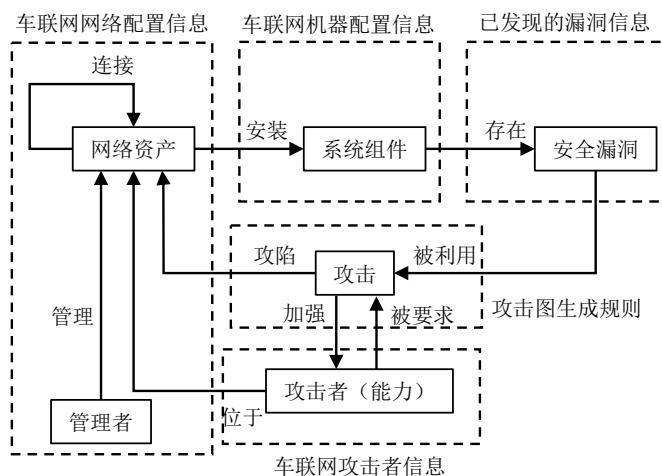


图 3-4 生成攻击图所需的输入信息

Figure 3-4 Information for Attack Graph Generation

结合车联网安全本体模型，本小节详述车联网攻击图生成所需的输入信息如下：

- **车联网网络配置信息**：车联网网络配置信息包括车联网安全本体模型的“网络资产”实体以及“网络资产”间的“连接”关系的信息，以及“管理者”实体和其与“网络资产”间的“管理”关系。包括网络拓扑（“连接”关系）、网络结构（子网划分等）、通信技术（蜂窝网络、Wi-Fi、蓝牙等，以及他们的具体版本）、通信协议（TCP、VPN、HTTP、FTP 等）、管理者账号等。
- **车联网通信节点配置信息**：车联网通信节点配置信息包括车联网安全本体模型的“系统组件”实体以及“网络资产”和“系统组件”间的“安装”关系的信息。这包括运行的程序、运行的服务、开启的接口、安装的固件、安装的硬件等，也包括“网络资产”实体“安装”“系统组件”实体的对应信息。
- **车联网已发现的漏洞信息**：车联网已发现的漏洞信息包括车联网安全本体模型的“安全漏洞”实体以及“系统组件”和“安全漏洞”间的“存在”关系的信息。这包括软件漏洞、硬件漏洞、固件漏洞、接口漏洞、服务漏洞等。
- **车联网攻击者信息**：车联网攻击者信息即车联网安全本体模型中的“攻击者（能力）”实体与攻击者“位于”什么“网络资产”的信息。在一般的攻击图技术中，攻击图生成时所需要的“攻击者”信息一般都是由网络安全管理人员设定的，本文的车联网攻击图生成也不例外。如未特殊说明，本文的“攻击者（能力）”信息均设定为“远程攻击者”与“物理攻击者”两类能力最弱的攻击者。

得到以上四种信息后，加之车联网攻击图规则集所代表的“攻击”实体和“攻陷”、“被利用”、“被要求”、“加强”关系信息，我们就能使用车联网安全本体模型来完整描述攻击路径，即进行攻击路径的生成。

### 3.3.2 端层实体的安全信息收集

依据小节 3.1.3 所提出的车联网攻击图生成系统模型，车联网端层实体，如移动智能终端、智能网联汽车、RSU 等，接收安全厂商的漏洞信息发布，或自主进行漏洞扫描，收集机器配置信息、已发现的漏洞信息和部分网络配置信息（通信技术和通信协议等，不包括网络拓扑和网络结构），分别维护端层实体自身的安全信息数据库。

### 3.3.3 蜂窝服务区的安全信息收集

系统模型中，MEC 服务区和蜂窝服务区重合。蜂窝服务区内的车联网端层实体上传自己的实时安全信息数据库给网络边缘的 MEC 服务器。各 MEC 服务器汇总服务区的端层实体的实时安全信息，组成各服务区子网的实时安全信息数据库。

对于智能网联汽车和移动智能终端，MEC 服务器维护服务区子网的实时安全信息数据库的流程如下：

1. 终端接入新蜂窝网后，立即与该蜂窝网 MEC 服务器建立 TCP 连接。
2. 终端与 MEC 服务器相互进行证书验证并协商密钥，之后的信息交换均在加密状态下进行。
3. MEC 服务器查询本地数据库中是否有该终端的相关条目：若没有，则请求终端发送终端的车辆当前安全信息集；若有，则将相关条目生效，并请求终端发送终端当前安全信息集的哈希值与本地的该终端的相关安全条目的哈希值进行对比，若不同则继续请求完整的终端当前安全信息集。

4. 若终端当前安全信息发生改变，则向 MEC 服务器请求更新对应安全条目并收到确认后，发送更新的条目，MEC 服务器进行相应条目更新。
5. 终端与 MEC 服务器建立安全连接后，每隔一段时间向 MEC 服务器发送状态信号（上锁、静止、或移动中）并接收确认信号；若连续三次 MEC 服务器未收到该终端的确认信号，则视为与该终端断开连接；若终端连续三次未收到 MEC 服务器的确认信号，则视为与 MEC 服务器断开连接。
6. 终端离开一个蜂窝服务区时，与该服务区的 MEC 服务器断开连接。MEC 服务器将断开连接的终端的对应条目为失效，并在一段时间后，若终端未再次建立连接，MEC 服务器删除对应条目。

通过维护蜂窝服务区子网的实时安全信息数据库，本方案能保证 MEC 服务器所存储的安全信息始终是实时的。

对于端层其他实体，由于其不具有和智能网联汽车一样的移动性，因此仅在自身安全信息变化时将新的安全信息上传至本服务区边缘的 MEC 服务器。

除汇总服务区内的车联网端层实体的实时安全信息之外，为了能在服务区 MEC 平台内进行局部攻击图的生成与分析，以及之后在云平台中使用拼接算法拼接生成全局攻击图，云平台更新自己的局部贝叶斯攻击图后，将会把攻击者在云平台中的某些特殊通信节点上能得到的最强能力，以及获得该能力的概率，发送至车联网所有服务区的 MEC 服务器中，作为“特殊攻击者（能力）”信息参与服务区实时局部攻击图的生成与分析，作为云平台攻击图与服务区攻击图的纽带，用于之后将局部贝叶斯攻击图拼接为全局贝叶斯攻击图。这些特殊通信节点一般为车联网端层实体可以访问到的服务器，或者可以连接到端层实体的服务器。

车联网安全管理人员也可以选择将整张云平台局部攻击图或云平台安全信息全部发送给各服务区 MEC 服务器，但这样不符合风险隔离的安全原则，会使云平台的安全将受到极大威胁。这是因为，只要一台 MEC 服务器被攻陷，云平台的全部安全信息甚至攻击路径都将暴露给攻击者。攻击者得到云平台攻击图后，可以轻易地攻陷云平台。所以方案设计中仅将“特殊攻击者（能力）”节点及其概率下发到各 MEC 服务器。这样，即使某一个服务区的 MEC 服务器被攻陷，攻击者得到的也只是该服务区的安全信息以及云平台的“特殊攻击者（能力）”信息，对其攻击云平台帮助不大。通过该方法，在提升攻击图生成与分析效率的同时，将攻击者获取网络安全信息的风险隔离在了各个服务区内。

### 3.3.4 云平台的安全信息收集

车联网云平台类似于传统的数据中心网络，其很少有拓扑变化，网络配置变化少，且机器配置变化也很少。其主要变化的安全信息只有漏洞信息。所以其安全信息时效性的优先度不高，可以按照传统网络的安全信息收集方式进行收集。同时，云平台收集各服务区的局部贝叶斯攻击图也可看作是在收集整个车联网的安全信息。

## 3.4 基于 MEC 的车联网实时攻击图生成与分析方案

车联网巨大的网络规模和网络复杂性对车联网攻击图生成与分析技术的性能提出了很高的要求，这要求攻击图的推导生成与量化分析需要有较低的时间复杂度，且在车联网网络规模继续增大的现状下仍能维持低延迟和高性能。因此，本章提出了基于 MEC 的车联网

实时攻击图分布式生成与分析的方案。其中，实时攻击图的分析使用了基于贝叶斯攻击图的量化风险评估方案，由于该方案较为复杂，将在章节 3.5 单独详细介绍。

本文的车联网实时攻击图生成与分析方案与 2015 年 Kaynar 和 Sivrikaya 设计的分布式攻击图并行生成算法<sup>[19]</sup>的思想类似。但不同的是，本文将分布式并行计算的思想与车联网的 C-V2X 通信架构以及 MEC 技术相结合，使得攻击图的分布式并行生成能适用于车联网。另外，本文在攻击图的分布式并行生成之上，还加入了攻击图的分布式并行分析，进一步提高了网络安全分析与评估的效率。

### 3.4.1 云平台的局部攻击图生成与分析

正如前文所述，车联网云平台类似于传统的数据中心网络，其网络中的变化很少，主要的安全信息变化是漏洞信息的变化。所以，车联网云平台局部攻击图很少有对时效性的要求。基于以上原因，车联网云平台的局部攻击图生成与分析结果采用固定时间间隔进行更新的方法。

在生成或更新车联网云平台的局部攻击图时，使用推导引擎，输入车联网攻击图生成规则集，推导生成云平台攻击图。然后使用量化风险评估方案将攻击图转化为贝叶斯攻击图，得到局部量化风险评估结果。在更新完成后到下一次更新前的这一段时间里，车联网云平台的局部贝叶斯攻击图将存储在云平台中，供以拼接生成全局贝叶斯攻击图。

云平台更新自己的局部贝叶斯攻击图后，将会把攻击者在云平台中某些特殊通信节点上能得到的最强能力及其概率发送至车联网所有服务区的 MEC 服务器中，作为“特殊攻击者（能力）”信息参与车联网各服务区局部贝叶斯攻击图的生成。

### 3.4.2 蜂窝服务区的局部攻击图生成与分析

车联网服务区的局部贝叶斯攻击图生成是整个车联网实时攻击图分布式生成与分析方案的核心。各服务区的局部贝叶斯攻击图由每个服务区的 MEC 服务器负责实时生成。

每个 MEC 服务器使用实时安全信息收集方案收集本服务区的实时安全信息（包括来自云平台的“特殊攻击者（能力）”信息）。若发现本服务区的安全信息发生改变，如某条安全信息被删除、某条安全信息失效、或某条安全信息重新生效等，则立即使用推导引擎，输入实时安全信息（已标准化）和车联网攻击图生成规则集，生成实时局部攻击图。再使用量化风险评估方案，将攻击图转化为贝叶斯攻击图。换句话说，服务区的安全信息变化将触发 MEC 服务器更新局部贝叶斯攻击图。服务区的局部贝叶斯攻击图更新后，将立即上传至云平台以供更新全局贝叶斯攻击图。

由于各服务区的 MEC 服务器是分布式独立生成局部贝叶斯攻击图的，因此生成整个车联网各服务区子网的所有局部贝叶斯攻击图的平均时间复杂度接近单个服务区子网的时间复杂度。若单个服务区子网的网络规模不变，生成局部贝叶斯攻击图和全局贝叶斯攻击图的时间复杂度将为常数，不论车联网的整体规模有多大，攻击图的生成延迟都很低。

### 3.4.3 车联网实时量化风险评估结果的生成

车联网的某个 MEC 服务器更新局部贝叶斯攻击图后，会立刻将更新的贝叶斯攻击图以文本的形式，如 XML 格式，上传至云平台。云平台接收到更新的服务区局部贝叶斯攻击图后，将立刻进行全局贝叶斯攻击图的更新。其更新并不像局部贝叶斯攻击图那样使用攻击

图推导引擎进行推导后再进行分析，而是直接将已保存在云平台的全局贝叶斯攻击图与新的服务区局部贝叶斯攻击图使用拼接算法进行拼接。

**基于拼接的全局贝叶斯攻击图生成算法。** 如前文所述，攻击者在云平台中的某些特殊通信节点上能得到的最强能力及其概率将被发送至各 MEC 服务器，作为“特殊攻击者（能力）”信息参与各服务区局部贝叶斯攻击图生成。因此，这些“特殊攻击者（能力）”便成为了云平台局部贝叶斯攻击图与各服务区局部贝叶斯攻击图间的纽带。特殊“攻击者（能力）”作为最深节点存在于云平台局部贝叶斯攻击图中，也作为一种“攻击者”节点，即起始节点，存在于各服务区局部贝叶斯攻击图中。因此，可以通过将云平台局部贝叶斯攻击图和服务区局部贝叶斯攻击图的“特殊攻击者（能力）”节点进行一一对应的合并，从而实现云平台局部贝叶斯攻击图和服务区局部贝叶斯攻击图的拼接。另外，若已经存在全局贝叶斯攻击图，并且只是需要更新全局贝叶斯攻击图中某个服务区的部分，可以通过只删除全局贝叶斯攻击图中的该服务区部分，并拼接入该服务区的最新局部贝叶斯攻击图，进行局部更新，而不需要改变攻击图的其他部分，进一步减小了计算复杂度。局部拼接更新算法如 3-1 所示。

---

### 算法 3-1 全局贝叶斯攻击图局部拼接更新算法

**Data:** 全局贝叶斯攻击图 fullgraph, 旧的局部攻击图 graph, 新的局部攻击图 newgraph, “特殊攻击者”节点集 pvlist

**Result:** 新的全局贝叶斯攻击图 newfullgraph

```

1 newfullgraph = fullgraph;
2 newfullgraph.remove(graph);
3 newfullgraph.append(newgraph);
4 for node in newgraph.pvlist do
5     newfullgraph.remove(node);
6     for nod in node.next do
7         nod.prior.remove(node);
8         node.next.remove(nod);
9         for nd in newfullgraph.pvlist do
10            if nd.fact == node.fact then
11                nd.next.append(nod);
12                nod.prior.append(nd);
13            end
14        end
15    end
16 end

```

---

## 3.5 基于贝叶斯攻击图的量化风险评估方案

由于车联网规模巨大且复杂度高，其生成的攻击图也非常复杂，节点众多，攻击路径众多，且常包含环路。这抵消了攻击图所具有的直观性，非常不利于阅读和理解。且单纯的攻击图只能展示出攻击者可能利用的攻击路径，并不能量化地表示车联网的安全风险，所以需要对原始攻击图进行进一步的分析和量化。

本文对各局部攻击图的分析以及对车联网整体的量化风险评估使用了基于贝叶斯攻击图的量化风险评估方案，本节将详细介绍该方案。与王秀娟等人于 2015 年提出的贝叶斯属

性攻击图网络脆弱性评估<sup>[20]</sup> 方案相比，本文的消环算法基于最深原子攻击消环的思想，且本文在贝叶斯攻击图之上提出了基于风险节点风险值的量化风险评估模型。

### 3.5.1 攻击图的预处理

在攻击图推导中经常会出现攻击环路，这是因为若攻击者获得了更强的能力，那其自然就能再去获得已经获得过的能力，但这对攻击者来说是毫无意义的。这也是逻辑推导生成攻击图的一个常见的缺陷。对于安全人员来说，攻击环路也是无用的路径，因为其对预测攻击与评估风险没有帮助，反而会增大攻击图的复杂度，使攻击图难以阅读。因此，攻击环路不便于对攻击图的分析和理解。所以在对攻击图进行分析前，首先要进行消环的预处理。

**最深原子攻击消环思想。** 合适的攻击图消环算法必须控制其对攻击图的影响。最理想的消环不会影响网络安全分析与评估的结果。由于攻击图是用以发现网络中可能被攻击者利用的攻击路径的技术，因此若通过移除攻击者利用可能性最小的原子攻击来消去攻击图中的环路，就能将消环对攻击图的影响降至最低。在一条攻击路径中，攻击者最不可能利用的原子攻击是路径中最深的原子攻击，因为进行该原子攻击需要成功完成之前的一整条攻击路径。所以，攻击者成功进行路径中最深的原子攻击的概率最低，切除最深原子攻击节点的消环方法对网络安全风险评估的影响最小。

根据上述思想，本文提出了基于深度优先搜索（Depth-First-Search, DFS）的最深原子攻击消环算法。该算法的输入是一个原始攻击图。首先初始化一个栈用来存放已经经过的DFS 路径。从攻击者节点，即攻击图入口节点开始进行 DFS。DFS 访问栈顶节点的子节点列表，将访问到的子节点压入栈中，并删除子节点列表中被压入栈的节点。在 DFS 的过程中，当发现将要压入栈的节点在栈中已经存在时，说明栈中已经存储了一个环路。该环路的入口节点便是该已存在的节点。此时删除栈中最晚出现的攻击节点，并把栈中该节点之上的节点全部出栈，即可消除环路。当遍历到的栈顶节点不存在子节点时，将栈顶节点出栈。当栈为空时，则以该攻击者节点为起点的 DFS 完成，从下一个新的攻击者节点开始按同样的方法 DFS。直到所有的攻击者节点都遍历完，则整个攻击图消环完成。此时得到一个无环的攻击图。但需要注意的是，该算法正确的前提是 DFS 的路径与实际的攻击路径相同，否则该算法需要进一步修改。在之后的小节 4.1.4 中，由于使用了 MuVAL 推导引擎生成攻击图，我们发现 DFS 的路径和攻击图推导时的路径可能不一致，因此对该算法进行了修改。

### 3.5.2 贝叶斯攻击图的计算

贝叶斯攻击图由属性攻击图转化而来。贝叶斯网络是一个有向无环图，具有因果关系和概率语义，可以根据已知的信息对未知的结果或原因进行推理和预测，因此可以将贝叶斯网络与去环后攻击图进行结合，对网络的脆弱性进行量化评估，即贝叶斯攻击图<sup>[20]</sup>。

计算贝叶斯攻击图时，除了需要输入无环的属性攻击图外，还需要得到各初始属性节点（网络的安全信息）的概率，即叶子节点被利用成功的先验概率，才能计算出一次原子攻击的成功概率，并接着进行攻击路径概率的贝叶斯概率计算。由于一次原子攻击中，最关键的是漏洞的利用，且其他的安全信息与预备步骤大多是为漏洞利用服务的，因此本文不考虑漏洞信息和“特殊攻击者（能力）”以外的叶子结点的先验概率计算，而是将它们都默认设为 100%，即攻击者成功实施漏洞信息利用以外的动作的成功概率为 100%。“特殊攻击者

(能力)”的概率由云平台下发，所以已知。漏洞信息节点的利用成功概率可以使用 CVSS 评分计算得到。

CVSS 是一个行业公开标准，被设计用来评测漏洞的严重程度。CVSS 使用三个指标来描述漏洞的利用难度：访问向量（Access Vector, AV）、访问复杂性（Access Complexity, AC）、认证方式（Authentication, AU）。它们的度量值都可分为三个等级：低、中、高。分值越高，该漏洞越容易利用。由低到高，AV 值依次为 0.359、0.646、1.0；AC 值依次为 0.35、0.61、0.71；AU 值依次为 0.45、0.56、0.704<sup>[17]</sup>。不失一般性地，我们基于 CVSS 用公式 3-1 计算一条漏洞信息被攻击者利用成功的概率。

$$P = 2 \cdot AV \cdot AC \cdot AU, \quad 0 < P < 1 \quad (3-1)$$

计算得到漏洞信息被攻击者利用成功的概率后，我们通过攻击路径的节点间的父子因果关系和逻辑关系进行攻击路径的攻击成功概率的计算，得到贝叶斯攻击图。

### 3.5.3 量化风险评估模型

得到攻击路径的贝叶斯概率只能量化各攻击路径的风险，并不足以对车联网整体进行量化风险评估。所以我们在贝叶斯攻击图的基础上，提出一种基于风险节点风险值的车联网量化风险评估模型。

攻击图中，攻击者经过原子攻击后可以得到的能力有很多种，但从车联网面临的主要安全风险的角度考虑，对车联网产生直接威胁的能力有控制某辆车、在某网络资产的系统内以某一权限执行任意代码、以及进入某辆车等几种，我们把代表这些能力的节点作为攻击图中的特殊节点，称为“风险节点”，并基于风险节点对车联网进行量化风险评估。

我们使用风险节点“风险值”来量化表示该风险节点能对车联网造成的直接损害，用攻击路径“风险值”来量化表示该攻击路径能对车联网造成的直接损害，用车联网整体的“风险值”来量化表示车联网所可能收到的最大损害，即量化风险。攻击路径“风险值”是该条攻击路径上所有风险节点“风险值”之和，车联网整体“风险值”等于车联网全局贝叶斯攻击图中的所有风险节点的“风险值”之和。我们用  $P(x)$  表示攻击者成功得到风险节点  $x$  的概率，也即以其为末尾节点的攻击路径的贝叶斯概率；用  $num(x.next)$  表示风险节点  $x$  的出度。经验性地，我们给出风险节点  $x$  的风险值  $Risk(x)$  的计算方法，如算式 3-2 所示。

$$Risk(x) = a_x \cdot P(x) \cdot (1 + num(x.next)) \quad (3-2)$$

算式 3-2 中，危害系数  $a_x$  量化表示了风险节点本身对车联网安全的公认的危险程度，如控制一辆车的危害性大于进入一辆车，则前者的危害系数大于后者的危害系数； $(1 + num(x.next))$  是对风险节点在攻击图中的重要性的量化表示，因为风险节点的出度等于从该风险节点能继续延伸出的攻击路径的数量；风险节点概率  $P(x)$  则量化表示了得到该节点的难易程度，得到该节点越容易，车联网也就越容易受到危害，节点对车联网造成的风险也就越大。

云平台和各 MEC 服务器计算局部贝叶斯攻击图时，也将同时计算攻击图中的风险节点的风险值。拼接局部贝叶斯攻击图时，除了“特殊攻击者（能力）”节点需要重新计算风险值外（因为“特殊攻击者（能力）”节点在拼接生成全局贝叶斯攻击图后其出度一般会增大），其他风险节点的风险值不变。这种分布式算法将风险值的计算成本也分摊到了各服务区中。

### 3.6 本章小结

本章提出了车联网攻击图生成与分析方案的系统模型，该模型基于车联网三层架构和 C-V2X 车联网通信架构，利用车联网边缘计算，通过车联网实时安全信息收集、实时攻击图分布式生成与分析、全局贝叶斯攻击图的拼接三步，来得到车联网实时贝叶斯攻击图，并据此对车联网进行量化风险评估。

首先，使用车联网安全本体模型，形式化描述已有的车联网安全知识库中的攻击方法，构建了车联网攻击图生成规则集，这是推导生成攻击图的核心。其次，基于 C-V2X 通信架构和 MEC 架构，从端层实体、蜂窝服务区、云平台三个层面阐述了车联网的实时安全信息收集方案。再次，基于系统模型设计了基于 MEC 的实时攻击图分布式生成与分析方案，并对云平台局部攻击图、蜂窝服务区局部攻击图、以及全局贝叶斯攻击图的生成与分析过程进行了详细介绍。最后，进一步对基于贝叶斯攻击图的量化风险评估方案进行了单独地详细介绍。该方案分为三步，第一步为攻击图的预处理，即攻击图的去环；第二步为基于 CVSS 的攻击路径贝叶斯概率计算；最后，依据本章提出的量化风险评估模型，基于贝叶斯攻击图来对车联网进行局部和整体的风险值计算。

## 第四章 车联网实时量化风险评估原型系统的实现与测试

### 4.1 原型系统实现

本节基于车联网实时攻击图生成与分析的方案设计，实现了车联网实时量化风险评估原型系统。本文的攻击图生成使用 MulVAL 作为推导引擎，因此本节首先对 MulVAL 进行了简要介绍。之后，将小节 3.2 中构建的原始攻击图生成规则集标准化为 MulVAL 所要求的 Datalog 语言规则集。然后详细介绍了车联网实时量化风险评估原型系统的实现，包括局部贝叶斯攻击图和全局贝叶斯攻击图的生成，以及攻击图分析所使用的基于贝叶斯攻击图的量化风险评估方案的实现。

#### 4.1.1 攻击图生成工具 MulVAL 简介

MulVAL (multihost, multistage, vulnerability analysis) 是由 Ou 等开发的 Linux 平台开源攻击图生成工具<sup>[16, 22]</sup>。MulVAL 对于具有  $n$  个节点的网络生成攻击图，其时间复杂度为  $O(n^2)$ 。MulVAL 使用 Datalog 语言作为模型语言（包括漏洞描述，规则描述，配置描述，权限系统等）。其将 Nessus/OVAL 扫描器报告、防火墙管理工具提供的网络拓扑信息、网络管理员提供的网络管理策略等转化为 Datalog 语言的事实作为输入，交由内部的推导引擎进行攻击过程推导。推导引擎由 Datalog 攻击图生成规则组成，这些规则捕获操作系统行为和网络中各个组件的交互。最后由可视化工具 graphviz 将推导引擎得到的攻击树可视化形成攻击图。一个攻击图的例子如图 4-1 所示，其描述了从互联网通过远程提权漏洞攻陷内网的一台 Web 服务器的过程。

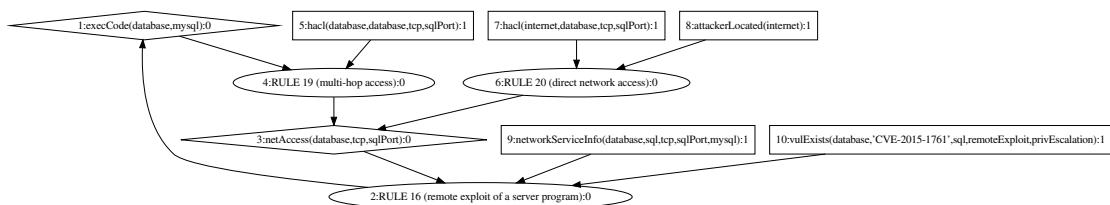


图 4-1 一个 MulVAL 攻击图的例子  
Figure 4-1 An example of MulVAL attack graph

可视化的攻击图中，有三类节点：

- 方形节点为 LEAF 型节点，即叶子节点，是一种属性节点，可用作原子攻击的条件。其内容为攻击图生成所需的网络安全信息。从本文构建的车联网安全本体模型的视角看，其包含了“网络资产”、“系统组件”、“安全漏洞”、“用户实体”和初始“攻击者（能力）”实体以及它们之间的关系。
- 椭圆节点为 AND 型节点，即“与”节点，又称原子攻击节点，其表示单步攻击。在 MulVAL 的推导引擎中，它又表示单步基于攻击图生成规则的推导，是攻击图生成规则的实例。其为真的条件是其所有父节点均为真，即该原子攻击的所有条件都满足才能进行一步原子攻击。从本文构建的车联网安全本体模型的视角看，其包含了“攻击”实体以及“被要求”、“被利用”、“攻陷”、“加强”的关系。

- 棱形节点为 OR 型节点，即“或”节点，和 LEAF 型节点一样是一种属性节点，可用作原子攻击的条件。其表示一次原子攻击后的攻击结果，即经过原子攻击加强后的攻击者能力。其为真的条件是任一父节点为真，即任一父节点的原子攻击成功后都可得到该节点。从本文构建的车联网安全本体模型的视角看，其包含了攻击过程中的“攻击者（能力）”实体。

#### 4.1.2 车联网攻击图的 Datalog 规则集构建

本小节介绍从章节 3.2 的原始规则集转化来的 Datalog 语言攻击图生成规则。其规则语义与原始规则集相同，所以不加赘述。由于原始规则集已经经过车联网安全本体模型的形式化表述，因此可以非常容易地转化为 Datalog 规则集。之后可以直接用于输入攻击图生成工具 MulVAL 生成攻击图，且方便进行修改和扩充。本小节的 Datalog 规则集并未也很难包括车联网中所有的攻击规则。本文旨在通过将主要的几种攻击方法用车联网安全本体模型形式化表述后，再转化为 Datalog 规则的方式，来指出一种标准化且流程化地得到攻击图生成规则的方法。之后的研究可以轻易地通过这种方法扩充攻击图生成规则集。

本小节展示的规则集只是规则集中的主要部分，即与漏洞利用相关的规则，而实际上规则集中的一大部分都是描述漏洞利用前的准备动作，如建立连接、访问主机等规则。

##### (1) 车联网云平台的攻击图生成规则集

本部分将车联网云平台的原始攻击图生成规则集标准化为 Datalog 语言描述的规则集。云平台的 Datalog 规则集在 MulVAL 的默认规则集中已经存在<sup>[22, 34]</sup>，即传统网络的攻击图生成规则集，此处将其中重要的几条进行总结。

- 远程提权漏洞利用规则。

```
interaction_rule(
    (execCode(H, Perm) :- 
        vulExists(H, _, Software, remoteExploit, privEscalation),
        networkServiceInfo(H, Software, Protocol, Port, Perm),
        netAccess(H, Protocol, Port)),
    rule_desc('remote exploit of a server program')).
```

- 本地提权漏洞利用规则。

```
interaction_rule(
    (execCode(Host, root) :-
        execCode(Host, _Perm2),
        vulExists(Host, _, Software, localExploit, privEscalation)),
    rule_desc('local exploit')).
```

- 网络管理漏洞利用规则。

```
interaction_rule(
    (execCode(Host, Perm) :-
        principalCompromised(Victim),
        hasAccount(Victim, Host, Perm),
        canAccessHost(Host)),
    rule_desc('When a principal is compromised any machine he has an account on will also
be compromised')).
```

- 客户端远程提权漏洞利用规则。

```
interaction_rule(
```

```
(execCode(H, Perm) :-  
    vulExists(H, _, Software, remoteClient, privEscalation),  
    hasAccount(Victim, H, Perm),  
    accessMaliciousInput(H, Victim, Software)),  
    rule_desc('remote exploit for a client program')).
```

## (2) 车联网通信管道的攻击图生成规则集

将车联网通信管道的原始攻击图生成规则集标准化为 Datalog 语言描述的规则集：

- 通信信息泄露漏洞利用规则。

```
interaction_rule(  
    (inside(V) :-  
        pair(_, V, Comm),  
        attackerLocated(physics),  
        vulExists(Comm, _, Component, phyShortExploit, infoLeak)  
    ),  
    rule_desc('Sniff infomation to unlock')).
```

- 通信验证绕过漏洞利用规则。

```
interaction_rule(  
    (execCode(V, Vuln) :-  
        pair(_, V, Comm),  
        attackerLocated(physics),  
        vState(V, locked),  
        vulExists(Comm, Vuln, Component, phyShortExploit, verifiBypass)  
    ),  
    rule_desc('Short range channel bypass to execute')).
```

## (3) 车联网终端的攻击图生成规则集

将车联网终端的原始攻击图生成规则集标准化为 Datalog 语言描述的规则集：

- 车联网物理近程信息泄露漏洞利用规则。

```
interaction_rule(  
    (inside(V) :-  
        installed(V, Component),  
        attackerLocated(physics),  
        vState(V, locked),  
        vulExists(V, _, Component, phyShortExploit, infoLeak)  
    ),  
    rule_desc('Leak the password to unlock')).
```

- 车联网物理近程验证绕过漏洞利用规则。

```
interaction_rule(  
    (control(V) :-  
        installed(V, Component),  
        attackerLocated(physics),  
        vState(V, locked),  
        vulExists(V, _, Component, phyShortExploit, verifiBypass)  
    ),  
    rule_desc('Short range bypass to control')).  
  
interaction_rule(  
    (control(V) :-  
        installed(V, Component),
```

```

inside(V),
vulExists(V, _, Component, phyShortExploit, verifiBypass)
),
rule_desc('Short range bypass to control')).  
  

interaction_rule(
(control(V) :-
    installed(V, Component),
    execCode(V, Perm),
    vState(V, _),
    vulExists(V, _, Component, phyShortExploit, verifiBypass)
),
rule_desc('Leverage the local OS to bypass to control')).  
  

interaction_rule(
(control(V) :-
    execCode(X, Perm),
    hacl(X, V, _, _),
    installed(V, Component),
    vState(V, _),
    vulExists(V, _, Component, phyShortExploit, verifiBypass)
),
rule_desc('V2X communication exploit')).
```

- 车联网物理本地验证绕过漏洞利用规则。

```

interaction_rule(
(control(V) :-
    installed(V, Component),
    inside(V),
    vulExists(V, _, Component, phyLocalExploit, verifiBypass)
),
rule_desc('Local bypass to unlock')).  
  

interaction_rule(
(control(V) :-
    installed(V, Component),
    execCode(V, Perm),
    vState(V, _),
    vulExists(V, _, Component, phyLocalExploit, verifiBypass)
),
rule_desc('Leverage the local OS to bypass to control')).
```

- 车联网车载操作系统本地验证绕过漏洞利用规则。

```

interaction_rule(
(control(V) :-
    installed(V, Component),
    execCode(V, Perm),
    vState(V, _),
    vulExists(V, _, Component, localExploit, verifiBypass)
),
rule_desc('Leverage local OS to bypass to control')).
```

- 移动智能终端远程/近程车控 APP 验证绕过漏洞利用规则。

```

interaction_rule(
(control(V) :-
    installed(M, APP),
    pair(M, V, Comm),
    execCode(M, Perm),
```

```
vulExists(M, _, APP, localExploit, verifiBypass)
),
rule_desc('Compromised APP control vehicle')).
```

- 移动智能终端直接控制利用规则。

```
interaction_rule(
control(V) :-
pair(M, V, Comm),
execCode(M, root)
),
rule_desc('Compromised Mobile control vehicle')).
```

#### 4.1.3 车联网实时攻击图生成与分析的实现

根据章节 3.4 的设计，为了保证攻击图生成与分析的低延迟，我们采用分布式并行计算的方法。各蜂窝服务区 MEC 服务器独立并行生成实时局部贝叶斯攻击图，再上传到云平台，与云平台的局部贝叶斯攻击图拼接生成实时全局贝叶斯攻击图，进行车联网整体的量化风险评估。

由于设备和时间的限制，我们不能在实际车联网网上进行实验，所以我们使用多个虚拟机来作为 MEC 服务器和云平台的替代。一个虚拟机作为云平台中负责贝叶斯攻击图生成的服务器，其他虚拟机作为各服务区的 MEC 服务器。所有 MEC 服务器虚拟机中都各运行一个实时安全信息收集进程和一个局部贝叶斯攻击图生成进程，云平台虚拟机中运行一个攻击图规则集更新进程、一个局部贝叶斯攻击图生成进程和一个全局贝叶斯攻击图拼接生成进程。本章的所有程序均由 Python 实现。云平台虚拟机和其他 MEC 服务器虚拟机的进程间的关系如图 4-2 所示。

##### (1) MEC 服务器虚拟机实时安全信息收集进程

每个 MEC 服务器虚拟机中都运行着一个实时安全信息收集进程，该进程收集车联网的实时安全信息与来自云平台的“特殊攻击者（能力）”信息，并将这些信息整合和标准化为 Datalog 语言描述的安全信息后递交给实时局部攻击图生成进程生成实时局部攻击图。本文采用的方法是事先准备好服务区的 Datalog 描述的安全信息数据文件，通过修改文件内容的方式模拟车联网蜂窝服务区中的实时安全信息变化。实现中，实时安全信息收集进程的一个线程通过循环检查文件修改时间戳检查该安全信息数据文件内容是否有变化，若发现文件内容改变，则说明模拟的车联网蜂窝服务区的实时安全信息发生了改变，于是该线程将最新的服务区安全信息和最新的来自云平台的特殊“攻击者（能力）”信息拼接，递交给实时局部攻击图生成进程监听的特定端口（21451）。同时，进程也维护来自云平台的特殊“攻击者（能力）”信息。该进程监听虚拟机的特定端口（21450），若监听到该端口接收到了新的连接，则开启新线程处理该连接。首先判断该信息的来源，若信息并非来自于云平台，则丢弃；若信息来自于云平台，则说明这是云平台发送的最新的“特殊攻击者（能力）”信息。进程将该信息与服务区实时安全信息拼接，递交给实时局部攻击图生成进程监听的特定端口（21451）。

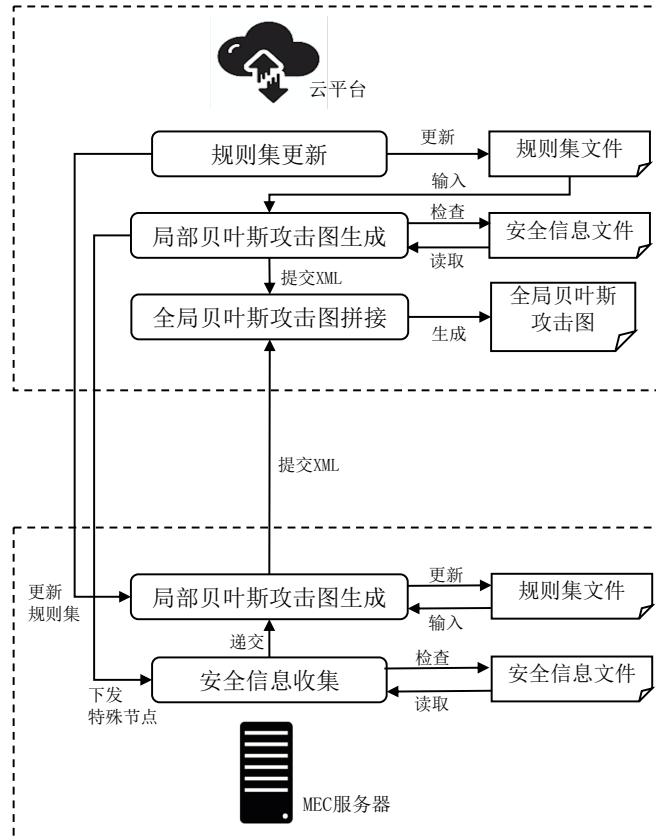


图 4-2 原型系统进程间关系

Figure 4-2 Relationships between processes in prototype system

### (2) MEC 服务器虚拟机局部贝叶斯攻击图生成进程

每个 MEC 服务器虚拟机中都运行着一个实时局部攻击图生成与分析进程，即局部贝叶斯攻击图生成进程。该进程实时监听虚拟机的特定端口（21451），若监听到该端口接收到了新的连接，则开启新线程处理该连接。首先判断该信息的来源。若信息来源于云平台，则说明这是一条云平台提供的攻击图生成规则集的更新信息，用该信息的内容更新本地的攻击图生成规则集文件；若该信息来源于本地，则说明该 MEC 服务器所在的蜂窝服务区的实时安全信息发生了变化，进程立刻调用 MulVAL 攻击图推导生成引擎，输入已存储好的最新的攻击图生成规则集和接收到的实时安全信息，推导生成实时局部攻击图，然后使用量化风险评估方案转化为贝叶斯攻击图（基于贝叶斯攻击图的量化风险评估方案的实现将在小节 4.1.4 详细介绍），并将描述该贝叶斯攻击图的 XML 格式的文件上传给云平台虚拟机的特定端口（21451）。处理该连接的同时进程继续监听特定端口（21451）。

### (3) 云平台虚拟机攻击图规则集更新进程

车联网技术在不断更新，针对车联网的新的攻击方法和手段也在不断出现。因此，想要构建一个包含车联网所有攻击方法和手段的攻击图生成规则集是不现实的。所以本文的原型系统提供了一个供以及时扩充车联网攻击图生成规则集的接口，即在云平台虚拟机中加入一个攻击图规则集更新进程。该进程提供手动进行车联网攻击图生成规则集增加、删除、修改的命令。该进程维护了一个列表，该列表存储了整个车联网攻击图生成规则集，列表的

元素就是车联网攻击图生成规则。通过命令对该列表元素进行增、删、改，即能更新车联网攻击图生成规则集。车联网攻击图生成规则集列表被更新后，该进程立即依据该列表更新本地的攻击图生成规则集文件，并把更新后的文件发送至所有 MEC 服务器虚拟机的特定端口（21451），实现整个车联网的攻击图生成规则集的更新。

#### (4) 云平台虚拟机局部贝叶斯攻击图生成进程

正如小节 3.4.1 所述，云平台局部攻击图生成与分析不追求实时性，因此云平台虚拟机的局部贝叶斯攻击图采用固定时间间隔进行更新的方法。在本小节中具体来说是每隔五分钟检查云平台攻击图安全信息数据文件是否有变化：若安全信息有变化，则输入变化后的安全信息和最新的攻击图生成规则集，更新一次云平台局部攻击图，再使用量化风险评估方案转化为贝叶斯攻击图，将最新的云平台局部贝叶斯攻击图以 XML 格式存储在本地，并同时通知本地的全局贝叶斯攻击图拼接生成进程监听的特定端口（21451）；若安全信息无变化，则不更新局部贝叶斯攻击图。由于攻击者对云平台成功进行网络攻击一般需要较长的时间，所以此处认为最坏情况下五分钟的云平台局部攻击图更新延迟是可以接受的。

在云平台更新自己的局部贝叶斯攻击图后，由于拼接算法的需要，将会把“特殊攻击者（能力）”信息（节点内容与节点概率）发送至所有 MEC 服务器虚拟机的安全信息收集进程监听的特定端口（21450）。这些特殊通信节点一般为车联网端层实体可以连接到的服务器，或者可以连接到端层实体的服务器。本小节的具体实现中由网络安全管理人员事先选定。

#### (5) 云平台虚拟机全局攻击图拼接生成进程

该进程监听本地特定端口（21451）的连接，若监听到该端口接收到了新的连接，则开启新线程处理该连接。若连接来自于本地，则说明是云平台的局部贝叶斯攻击图进行了更新，进程将会清空当前的云平台贝叶斯攻击图结构和全局贝叶斯攻击图结构，读取最新的云平台局部贝叶斯攻击图的 XML 文件，并将其解析存入云平台图结构，然后再与进程中保存的服务区局部贝叶斯攻击图结构通过拼接算法重新拼接成实时全局攻击图。若连接来自于 MEC 服务器虚拟机，则说明是该 MEC 服务器所在的服务区的安全信息发生了变化。进程则会使用局部拼接更新算法（算法 3-1）将新的服务区局部贝叶斯攻击图拼接入全局贝叶斯攻击图中，进行全局贝叶斯攻击图的局部更新。

### 4.1.4 基于贝叶斯攻击图的量化风险评估方案的实现

贝叶斯攻击图计算流程如图 4-3 所示。其从 MulVAL 生成的原始攻击图的 XML 描述文件出发，经过读取 XML 文件、解构攻击图、消除环路、贝叶斯分析、攻击路径可视化等几个步骤，最后输出贝叶斯攻击路径的可视化 PDF 文件，供车联网安全分析和量化风险评估。

#### (1) 数据结构

MulVAL 的贝叶斯攻击图表示为  $BAG < A, T, E, P >$ 。其中， $A$  表示属性节点集合； $T$  表示原子攻击节点集合； $E$  表示有向边集合； $P$  表示属性节点被利用的条件概率表。 $BAG$  满足以下条件：

1.  $\forall T \subseteq A \times A$  对  $\forall t \in T$ ，都有  $t = T_{in}(t) \rightarrow T_{out}(t)$ ，其中  $T_{in}(t)$  表示原子攻击  $t$  的一个条件属性节点， $T_{out}(t)$  表示原子攻击的结果属性节点，“ $\rightarrow$ ” 表示两个属性节点之间的因果关系。

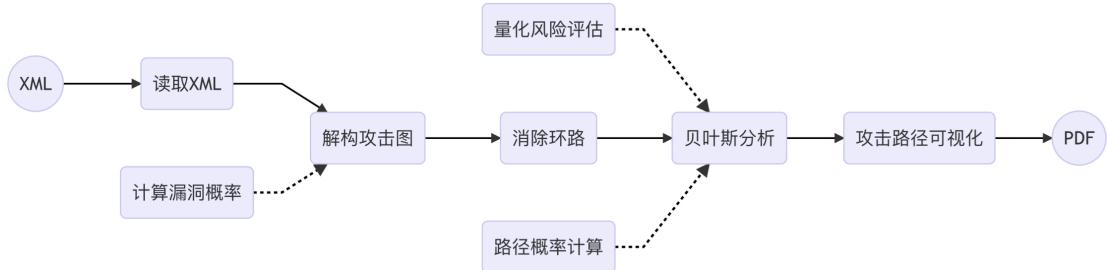


图 4-3 贝叶斯攻击图计算流程

Figure 4-3 Process of quantitative IoV risk assessment

2. 条件独立假设：任意节点只与其父节点有关，与其非父节点均相互独立。

### (2) 解析攻击图的 XML 描述文件

使用 Python 的 xml 库解析攻击图的 XML 文件生成有向图。其中攻击图的 XML 描述文件描述了两种实体：边（arc）和点（vertex）。vertex 有四种属性：编号（id），内容（fact），状态（metric），类型（type）；arc 有两种属性：起点编号（src），终点编号（dst）。用初始化的点类实例存储 vertex 信息，用初始化的边类实例存储 arc 信息，在用初始化的图类实例存储所有点和边的信息。

### (3) 解构攻击图

该步骤解构攻击图，将图的边与节点关联，节点相互关联并分类。具体来说，由于上一步存储在各实例中的属性信息只是文本描述的原始信息，需要在本步骤中替换为文本信息对应的实例。另外，由于之后分析攻击图时需要明确攻击图的“攻击者”节点和“目标节点”，因此本步骤也负责将“攻击者”节点和“目标节点”存入相应的集合中。

另外，计算单个漏洞的利用成功概率也在本步骤完成。如小节 3.5.2 所述，本文采用 CVSS 进行漏洞利用难度的量化估计，单个漏洞  $x$  的利用成功概率为  $P(x) = 2 \cdot AV_x \cdot AC_x \cdot AU_x$ 。在解析攻击图节点的同时判断节点是否为漏洞描述节点，若为漏洞描述节点，则根据其 CVE 编号查询漏洞数据库中该节点的 AV、AC、AU 属性进行利用成功概率计算，计算完成后将漏洞的利用成功概率存储到节点结构的利用概率属性中。漏洞 CVSS 的 AV、AC、AU 属性从美国国家漏洞数据库获得。在实现中，本文使用了实验室收集和维护的车联网安全知识库作为查询对象。

### (4) 结构转换：消除含圈路径

如小节 3.5.1 所述，MulVAL 同样具有会产生攻击环路的缺陷。为了使攻击图更易于分析，我们需要对攻击图进行去除攻击环路的预处理，对 MulVAL 的推导结果进行改进。本小节对该步骤的实现也是基于小节 3.5.1 的最深原子攻击消环思想。在针对 MulVAL 攻击图的具体实现中，我们发现，由于 DFS 路径不一定和真实的攻击图推导路径相同，所以 DFS 路径中最深的攻击节点不一定是最深的攻击路径节点。进一步，我们发现在 MulVAL 攻击图上实现基于 DFS 的最深原子攻击消环算法需要处理两类攻击路径环路，分别如图 4-4 的 (a) 和 (b) 所示，前者的环路入口节点为 OR 节点（菱形 3），后者的环路入口节点为 AND 节点（椭圆 2）。由于 AND 和 OR 节点逻辑含义不同，对这两种环路的处理也不同。具体算法

如 4-1 所示。由于去环采用的是使用临接表的 DFS 方法，遍历节点次数为有限次，与节点数量无关，所以攻击图去环算法的时间复杂度为  $O(n)$ 。

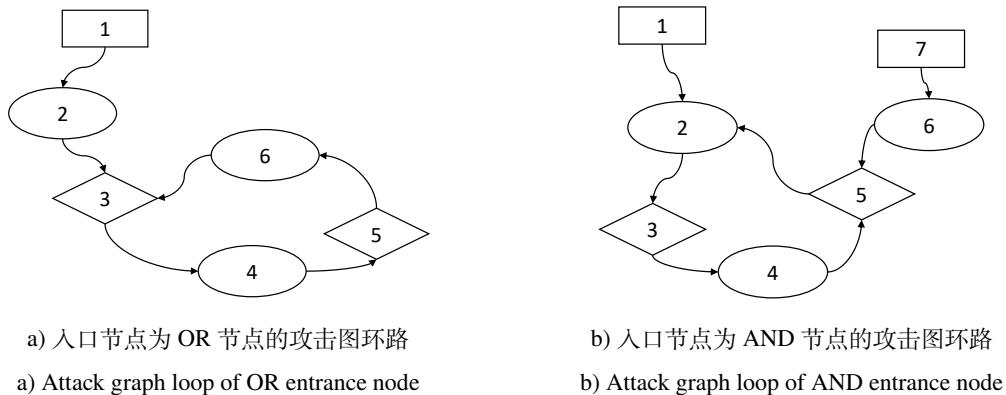


图 4-4 两种攻击图环路

Figure 4-4 Two kinds of attack graph loop

- 入口节点为 OR 节点的攻击图环路处理：如图 4-4 (a) 所示，DFS 寻找环路时经过的路径为  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 3$ 。但是，由于攻击图中 OR 节点为真的条件是其任一父节点为真，节点 3 可能先由节点 2 推导生成，也可能先由节点 6 推导生成，这两种可能导致了两种不同的攻击路径。事实上，我们可以用反证法证明，该攻击图的节点推导生成顺序是  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 3$ 。在该环路中，节点 6 是最深原子攻击节点，而节点 5 是最深攻击者能力节点。依据最远原子攻击消环思想，消去节点 6 即可在对攻击图整体影响最小的情况下消除该环路。事实上，我们可以用相同方法证明，入口节点为 OR 节点的攻击图环路的 DFS 路径与实际推导过程中的攻击路径相同，且实际的攻击路径若出现环路，则该环路一定是入口节点为 OR 节点的攻击图环路。因此在 DFS 检测到入口节点为 OR 节点的环路后，删除离栈顶最近的原子攻击节点，即 AND 节点即可消除环路。
- 入口节点为 AND 节点的攻击图环路处理：如图 4-4 (b) 所示，DFS 寻找环路时经过的路径为  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 2$ 。依据攻击图中节点的逻辑关系，可以得出攻击图生成推导时入口节点 2 是由节点 1 和节点 5 推导生成的。然而，节点 5 的生成却有两种可能，其一是由节点 6 先推导生成，其二是先由节点 4 推导生成，这两种可能导致了两种不同的攻击路径。实际上，我们可以用反证法证明，该攻击图的节点推导生成顺序是  $7 \rightarrow 6 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 4$ ，与 DFS 的结果不同。因此，在该环路中，节点 4 是真正的最深原子攻击节点，而节点 3 是最深攻击者能力节点。依据最远原子攻击消环思想，消去节点 4 即可在对攻击图整体影响最小的情况下消除该环路。事实上，我们可以用相同方法证明，入口节点为 AND 节点的攻击图环路一定可以转化为入口节点为 OR 节点的攻击图环路。在本例中由于 DFS 算法的不稳定性，导致  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 2$  的 DFS 路径先于  $7 \rightarrow 6 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 4$  出现。为了处理这种情况，我们给出了一种确定性的方法，即找到沿 DFS 路径找到环路中第一个入度大于 1 的 OR 节点，删除其前一个节点即可消除环路。这种方法实际上是把入口节点为 AND 节点的攻击图环路转化为了入口节点为 OR 节点的攻击图环路进行处理，此时第一个入度大于 1 的 OR 节点就是入口节点为 OR 节点的攻击图环路的入口节点。

---

### 算法 4-1 基于 DFS 的最深原子攻击消环算法

---

```

Data: 含环路的攻击图 graph
Result: 无环攻击图 graph

1 stack = Stack();
2 for attacker in graph.attacker do
3   | stack.push(attacker);
4 end
5 while stack is not empty do
6   | if stack.peek().childnode then
7     |   | temp = stack.peek().childnode.pop();
8     |   | if temp in stack then
9       |     |     | if temp is an OR node then
10      |       |       |   | graph removes stack.peek();
11      |       |       |   | stack.pop();
12      |     |     | else
13      |       |       |   |   find the first OR node whose in-degree > 1 in the loop;
14      |       |       |   |   graph removes its parent node in the loop;
15      |       |       |   |   stack removes all nodes after the cut point in the loop;
16      |     |     | end
17      |   | else
18      |     |   | stack.push(temp);
19      |   | end
20    | else
21    |   | stack.pop();
22  | end
23 end

```

---

### (5) 攻击路径贝叶斯概率计算

在计算攻击路径贝叶斯概率时，采用递归计算的方法，计算一条攻击路径的贝叶斯概率等于计算该攻击路径末尾节点的概率，也即成功走完该攻击路径到达最终节点的概率。由于递归计算时遍历每个节点次数均为一次，与节点数量无关，所以计算攻击路径贝叶斯概率的时间复杂度为 O(n)。计算涉及到 LEAF、AND、OR 三种类型的节点。由于这三种节点具有的逻辑属性各不相同，所以需要不同的贝叶斯概率计算方法。

**LEAF 型节点** MulVAL 生成的攻击图中的 LEAF 型节点为方形节点，为初始属性节点，内容为攻击图生成所需的网络初始信息，即除攻击图生成规则集外的所有输入信息。如图 4-5，计算贝叶斯攻击图中  $a_1$  节点（LEAF 节点，方形）的概率如算式 4-1 所示。

$$P(a_1) = \begin{cases} 2 \cdot AV_{a_1} \cdot AC_{a_1} \cdot AU_{a_1}, & a_1 = \text{vulnerability information} \\ 1, & a_1 \neq \text{vulnerability information} \end{cases} \quad (4-1)$$

**AND 型节点** MulVAL 生成的攻击图中的 AND 型节点为椭圆形节点，为原子攻击节点，指攻击图生成过程中的一步推导，即一次原子攻击，如一次漏洞利用或一次登陆行为。该节点为真的条件是其所有父节点均为真。如图 4-6，计算贝叶斯攻击图中  $a_3$  节点（AND 节点，

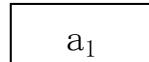


图 4-5 LEAF 型节点  
Figure 4-5 LEAF Node

椭圆形) 的概率如算式 4-2 所示。

$$P(a_3) = P(a_3|a_1, a_2)P(a_1, a_2) \quad (4-2)$$

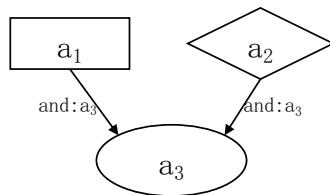


图 4-6 AND 型节点  
Figure 4-6 AND Node

**OR 型节点** MulVAL 生成的攻击图中的 OR 型节点为棱形节点, 和 LEAF 型节点一样为属性节点, 可用作原子攻击的条件。其表示一次原子攻击后的攻击结果, 即经过原子攻击加强后的攻击者能力。该节点为真的条件是其任意一个父节点为真。如图 4-7, 计算贝叶斯攻击图中  $a_3$  节点 (OR 节点, 棱形) 的概率如算式 4-3 所示。

$$P(a_3) = P(a_3|a_1, a_2)P(a_1, a_2) + P(a_3|a_1, \bar{a}_2)P(a_1, \bar{a}_2) + P(a_3|\bar{a}_1, a_2)P(\bar{a}_1, a_2) \quad (4-3)$$

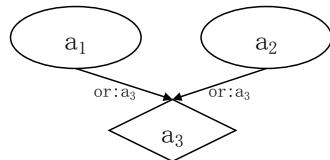


图 4-7 OR 型节点  
Figure 4-7 OR Node

## (6) 风险值计算与车联网量化风险评估

根据 Datalog 攻击图生成规则集, MulVAL 生成的攻击图中的 OR 节点, 即攻击者经过原子攻击后可以得到的能力由很多种, 但从车联网面临的主要安全风险的角度考虑, 其造成危害的能力有三种, 即 “control( )”、“execCode( , )” 和 “inside( )”, 分别表示获得某辆车的控制权、能在某台机器上以某一权限执行任意代码、和能进入某辆车。其中 “execCode” 又存在一种特殊情况 “execCode( , root)”, 表示能在某台机器上以 root 权限执行任意代码。因此, 我们把这四种节点作为攻击图的风险节点, 并基于这四种节点对车联网进行量化风险评估。

如小节 3.5.3 所述，我们用  $P(x)$  表示成功拿到风险节点  $x$  的概率；用  $num(x.next)$  风险节点  $x$  的出度。经验性地，我们给出风险节点  $x$  的风险值  $Risk(x)$  的计算方法，如算式 4-4 所示。

$$Risk(x) = \begin{cases} 10P(x)(1 + num(x.next)), & x = control(\_) \\ 10P(x)(1 + num(x.next)), & x = execCode(\_, root) \\ 6P(x)(1 + num(x.next)), & x = execCode(\_, X), X \neq root \\ 3P(x)(1 + num(x.next)), & x = inside(\_) \end{cases} \quad (4-4)$$

进一步，依据风险值计算，本文给出车联网整体、攻击路径、以及风险节点的安全风险等级评估参考表，如表 4-1 所示，其中  $x$  指车联网规模（通信节点数量）。

表 4-1 安全风险等级评估参考表  
Table 4-1 Reference table for risk level assessment

| 风险值上限<br>/ 评估对象 | 全局车联网  | 攻击路径   | 风险节点   |
|-----------------|--------|--------|--------|
| 风险等级            |        |        |        |
| 极高              | $> 3x$ | $> 30$ | $> 15$ |
| 很高              | $3x$   | $30$   | $15$   |
| 高               | $2x$   | $20$   | $10$   |
| 中               | $x$    | $10$   | $6$    |
| 低               | $0.5x$ | $5$    | $2$    |

## (7) 可视化贝叶斯攻击图

使用 Python 的 graphviz 库将整个贝叶斯攻击图录入一个.dot 文件结构中，调用 dot.render 函数生成描述攻击图的 PDF 文件。可视化时，除了 MulVAL 原来默认的椭圆、棱形、方形三种节点外，为了直观看到风险节点和风险值，我们设定粉色双层节点虚线连接到的节点为风险节点，粉色双层八边形节点内的值表示该风险节点的风险值。由虚线连接到图标标题（蓝色三层八边形节点）的红色三层八边形节点值为图中所有风险节点的风险值之和，表示整张全局攻击图的风险值，也即车联网整体的风险值。

另外，为了便于理解，对全局贝叶斯攻击图的节点编号也做了规范。全局贝叶斯攻击图的节点编号均为 “X[Y]” 的格式，其中 “X” 是该节点在对应的局部贝叶斯攻击图内的编号，“[Y]” 是节点所在的子网的编号，如云平台的编号是 “[0]”，若云平台局部贝叶斯攻击图中有节点 “1”，那它在拼接入全局贝叶斯攻击图后的编号为 “1[0]”。

## 4.2 原型系统测试

本节构建了一个较为复杂的攻击测试场景对原型系统进行了正确性和有效性的测试，并进一步对原型系统的实时性进行了测试。

#### 4.2.1 攻击测试场景构建

本小节构造攻击场景测试用例来检验本章实现的原型系统的正确性与有效性。该攻击场景测试用例为一个较为复杂的横跨“云”、“管”、“端”三层的车联网系统。该车联网系统包括一个云平台与三个蜂窝服务区。云平台及各蜂窝服务区的初始用户实体、网络资产、组件、安全漏洞及其关系的安全信息如表 4-2 所示。另外，表中还加入了车辆的初始状态的信息，如已上锁（静止）、已解锁（静止）、和移动中，作为攻击者初始能力的一部分。表中的漏洞信息选自车联网安全研究小组收集和维护的车联网漏洞数据库<sup>[40]</sup>。在生成攻击图前，我们将表中的安全信息用 Datalog 语言标准化表达。另外，由于实时攻击图生成的拼接算法要指定云平台需要下发到个蜂窝服务区的特殊节点，这里我们设定云平台的 cloudplat\_webserver1 为特殊节点，在云平台虚拟机生成局部攻击图后，会将攻击者可能在这些特殊节点得到的最强能力节点，即“特殊攻击者（能力）”信息下发到各蜂窝服务区 MEC 服务器虚拟机中，以辅助蜂窝服务区实时局部攻击图的生成。

表 4-2 攻击测试场景安全信息

Table 4-2 Security information of attack test scenario

| 位置    | 用户实体      | 网络资产                 | 车辆状态   | 系统组件    | 安全漏洞           |
|-------|-----------|----------------------|--------|---------|----------------|
| 云平台   | employee1 | cloudplat_webserver1 | -      | httpd   | -              |
| 云平台   | employee1 | cloudplat_webserver1 | -      | sshd    | -              |
| 云平台   | employee1 | cloudplat_database1  | -      | sql     | CVE-2015-1761  |
| 云平台   | employee2 | cloudplat_TSPserver3 | -      | sudo    | CVE-2021-3156  |
| 云平台   | employee2 | cloudplat_OTAserver  | -      | sudo    | CVE-2021-3156  |
| 服务区 1 | -         | vsubnet1_MECSERVER   | -      | -       | -              |
| 服务区 1 | owner1    | vehicle1             | locked | keyFobs | CVE-2020-29438 |
| 服务区 1 | owner1    | vehicle1             | locked | obd     | CVE-2020-29440 |
| 服务区 1 | owner1    | mobile1              | -      | -       | -              |
| 服务区 1 | owner1    | bluetooth1           | -      | daemons | CVE-2020-0022  |
| 服务区 1 | owner2    | vehicle2             | move   | obd     | CVE-2018-11478 |
| 服务区 1 | owner2    | mobile2              | -      | -       | -              |
| 服务区 1 | owner2    | bluetooth2           | -      | -       | -              |
| 服务区 1 | owner3    | vehicle3             | move   | chrome  | CVE-2021-21220 |
| 服务区 1 | owner3    | vehicle3             | move   | micomd  | CVE-2020-8539  |
| 服务区 1 | owner3    | mobile3              | -      | -       | -              |
| 服务区 1 | owner3    | bluetooth3           | -      | -       | -              |
| 服务区 1 | owner4    | vehicle4             | move   | -       | -              |
| 服务区 1 | owner4    | mobile4              | -      | -       | -              |
| 服务区 1 | owner4    | bluetooth4           | -      | -       | -              |
| 服务区 1 | -         | vsubnet1_RSU1        | -      | -       | -              |
| 服务区 2 | -         | vsubnet2_MECSERVER   | -      | -       | -              |
| 服务区 2 | owner5    | vehicle5             | move   | keyFobs | CVE-2020-29438 |
| 服务区 2 | owner5    | vehicle5             | move   | obd     | CVE-2020-29440 |

续下页

续表 4-2

| 位置    | 管理者     | 网络资产               | 车辆状态     | 系统组件      | 安全漏洞           |
|-------|---------|--------------------|----------|-----------|----------------|
| 服务区 2 | owner5  | mobile5            | -        | -         | -              |
| 服务区 2 | owner5  | bluetooth5         | -        | -         | -              |
| 服务区 2 | owner6  | vehicle6           | locked   | -         | -              |
| 服务区 2 | owner6  | mobile6            | -        | -         | -              |
| 服务区 2 | owner6  | wifi6              | -        | interface | CVE-2018-11477 |
| 服务区 2 | owner7  | vehicle7           | move     | -         | -              |
| 服务区 2 | owner7  | mobile7            | -        | chrome    | CVE-2021-21220 |
| 服务区 2 | owner7  | mobile7            | -        | teslaAPP  | vulIID         |
| 服务区 2 | owner7  | bluetooth7         | -        | -         | -              |
| 服务区 2 | -       | vsubnet2_RSU       | -        | -         | -              |
| 服务区 3 | -       | vsubnet3_MECSERVER | -        | -         | -              |
| 服务区 3 | owner8  | vehivle8           | move     | -         | -              |
| 服务区 3 | owner8  | mobile8            | -        | chrome    | CVE-2021-21220 |
| 服务区 3 | owner8  | mobile8            | -        | driver    | CVE-2017-6424  |
| 服务区 3 | owner8  | bluetooth8         | -        | -         | -              |
| 服务区 3 | owner9  | vehivle9           | move     | chrome    | CVE-2021-21220 |
| 服务区 3 | owner9  | vehivle9           | move     | sudo      | CVE-2021-3156  |
| 服务区 3 | owner9  | mobile9            | -        | -         | -              |
| 服务区 3 | owner9  | wifi9              | -        | -         | -              |
| 服务区 3 | owner10 | vehicle10          | move     | obd       | CVE-2016-2354  |
| 服务区 3 | owner11 | vehicle11          | unlocked | ivi       | CVE-2018-9314  |
| 服务区 3 | -       | vsubnet3_RSU       | -        | -         | -              |

表 4-2 中的漏洞选自实验室收集和维护的车联网漏洞数据库以及 NVD。漏洞的具体信息如表 4-3 所示，其中包含了云平台服务器常见的远程或本地利用的提权漏洞、智能网联汽车中最常见的信息泄露漏洞和验证绕过漏洞、以及通信管道中常见的信息泄露漏洞和验证绕过漏洞。

攻击场景中还包含攻击者。本攻击测试场景中有两种攻击者，一种是 internet 网络攻击者，另一种是 physics 物理攻击者。internet 网络攻击者可以访问开启了网络服务的主机，而 physics 物理攻击者有对道路上的所有车辆展开攻击的能力。

#### 4.2.2 静态车联网攻击图生成与分析测试

将上一小节中构建的攻击测试场景的安全信息用 Datalog 语言标准化表示，与构建好的车联网攻击图生成规则集一起输入 MulVAL 推导引擎即可生成车联网攻击图。由于 MulVAL 攻击图较为复杂，为了更清楚地展示攻击图的结构，本文正文中将攻击图的节点内容均略去。详细的攻击图节点内容表请参考附录 A，完整的攻击图请参考 [https://github.com/MekAk/UActOR/Distributed\\_Bayesian\\_Attack\\_Graph](https://github.com/MekAk/UActOR/Distributed_Bayesian_Attack_Graph)。

本节测试攻击图生成、局部贝叶斯攻击图生成、全局贝叶斯攻击图生成与量化风险评估的准确性。首先使用 MulVAL 直接生成整个车联网测试场景的全局攻击图，如图 4-8 所示。

表 4-3 漏洞具体信息  
Table 4-3 Vulnerability information

| 漏洞 CVE 编号          | 脆弱性组件     | 利用结果 <sup>a</sup> | AV | AC | AU |
|--------------------|-----------|-------------------|----|----|----|
| CVE-2015-1761      | SQL       | 提权                | N  | L  | S  |
| CVE-2021-3156      | sudo      | 提权                | L  | L  | N  |
| CVE-2017-6424      | driver    | 提权                | L  | M  | N  |
| CVE-2021-21220     | Chrome    | 客户端提权             | N  | M  | N  |
| CVE-2020-29438     | key fobs  | 信息泄露              | A  | L  | N  |
| CVE-2018-11477     | wifi      | 信息泄露              | A  | L  | N  |
| CVE-2020-0022      | Bluetooth | 验证绕过              | A  | L  | N  |
| CVE-2020-29440     | BCM       | 验证绕过              | L  | L  | N  |
| CVE-2018-11478     | OBD2      | 验证绕过              | A  | L  | N  |
| CVE-2020-8539      | deamon    | 验证绕过              | L  | L  | N  |
| CVE-2018-9314      | IVI       | 验证绕过              | L  | L  | N  |
| CVE-2016-2354      | OBD2      | 验证绕过              | A  | L  | N  |
| vulID <sup>b</sup> | TeslaAPP  | 验证绕过              | L  | L  | N  |

<sup>a</sup> 利用结果指该漏洞利用的一般最终结果，而不考虑利用时的细节。

<sup>b</sup> 该漏洞是蜚语安全于 2021 年发现的一组漏洞，暂未有 CVE 编号<sup>[41]</sup>。

可见本文构建的攻击图生成规则集可以通过 MulVAL 推导引擎成功生成攻击测试场景的全局攻击图。由于原始攻击图过于复杂，难以直接分析，此处我们不对该攻击图进行分析。

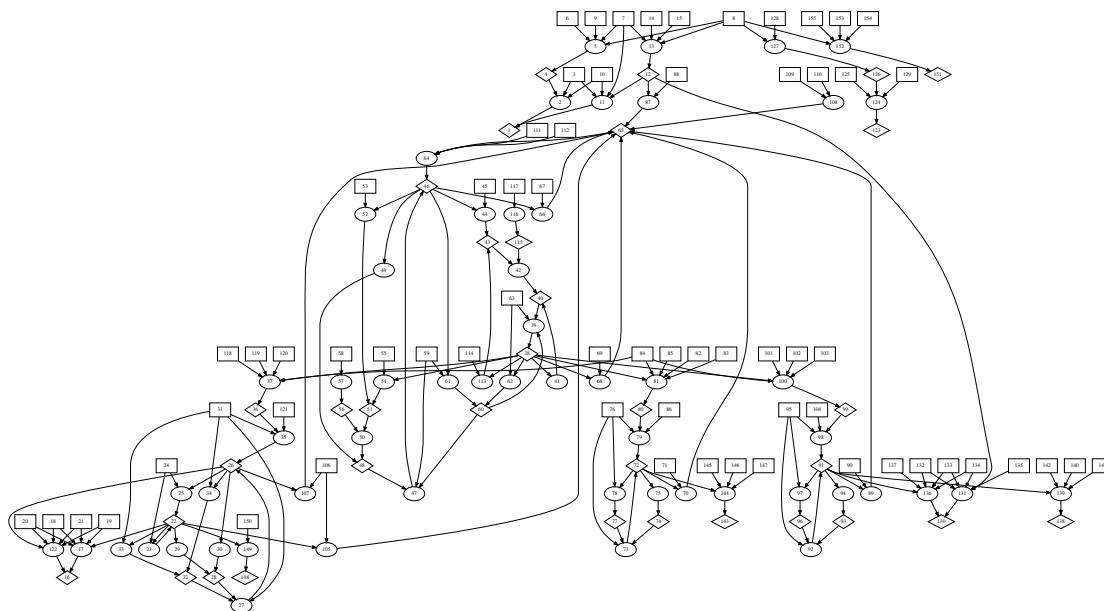


图 4-8 MulVAL 直接生成的车联网全局攻击图  
 Figure 4-8 Global attack graph generated directly by MulVAL

接下来我们通过对各局部贝叶斯攻击图的攻击路径进行分析，来验证本文的攻击图生成和去环算法的正确性。图 4-9、4-10、4-11 和 4-12 分别是由局部贝叶斯攻击图生成进程

的云平台、服务区 1、服务区 2 和服务区 3 的局部贝叶斯攻击图。其中云平台贝叶斯攻击图中的节点 “11:execCode(cloudplat\_webserver1,user)”（攻击者能在 cloudplat\_webserver1 以 user 用户的权限执行任意代码）是攻击者在云平台在 cloudplat\_webserver1 上所能获得的最强能力，因此将该节点作为“特殊攻击者（能力）”节点。由于 MulVAL 的推导引擎所限，将该“特殊攻击者（能力）”节点下发至各蜂窝服务区之前，需先将该节点从 OR 型节点转化为 LEAF 型节点。本文的处理方式是在“特殊攻击者（能力）”节点的信息描述前加上“pv”（primitive vector）字符串，表示其为“特殊攻击者（能力）”节点转化而来的 LEAF 节点。蜂窝服务区接收到该节点后，将通过规则 “pvexecCode(cloudplat\_webserver1,user)→RULE 38 (PV rule)→execCode(cloudplat\_webserver1,user)” 将其重新推导为原本的 OR 型节点参与攻击图推导。

对云平台局部贝叶斯攻击图从攻击者节点 “40:attackerLocated(internet)” 开始分析。实际上去除环路后，该贝叶斯攻击图只有一条主要攻击路径：40 → 38 → 33 → 32 → 1 → 26 → 25 → 12 → 11。图中还有一部分去除环路后的残余部分，但由于容易分辨出来（路径的末尾不是风险节点），因此不会对攻击图的分析造成很大的影响。

攻击者的初始位置在 Internet 上 (40)，由于 cloudplat\_database1 开放 SQL 服务 (39)，攻击者可以直接连接到 cloudplat\_database1 (38→33)。又因为 cloudplat\_database1 的 SQL 服务存在远程提权漏洞 CVE-2015-1761 (42)，攻击者得以远程攻陷服务器 cloudplat\_database1 (15)，获得 cloudplat\_database1 上以 mysql 账号的权限执行任意代码的能力 (1)。而 cloudplat\_database1 上的 mysql 账号由用户实体 employee1 拥有 (27)，所以攻击者可以通过密码嗅探 (26)，攻陷用户实体 employee1，获得 employee1 所管理的全部主机账号和密码 (25)。与此同时，cloudplat\_webserver1 开启了 sshd 服务 (23)，其提供了 (22) 攻击者登陆 cloudplat\_webserver1 的可能 (21)；而 cloudplat\_database1 与 cloudplat\_webserver1 在同一个内网中，可以连接 (18)，所以攻击者获得了通过多跳网络从 cloudplat\_database1 连接到 cloudplat\_webserver1 的能力 (16)。至此，攻击者同时具有了登陆 cloudplat\_webserver1 的可能 (21) 以及连接到 cloudplat\_webserver1 的能力 (16)，可以通过登陆行为 (15) 访问 cloudplat\_webserver1 (13)。到目前为止，攻击者同时具有了获得 employee1 所管理的全部主机账号和密码的能力 (24) 和访问 cloudplat\_webserver1 的能力 (13)，而用户实体 employee1 也同时管理 cloudplat\_webserver1 的 user 用户账号 (24)，所以，攻击者可以登陆进 cloudplat\_webserver1 (12)，得到 cloudplat\_webserver1 的 user 用户的权限 (11)。

对服务区 1 局部贝叶斯攻击图分别从物理攻击者节点 “8:attackerLocated(physics)” 和“特殊攻击者（能力）”节点 “34:execCode(cloudplat\_webserver1,user)” 开始分析。去除环路后，该贝叶斯攻击图有五条主要攻击路径。

- 34→33→32→17→24→43→42。攻击者在云平台得到 cloudplat\_webserver1 的 user 用户的权限后 (34)，由于 cloudplat\_webserver1 是 Web 服务器 (39)，vehicle3 上安装有客户端程序 chrome (38)，vehicle3 可以连接到 cloudplat\_webserver1 (37)，用户实体 vehicle3Owner 是一个不称职者 (40)，因此 vehicle3Owner 可能使用 vehicle3 访问已经被攻击者攻陷的 cloudplat\_webserver1 (33)，接收到攻击者构造的恶意数据 (32)。而不巧的是 vehicle3Owner 在 vehicle3 有 user 账号 (28)，且 vehicle3 的 chrome 上存在远程客户端利用提权漏洞 CVE-2021-21220 (41)，因此攻击者可以利用 vehicle3Owner 访问网页的行为利用该漏洞 (31)，得到 vehicle3 上以 user 用户的权限执行任意代码的能力 (24)。而 vehicle3 上安装的 micomd 进程 (1) 存在可以本地利用的验证绕过漏洞 CVE-2020-8539 (44)，因此已得到 vehicle3 的 user 用户的权限到攻击者可以利

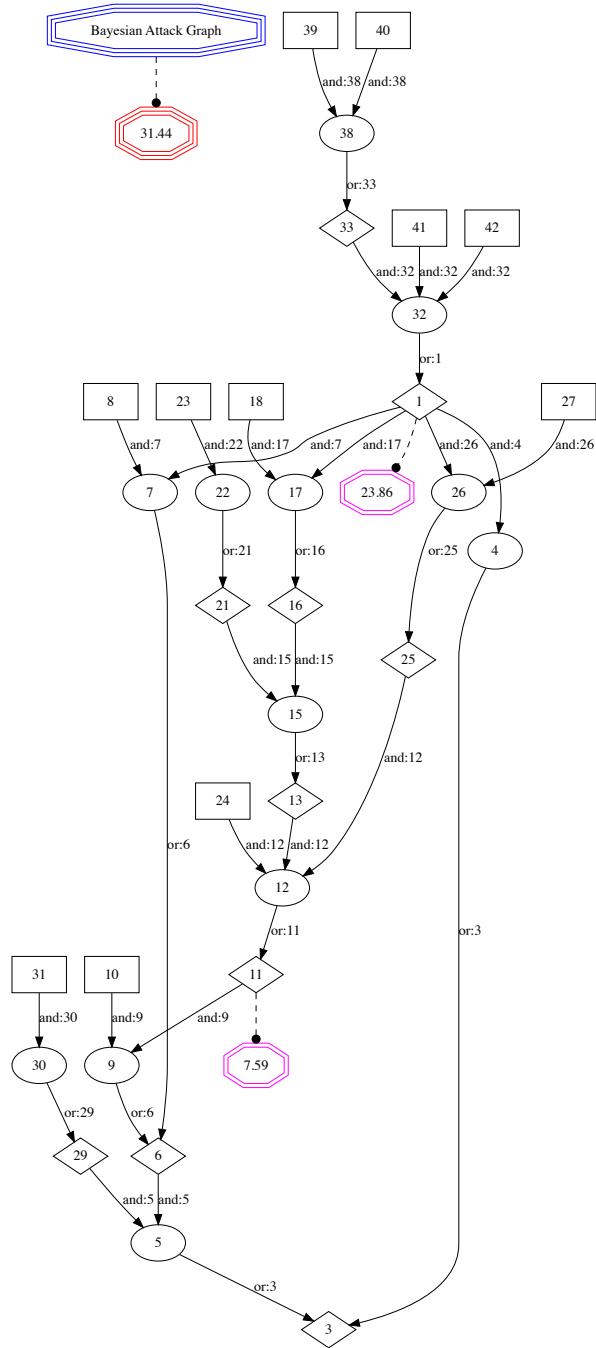


图 4-9 云平台局部贝叶斯攻击图  
Figure 4-9 Bayesian attack graph of cloud

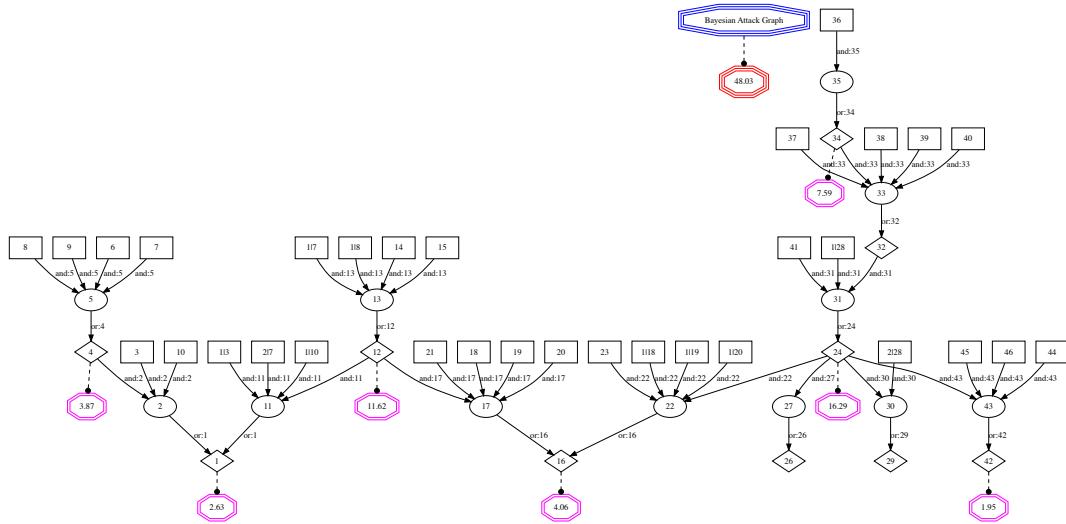


图 4-10 服务区 1 局部贝叶斯攻击图

Figure 4-10 Bayesian attack graph of cell1

用该漏洞进行 CAN 总线命令注入 (43)，得到 vehicle3 的控制权。该攻击路径是一条典型的通过网络攻击并控制智能网联汽车的“网络-物理”攻击路径。

- 34→33→32→17→24→22→16。由上一条攻击路径所述，攻击者可以得到 vehicle3 的 user 用户的权限 (24)。由于 vehicle3 和 vehicle2 距离较近 (在同一个服务区内)，因此他们可以通过 V2V 通信连接 (23)。而 vehicle2 安装的 obd2 组件 (20) 存在物理进程验证绕过漏洞 CVE-2018-11478 (18)，因此攻击者可以在 vehicle3 上 (24) 通过 V2V 通信近程利用该漏洞向移动中的 vehicle2 的 CAN 总线进行命令注入 (22)，得到 vehicle2 的控制权 (16)。该路径利用了 C-V2X 通信建构的短程 D2D 通信。
- 8→13→12→17→16。物理攻击者在服务区 1 内 (8)，由于 mobile1 和 vehicle1 直接的 bluetooth1 连接 (15) 的 daemons 组件存在物理进程验证绕过漏洞 CVE-2020-0022 (14)，且 vehicle1 的状态是静止 (7)，所以物理攻击者可以利用该通信管道的漏洞绕过蓝牙验证 (13)，通过发送构造的恶意报文在蓝牙接收端 vehicle1 上以蓝牙程序的权限执行任意代码 (12)。由于 vehicle1 和 vehicle2 距离较近 (在同一个服务区内)，因此他们可以通过 V2V 通信连接 (21)。而 vehicle2 安装的 obd2 组件 (20) 存在可以物理本地利用的验证绕过漏洞 CVE-2018-11478 (18)，因此攻击者可以在 vehicle1 上 (12) 通过 V2V 通信近程利用该漏洞向 vehicle2 的 CAN 总线进行命令注入 (17)，得到 vehicle2 的控制权 (16)。该路径中利用了通信管道的漏洞，这也是车联网面临的一个常见风险。
- 8→13→12→11→1。由上一条攻击路径所述，攻击者可以在 vehicle1 上以某一权限执行任意代码 (12)。由于 vehicle1 安装有 obdTeslaX 组件 (10)，该组件存在可本地利用的验证绕过漏洞 CVE-2020-29440 (3)，因此攻击者可以利用该漏洞，向 CAN 总线进行命令注入 (11)，得到 vehicle1 的控制权 (1)。
- 8→5→4→2→1。物理攻击者在服务区 1 内 (8)，vehicle1 上安装的 keyFobs 组件 (9) 存在物理近程信息泄露漏洞 CVE-2020-29438 (6)，且车辆的状态是上锁中 (7)，因此攻击者可以利用该漏洞得到车辆的开锁密钥 (5) 从而进入车 vehicle1 内 (4)。由



于 vehicle1 安装有 obdTeslaX 组件 (10)，该组件存在可物理本地利用的验证绕过漏洞 CVE-2020-29440 (3)，因此 vehicle1 内的攻击者可以通过物理插入 OBD 接口的方式利用该漏洞，向 CAN 总线进行命令注入 (2)，得到 vehicle1 的控制权 (1)。该路径是一条典型的车联网物理攻击路径，同时也是 2020 年 11 月 23 日 Lennert Wouters 通过组合利用特斯拉 Model X 汽车上的几个信息泄露和验证绕过漏洞控制特斯拉汽车的攻击路线。这条路径在小节 2.3.2 中也作为例子用于验证车联网本体模型的正确性。

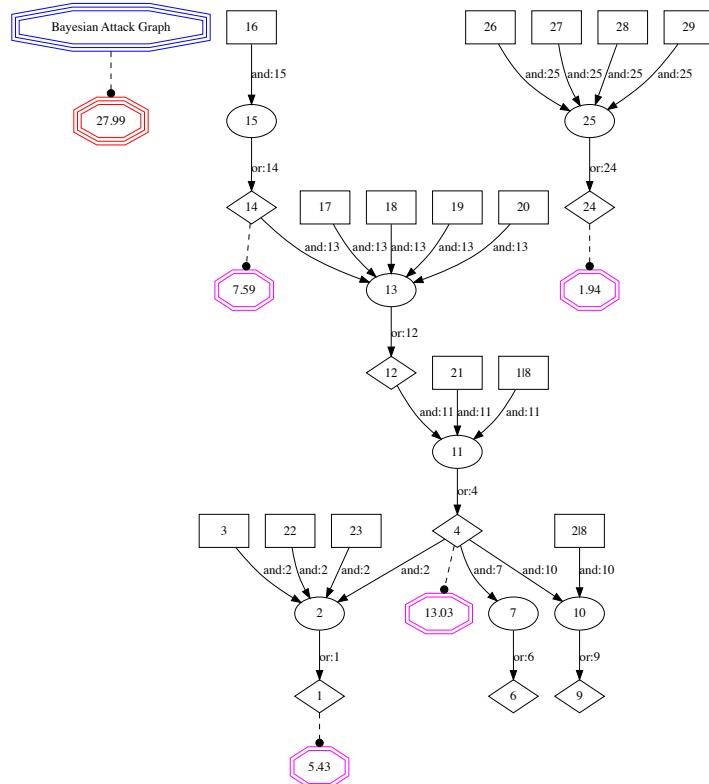


图 4-11 服务区 2 局部贝叶斯攻击图  
Figure 4-11 Bayesian attack graph of cell2

对服务区 2 局部贝叶斯攻击图也是分别从物理攻击者节点 “28:attackerLocated(physics)” 和 “特殊攻击者 (能力)” 节点 “14:execCode (cloudplat\_webserver1,user)” 开始分析。去除环路后，该贝叶斯攻击图有两条主要攻击路径。此时 vehicle5 的状态是移动中，vehicle5 上的 headUnitHU\_NBT 组件存在的可以物理近程利用的信息泄露漏洞 CVE-2020-29438 与可以物理本地利用的验证绕过漏洞 CVE-2018-9314 并不能得到利用，因此在服务区 2 的局部攻击图中没有出现利用这些漏洞的路径。这也证明了攻击图生成的正确性。

- 14→13→12→11→4→2→1。该路径的前期与服务区 1 局部攻击图的第一条路径相似，利用了智能移动终端 mobile7 的用户实体 vehicle7Owner 访问不安全的网站 (cloudplat\_webserver1) 的行为，以及 chrome 浏览器上的远程客户端利用提权漏洞 CVE-2021-21220 攻陷了 mobile7，得到在 mobile7 上以 user 用户的权限执行任意代码的能力 (4)。而 mobile7 上安装的车辆控制软件 TeslaAPP (23) 存在可以本地利用的身



份验证绕过漏洞 vulID (3) (该漏洞虽然已被发现，但还未被分配 CVE 编号)。另外 mobile7 与智能网联汽车 vehicle7 配对 (22)，因此攻击者可通过在 mobile7 上以 user 用户权限执行任意代码来利用该漏洞 (2)，绕过用户身份验证并控制汽车 vehicle7 (1)。该路径是一条典型的通过智能移动终端攻击并控制车辆的攻击路径。

- 28→25→24。物理攻击者在服务区 2 内 (28)，vehicle6 的状态是被上锁 (27)，而在 mobile6 与 vehicle6 通信的 wifi6 通信管道安装的 vgateiCar2WiFi 接口上存在可物理远程利用的信息泄露漏洞 CVE-2018-11477 (26)，因此道路上的物理攻击者可以通过该漏洞泄露 mobile6 与 vehicle6 之间的通信信息 (25)，获得 vehicle6 的开锁密钥并进入车辆 (24)。

对蜂窝服务区 3 局部攻击图也是分别从物理攻击者节点 37: attackerLocated(physics) 和“特殊攻击者（能力）”节点 23: execCode(cloudplat\_webserver1,user) 开始分析。去除环路后，服务区 3 局部攻击图有四条主要攻击路径。

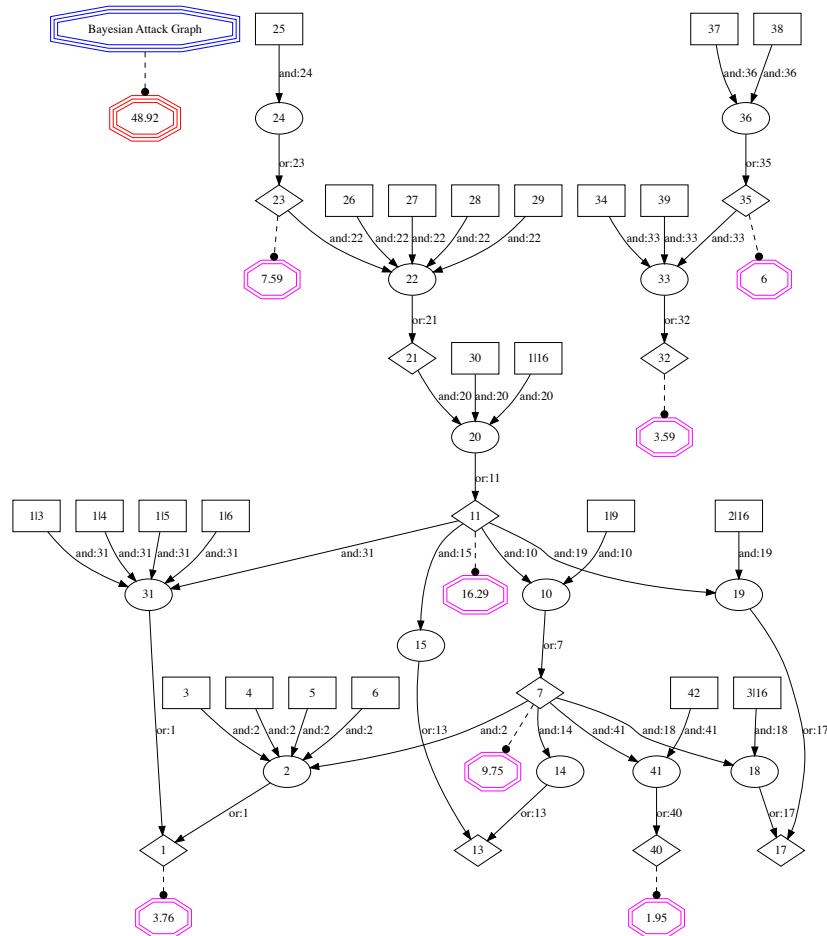


图 4-12 服务区 3 局部贝叶斯攻击图

Figure 4-12 Bayesian attack graph of cell3

- 23 → 22 → 21 → 20 → 11 → 31 → 1。该路径的前期也与服务区 1 局部攻击图的第一条路径相似，利用了智能网联汽车 vehicle9 的用户实体 vehicle9Owner 访问不安全的网站 (cloudplat\_webserver1) 的行为，以及 chrome 浏览器上的远程客户端利用提权

漏洞 CVE-2021-21220 攻陷了 vehicle9，得到在 vehicle9 上以 user 用户的权限执行任意代码的能力 (11)。而 vehicle10 上安装的 obd2 组件 (5) 存在可以物理远程利用的验证绕过漏洞 CVE-2016-2354 (3)。由于 vehicle9 和 vehicle10 距离较近 (在同一个服务区内)，因此他们可以通过 V2V 通信连接 (6)。因此攻击者可以在 vehicle9 上以 user 权限执行恶意代码 (11)，通过 V2V 通信远程利用该漏洞向移动中的 vehicle10 的 CAN 总线进行命令注入 (31)，得到 vehicle10 的控制权 (1)。

- $23 \rightarrow 22 \rightarrow 21 \rightarrow 20 \rightarrow 11 \rightarrow 10 \rightarrow 7 \rightarrow 2 \rightarrow 1$ 。如上条攻击路径所述，攻击者通过  $23 \rightarrow 22 \rightarrow 21 \rightarrow 20 \rightarrow 11$  得到了在 vehicle9 上以 user 用户的权限执行任意代码的能力 (11)。而 vehicle9 上安装的 sudo 程序存在可以本地利用的提权漏洞 CVE-2021-3156 (9)，因此攻击者可以通过以 user 用户的权限执行任意代码利用该漏洞进行提权 (10)，获得 vehicle9 上以 root 权限执行任意代码的能力 (7)。而 vehicle10 上安装的 obd2 组件 (5) 存在可以物理远程利用的验证绕过漏洞 CVE-2016-2354 (3)。由于 vehicle9 和 vehicle10 距离较近 (在同一个服务区内)，因此他们可以通过 V2V 通信连接 (6)。因此攻击者可以在 vehicle9 上以 root 权限执行恶意代码 (7)，通过 V2V 通信远程利用该漏洞向移动中的 vehicle10 的 CAN 总线进行命令注入 (2)，得到 vehicle10 的控制权 (1)。可以看到这条路径的  $10 \rightarrow 7 \rightarrow 2 \rightarrow 1$  部分其实是上一条路径绕了一个弯路，这也是受限一阶谓词逻辑推导不可避免的现象。
- $23 \rightarrow 22 \rightarrow 21 \rightarrow 20 \rightarrow 11 \rightarrow 10 \rightarrow 7 \rightarrow 41 \rightarrow 40$ 。如上条攻击路径所述，攻击者通过  $23 \rightarrow 22 \rightarrow 21 \rightarrow 20 \rightarrow 11 \rightarrow 10 \rightarrow 7$  得到了在 vehicle9 上以 root 权限执行任意代码的能力 (7)。而获得了车载操作系统的 root 权限后，攻击者也就获得了正在移动中的 vehicle9 的控制权 ( $41 \rightarrow 40$ )。这条路径是一个通过完全控制车载操作系统来控制汽车的攻击路径的例子。
- $37 \rightarrow 36 \rightarrow 35 \rightarrow 33 \rightarrow 32$ 。物理攻击者位于服务区 3 内 (37)，而 vehicle11 处于未上锁的状态 (38)，因此攻击者可以直接进入 vehicle11 内 ( $36 \rightarrow 35$ )。而 vehicle11 上安装的 headUnitHU\_NBT 组件 (39) 存在可以物理本地利用的验证绕过漏洞 CVE-2018-9314 (34)，因此车内的攻击者可以通过接入物理接口的方式利用该漏洞向 CAN 总线注入命令 (33)，得到 vehicle11 的控制权 (32)。这条路径也是典型的通过利用车辆状态的攻击路径。在本文构建的车联网安全本体模型中，车辆状态也是一种攻击者 (能力) 实体。

接下来我们验证实时全局贝叶斯攻击图生成所使用的拼接算法的正确性。图 4-13 和 4-14 是云平台虚拟机实时全局攻击图拼接生成与分析进程读取四个局部贝叶斯攻击图的 XML 文件后生成的最终的贝叶斯攻击图。由于全局攻击图规模较大，为了便于观察，此处将图分为左右两部分，左半部分和右半部分通过  $33[1] \rightarrow 32[1]$  连接。

首先直观上看，贝叶斯攻击图中已无任何攻击环路，从攻击者节点开始可以轻松地遍历攻击路径，且绝大多数攻击路径都是有效的攻击路径，满足了便于分析的要求。接下来我们通过指出全局贝叶斯攻击图中与局部贝叶斯攻击图攻击路径的组合相匹配的完整攻击路径，来证明拼接算法和去环算法的正确性。

1. 攻击路径  $40[0] \rightarrow 38[0] \rightarrow 33[0] \rightarrow 32[0] \rightarrow 1[0] \rightarrow 26[0] \rightarrow 25[0] \rightarrow 12[0] \rightarrow \mathbf{11[0]} \rightarrow 33[1] \rightarrow 32[1] \rightarrow 17[1] \rightarrow 24[1] \rightarrow 43[1] \rightarrow 42[1]$  即为云平台攻击图的攻击路径与服务区 1 局部攻击图的第一条攻击路径的组合。将该路径上所有风险节点的风险值相加，得到这条攻击路径的风险值为 61.07。
2. 攻击路径  $40[0] \rightarrow 38[0] \rightarrow 33[0] \rightarrow 32[0] \rightarrow 1[0] \rightarrow 26[0] \rightarrow 25[0] \rightarrow 12[0] \rightarrow \mathbf{11[0]} \rightarrow$

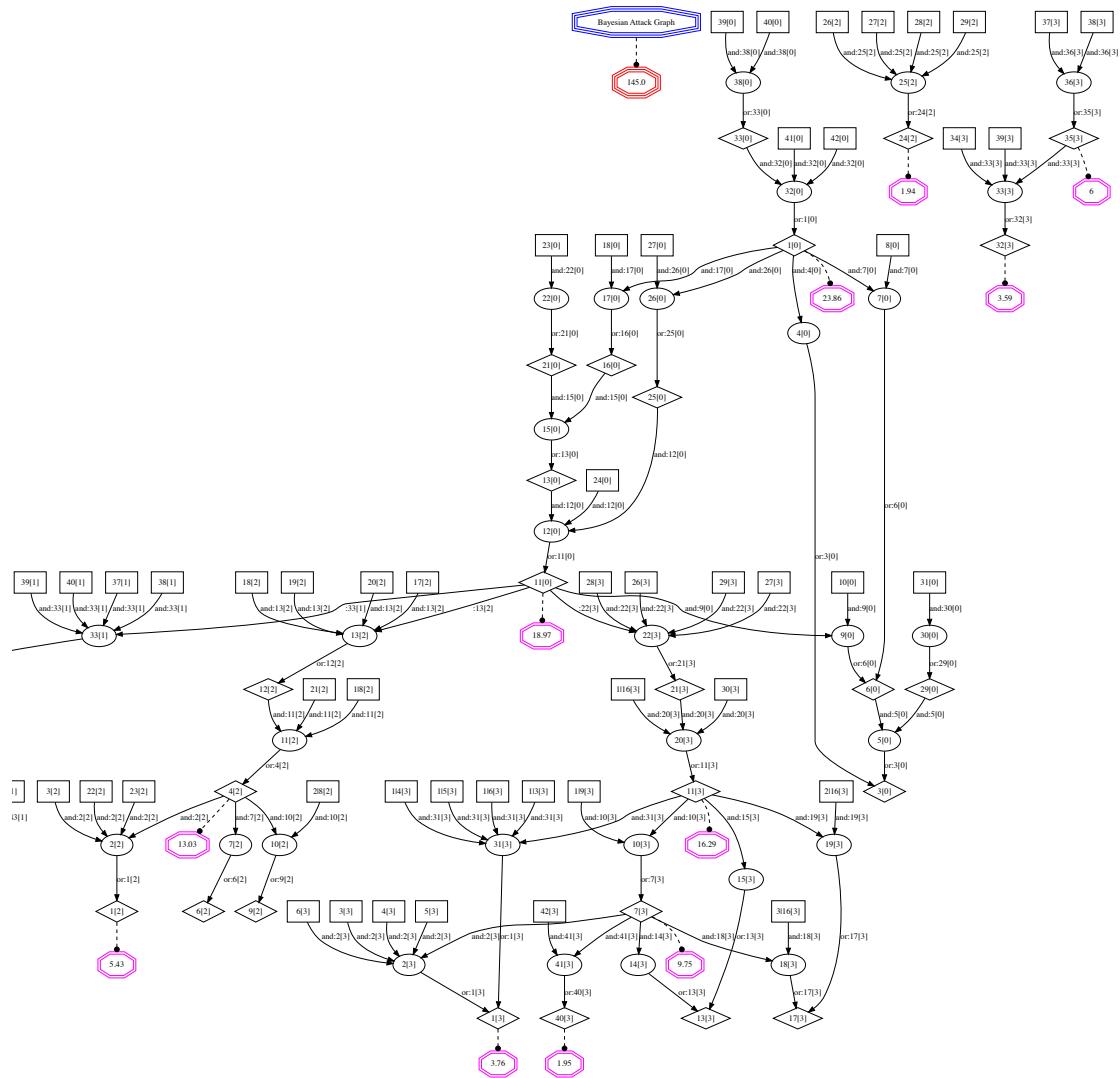


图 4-13 全局贝叶斯攻击图 (右半部分)  
Figure 4-13 Global Bayesian attack graph (right part)

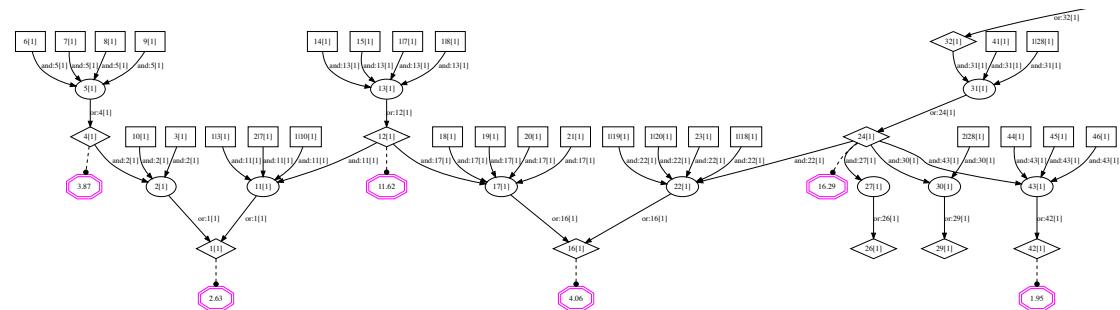


图 4-14 全局贝叶斯攻击图（左半部分）  
Figure 4-14 Global Bayesian attack graph (left part)

- 33[1] → 32[1] → 17[1] → 24[1] → 22[1] → 16[1] 即为云平台攻击图的攻击路径与服务区 1 局部攻击图的第二条攻击路径的组合。这条攻击路径的风险值为 63.18。
3. 攻击路径 8[1] → 13[1] → 12[1] → 17[1] → 16[1] 即为服务区 1 局部攻击图的第三条攻击路径。这条攻击路径的风险值为 15.68。
  4. 攻击路径 8[1] → 13[1] → 12[1] → 11[1] → 1[1] 1 即为服务区 1 局部攻击图的第四条攻击路径。这条攻击路径的风险值为 14.25。
  5. 攻击路径 8[1] → 5[1] → 4[1] → 2[1] → 1[1] 即为服务器 1 局部攻击图的第五条攻击路径。这条攻击路径的风险值为 6.5。
  6. 攻击路径 40[0] → 38[0] → 33[0] → 32[0] → 1[0] → 26[0] → 25[0] → 12[0] → **11[0]** → 13[2] → 12[2] → 11[2] → 4[2] → 2[2] → 1[2] 即为云平台攻击图的攻击路径与服务区 2 局部攻击图的第一条攻击路径的组合。这条攻击路径的风险值为 61.29。
  7. 攻击路径 28[2] → 25[2] → 24[2] 即服务区 2 局部攻击图的第二条攻击路径。这条攻击路径的风险值为 1.94。
  8. 攻击路径 40[0] → 38[0] → 33[0] → 32[0] → 1[0] → 26[0] → 25[0] → 12[0] → **11[0]** → 22[3] → 21[3] → 20[3] → 11[3] → 31[3] → 1[3] 即为云平台攻击图的攻击路径与服务区 3 局部攻击图的第一条攻击路径的组合。这条攻击路径的风险值为 62.88。
  9. 攻击路径 40[0] → 38[0] → 33[0] → 32[0] → 1[0] → 26[0] → 25[0] → 12[0] → **11[0]** → 22[3] → 21[3] → 20[3] → 11[3] → 10[3] → 7[3] → 2[3] → 1[3] 即为云平台攻击图的攻击路径与服务区 3 局部攻击图的第二条攻击路径的组合。这条攻击路径的风险值为 72.63。
  10. 攻击路径 40[0] → 38[0] → 33[0] → 32[0] → 1[0] → 26[0] → 25[0] → 12[0] → **11[0]** → 22[3] → 21[3] → 20[3] → 11[3] → 10[3] → 7[3] → 41[3] → 40[3] 即为云平台攻击图的攻击路径与服务区 3 局部攻击图的第三条攻击路径的组合。这条攻击路径的风险值为 70.82。
  11. 攻击路径 37[3] → 36[3] → 35[3] → 33[3] → 32[3] 即服务区 3 局部攻击图的第四条攻击路径。这条攻击路径的风险值为 9.59。

从以上全局贝叶斯攻击图分解可以看到，经过拼接后得到的全局贝叶斯攻击图，其攻击路径数量等于原来四个局部贝叶斯攻击图所有局部攻击路径的组合的数量，所以拼接算法并没有减少攻击场景中有效攻击路径的数量，保持了攻击图的正确性，也证明了拼接算法的正确性。

另外，经过基于攻击路径贝叶斯概率的车联网量化风险评估，该全局车联网当前的风险值为 145.0；风险值最大的攻击路径为 40[0] → 38[0] → 33[0] → 32[0] → 1[0] → 26[0] → 25[0] → 12[0] → **11[0]** → 22[3] → 21[3] → 20[3] → 11[3] → 10[3] → 7[3] → 2[3] → 1[3]，其风险值为 72.63；而前三个风险值最大的节点分别为 1[0]、11[0]，24[1] 与 11[3] 均为第三，它们的风险值分别为 23.86、18.97、16.29。根据表 4-1，攻击场景测试用例中的车联网此时整体风险等级为极高，极高风险等级的攻击路径有 6 条，极高风险等级的风险节点有 4 个。其中高风险节点一般在高风险攻击路径的前端，这也是风险评估所基于的贝叶斯概率计算的特性。车联网网络安全管理人员可以通过处理高风险攻击路径上的前端节点，或直接处理高风险节点来降低整个车联网的风险等级。

### 4.2.3 动态车联网实时攻击图生成与分析测试

上一小节我们验证了本文提出的车联网攻击图生成与分析原型系统的正确性。这一小节，我们将通过更改车联网安全信息来模拟车联网的实时变化，从而来验证实时攻击图生成与分析功能的正确性。车联网的初始场景仍如表 4-2 所示。

#### (1) 拓扑变化

在某一时刻，原来在服务区 3 的 vehicle9 与 mobile9 移动到了服务区 2。根据本文提出的实时安全信息收集方案，这一拓扑变化将会被服务区 2 和服务区 3 的 MEC 服务器收集。vehicle9 与 mobile9 的安全信息将会在服务区 3 的 MEC 服务器中被标为失效，而服务区 2 的 MEC 服务器将会收集到 vehicle9 与 mobile9 的安全信息并标为有效。实际实现时，我们通过删除服务区 2 的 Datalog 安全信息文件中与 vehicle9 与 mobile9 相关的安全信息条目，并在服务区 3 的 Datalog 安全信息文件中加入 vehicle9 与 mobile9 的安全信息条目，来模拟实现这一过程。服务区的 Datalog 安全信息文件被更改代表收集到了车联网蜂窝服务区的安全信息变化。我们可以立即看到，云平台虚拟机更新了实时全局攻击图和描述服务区 2 与服务区 3 攻击图的 XML 文件，其延时将在小节 4.2.5 讨论。攻击图变化的部分如图 4-15 所示。

从拓扑变化后的全局贝叶斯攻击图可见，车联网在拓扑变化后风险值降低为了 136.03，仍为极高风险等级；极高风险等级的攻击路径减少为了 4 条，最高风险路径与初始场景时相同；极高风险等级的节点减少为了 3 个。可以看到，由于 vehicle9 移动到了蜂窝服务区 2 中，攻击者无法通过近程 V2V 与仍在蜂窝服务区 3 中的 vehicle10 通信，也就无法利用 vehicle10 上的可物理近程利用的验证绕过节点，所以更新后的攻击图中经过 vehicle10 的两条攻击路径消失（原全局攻击图的第 8、第 9 条攻击路径），服务区 3 只剩下一条攻击路径（6[3] 以下部分）。由于 vehicle9 上的可远程利用的漏洞仍存在，经过 vehicle9 的攻击路径不变（节点 42[2] 以下部分，从节点编号可以看到 vehicle9 的攻击路径现在位于服务区 2 中），但风险值减少了，这是因为其对车联网造成的直接损害降低了（无法再攻陷 vehicle10）。没有新的攻击路径出现。

#### (2) 漏洞变化

在 vehicle9 与 mobile9 移动到了服务区 2 后的某一时刻，服务区 3 的智能移动终端 mobile8 从网上安装了一个老版本的 chrome 浏览器，该 chrome 浏览器存在可被远程利用的客户端提权漏洞 CVE-2021-21220。通过与之前同样的方法，我们模拟车联网实时安全信息变化，并在云平台虚拟机得到了更新后的实时攻击图，如图 4-16 所示。

从漏洞增加后的全局贝叶斯攻击图可见，车联网在漏洞变化后风险值升高为了 169.9，仍为极高风险等级；极高风险等级的攻击路径增加为了 7 条，经过 mobile8 的攻击路径代替了 vehicle9 成为了最高风险的攻击路径；极高风险等级的节点增加为了 4 个。由于 mobile8 上新增的可被远程利用的客户端提权漏洞 CVE-2021-21220，可在云平台 Web 服务器以 user 权限执行任意代码的攻击者得以攻陷 mobile8，在 mobile8 上以 user 权限执行任意代码。这也使得 mobile8 上原本就存在于组件 wifidriver 的本地提权漏洞 CVE-2017-6424 暴露出来。攻击者在 mobile8 本地利用该漏洞使自己得到了 mobile8 的 root 权限。由于 mobile8 通过蓝牙与 vehicle8 配对，拥有 mobile8 的 root 权限的攻击者也就获得了 vehicle8 的控制权。同时，mobile8 也代替了刚离开服务区 3 的 vehicle9，通过 V2X 通信物理近程利用 vehicle10 上的验证绕过漏洞 CVE-2016-2354，获得了 vehicle10 的控制权。相比于更新前的攻击图，增加一

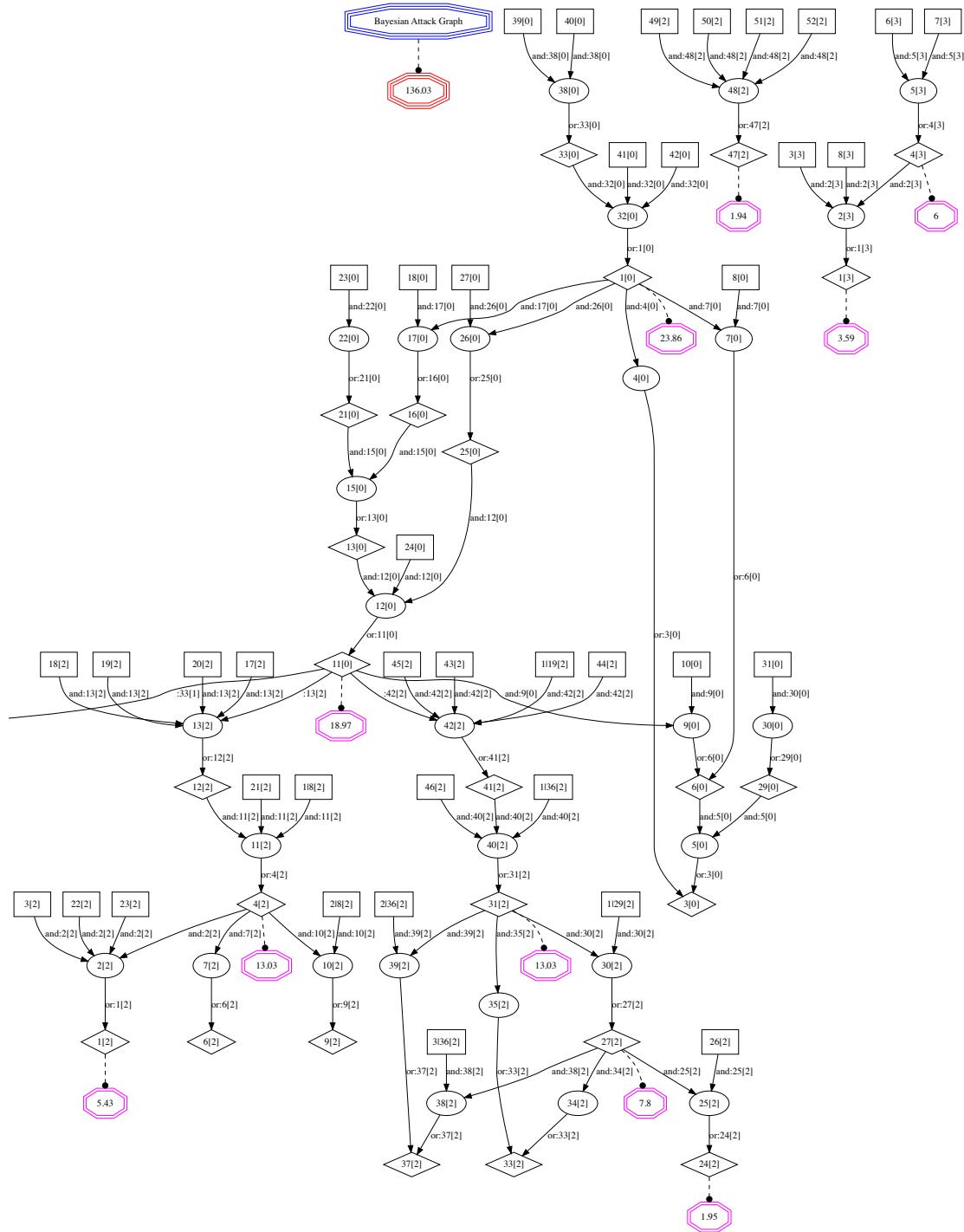


图 4-15 拓扑变化后的全局贝叶斯攻击图（变化部分）  
Figure 4-15 Global Bayesian attack graph after topology change



个漏洞使本次更新后的攻击图增加了三条路径（新攻击图的节点 22[3] 以下部分），多获得了两辆汽车的控制权，风险值增加了 33.9。其中“execCode(cloudplat\_webserver1,user)”节点，也即最新攻击图中的节点 38 的风险值显著提高为 22.76，这是因为它对车联网造成的直接损害增大的缘故（利用了 mobile8 的新漏洞）。

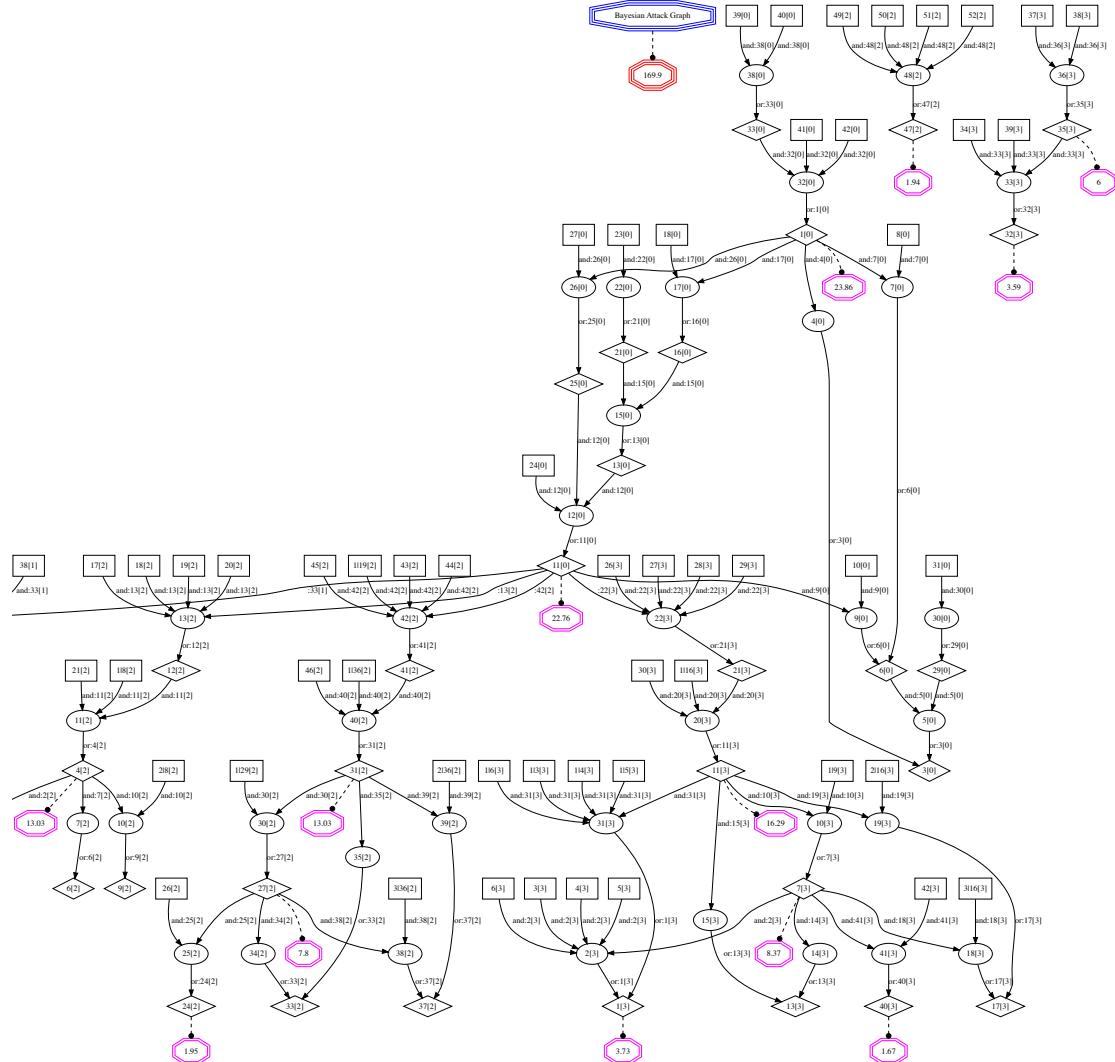


图 4-16 漏洞变化后的全局贝叶斯攻击图（变化部分）

Figure 4-16 Global Bayesian attack graph after vulnerability change

### (3) 车辆状态变化

在服务区 3 的 mobile8 安装了一个存在漏洞的浏览器后的某个时刻，服务区 2 的特斯拉汽车 vehicle5 停止了运动并被锁上。通过与之前同样的方法，我们模拟车联网实时安全信息变化，并在云平台虚拟机得到了更新后的实时攻击图，如图 4-17 所示。

从车辆状态变化后的全局贝叶斯攻击图可见，车联网在车辆状态变化后风险值升高为了 176.09，仍为极高风险等级；极高风险等级的攻击路径保持为 7 条，经过 mobile8 的攻击路径仍为最高风险的攻击路径；极高风险等级的节点保持为 4 个。vehicle5 停止了运动并被锁上后，道路上的物理攻击者获得了物理近程接触 vehicle5 的能力，因此原来由于车

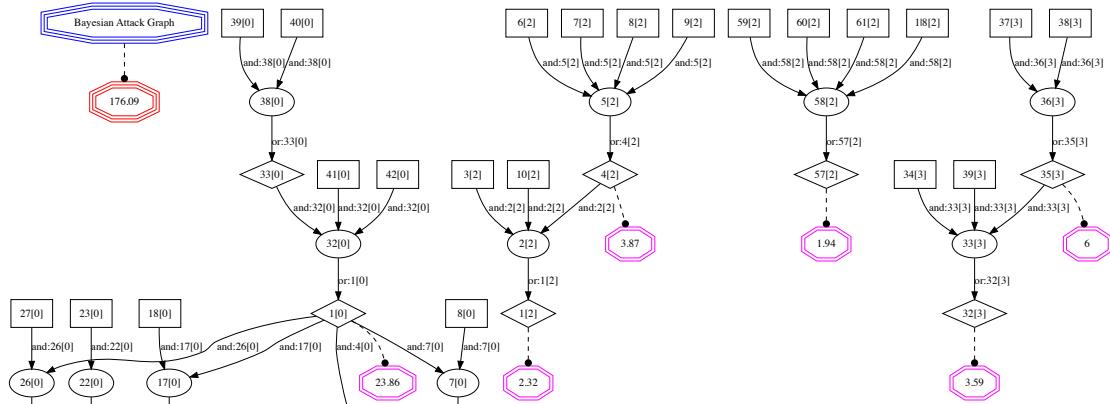


图 4-17 车辆状态变化后的全局贝叶斯攻击图（变化部分）

Figure 4-17 Global Bayesian attack graph after vehicle state change

辆运动而无法利用的 vehicle5 上的 keyFobs 组件中存在的可物理近程利用的信息泄露漏洞 CVE-2020-29438 暴露给了攻击者，攻击者可以通过利用该漏洞获取车辆开锁密钥，进入车内。而 vehicle5 上的 obdTeslaX 组件存在可物理本地利用的验证绕过漏洞 CVE-2020-29440，攻击者利用该漏洞进行 CAN 总线命令注入，获得了 vehicle5 的控制权。相比于更新前的攻击图，vehicle5 停止运动并上锁使攻击图增加了一条攻击路径 ( $6[2] \rightarrow 5[2] \rightarrow 4[2] \rightarrow 2[2] \rightarrow 1[2]$ )，多获得了一辆车的控制权，风险值增加了 6.19。攻击图其余部分未发生变化，其他攻击路径与风险节点对车联网造成的直接损失不变，因此风险值也不变。

#### 4.2.4 车联网攻击图应用示例

基于攻击图的网络安全量化风险评估技术通过计算网络局部和整体的量化风险，给出安全管理的建议。本小节将基于初始攻击测试场景的全局贝叶斯攻击图，给出对该测试场景中车联网的合理的安全管理建议，来降低其安全风险，作为本文车联网量化风险评估系统的一个应用示例，证明原型系统对于辅助车联网安全管理的有效性。

根据图 4-13、4-14，此时的车联网整体风险值为 145.0，风险等级为极高。要降低该车联网的安全风险，需要首先从风险值最高的攻击路径和风险节点入手，因为它们对车联网造成的损害最大，造成的次生攻击路径也最多；因此处理它们，对降低车联网安全风险的效果也就最大。另外，由于风险评估是基于贝叶斯概率计算，一般来说在攻击路径上越靠前端的风险节点的风险值越大，越靠末端的风险节点的风险值越小。处理风险值更高的风险节点，有助于从源头切断更多的攻击路径。

根据小节 4.2.2 的分析，攻击测试场景中风险值最高的攻击路径为  $40[0] \rightarrow 38[0] \rightarrow 33[0] \rightarrow 32[0] \rightarrow 1[0] \rightarrow 26[0] \rightarrow 25[0] \rightarrow 12[0] \rightarrow 11[0] \rightarrow 22[3] \rightarrow 21[3] \rightarrow 20[3] \rightarrow 11[3] \rightarrow 10[3] \rightarrow 7[3] \rightarrow 2[3] \rightarrow 1[3]$ ，其风险值为 72.63。这条攻击路径上的风险值最大的风险节点为节点  $1[0]$ ，内容为“execCode(cloudplat\_database1,mysql)”，即攻击者得到了在 cloudplat\_database1 上以 mysql 用户权限执行任意代码的能力。攻击图中也可见该节点上延伸出了多条攻击路径，对车联网造成的损害最大，所以其风险值也最大。我们考虑处理该节点。首先找到其父节点，其只有一个父节点  $32[0]$ ，这是一个原子攻击节点。使该原子攻击节点的任意前提条件失效则可以阻止该原子攻击发生，攻击者也就得不到节点  $1[0]$ 。我们找到节点  $32[0]$  的前提条件，也即它的父节点，分别为“ $33[0]:netAccess(cloudplat_database1,tcp,sqlPort)$ ”（攻

击者可以用 TCP 协议连接到 cloudplat\_database1 的 sqlPort 端口) 、“41[0]:networkServiceInfo(cloudplat\_database1,sql,tcp,sqlPort,mysql)” (cloudplat\_database1 上运行着 SQL 网络服务, 服务端口为 sqlPort) 、“42[0]:vulExists(cloudplat\_database1,’CVE-2015-1761’,sql,remoteExploit,privEscalation)” (cloudplat\_database1 上的 SQL 服务存在远程提权漏洞 CVE-2015-1761)。这里我们假设车联网云平台因为业务要求无法停止 cloudplat\_database1 的 SQL 服务, 那么网络安全管理人员可以处理的只有节点 42[0], 即修复漏洞 CVE-2015-1761。修复该漏洞后的车联网攻击图如图 4-18 所示。

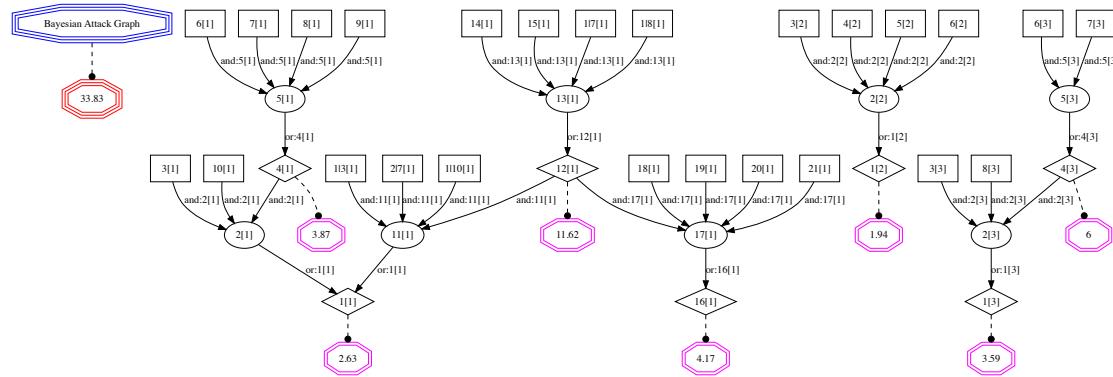


图 4-18 修复 SQL 服务漏洞后的全局贝叶斯攻击图

Figure 4-18 Global Bayesian attack graph after fixing the vulnerability in SQL service

从更新后的攻击图可见, 修复 SQL 服务漏洞后, 原来的最高风险节点 42[0] 消失, 车联网所有通过云端的攻击路径全部被切断, 只剩下了可以由道路上的物理攻击者利用的攻击路径。此时车联网整体的风险值降为了 33.83, 由极高风险等级降为了中风险等级; 无极高和很高风险等级的攻击路径, 高风险等级路径有 1 条; 无极高风险等级的风险节点, 很高风险等级的节点有 1 个。可以看到, 运用车联网贝叶斯攻击图可以非常有效地找到车联网中的关键节点。网络管理员对这些关键节点进行处理, 就可以有效降低车联网安全风险等级。

进一步, 若网络安全管理人员希望进一步降低车联网安全风险等级, 可以继续在新的贝叶斯攻击图中寻找风险值最大的攻击路径。最新攻击图中的最高风险路径为  $14[1] \rightarrow 13[1] \rightarrow 12[1] \rightarrow 17[1] \rightarrow 16[1]$ , 该路径上的最高风险节点为 12[1], 表示攻击者可以在 vehicle1 上以蓝牙进程所拥有的用户权限执行任意代码。其只有一个父节点, 即原子攻击节点 13[1], 该节点的所有前提条件为节点 14[1]、15[1]、117[1]、118[1], 其中 15[1]、117[1]、118[1] 难以改变, 所以网络安全管理人员可以通过向控制 bluetooth1 的用户实体 vehicle1Owner 发出告警, 要求其修复 bluetooth1 上的可物理远程利用的验证绕过漏洞 CVE-2020-0022 (14[1])。假设用户实体 vehicle1Owner 接受了告警建议并修复了该漏洞, 则新的车联网贝叶斯攻击图如图 4-19 所示。

从更新后的攻击图可见, 修复 SQL 服务漏洞后, 原来的最高风险节点 12[1] 消失, 只剩下了三条攻击路径, 分别为两条中风险路径和一条低风险路径。此时车联网整体的风险值降为了 17.72, 由中风险等级降为了低风险等级; 中风险节点有 4 个, 低风险节点有 1 个。值得一提的是, 到目前为止, 我们基于车联网贝叶斯攻击图只处理了两个漏洞, 就将车联网的风险值从 145.0 降到了 17.72, 从极高风险等级降为了低风险等级, 车联网内的攻击路径数从 11 条降为了风险值不高的 3 条, 风险节点的数量从 17 个降为了风险值不高的 5 个。而车联网中此时的漏洞数为 18 个, 却大多数无法再被攻击者利用。可见, 运用车联网贝叶斯

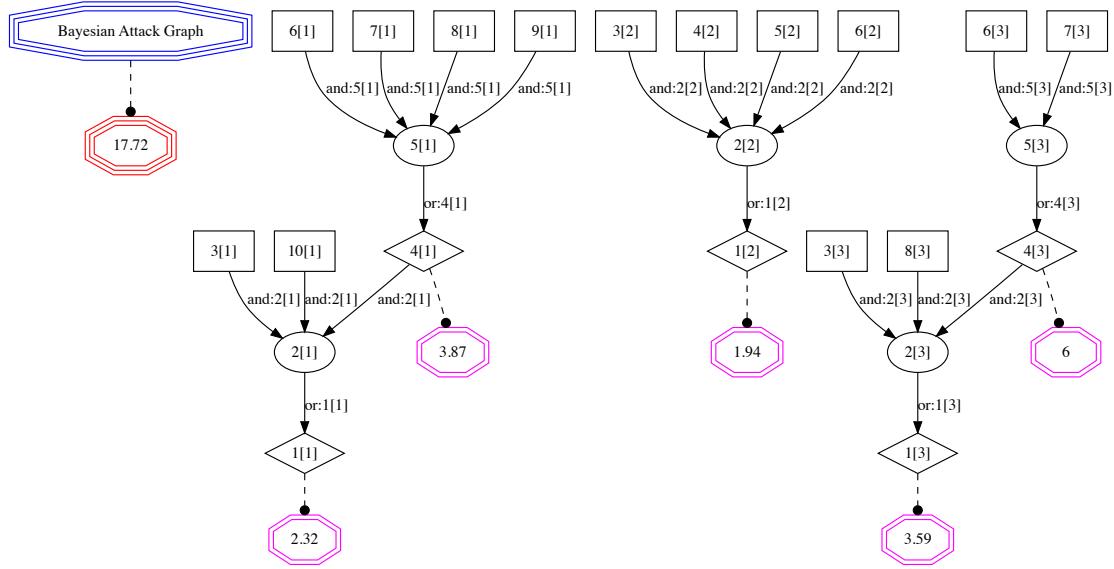


图 4-19 修复蓝牙漏洞后的全局贝叶斯攻击图

Figure 4-19 Global Bayesian attack graph after fixing the vulnerability in bluetooth

攻击图来辅助网络安全管理，可以准确地评估和有效地降低车联网网络安全风险。

#### 4.2.5 车联网实时量化风险评估性能测试

由小节 3.1.2 所述，车联网信息安全分析与评估面临的挑战中重要的一个挑战就是车联网动态性和网络复杂性对车联网安全分析提出的的实时性的性能要求。车联网安全风险分析的性能必须很高，能够快速对大规模的车联网安全信息进行分析，并低延时地给出分析结果。本小节将测试本文车联网实时攻击图生成与分析系统的性能，来验证该系统是否可以满足车联网的实时性要求。

本小节将以车联网中的主机数量作为网络规模的衡量指标，通过将本文的基于 MEC 的车联网实时攻击图分布式生成与分析的延迟 (CPU 时间，未包括不同虚拟机间的通信延迟)，与直接用 MulVAL 生成全局车联网攻击图再进行整体分析的延迟 (CPU 时间) 进行比较，来说明本文的车联网量化风险评估系统的性能优势。测试中，模拟的车联网中每个蜂窝服务区中的机器数量均为 10 个，用增加服务区数量的方法来增大网络规模，比较相同车联网规模下两种方法进行车联网量化风险评估的延迟。其生成全局攻击图的延迟随网络规模的变化曲线如图 4-20 所示。

可以看到，在单个服务区网络规模不变的情况下，不论车联网规模如何增大，本文的原型系统的量化风险评估延迟始终稳定在 0.9s 左右。这是由于采用 MEC 的方法，所有蜂窝服务区可以独立并行生成局部贝叶斯攻击图，因此生成所有局部贝叶斯攻击图的最大延迟只与最大的服务区网络规模有关。另外，由于全局贝叶斯攻击图生成采用局部攻击图拼接的更新方法，除了云平台局部攻击图发生变化的情况外，每次更新全局贝叶斯攻击图只需要把全局攻击图中旧的局部攻击图部分删除，然后拼接上改变后的局部攻击图，因此每次更新全局攻击图的时间复杂度也只和安全信息变化了的服务区内的网络规模有关，而与整个车联网的规模无关。本小节测试中所有服务区的网络规模均为常数 10，所以每次更新全局贝叶斯攻击图的时间复杂度也为常数。虽然图中的延迟并未包括 MEC 服务器与云平台的通信延迟，但是由于 MEC 服务器上传的攻击图描述文件大小也与服务区的网络规模成正比，

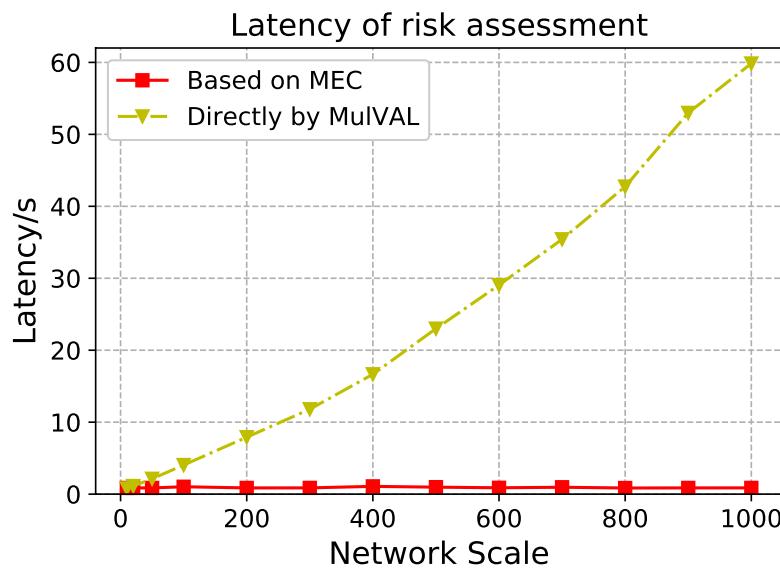


图 4-20 车联网量化风险评估延迟

Figure 4-20 Latency of quantitative IoV risk assessment

所以通信延迟在服务区网络规模不变的情况下也为常数。

然而，如果直接使用 MulVAL 生成全局攻击图，再进行整体分析，MulVAL 推导生成的时间复杂度  $O(n^2)$  与贝叶斯攻击图分析的时间复杂度  $O(n)$  将会非常明显地体现出来，尤其是在网络规模很大的时候，造成了非常大的延时。因此本文的基于 MEC 的车联网量化风险评估系统继承了 MEC 的低延迟的优点，相比于传统的直接生成攻击图的方法，其性能很好，且在网络规模持续增大的情况下，若单个服务区内的网络规模不变，仍能保持常数时间复杂度和稳定的低延迟。

### 4.3 本章小结

本章依据设计方案实现了基于 MEC 的车联网实时攻击图分布式生成和分析的原型系统，并对该原型系统进行了测试。首先，对攻击图生成工具 MulVAL 进行了简单介绍，并将基于车联网安全本体模型构建的原始规则集转化为了 Datalog 语言描述的规则集。其次，详细介绍了车联网实时攻击图生成与分析的实现，包括 MEC 服务器虚拟机中的实时安全信息收集、局部贝叶斯攻击图生成，以及云平台虚拟机中的攻击图规则集更新、局部贝叶斯攻击图生成、和全局贝叶斯攻击图拼接生成五个主要进程，这五个进程在不同的虚拟机中相互配合生成实时全局贝叶斯攻击图，进行车联网的量化风险评估。再次，本章着重介绍了基于贝叶斯攻击图的车联网量化风险评估的实现过程，包括攻击图去环、基于 CVSS 的攻击路径贝叶斯概率计算、以及风险值计算等几个重要步骤。然后，本章构建了一个较为复杂的攻击测试场景，对原型系统进行了正确性和有效性的测试。最后，本章还进一步对原型系统的实时性进行了测试，测试结果表明，基于 MEC 的车联网实时攻击图生成与分析系统相比于直接使用 MulVAL 生成攻击图再整体分析表现出了优越的性能。

## 第五章 总结与展望

### 5.1 本文主要内容总结

本文针对车联网中的网络安全分析与评估问题，提出了车联网安全本体模型，并使用该模型构建了车联网攻击图生成规则集，设计了基于车联网实时攻击图生成与分析的量化风险评估方案。本文基于该方案实现了原型系统，并设计了攻击场景测试用例，验证了原型系统的正确性、有效性和实时性，实现了预设的研究目标。本文主要创新点与贡献如下：

- 本文依据车联网的“云”、“管”、“端”三层架构，结合车联网面临的安全风险与安全需求，在通用网络安全本体模型的基础上构建了车联网安全本体模型。利用该本体模型可以形式化地描述车联网中的实体和实体间的复杂关系，以及车联网中的攻击路径。本文设计了典型的车联网攻击场景用例，验证了车联网安全本体模型的正确性和有效性。
- 基于所构建的车联网安全本体模型，本文对车联网安全知识库中的漏洞利用方法和攻击方法进行了形式化描述，构建了原始的车联网攻击图生成规则集，并在实现原型系统时将原始的规则集转化为 Datalog 表达的规则集。
- 通过总结车联网信息安全管理与评估技术所面临的挑战，本文创新性地提出了基于 C-V2X 通信架构与车联网边缘计算的实时攻击图生成与分析方案。该方案采用边缘计算的方式分布式生成局部攻击图，并通过基于贝叶斯攻击图的量化风险评估方案进行分析，得到局部贝叶斯攻击图，再上传到云平台中拼接成全局贝叶斯攻击图，来量化风险评估，将计算成本分摊到各服务区中，有效地降低了量化风险评估延迟。
- 本文在车联网实时攻击图生成的基础上提出了基于贝叶斯攻击图的车联网量化风险评估方案。其中，在攻击图预处理时提出了基于 DFS 的切除最深原子攻击节点的攻击图去环算法，然后在贝叶斯攻击图计算的基础上建立了车联网量化风险评估模型。
- 本文基于车联网实时攻击图生成与分析方案实现了原型系统，并设计了车联网攻击场景用例对该原型系统进行了测试，验证了其正确性、有效性和实时性。

### 5.2 局限性及展望

虽然本文提出的车联网实时攻击图生成与分析方案实现了预设的研究目标，但本文仍存在一定的局限性。

在构建车联网攻击图生成规则集时，规则集中的规则虽然能覆盖很多车联网中的攻击与漏洞利用，但仍不全面。后续研究还需要不断利用车联网安全本体模型来总结出更多的攻击图生成规则来补充规则集，使规则集更全面。

此外，虽然使用 MEC 的方法分布式生成局部攻击图，可以有效地提高生成攻击图的效率，但将车联网按服务区分割成子网独立生成攻击图，且仅允许服务区知晓云平台的部分安全信息（“特殊攻击者”信息），也使得该方法无法推导出从车联网端层到云平台方向的攻击路径和跨服务区的攻击路径。之后的研究将关注如何合理选择共享至服务区的云平台安全信息和服务区之间共享的安全信息，以及如何提升拼接算法的安全性和有效性，使系统能生成从端层到云平台方向以及跨服务区的攻击路径，从而更全面地进行车联网量化风险评估。

另外，虽然在测试中原型系统相比于直接生成全局攻击图表现出了很好的实时性，但该实时性仅存在于车联网的通信节点密度在整个车联网均匀分布，各服务区网络规模均不大的情况下。若将本文的方案应用到实际，可能会出现某个服务区网络规模明显很大的情况，如堵车。那么在更新该服务区的局部贝叶斯攻击图时，本文的方案仍会产生较明显的延迟。后续研究可以讨论在某服务区出现异常大网络规模时的处理方法。

最后，由于时间、精力和设备所限，现阶段采用的是虚拟机来实现和测试原型系统。在今后的研究中，可以在真实的车联网场景中实现并测试本文提出的方案，以得到更实用、科学的结果。

## 附录 A 攻击图节点完整信息

表 A-1 攻击图节点完整信息

Table A-1 Complete information of attack graph nodes

| 位置    | 节点编号      |           |          |          |            |          |              | 节点内容  |
|-------|-----------|-----------|----------|----------|------------|----------|--------------|---|
|       | 局部<br>攻击图 | 拼接<br>攻击图 | 拓扑<br>变化 | 漏洞<br>变化 | 车辆状<br>态变化 | 修复<br>漏洞 | 修复 SQL<br>漏洞 |   |
|       | 1         | 1[0]      | 1[0]     | 1[0]     | 1[0]       | -        | -            | execCode(cloudplat_database1.mysql)   |
|       | 3         | 3[0]      | 3[0]     | 3[0]     | 3[0]       | -        | -            | canAccessHost(cloudplat_database1)  |
|       | 4         | 4[0]      | 4[0]     | 4[0]     | 4[0]       | -        | -            | RULE 22 (Access a host through executing code on the machine)                                       |
|       | 5         | 5[0]      | 5[0]     | 5[0]     | 5[0]       | -        | -            | RULE 23 (Access a host through a log-in service)  |
|       | 6         | 6[0]      | 6[0]     | 6[0]     | 6[0]       | -        | -            | netAccess(cloudplat_database1.tcp.sshPort)  |
|       | 7         | 7[0]      | 7[0]     | 7[0]     | 7[0]       | -        | -            | RULE 19 (multi-hop access)  |
|       | 8         | 8[0]      | 8[0]     | 8[0]     | 8[0]       | -        | -            | hacl(cloudplat_database1,cloudplat_database1.tcp.sshPort)   |
|       | 9         | 9[0]      | 9[0]     | 9[0]     | 9[0]       | -        | -            | RULE 19 (multi-hop access)  |
|       | 10        | 10[0]     | 10[0]    | 10[0]    | 10[0]      | -        | -            | hacl(cloudplat_webserver1,cloudplat_database1.tcp.sshPort)  |
|       | 11        | 11[0]     | 11[0]    | 11[0]    | 11[0]      | -        | -            | execCode(cloudplat_webserver1.user)   |
|       | 12        | 12[0]     | 12[0]    | 12[0]    | 12[0]      | -        | -            | RULE 14 (When a principal is compromised any machine he has an account on will also be compromised) |
|       | 13        | 13[0]     | 13[0]    | 13[0]    | 13[0]      | -        | -            | canAccessHost(cloudplat_webserver1)   |
|       | 15        | 15[0]     | 15[0]    | 15[0]    | 15[0]      | -        | -            | RULE 23 (Access a host through a log-in service)  |
|       | 16        | 16[0]     | 16[0]    | 16[0]    | 16[0]      | -        | -            | netAccess(cloudplat_webserver1.tcp.sshPort)   |
|       | 17        | 17[0]     | 17[0]    | 17[0]    | 17[0]      | -        | -            | RULE 19 (multi-hop access)  |
|       | 18        | 18[0]     | 18[0]    | 18[0]    | 18[0]      | -        | -            | hacl(cloudplat_database1,cloudplat_webserver1.tcp.sshPort)  |
| 云平台   | 21        | 21[0]     | 21[0]    | 21[0]    | 21[0]      | -        | -            | logInService(cloudplat_webserver1.tcp.sshPort)  |
|       | 22        | 22[0]     | 22[0]    | 22[0]    | 22[0]      | -        | -            | RULE 27 (Provide login service ssh)   |
|       | 23        | 23[0]     | 23[0]    | 23[0]    | 23[0]      | -        | -            | networkServiceInfo(cloudplat_webserver1.sshd.tcp.sshPort,ssh)                                       |
|       | 24        | 24[0]     | 24[0]    | 24[0]    | 24[0]      | -        | -            | hasAccount(employee1,cloudplat_webserver1.user)   |
|       | 25        | 25[0]     | 25[0]    | 25[0]    | 25[0]      | -        | -            | principalCompromised(employee1)   |
|       | 26        | 26[0]     | 26[0]    | 26[0]    | 26[0]      | -        | -            | RULE 26 (password sniffing)   |
|       | 27        | 27[0]     | 27[0]    | 27[0]    | 27[0]      | -        | -            | hasAccount(employee1,cloudplat_database1.mysql)   |
|       | 29        | 29[0]     | 29[0]    | 29[0]    | 29[0]      | -        | -            | logInService(cloudplat_database1.tcp.sshPort)   |
|       | 30        | 30[0]     | 30[0]    | 30[0]    | 30[0]      | -        | -            | RULE 27 (Provide login service ssh)   |
|       | 31        | 31[0]     | 31[0]    | 31[0]    | 31[0]      | -        | -            | networkServiceInfo(cloudplat_database1.sshd.tcp.sshPort,ssh)  |
|       | 32        | 32[0]     | 32[0]    | 32[0]    | 32[0]      | -        | -            | RULE 16 (remote exploit of a server program)  |
|       | 33        | 33[0]     | 33[0]    | 33[0]    | 33[0]      | -        | -            | netAccess(cloudplat_database1.tcp.sqlPort)  |
|       | 38        | 38[0]     | 38[0]    | 38[0]    | 38[0]      | -        | -            | RULE 20 (direct network access)   |
|       | 39        | 39[0]     | 39[0]    | 39[0]    | 39[0]      | -        | -            | hacl(internet,cloudplat_database1.tcp.sqlPort)  |
|       | 40        | 40[0]     | 40[0]    | 40[0]    | 40[0]      | -        | -            | attackerLocated(internet)   |
|       | 41        | 41[0]     | 41[0]    | 41[0]    | 41[0]      | -        | -            | networkServiceInfo(cloudplat_database1.sql.tcp.sqlPort,mysql)                                       |
|       | 42        | 42[0]     | 42[0]    | 42[0]    | 42[0]      | -        | -            | vulExists(cloudplat_database1,'CVE-2015-1761',sql.remoteExploit.privEscalation)                     |
|       | 1         | 1[1]      | 1[1]     | 1[1]     | 1[1]       | 1[1]     | 1[1]         | control(vehicle1)   |
|       | 2         | 2[1]      | 2[1]     | 2[1]     | 2[1]       | 2[1]     | 2[1]         | RULE 4 (Local bypass to control)  |
|       | 3         | 3[1]      | 3[1]     | 3[1]     | 3[1]       | 3[1]     | 3[1]         | vulExists(vehicle1,'CVE-2020-29440',obdTeslaX.phyLocalExploit,verifiBypass)                         |
|       | 4         | 4[1]      | 4[1]     | 4[1]     | 4[1]       | 4[1]     | 4[1]         | inside(vehicle1)  |
|       | 5         | 5[1]      | 5[1]     | 5[1]     | 5[1]       | 5[1]     | 5[1]         | RULE 1 (Leak the password to unlock)  |
|       | 6         | 6[1]      | 6[1]     | 6[1]     | 6[1]       | 6[1]     | 6[1]         | vulExists(vehicle1,'CVE-2020-29438',keyFobs,phyShortExploit,infoLeak)                               |
|       | 7         | 7[1]      | 7[1]     | 7[1]     | 7[1]       | 7[1]     | 7[1]         | vState(vehicle1.locked)   |
|       | 8         | 8[1]      | 8[1]     | 8[1]     | 8[1]       | 8[1]     | 8[1]         | attackerLocated(physics)  |
|       | 9         | 9[1]      | 9[1]     | 9[1]     | 9[1]       | 9[1]     | 9[1]         | installed(vehicle1,keyFobs)   |
|       | 10        | 10[1]     | 10[1]    | 10[1]    | 10[1]      | 10[1]    | 10[1]        | installed(vehicle1,obdTeslaX)   |
|       | 11        | 11[1]     | 11[1]    | 11[1]    | 11[1]      | 11[1]    | -            | RULE 6 (Leverage the local OS to bypass to control)   |
|       | 12        | 12[1]     | 12[1]    | 12[1]    | 12[1]      | 12[1]    | -            | execCode(vehicle1,'CVE-2020-0022')  |
|       | 13        | 13[1]     | 13[1]    | 13[1]    | 13[1]      | 13[1]    | -            | RULE 13 (Short range channel bypass to execute)   |
|       | 14        | 14[1]     | 14[1]    | 14[1]    | 14[1]      | 14[1]    | -            | vulExists(blueooth1,'CVE-2020-0022',daemons,phyShortExploit,verifiBypass)                           |
| 服务区 1 | 15        | 15[1]     | 15[1]    | 15[1]    | 15[1]      | 15[1]    | -            | pair/mobile1,vehicle1,bluetooth1)   |
|       | 16        | 16[1]     | 16[1]    | 16[1]    | 16[1]      | 16[1]    | -            | control(vehicle2)   |
|       | 17        | 17[1]     | 17[1]    | 17[1]    | 17[1]      | 17[1]    | -            | RULE 8 (V2X communication exploit)  |
|       | 18        | 18[1]     | 18[1]    | 18[1]    | 18[1]      | 18[1]    | -            | vulExists(vehicle2,'CVE-2018-11478',obd2,phyShortExploit,verifiBypass)                              |
|       | 19        | 19[1]     | 19[1]    | 19[1]    | 19[1]      | 19[1]    | -            | vState(vehicle2.mov)  |
|       | 20        | 20[1]     | 20[1]    | 20[1]    | 20[1]      | 20[1]    | -            | installed(vehicle2,obd2)  |
|       | 21        | 21[1]     | 21[1]    | 21[1]    | 21[1]      | 21[1]    | -            | hacl(vehicle1,vehicle2,_)   |
|       | 22        | 22[1]     | 22[1]    | 22[1]    | 22[1]      | -        | -            | RULE 8 (V2X communication exploit)  |
|       | 23        | 23[1]     | 23[1]    | 23[1]    | 23[1]      | -        | -            | hacl(vehicle3,vehicle2,_)   |
|       | 24        | 24[1]     | 24[1]    | 24[1]    | 24[1]      | -        | -            | execCode(vehicle3,user)   |
|       | 26        | 26[1]     | 26[1]    | 26[1]    | 26[1]      | -        | -            | canAccessHost(vehicle3)   |
|       | 27        | 27[1]     | 27[1]    | 27[1]    | 27[1]      | -        | -            | RULE 22 (Access a host through executing code on the machine)                                       |
|       | 29        | 29[1]     | 29[1]    | 29[1]    | 29[1]      | -        | -            | principalCompromised(vehicle3Owner)   |
|       | 30        | 30[1]     | 30[1]    | 30[1]    | 30[1]      | -        | -            | RULE 26 (password sniffing)   |
|       | 31        | 31[1]     | 31[1]    | 31[1]    | 31[1]      | -        | -            | RULE 17 (remote exploit for a client program)   |
|       | 32        | 32[1]     | 32[1]    | 32[1]    | 32[1]      | -        | -            | accessMaliciousInput(vehicle3,vehicle3Owner,chrome)   |

续下页



续表 A-1

|      |         |         |         |         |         |      |   |
|------|---------|---------|---------|---------|---------|------|---|
| 33   | 33[1]   | 33[1]   | 33[1]   | 33[1]   | -       | -    | RULE 38 (Browsing a compromised website)                                    |
| 37   | 37[1]   | 37[1]   | 37[1]   | 37[1]   | -       | -    | hal(vehicle3,cloudplat_webserver1,httpProtocol,httpPort)                    |
| 38   | 38[1]   | 38[1]   | 38[1]   | 38[1]   | -       | -    | clientProgram(vehicle3,chrome)  |
| 39   | 39[1]   | 39[1]   | 39[1]   | 39[1]   | -       | -    | isWebServer(cloudplat_webserver1)   |
| 40   | 40[1]   | 40[1]   | 40[1]   | 40[1]   | -       | -    | inCompetent(vehicle3Owner)  |
| 41   | 41[1]   | 41[1]   | 41[1]   | 41[1]   | -       | -    | vulExists(vehicle3,'CVE-2021-21220',chrome,remoteClient,privEscalation)     |
| 42   | 42[1]   | 42[1]   | 42[1]   | 42[1]   | -       | -    | control(vehicle3)   |
| 43   | 43[1]   | 43[1]   | 43[1]   | 43[1]   | -       | -    | RULE 7 (Leverage local OS to bypass to control)                             |
| 44   | 44[1]   | 44[1]   | 44[1]   | 44[1]   | -       | -    | vulExists(vehicle3,'CVE-2020-8539',micomd,localExploit,verifiBypass)        |
| 45   | 45[1]   | 45[1]   | 45[1]   | 45[1]   | -       | -    | vState(vehicle3,mov)  |
| 46   | 46[1]   | 46[1]   | 46[1]   | 46[1]   | -       | -    | installed(vehicle3,micomd)  |
| 113  | 113[1]  | 113[1]  | 113[1]  | 113[1]  | 113[1]  | -    | vulExists(vehicle1,'CVE-2020-29440',obdTeslaX,phyLocalExploit,verifiBypass) |
| 117  | 117[1]  | 117[1]  | 117[1]  | 117[1]  | 117[1]  | -    | vState(vehicle1,locked)   |
| 217  | 217[1]  | 217[1]  | 217[1]  | 217[1]  | 217[1]  | -    | vState(vehicle1,locked)   |
| 118  | 118[1]  | 118[1]  | 118[1]  | 118[1]  | 118[1]  | -    | attackerLocated(physics)  |
| 1110 | 1110[1] | 1110[1] | 1110[1] | 1110[1] | 1110[1] | -    | installed(vehicle1,obdTeslaX)   |
| 1118 | 1118[1] | 1118[1] | 1118[1] | 1118[1] | -       | -    | vulExists(vehicle2,'CVE-2018-11478',obd2,phyShortExploit,verifiBypass)      |
| 1119 | 1119[1] | 1119[1] | 1119[1] | 1119[1] | -       | -    | vState(vehicle2,mov)  |
| 1120 | 1120[1] | 1120[1] | 1120[1] | 1120[1] | -       | -    | installed(vehicle2,obd2)  |
| 1128 | 1128[1] | 1128[1] | 1128[1] | 1128[1] | -       | -    | hasAccount(vehicle3Owner,vehicle3,user)                                     |
| 2128 | 2128[1] | 2128[1] | 2128[1] | 2128[1] | -       | -    | hasAccount(vehicle3Owner,vehicle3,user)                                     |
| 36   | -       | -       | -       | -       | -       | -    | pexecCode(cloudplat_webserver1,user)  |
| 35   | -       | -       | -       | -       | -       | -    | RULE 39 (PV rule)   |
| 34   | -       | -       | -       | -       | -       | -    | execCode(cloudplat_webserver1,user)   |
| 1    | 1[2]    | 1[2]    | 1[2]    | 11[2]   | -       | -    | control(vehicle7)   |
| 2    | 2[2]    | 2[2]    | 2[2]    | 12[2]   | -       | -    | RULE 9 (Compromised APP control vehicle)                                    |
| 3    | 3[2]    | 3[2]    | 3[2]    | 13[2]   | -       | -    | vulExists(mobile7,vulID,teslaAPP,localExploit,verifiBypass)                 |
| 4    | 4[2]    | 4[2]    | 4[2]    | 14[2]   | -       | -    | execCode(mobile7,user)  |
| 6    | 6[2]    | 6[2]    | 6[2]    | 16[2]   | -       | -    | canAccessHost(mobile7)  |
| 7    | 7[2]    | 7[2]    | 7[2]    | 17[2]   | -       | -    | RULE 22 (Access a host through executing code on the machine)               |
| 9    | 9[2]    | 9[2]    | 9[2]    | 19[2]   | -       | -    | principalCompromised(vehicle7Owner)   |
| 10   | 10[2]   | 10[2]   | 10[2]   | 20[2]   | -       | -    | RULE 26 (password sniffing)   |
| 11   | 11[2]   | 11[2]   | 11[2]   | 21[2]   | -       | -    | RULE 17 (remote exploit for a client program)                               |
| 12   | 12[2]   | 12[2]   | 12[2]   | 22[2]   | -       | -    | accessMaliciousInput(mobile7,vehicle7Owner,chrome)                          |
| 13   | 13[2]   | 13[2]   | 13[2]   | 23[2]   | -       | -    | RULE 38 (Browsing a compromised website)                                    |
| 17   | 17[2]   | 17[2]   | 17[2]   | 27[2]   | -       | -    | hal(mobile7,cloudplat_webserver1,httpProtocol,httpPort)                     |
| 18   | 18[2]   | 18[2]   | 18[2]   | 28[2]   | -       | -    | clientProgram(mobile7,chrome)   |
| 19   | 19[2]   | 19[2]   | 19[2]   | 29[2]   | -       | -    | isWebServer(cloudplat_webserver1)   |
| 20   | 20[2]   | 20[2]   | 20[2]   | 30[2]   | -       | -    | inCompetent(vehicle7Owner)  |
| 21   | 21[2]   | 21[2]   | 21[2]   | 31[2]   | -       | -    | vulExists(mobile7,'CVE-2021-21220',chrome,remoteClient,privEscalation)      |
| 22   | 22[2]   | 22[2]   | 22[2]   | 32[2]   | -       | -    | pair(mobile7,vehicle7,bluetooth7)   |
| 23   | 23[2]   | 23[2]   | 23[2]   | 33[2]   | -       | -    | installed(mobile7,teslaAPP)   |
| 24   | 24[2]   | 47[2]   | 47[2]   | 57[2]   | 1[2]    | 1[2] | inside(vehicle6)  |
| 25   | 25[2]   | 48[2]   | 48[2]   | 58[2]   | 2[2]    | 2[2] | RULE 12 (Sniff infomation from channel to unlock)                           |
| 26   | 26[2]   | 49[2]   | 49[2]   | 59[2]   | 3[2]    | 3[2] | vulExists(wifi6,'CVE-2018-11477',vigateCar2WiFi,phyShortExploit,infoLeak)   |
| 27   | 27[2]   | 50[2]   | 50[2]   | 60[2]   | 4[2]    | 4[2] | vState(vehicle6,locked)   |
| 28   | 28[2]   | 51[2]   | 51[2]   | 61[2]   | 5[2]    | 5[2] | attackerLocated(physics)  |
| 29   | 29[2]   | 52[2]   | 52[2]   | 62[2]   | 6[2]    | 6[2] | pair(mobile6,vehicle6,wifi6)  |
| 118  | 118[2]  | 21[2]   | 21[2]   | 31[2]   | -       | -    | hasAccount(vehicle7Owner,mobile7,user)                                      |
| 218  | 218[2]  | 218[2]  | 218[2]  | 218[2]  | -       | -    | hasAccount(vehicle7Owner,mobile7,user)                                      |
| 16   | -       | -       | -       | -       | -       | -    | pexecCode(cloudplat_webserver1,user)  |
| 15   | -       | -       | -       | -       | -       | -    | RULE 39 (PV rule)   |
| 14   | -       | -       | -       | -       | -       | -    | execCode(cloudplat_webserver1,user)   |
| -    | -       | 45[2]   | 45[2]   | 55[2]   | -       | -    | inCompetent(vehicle9Owner)  |
| -    | -       | 1119[2] | 1119[2] | 1129[2] | -       | -    | isWebServer(cloudplat_webserver1)   |
| -    | -       | 43[2]   | 43[2]   | 53[2]   | -       | -    | hal(vehicle9,cloudplat_webserver1,httpProtocol,httpPort)                    |
| -    | -       | 44[2]   | 44[2]   | 54[2]   | -       | -    | clientProgram(vehicle9,chrome)  |
| -    | -       | 42[2]   | 42[2]   | 52[2]   | -       | -    | RULE 38 (Browsing a compromised website)                                    |
| -    | -       | 46[2]   | 46[2]   | 56[2]   | -       | -    | vulExists(vehicle9,'CVE-2021-21220',chrome,remoteClient,privEscalation)     |
| -    | -       | 41[2]   | 41[2]   | 51[2]   | -       | -    | accessMaliciousInput(vehicle9,vehicle9Owner,chrome)                         |
| -    | -       | 40[2]   | 40[2]   | 50[2]   | -       | -    | RULE 17 (remote exploit for a client program)                               |
| -    | -       | 1129[2] | 1129[2] | 1139[2] | -       | -    | vulExists(vehicle9,'CVE-2021-3156',sudo,localExploit,privEscalation)        |
| -    | -       | 31[2]   | 31[2]   | 41[2]   | -       | -    | execCode(vehicle9,user)   |
| -    | -       | 1136[2] | 1136[2] | 1146[2] | -       | -    | hasAccount(vehicle9Owner,vehicle9,user)                                     |
| -    | -       | 2136[2] | 2136[2] | 2146[2] | -       | -    | hasAccount(vehicle9Owner,vehicle9,user)                                     |
| -    | -       | 3136[2] | 3136[2] | 3146[2] | -       | -    | hasAccount(vehicle9Owner,vehicle9,user)                                     |
| -    | -       | 39[2]   | 39[2]   | 49[2]   | -       | -    | RULE 26 (password sniffing)   |
| -    | -       | 30[2]   | 30[2]   | 40[2]   | -       | -    | RULE 15 (local exploit)   |
| -    | -       | 26[2]   | 26[2]   | 36[2]   | -       | -    | vState(vehicle9,mov)  |
| -    | -       | 27[2]   | 27[2]   | 37[2]   | -       | -    | execCode(vehicle9,root)   |
| -    | -       | 38[2]   | 38[2]   | 48[2]   | -       | -    | RULE 25 (password sniffing)   |
| -    | -       | 34[2]   | 34[2]   | 44[2]   | -       | -    | RULE 22 (Access a host through executing code on the machine)               |
| -    | -       | 25[2]   | 25[2]   | 35[2]   | -       | -    | RULE 0 (The vehicle is controlled when the OS is compromised)               |
| -    | -       | 24[2]   | 24[2]   | 34[2]   | -       | -    | control(vehicle9)   |
| -    | -       | 33[2]   | 33[2]   | 43[2]   | -       | -    | canAccessHost(vehicle9)   |
| -    | -       | 37[2]   | 37[2]   | 47[2]   | -       | -    | principalCompromised(vehicle9Owner)   |
| -    | -       | -       | -       | 1[2]    | -       | -    | control(vehicle5)   |

续下页

续表 A-1

|       |         |      |         |         |      |      |  |
|-------|---------|------|---------|---------|------|------|--|
| -     | -       | -    | -       | 2[2]    | -    | -    | RULE 4 (Local bypass to control)   |
| -     | -       | -    | -       | 3[2]    | -    | -    | vulExists(vehicle5,'CVE-2020-29440',obdTeslaX,phyLocalExploit,verifiBypass)      |
| -     | -       | -    | -       | 4[2]    | -    | -    | inside(vehicle5)   |
| -     | -       | -    | -       | 5[2]    | -    | -    | RULE 1 (Leak the password to unlock)   |
| 服务区 2 | -       | -    | -       | 6[2]    | -    | -    | vulExists(vehicle5,'CVE-2020-29438',keyFobs,phyShortExploit,infoLeak)            |
|       | -       | -    | -       | 7[2]    | -    | -    | vState(vehicle5,locked)  |
|       | -       | -    | -       | 8[2]    | -    | -    | attackerLocated(physics)   |
|       | -       | -    | -       | 9[2]    | -    | -    | installed(vehicle5,keyFobs)  |
|       | -       | -    | -       | 10[2]   | -    | -    | installed(vehicle5,obdTeslaX)  |
| 1     | 1[3]    | -    | 1[3]    | 1[3]    | -    | -    | control(vehicle10)   |
| 2     | 2[3]    | -    | 2[3]    | 2[3]    | -    | -    | RULE 8 (V2X communication exploit)   |
| 3     | 3[3]    | -    | 3[3]    | 3[3]    | -    | -    | vulExists(vehicle10,'CVE-2016-2354',obd2,phyShortExploit,verifiBypass)           |
| 4     | 4[3]    | -    | 4[3]    | 4[3]    | -    | -    | vState(vehicle10,mov)  |
| 5     | 5[3]    | -    | 5[3]    | 5[3]    | -    | -    | installed(vehicle10,obd2)  |
| 6     | 6[3]    | -    | -       | -       | -    | -    | hacl(vehicle9,vehicle10,_,_)   |
| 7     | 7[3]    | -    | -       | -       | -    | -    | execCode(vehicle9,root)  |
| 10    | 10[3]   | -    | 10[3]   | 10[3]   | -    | -    | RULE 15 (local exploit)  |
| 11    | 11[3]   | -    | -       | -       | -    | -    | execCode(vehicle9,user)  |
| 13    | 13[3]   | -    | -       | -       | -    | -    | canAccessHost(vehicle9)  |
| 14    | 14[3]   | -    | 14[3]   | 14[3]   | -    | -    | RULE 22 (Access a host through executing code on the machine)                    |
| 15    | 15[3]   | -    | -       | -       | -    | -    | RULE 22 (Access a host through executing code on the machine)                    |
| 17    | 17[3]   | -    | -       | -       | -    | -    | principalCompromised(vehicle9Owner)  |
| 18    | 18[3]   | -    | 18[3]   | 18[3]   | -    | -    | RULE 25 (password sniffing)  |
| 19    | 19[3]   | -    | 19[3]   | 19[3]   | -    | -    | RULE 26 (password sniffing)  |
| 20    | 20[3]   | -    | 20[3]   | 20[3]   | -    | -    | RULE 17 (remote exploit for a client program)                                    |
| 21    | 21[3]   | -    | -       | -       | -    | -    | accessMaliciousInput(vehicle9,vehicle9Owner,chrome)                              |
| 22    | 22[3]   | -    | 22[3]   | 22[3]   | -    | -    | RULE 38 (Browsing a compromised website)   |
| 26    | 26[3]   | -    | -       | -       | -    | -    | hacl(vehicle9,cloudplat_webserver1,httpProtocol,httpPort)                        |
| 27    | 27[3]   | -    | -       | -       | -    | -    | clientProgram(vehicle9,chrome)   |
| 28    | 28[3]   | -    | 28[3]   | 28[3]   | -    | -    | isWebServer(cloudplat_webserver1)  |
| 29    | 29[3]   | -    | -       | -       | -    | -    | inCompetent(vehicle9Owner)   |
| 30    | 30[3]   | -    | -       | -       | -    | -    | vulExists(vehicle9,'CVE-2021-21220',chrome,remoteClient,privEscalation)          |
| 31    | 31[3]   | -    | 31[3]   | 31[3]   | -    | -    | RULE 8 (V2X communication exploit)   |
| 32    | 32[3]   | 1[3] | 32[3]   | 32[3]   | 1[3] | 1[3] | control(vehicle11)   |
| 33    | 33[3]   | 2[3] | 33[3]   | 33[3]   | 2[3] | 2[3] | RULE 4 (Local bypass to control)   |
| 34    | 34[3]   | 3[3] | 34[3]   | 34[3]   | 3[3] | 3[3] | vulExists(vehicle11,'CVE-2018-9314',headUnitHU_NBT,phyLocalExploit,verifiBypass) |
| 35    | 35[3]   | 4[3] | 35[3]   | 35[3]   | 4[3] | 4[3] | inside(vehicle11)  |
| 36    | 36[3]   | 5[3] | 36[3]   | 36[3]   | 5[3] | 5[3] | RULE 11 (Enter a vehicle when the vehicle is unlocked)                           |
| 37    | 37[3]   | 6[3] | 37[3]   | 37[3]   | 6[3] | 6[3] | attackerLocated(physics)   |
| 38    | 38[3]   | 7[3] | 38[3]   | 38[3]   | 7[3] | 7[3] | vState(vehicle11,unlocked)   |
| 39    | 39[3]   | 8[3] | 39[3]   | 39[3]   | 8[3] | 8[3] | installed(vehicle11,headUnitHU_NBT)  |
| 40    | 40[3]   | -    | -       | -       | -    | -    | control(vehicle9)  |
| 41    | 41[3]   | -    | -       | -       | -    | -    | RULE 0 (The vehicle is controlled when the OS is compromised)                    |
| 42    | 42[3]   | -    | -       | -       | -    | -    | vState(vehicle9,mov)   |
| 113   | 113[3]  | -    | 113[3]  | 113[3]  | -    | -    | vulExists(vehicle10,'CVE-2016-2354',obd2,phyShortExploit,verifiBypass)           |
| 114   | 114[3]  | -    | 114[3]  | 114[3]  | -    | -    | vState(vehicle10,mov)  |
| 115   | 115[3]  | -    | 115[3]  | 115[3]  | -    | -    | installed(vehicle10,obd2)  |
| 116   | 116[3]  | -    | -       | -       | -    | -    | hacl(vehicle9,vehicle10,_,_)   |
| 119   | 119[3]  | -    | -       | -       | -    | -    | vulExists(vehicle9,'CVE-2021-3156',sudo,localExploit,privEscalation)             |
| 1116  | 1116[3] | -    | -       | -       | -    | -    | hasAccount(vehicle9Owner,vehicle9,user)  |
| 2116  | 2116[3] | -    | -       | -       | -    | -    | hasAccount(vehicle9Owner,vehicle9,user)  |
| 3116  | 3116[3] | -    | -       | -       | -    | -    | hasAccount(vehicle9Owner,vehicle9,user)  |
| 25    | -       | -    | -       | -       | -    | -    | pexecCode(cloudplat_webserver1,user)   |
| 24    | -       | -    | -       | -       | -    | -    | RULE 39 (PV rule)  |
| 23    | -       | -    | -       | -       | -    | -    | execCode(cloudplat_webserver1,user)  |
| -     | -       | -    | 26[3]   | 26[3]   | -    | -    | hacl(mobile8,cloudplat_webserver1,httpProtocol,httpPort)                         |
| -     | -       | -    | 27[3]   | 27[3]   | -    | -    | clientProgram(mobile8,chrome)  |
| -     | -       | -    | 29[3]   | 29[3]   | -    | -    | inCompetent(vehicle8Owner)   |
| -     | -       | -    | 30[3]   | 30[3]   | -    | -    | vulExists(mobile8,'CVE-2021-21220',chrome,remoteClient,privEscalation)           |
| -     | -       | -    | 1116[3] | 1116[3] | -    | -    | hasAccount(vehicle8Owner,mobile8,user)   |
| -     | -       | -    | 2116[3] | 2116[3] | -    | -    | hasAccount(vehicle8Owner,mobile8,user)   |
| -     | -       | -    | 3116[3] | 3116[3] | -    | -    | hasAccount(vehicle8Owner,mobile8,user)   |
| -     | -       | -    | 21[3]   | 21[3]   | -    | -    | accessMaliciousInput(mobile8,vehicle8Owner,chrome)                               |
| -     | -       | -    | 6[3]    | 6[3]    | -    | -    | hacl(mobile8,vehicle10,_,_)  |
| -     | -       | -    | 116[3]  | 116[3]  | -    | -    | hacl(mobile8,vehicle10,_,_)  |
| -     | -       | -    | 11[3]   | 11[3]   | -    | -    | execCode(mobile8,user)   |
| -     | -       | -    | 19[3]   | 19[3]   | -    | -    | vulExists(mobile8,'CVE-2017-6424',wifidriver,localExploit,privEscalation)        |
| -     | -       | -    | 7[3]    | 7[3]    | -    | -    | execCode(mobile8,root)   |
| -     | -       | -    | 42[3]   | 42[3]   | -    | -    | pair(mobile8,vehicle8,bluetooth8)  |
| -     | -       | -    | 41[3]   | 41[3]   | -    | -    | RULE 10 (Compromised Mobile control vehicle)                                     |
| -     | -       | -    | 40[3]   | 40[3]   | -    | -    | control(vehicle8)  |
| -     | -       | -    | 13[3]   | 13[3]   | -    | -    | canAccessHost(mobile8)   |
| -     | -       | -    | 17[3]   | 17[3]   | -    | -    | principalCompromised(vehicle8Owner)  |



## 参考文献

- [1] Abboud K, Omar H A, Zhuang W. Interworking of DSRC and cellular network technologies for V2X communications: A survey[J]. IEEE transactions on vehicular technology, 2016, 65(12): 9457-9470.
- [2] MacHardy Z, Khan A, Obana K, et al. V2X access technologies: Regulation, research, and remaining challenges[J]. IEEE Communications Surveys & Tutorials, 2018, 20(3): 1858-1877.
- [3] Arena F, Pau G, Severino A. V2X communications applied to safety of pedestrians and vehicles[J]. Journal of Sensor and Actuator Networks, 2020, 9(1): 3.
- [4] Alnasser A, Sun H, Jiang J. Cyber security challenges and solutions for V2X communications: A survey[J]. Computer Networks, 2019, 151: 52-67.
- [5] 王璐, 庞昕熠. 智慧交通建设掀热潮产业链机遇倍增[EB/OL]. (2020-12-23) [2021-05-27]. <https://baijiahao.baidu.com/s?id=1686836724709901748>.
- [6] Contreras-Castillo J, Zeadally S, Guerrero-Ibañez J A. Internet of vehicles: architecture, protocols, and security[J]. IEEE internet of things Journal, 2017, 5(5): 3701-3709.
- [7] Sharma S, Kaushik B. A survey on internet of vehicles: Applications, security issues & solutions[J]. Vehicular Communications, 2019, 20: 100182.
- [8] Ghosal A, Conti M. Security issues and challenges in V2X: A Survey[J]. Computer Networks, 2020, 169: 107093.
- [9] Arora A, Yadav S K. Block chain based security mechanism for internet of vehicles (IoV)[C]. in: Proceedings of 3rd International Conference on Internet of Things and Connected Technologies (ICIoTCT). 2018: 26-27.
- [10] Liu Y, Wang Y, Chang G. Efficient privacy-preserving dual authentication and key agreement scheme for secure V2V communications in an IoV paradigm[J]. IEEE Transactions on Intelligent Transportation Systems, 2017, 18(10): 2740-2749.
- [11] Mendiboure L, Chalouf M A, Krief F. Towards a blockchain-based SD-IoV for applications authentication and trust management[C]. in: International Conference on Internet of Vehicles. 2018: 265-277.
- [12] Wazid M, Bagga P, Das A K, et al. AKM-IoV: Authenticated key management protocol in fog computing-based Internet of vehicles deployment[J]. IEEE Internet of Things Journal, 2019, 6(5): 8804-8817.
- [13] Poomagal C, Sathish Kumar G. ECC Based Lightweight Secure Message Conveyance Protocol for Satellite Communication in Internet of Vehicles (IoV)[J]. Wireless Personal Communications, 2020, 113: 1359-1377.
- [14] Xiao Y, Liu Y, Li T. Edge computing and blockchain for quick fake news detection in IoV[J]. Sensors, 2020, 20(16): 4360.

- [15] Alladi T, Chakravarty S, Chamola V, et al. A Lightweight Authentication and Attestation Scheme for In-Transit Vehicles in IoV Scenario[J]. IEEE Transactions on Vehicular Technology, 2020, 69(12): 14188-14197.
- [16] 叶子维, 郭渊博, 王宸东, 等. 攻击图技术应用研究综述[J]. 通信学报, 2017, 38(011): 121-132.
- [17] 常昊, 秦元庆, 周纯杰. 基于贝叶斯攻击图的工控系统动态风险评估[J]. 信息技术, 2018, 42(10): 70-75+80.
- [18] Che B, Liu L, Zhang H. KNEMAG: Key Node Estimation Mechanism Based on Attack Graph for IoT Security[J]. Journal on Internet of Things, 2020, 2(4): 145-162.
- [19] Kaynar K, Sivrikaya F. Distributed attack graph generation[J]. IEEE Transactions on Dependable and Secure Computing, 2015, 13(5): 519-532.
- [20] 王秀娟, 孙博, 廖彦文, 等. 贝叶斯属性攻击图网络脆弱性评估[J]. 北京邮电大学学报, 2015, 38(4): 110-116.
- [21] Fadlallah A, Sbeity H, Malli M, et al. Application of attack graphs in intrusion detection systems: an implementation[J]. International Journal of Computer Networks, 2016, 8(1): 1-12.
- [22] Ou X, Govindavajhala S, Appel A W. MulVAL: A Logic-based Network Security Analyzer.[C]. in: USENIX security symposium: vol. 8. 2005: 113-128.
- [23] 徐丽娟. 基于攻击图的工业控制网络安全隐患分析[D]. 北京邮电大学, 2015: 1-65.
- [24] 李兴华, 钟成, 陈颖, 等. 车联网安全综述[J]. 信息安全学报, 2019, 4(3): 17-33.
- [25] Miller C, Valasek C. Car hacking: for poories[R]. Tech. rep., IOActive Report, 2015: 1-25.
- [26] Chirgwin R. Big bimmer bummer: Bavaria's BMW buggies battered by bad bugs[EB/OL]. (2018-05-23) [2021-05-26]. [https://www.theregister.com/2018/05/23/bmw\\_security\\_bugs/](https://www.theregister.com/2018/05/23/bmw_security_bugs/).
- [27] Greenberg A. This Bluetooth Attack Can Steal a Tesla Model X in Minutes[EB/OL]. (2020-11-23) [2021-05-26]. <https://www.wired.com/story/tesla-model-x-hack-bluetooth/>.
- [28] Hossain M, Hasan R, Zawoad S. Trust-IoV: A Trustworthy Forensic Investigation Framework for the Internet of Vehicles (IoV)[C]. in: 2017 IEEE International Congress on Internet of Things (ICIOT). 2017: 25-32.
- [29] Garg S, Kaur K, Kaddoum G, et al. Sec-IoV: A multi-stage anomaly detection scheme for Internet of vehicles[C]. in: Proceedings of the ACM MobiHoc Workshop on Pervasive Systems in the IoT Era. 2019: 37-42.
- [30] Miller C, Valasek C. A survey of remote automotive attack surfaces[J]. Black hat USA, 2014, 2014: 1-94.
- [31] Kelarestaghi K B, Foruhandeh M, Heaslip K, et al. Intelligent Transportation System Security: Impact-Oriented Risk Assessment of in-Vehicle Networks[J]. IEEE Intelligent Transportation Systems Magazine, 2021, 13(2): 91-104.
- [32] Phillips C, Swiler L P. A graph-based system for network-vulnerability analysis[C]. in: Proceedings of the 1998 workshop on New security paradigms. 1998: 71-79.

- [33] Zhang S, Ou X, Homer J. Effective network vulnerability assessment through model abstraction[C]. in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. 2011: 17-34.
- [34] Ou X, Boyer W F, McQueen M A. A scalable approach to attack graph generation[C]. in: Proceedings of the 13th ACM conference on Computer and communications security. 2006: 336-345.
- [35] 张勇, 丁建林. 赛博空间态势感知技术研究[J]. 信息网络安全, 2012(3): 42-44.
- [36] 中国信息通信研究院. 车联网网络安全白皮书 (2017) [R/OL]. 中国信息通信研究院. 2017. <http://www.caict.ac.cn/kxyj/qwfb/bps/201804/P020170921430215345026.pdf>.
- [37] 魏忠, 张保稳. 一种基于本体的潜在多步网络攻击发现方法[J]. 通信技术, 2018, 051(002): 419-424.
- [38] Duan W, Gu J, Wen M, et al. Emerging technologies for 5G-IoV networks: Applications, trends and opportunities[J]. IEEE Network, 2020, 34(5): 283-289.
- [39] Lallie H S, Debattista K, Bal J. A review of attack graph and attack tree visual syntax in cyber security[J]. Computer Science Review, 2020, 35: 100219.
- [40] 上海交通大学网络安全技术研究院车联网安全研究小组. 汽车漏洞库[EB/OL]. (2021-05-28) [2021-05-30]. <https://shimo.im/sheets/DvJHDJkCQHvpP8Tc/MODOC>.
- [41] 蜚语安全. 特斯拉又双叒出问题了? 这一次可是手机远程启动的安全风险! [R/OL]. 蜚语安全. 2020. <https://mp.weixin.qq.com/s/kXkZt3D-vYYnTTyji6sH3w>.
- [42] Garg T, Kagalwalla N, Churi P, et al. A survey on security and privacy issues in IoV[J]. International Journal of Electrical & Computer Engineering, 2020, 10(5): 5409-5419.
- [43] 程叶霞, 姜文, 薛质, 等. 基于攻击图模型的多目标网络安全评估研究[J]. 计算机研究与发展, 2012(S2): 23-31.
- [44] 陈锋, 毛捍东, 张维明, 等. 攻击图技术研究进展[J]. 计算机科学, 2011, 38(011): 12-18.
- [45] Sheyner O, Haines J, Jha S, et al. Automated generation and analysis of attack graphs[C]. in: Proceedings 2002 IEEE Symposium on Security and Privacy. 2002: 273-284.
- [46] Lippmann R, Ingols K, Scott C, et al. Validating and restoring defense in depth using attack graphs[C]. in: MILCOM 2006 - 2006 IEEE Military Communications conference. 2006: 1-10.
- [47] Zhou H, Xu W, Chen J, et al. Evolutionary V2X technologies toward the internet of vehicles: challenges and opportunities[J]. Proceedings of the IEEE, 2020, 108(2): 308-323.
- [48] Gyawali S, Xu S, Qian Y, et al. Challenges and Solutions for Cellular Based V2X Communications[J]. IEEE Communications Surveys & Tutorials, 2021, 23(1): 222-255.



## 谢 辞

我首先要感谢网络安全技术研究院的马进老师，作为我的毕业设计指导老师，对我的论文中的研究提出了许多重要的建议，也激发了我许多的灵感。不管是设计方案的过程中，构建原型系统的过程中，还是撰写论文的过程中，马进老师都非常耐心地给予我指导。此外，在平时，马进老师也很关心我的个人情况。我经常和马进老师探讨关于研究人员的成长、研究方向的选择与职业生涯规划的话题，在这样的交流中马进老师也给了我很多实用的人生建议。

我还要感谢车联网安全研究组的其他老师和同学们。陈秀真老师虽然比较严厉，但提出的问题总是切中要害，为我的方案设计的完善提供了非常大的帮助。周志洪老师则经常为我们争取到去车厂参观、与车厂合作的机会，让我们实地参与车联网的研究，加深了我们对车联网安全的理解。侯书凝学姐与我的研究方向类似，因此我与侯书凝学姐经常有关于车联网漏洞利用和车联网架构的讨论。在这些讨论中侯书凝学姐给了我非常多的建议和帮助。其他学长学姐和同组做毕设的同学虽然研究方向与我不同，但在每周的组会和平常的讨论中，我从他们那里学习到了很多关于车联网安全的其他方向的知识，拓宽了我的视野，也加深了我对车联网安全的理解。此外，我们一起学习车联网安全知识，互相鼓励，共同进步，结下了深厚的友谊。另外，本文中所使用的车联网安全知识库也是车联网安全研究组的老师和同学们合作搜集整理的，极大地便利了我对车联网攻击图生成规则集的构建。

此外，我还要感谢网络空间安全学院与网络安全技术研究院的所有老师。在四年的大学课程中，你们耐心细致的讲解与教导让我心中知识的种子生根发芽，茁壮成长，让我牢固地掌握了计算机科学与信息安全的基础知识，让我可以在社会中实现更大的自我价值。关于在大学的科研活动，我尤其要感谢易平老师，他是我在信息安全研究领域的引路人。在易平老师的指导下，我在对抗样本检测与防御方法上进行了一年多的研究。在这一年多研究里，虽然取得了一些科研成果，但最重要的是为我打下了科学研究的基础，熟悉了科研的过程，让我能更从容自信地参加之后的科研活动。

另外，还要感谢 2017 级信息安全专业的其他同学。三年、甚至四年的一起学习、一起生活让我们结下了深厚的友谊。我们在互相鼓励中得到开始新一天的力量，在互相竞争中认识到自己的不足，在共同成长时发现自我，在共同进步中超越自我。郭建铭同学为人温和谦逊，乐于助人，具有优秀的交际能力，交友圈广泛，也是信息“小灵通”，有问题，问他准能行；他也有很强的领导能力，善于交流，懂得用人，在团队中能合理分配时间与任务。陈子昂同学作为我们的团支部书记非常认真负责，为人沉稳踏实，任劳任怨；他还有扎实的知识基础和很强的技术能力，对代码和工具实现有着执着的热情。江浩宇同学是我进入网络空间安全学院后交到的第一个朋友，他大方随和，开朗阳光，并用他的阳光感染周围的人，是一位不可多得的好友。谢宇翀同学聪明能干，学习成绩优异，积极参加科研活动，他的勤奋好学与积极上进值得我不断学习与看齐。你们充实了我四年的大学生活，也更让我明白了“优秀”二字的含义。

我还要感谢我的父母，感谢你们在大学四年对我的理解、支持和鼓励。大学中，与如此多优秀的同学在一起，难免有时会变得焦虑，偶尔繁重的任务又让人神经紧绷。因此我常把焦虑与委屈带入与你们的对话中，有时甚至把脾气发在你们身上。对此我深感愧疚，也感谢你们的理解与忍耐。生活和学习中总是会遇到很多困难，但你们的支持和鼓励是我面对困

难最大的力量。

最后，我还要向评审这篇论文的老师表示感谢。由于本人时间和能力有限，对本文的研究内容还是留下了一些遗憾，你们的批评和建议将成为我进一步研究的动力。

# ATTACK GRAPH GENERATION TECHNIQUE FOR V2X INTERNET OF VEHICLES

Recent years, with the rapid development of Internet of Vehicles (IoV) technology, Vehicle-to-Everything (V2X) communication is gradually becoming a reality. However, compared with traditional networks, the composition and structure of the IoV are much more complex and vulnerable. What's worse, the dynamicity, timeliness, complexity, and large-scale of the IoV make it difficult for people to analyze the overall security risk of the IoV effectively. Therefore, there is an urgent need for an IoV security risk analysis and assessment technique to help network security personnel obtain real-time vehicle network security status, assist network administrator to make security decisions, and efficiently maintain IoV security. So in this paper, we try to apply the attack graph technique to IoV security analysis and assessment.

This paper aims to answer four questions when applying the attack graph technique to IoV: 1) how to obtain the rapidly changing network information of the IoV in real time? 2) how to design the unique attack graph generation rules in the IoV? 3) how to generate the IoV attack graph in real time with high efficiency and low latency? 4) how to use attack graphs to quantitatively analyze the security risks of the IoV? We provide answers by our proposed V2X IoV attack graph generation scheme for these questions: 1) design a real-time security information collection scheme adapting to the IoV; 2) apply ontology to model the entities and complex relationships in the IoV, standardize the description of the attack methods in the IoV, and summarize the rules for generating the IoV attack graph; 3) design a real-time attack graph generation and analysis scheme, build a quantitative risk assessment model for the IoV, and implement the attack graph generation and analysis prototype system.

Firstly, according to the composition and system architecture of the IoV and its information security risks and requirements, we propose an IoV security ontology model based on the general network security ontology. The security ontology model contains six security ontology classes: "network asset", "system component", "vulnerability", "attack", "attacker (capability)" and "principal", as well as nine relationships between ontology classes: "exploited by", "required by", "locate in", "compromise", "hasAccount", "connect", "install", "exist" and "deepen". Through this IoV security ontology model, we can standardize the description of entities and the complex relationships between entities in the IoV. Then, we designed a simple but typical attack scenario use case to verify the correctness and effectiveness of the IoV security ontology model. We use the IoV security ontology model to correctly and completely describe the three attack paths in the attack scenario of this use case, which proves that the IoV security ontology model can be used to describe the potential attack paths and method. We can use this ontology model to describe the vulnerability exploitation in the IoV standardly. Then, using the security ontology model to standardize the vulnerability exploitation methods and attack methods in and the IoV security knowledge database, we construct the original rule set of IoV attack graph generation.

Secondly, we summarize the challenges faced by information security analysis and assessment

in the IoV, and proposes a distributed real-time attack graph generation and analysis scheme for IoV based on the C-V2X communication architecture and IoV MEC. This solution uses MEC to generate local attack graphs and analyze them by a method based on Bayesian graph calculation in a distributed manner, and then uploads them to the cloud platform for combination into the global Bayesian attack graph. The calculation cost is allocated to each service area, which effectively reduces the latency of quantitative risk assessment. The scheme can be mainly divided into two parts: real-time security information collection and real-time attack graph generation and analysis. The network security information in the IoV mainly includes rapidly changing network configuration information, changing communication node configuration information, and changing vulnerability information. We design a real-time security information collection scheme for the IoV to collect the above-mentioned real-time security information, put those information into the existing attack graph derivation engine, and run the attack graph derivation according to the rules set to generate attack graph. In order to generate and analyze real-time IoV attack graphs efficiently and with low latency, according to the characteristics of IoV C-V2X communication architecture, we propose the real-time IoV attack graph generation and analysis scheme based on MEC. The MEC server in each cellular service area generates a real-time local attack graph and then analyze it. This paper first re-expresses the original vehicle network attack graph generation rule set based on the IoV security ontology model with the restricted first-order predicate logic language Datalog. MulVAL will use the attack graph generation rule set for attack graph generation. Then the MEC server will analyze the attack graph by a method based on Bayesian graph calculation, which convert the attack graph into the Bayesian attack graph, and uploads the local Bayesian attack graph to the cloud platform. The cloud platform generates its own local Bayesian attack graph, and then the real-time local Bayesian attack graph of each cellular service area and the local Bayesian attack graph of the cloud platform are spliced into a real-time global Bayesian attack graph through a simple splicing algorithm.

We explain the analysis method based on Bayesian attack graph calculation in detail. We propose a quantitative risk assessment scheme for the IoV based on the Bayesian attack graph technique. First, we propose an algorithm for removing the loop path in the attack graph by removing the deepest atom attack node by Depth-First-Search, and then we calculate the difficulty of atomic attacks by CVSS. Then we calculate the Bayesian probability of the attack path to obtain the Bayesian attack graph, based on which we propose a quantitative risk assessment model for the IoV.

Thirdly, we implement a prototype system based on the real-time attack graph generation and analysis scheme based on MulVAL, and conducts a comprehensive test on the prototype system to verify its correctness and effectiveness. First, we re-express the original IoV attack graph generation rule set with Datalog. MulVAL will use the Datalog attack graph generation rule set for attack graph generation. The prototype system is implemented using Python multi-process, mainly including multiple local Bayesian attack graph generation processes and a global Bayesian attack graph splicing process. Each local Bayesian attack graph generation process represents the attack graph generation process of the MEC server in a cellular service area. The security information is directly standardized in the form of Datalog files. The process monitors the TCP connection of a specific port. If that port receives data, it means that the security information of the cellular service area has changed, and the real-time security information has been collected and reported to the MEC server. After the local Bayesian attack graph generation process receives the real-time secu-

rity information, it calls the MulVAL command to process the data, and uses the Datalog IoV attack graph generation rule set to generate a real-time local attack graph and then the process analyze the attack graph by converting it to Bayesian attack graph by the quantitative risk assessment model. Then the process submit the local Bayesian attack graph to the global Bayesian attack graph splicing process. The global attack graph splicing process monitors TCP connections and receives real-time local Bayesian attack graphs. This process stitches all real-time local Bayesian attack graphs into real-time global Bayesian attack graphs through a stitching algorithm, and based on which we can assess the risk in IoV quantitatively.

Finally, we test the correctness, effectiveness, and timeliness of the prototype system by a test scenario use case. The scenario includes multiple vulnerabilities and attack methods in the cloud layer, channel layer, and terminal layer. The network topology, vulnerability information, and vehicle status are designed to dynamically change. We use the prototype system to generate real-time global Bayesian attack graphs and conduct quantitative risk assessments for the IoV attack scenario. We conduct a detailed analysis of the quantitative risk assessment results, and verify the correctness and effectiveness of the prototype system. After that, we test the performance of the prototype system, and find that the latency and computational cost of the attack graph generation meet the real-time requirements of the IoV.