ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 22-04-13
j.inst.Korean.electr.electron.eng.Vol.26,No.4,614~621,December 2022
https://doi.org/10.7471/ikeee.2022.26.4.614
91

RIDS: 랜덤 포레스트 기반 차량 내 네트워크 침입 탐지 시스템

RIDS: Random Forest-Based Intrusion Detection System for In-Vehicle Network

이 대 기*, 한 창 선*, 이 성 수*★

Daegi Lee*, Changseon Han*, and Seongsoo Lee*★

Abstract

This paper proposes RIDS (Random Forest-Based Intrusion Detection), which is an intrusion detection system to detect hacking attack based on random forest. RIDS detects three typical attacks i.e. DoS (Denial of service) attack, fuzzing attack, and spoofing attack. It detects hacking attack based on four parameters, i.e. time interval between data frames, its deviation, Hamming distance between payloads, and its diviation. RIDS was designed in memory-centric architecture and node information is stored in memories. It was designed in scalable architecture where DoS attack, fuzzing attack, and spoofing attack can be all detected by adjusting number and depth of trees. Simulation results show that RIDS has 0.9835 accuracy and 0.9545 F1 score and it can detect three attack types effectively.

요 약

본 논문은 CAN(Controller Area Network) 버스에서 해킹에 의한 공격을 탐지하기 위한 랜덤 포레스트 기반 침입 감지 시스템(RIDS: Random Forest-Based Intrusion Detection)을 제안한다. RIDS는 CAN 버스에서 나타날 수 있는 전형적인 세 가지 공격, 즉 DoS(Denial of Service) 공격, Fuzzing 공격, Spoofing 공격을 탐지하며, 데이터 프레임 사이의 시간 간격과 그 편차, 페이로드끼리의 해밍 거리와 그 편차의 네 가지 파라미터를 사용하여 공격을 판단한다. RIDS는 메모리 중심 방식의 아키텍쳐를 가지며 노드의 정보를 메모리에 저장하여 사용하며 트리의 개수와 깊이만 조절하면 DoS 공격, Fuzzing 공격, Spoofing 공격을 모두 탐지할 수 있도록 확장이 용이한 구조로 설계되었다. 시뮬레이션 결과 RIDS는 정확도 0.9835, F1 점수 0.9545로 세 가지 공격을 효과적으로 탐지할 수 있었다.

Key words : Controller Area Network, Intrusion Detection System, Random Forest, Machine Learning, Hacking, Automotive Cybersecurity, In-Vehicle Network

* Soongsil University (Master Student, Professor)
★ Corresponding author
E-mail : sslee@ssu.ac.kr, Tel : +82-2-820-0692

※ Acknowledgment
This work was supported by Industrial Technology Challenge Track of the Ministry of Trade, Industry and Energy (MOTIE) / Korea Evaluation Institute of Industrial Technology (KEIT). (20012624) It was supported by the R&D Program of the Ministry of Trade, Industry, and Energy (MOTIE) and Korea Evaluation Institute of Industrial Technology (KEIT). (RS-2022-00155731, RS-2022-00154973)
Manuscript received Dec. 13, 2022; revised Dec. 16, 2022; accepted Dec. 20, 2022.

Ⅰ. 서론

차량과 인프라 간의 통신을 위한 V2X 통신 시스템을 사용하는 커넥티드 카의 수가 증가함과 동시에 보안 위협도 증가하고 있으며 전자 부품간 보안 안정성이 요구되고 있다. 이러한 기술동향에 따라 국제 전기 통신 연합(ITU-T)의 SG17은 X.itssec 시리즈로 자동차에 대한 보안 문제 및 위협을 각각 정의했다[1]. 또한, 2021년 국제 자동차 기술 협회(Society of Automotive Engineers, SAE)와 국제 표준화 기구(International Organization for Standardization. ISO)는 자동차의 사이버 보안 대응 체제를 구축 및 운영하여 공격에 의한 피해 발생을 줄이는 것을 목적으로 한 ISO/SAE 21434를 발행하였다[2]. 이와 같이 보안 위협으로부터 대응하기 위해 연구가 진행되고 있으며 자동차 사이버 보안의 중요도가 매우 높아졌다. 전자 기술이 발전함에 따라 차량 내 전자 장비의 비중이 높아졌고 차량 제어도 기계식에서 전자식으로 변화하고 있다. 차량 내의 전자 장비들은 ECU (Electronic Control Unit)에 의해 제어되며 전자 장비의 수가 증가함에 따라 ECU의 수 또한 증가하고 있다.

CAN(Controller Area Network)[3][4]는 다수의 ECU를 연결되어 운행에 필요한 정보를 주고받을 수 있도록 지원하는 차량 내 네트워크(IVN: In-Vehicle Network)이다. CAN 버스는 잡음에 강한 특징을 가지고 있어 높은 신뢰성을 보장하므로 엔진 제어 등 운행에 직접 연관 있는 영역에서 거희 필수적으로 사용되며, 이외에도 다양한 영역에서 차량 내 네트워크의 주축을 담당하고 있다.

일반적으로 차량 내 네크워크는 구조상 외부의 접근이 허용되지 않는 폐쇄 구조를 가지고 있으나 최근 V2X (Vechile-to-Everything Communication)를 사용하는 커넥티드카의 등장으로 해킹의 위험성이 높아지고 있다. 특히 CAN 버스와 같이 운전자의 안전과 직결되는 영역에서 사용되는 네트워크가 해킹으로 공격당한다면 운전자의 안전에 위협이 될 수 있다. CAN 버스는 이더넷(Ethernet) 등 다른 버스와는 다르게 메시지가 모든 노드에 브로드캐스트되며 메시지 프레임에 송신자, 수신자 정보가 없기 때문에 보안에 매우 취약하다. 따라서 CAN 버스를 사용하는 차량 내 네트워크에는 해킹에 의한 침입이 일어났는지를 탐지하는 침입 탐지 시스템(IDS: Intrusion Detection System)이 필요하며, 차량 사이버보안에 관련된 국제 법규인 UNECE WP.29[5]에서도 최근 IDS의 사용을 의무화하고 있다.

본 논문에서는 공격자가 악의적인 목적으로 전송하는 메시지를 탐지하기 위한 IDS 기법인 RIDS(Random Forest-Based Intrusion Detection System)를 다음과 같이 제안한다.

RIDS는 CAN 버스에 전송되는 데이터 프레임을 확인하며, 머신러닝 기법의 하나인 랜덤 포레스트(Random Forest)[6] 분류기를 통해 메시지의 클래스를 정상 메시지와 해킹이 의심되는 메시지로 분류한다. RIDS는 Verilog HDL로 기술하였으며 제한적인 하드웨어 리소스를 효율적으로 사용하기 위해 메모리를 이용한 피드백 방식을 사용하였다.

본 논문에서는 CAN 제어기가 데이터 프레임을 수신하고 특징점(Feature)을 추출하여 RIDS에 입력한다고 가정한다. 이때 RIDS는 CAN 버스에서 일어나는 세 가지 전형적인 공격(DoS: Denial of Service, Fuzzing, Spoofing)[7]을 고려하여 해킹이 의심되는 메시지를 찾아내며, 해킹 탐지 성능은 정확도(Accuracy), 민감도(Sensitivity), 정밀도(Precision), F1 점수(F1 Score)를 계산하여 평가하였다. RIDS는 IDEC의 설계 도구 지원을 받아 Siemens Modelsim을 통해 Verilog HDL로 기술하고 동작을 검증하였으며 Xilinx Vivado Design Suite를 사용하여 합성하고 결과를 확인하였다.

Ⅱ. 공격 시나리오

차량 내 네트워크에서의 공격은 주로 OBD-II(On-Board Diagnostics-II)[8] 포트 연결 또는 ECU 해킹을 통해 CAN 버스에 악의적인 메시지를 주입함으로써 버스를 마비시키거나 잘못된 동작을 하도록 유도하는 방식으로 이루어진다. 그 예로 OBD-II 포트를 통해 CAN 버스에 액세스하여 RPM 화면에 잘못된 값을 표시하거나 우선 순위가 높은 메시지 ID를 가진 데이터 프레임을 빠르게 주입하여 자동차가 시동이 걸리지 않도록 하는 공격이 가능함이 확인되었다[9].

본 논문에서 제안한 RIDS는 CAN 버스에서 나타날 수 있는 전형적인 세 가지 공격, 즉 DoS 공격, Fuzzing 공격, Spoofing 공격을 탐지한다[7]. 그림 1은 이들 공격에서 CAN 버스의 트래픽을 나타낸 것이며 그 차이점은 다음과 같이 설명할 수 있다.

DoS 공격(또는 Flooding 공격)은 그림 1(a)와 같이 CAN 버스의 중재(Arbitration) 방식을 악이용한 방식이다. CAN 버스에서 메시지의 중요도가 높을수록 숫자가 낮은 메시지 ID를 갖게 되며 두 개 이상의 노드가 동시에 데이터 프레임을 전송하려고 할 때 CAN 버스에서는

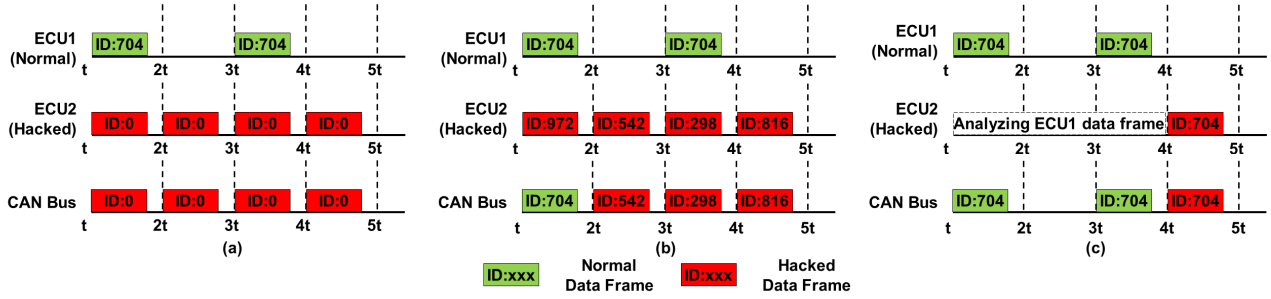

Fig. 1. Attack scenarios of in-vehicle networks (a) DoS attack (b) Fuzzing attack (c) Spoofing attack.
그림 1. 차량 내 네트워크에서의 공격 시나리오 (a) DoS 공격 (b) Fuzzing 공격 (c) Spoofing 공격

메시지 ID가 낮은 데이터 프레임이 우선적으로 전송된다. 해킹당한 ECU2가 숫자가 매우 낮은 메시지 ID로 해킹 데이터 프레임을 계속 보내게 되면 CAN 버스에서 ECU1이 보내는 정상 데이터 프레임은 중요도에서 밀려서 송신이 불가능하게 된다. DoS 공격은 이런 특성을 이용하여 숫자가 매우 낮은 메시지 ID로 데이터 프레임을 짧은 주기로 반복하여 전송함으로서 정상적인 메시지의 전송을 차단한다.

Fuzzing 공격은 그림 1(b)와 같이 메시지 ID와 페이로드를 랜덤하게 생성하여 CAN 버스에 보내는 방식이다. DoS 공격이 CAN 버스를 마비시켜서 전송 자체를 차단하는 것이 목적이라면 Fuzzing 공격은 차량의 제어를 지연 또는 오동작시키는 것이 목적이다. 해킹당한 ECU2가 랜덤하게 생성하여 보내는 데이터 프레임 중 일부는 정상 동작하는 ECU1의 데이터 프레임보다 낮은 숫자의 메시지 ID를 가지기 때문에 ECU1의 전송을 지연시키며, 일부는 다른 ECU들에게 거짓 메시지 ID와 페이로드를 전달하여 차량을 오동작시킨다. 이 방식은 해커의 입장에서 해킹 프로그램이 매우 간단하다는 장점이 있지만, 해킹된 ECU가 보내는 메시지 ID를 랜덤하게 생성하기 때문에 정상적인 차량 제어에는 사용하지 않는 메시지 ID가 나타나는 경우가 많아서 비교적 쉽게 탐지할 수 있다.

Spoofing 공격은 그림 1(c)와 같이 해킹된 ECU가 공격 시작 전에 일정 시간 동안 다른 ECU가 송신하는 데이터 프레임을 모니터링하고 분석한 다음에 해당 ECU가 송신하는 데이터 프레임과 동일한 메시지 ID를 사용하여 거짓 페이로드를 전송하는 방식이다. 해킹당한 ECU2는 정상 동작하는 ECU1이 송신하는 메시지 ID를 일정 시간 동안(t~4t) 분석하여 ECU1이 송신하지 않는 동안(4t~5t)에 동일한 메시지 ID로 거짓 페이로드를 보내게 되며 이때 차량 제어에 오동작이 발생하게 된다. 어떤 경우에는 Spoofing 공격에서 페이로드까지 동일하게, 즉 정상 동작하는 ECU가 보낸 데이터 프레임을 그대로 복사해서 보내기도 하는데 이를 수신하는 다른 ECU는 해킹되었다는 것을 전혀 파악할 수 없기 때문에 상당히 치명적일 수 있다.

Ⅲ. 머신러닝 모델

1. 랜덤 포레스트

랜덤 포레스트는 지도학습으로 훈련되는 앙상블 분류 모델이다. 랜덤하게 샘플을 선택하여 모델을 훈련하고, 여러 개의 작은 의사결정 트리(Decision Tree)로 구성되어 훈련 데이터 세트에 대한 과적합이 적은 장점이 있으며, CAN 메시지와 같이 형식이 정해져 있는 정형화 데이터에 효과적인 모델이다[10]. 또한 분류된 클래스를 최종적으로 한 번 더 종합하는 과정이 있어서 안전과 직결된 CAN 버스에서 IDS로 사용하기 적합하다. 본 논문에서 랜덤 포레스트 모델은 수신한 CAN 메시지가 악의적인 목적으로 전송된 메시지인지 일반적인 메시지인지를 분류하여 CAN 버스 상의 공격을 탐지하는 역할을 수행한다.

본 논문에서는 Python[11]의 머신러닝 라이브러리인 Scikit-Learn[12]을 사용하여 랜덤 포레스트 모델을 훈련했으며 그림 2와 유사한 구조를 가진 의사 결정 트리로 구성된다. 또한 머신러닝 모델을 테스트하여 최적의 하이퍼 파라미터를 출력하는 gridsearchCV[13] 기능을 사용하여 트리의 개수와 깊이를 탐색하였고 이를 훈련에 적용하였다.

2. 특징점 추출

CAN 버스에서 전송되는 데이터 프레임에서 해킹 침입을 판단하기 위해, 실제 차량에서 측정한 데이터 세트

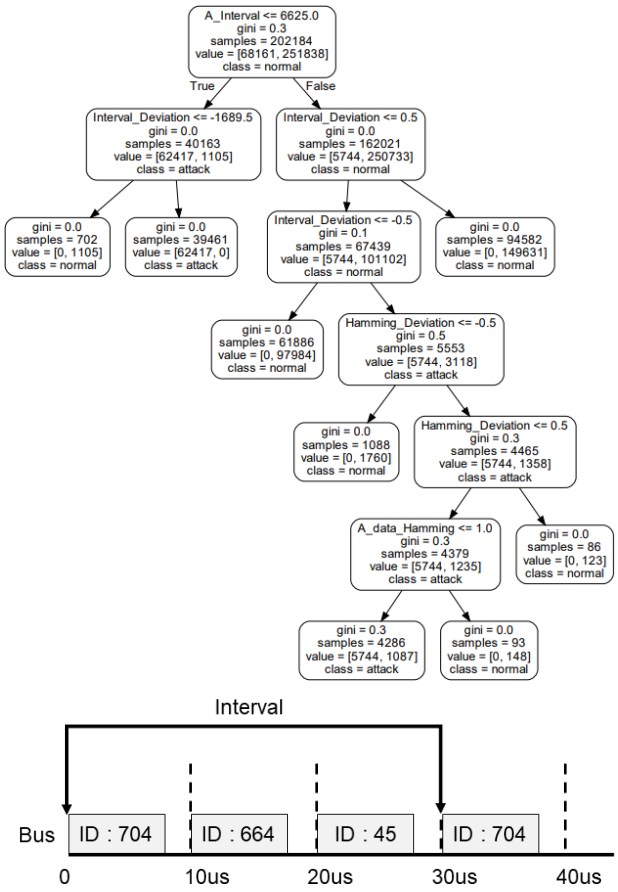

Fig. 2. Example of trained decision tree in random forest using Scikit-Learn.

그림 2. Scikit-Learn으로 훈련시킨 랜덤 포레스트의 의사결정 트리의 예시

[14][15]를 통해 특징점을 추출하였다. 이 데이터 세트는 메시지 전송 시간, 메시지 ID, 페이로드 길이, 페이로드 데이터로 구성되어 있다. 본 논문에서는 공격이 포함되어 있지 않은 정상 데이터 세트에서 정상적인 메시지 ID의 트래픽을 분석하고 공격이 포함된 데이터 세트와 비교하는 이상 탐지(Anomaly Detection) 방식을 사용하여 공격성을 확인할 수 있는 특징점을 추출하였다.

CAN 버스가 차량 내부에서 사용될 때에는 실시간성을 보장하기 위해 일정한 주기로 메시지를 전송하는 경우가 많아서 OBD-II 포트를 통한 진단 메시지 같은 비주기적 메시지보다 일정한 주기로 전송되는 주기적 메시지가 더 큰 비중을 차지한다. 또한, CAN 버스에서 DoS 공격과 Fuzzing 공격은 버스를 점유하여 메시지의 전송을 방해하기 위한 목적을 가지고 있으므로 주기성에 영향을 주기 때문에 같은 메시지 ID를 가진 데이터 프레임 사이의 시간 간격인 메시지 간격(Interval)을 계산하여 공격을 탐지하기에 적합하다. 이러한 특성을 이용한 침입 탐지 시스템이 제안되었다[16]. 그림 3은 메시지 간격

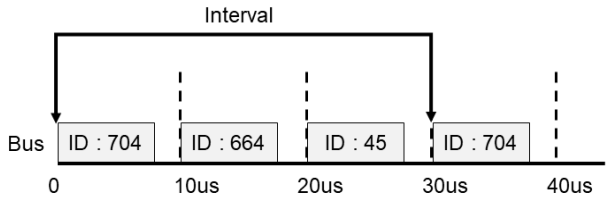

Fig. 3. Example of calculation interval in CAN bus.

그림 3. CAN 버스에서 메시지 간격 계산의 예시

을 계산하는 과정을 나타내며 메시지 ID가 704인 경우의 메시지 간격은 30 us로 계산된다.

또한 실제 차량에서 운전자의 조작 속도와 ECU의 동작 속도는 큰 차이가 나기 때문에 페이로드의 변화량은 많지 않다. 하지만 갑작스럽게 데이터가 큰 폭으로 변한다면 공격성이 높다고 판단할 수 있으며 이는 해밍 거리(Hamming Distance)[17]를 이용하여 페이로드의 변화량을 계산함으로서 해킹을 탐지할 수 있다. Fuzzing 공격은 무작위로 페이로드가 설정되어 CAN 버스에 주입되기 때문에 이에 대해 높은 탐지율을 가진다는 것이 확인되었다[18]. 해밍 거리는 길이가 같은 두 데이터 사이의 거리를 계산하는 것으로 간단한 알고리즘을 통해 구현할 수 있어 제한적인 리소스를 사용하는 하드웨어에서 사용하기에 적합하며 본 논문에서는 같은 메시지 ID를 가진 데이터 프레임의 페이로드끼리 각 비트별로 XOR 연산을 통해 계산한다.

RIDS에서는 정상 데이터 세트를 통해 계산된 정상적인 메시지 아이디의 주기 평균값과 해밍 거리 평균값을 공격 데이터 세트에서 측정한 값과 비교하여 얻은 편차값을 특징점으로 사용한다. 이 값은 메시지 ID를 가진 정상적인 메시지 흐름과 얼마나 차이가 나는지를 의미하므로 이 값이 클수록 공격성이 높다고 판단할 수 있다.

CAN 버스를 향한 악의적인 공격은 결과적으로 트래픽과 페이로드 값의 변화로 나타난다. 따라서 앞에서 설명한 특징점들은 본 논문에서 다루는 공격들을 탐지하기 위해 모두 적용될 수 있으며 RIDS는 모든 특징점들을 사용하여 훈련된다.

Ⅳ. 하드웨어 구현

1. 하드웨어 설계

하드웨어를 효율적으로 사용하기 위해 RIDS는 메모리 중심 방식(Memory-Centric)[19]으로 노드의 정보를 메모리에 저장하여 사용한다. 이는 메모리에 저장되는 값에 따라 트리의 구조를 수정할 수 있는 장점이 있다. 또한, 노드 번호를 메모리 주소로 사용함으로써 클래스 분

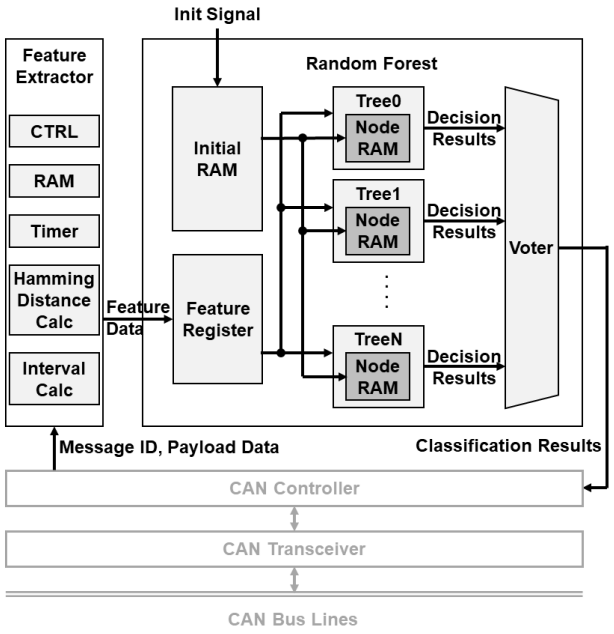

Fig. 4. Architecture of RIDS.
그림 4. RIDS 전체 아키텍쳐

류가 완료되지 않은 샘플을 트리에 다시 입력하여 분류하는 피드백 방식을 구현할 수 있다.

그림 4는 RIDS의 전체 아키텍쳐를 나타낸다. Random Forest는 여러 개의 작은 Tree로 구성되어있고 Tree들의 결과값을 Voter가 종합하여 최종 결정을 내리도록 설계되었다.

Initial RAM에는 학습된 랜덤 포레스트의 모든 의사결정 나무를 그림 5의 데이터 구조를 가지도록 분석하여 2진수로 변환된 데이터가 저장된다. ECU로부터 Init 신호를 받으면 Initial RAM에 저장된 데이터를 각 Tree 내부에 있는 Node RAM으로 노드 정보를 전송한다. 이후 수신받은 CAN 메시지로부터 Feature Extractor가 특징점 데이터를 추출하고 그 결과를 Freature Register에 그림 6과 같은 Compare Frame 포맷으로 저장한다. 각각의 Compare Frame은 각 Tree에 입력되고 Node RAM에 액세스하여 Decision Process를 진행하게 된다.

Tree 내부에서 Decision Process를 수행하는 Finite State Machine의 동작은 다음과 같다.

(1) s0: Compare Frame이 Tree에 입력된다.

(2) s1: Compare Frame의 Current Node로 node RAM에 액세스한다.

CAN ID (11bit)	Current Node (9bit)	Feature (93bit)

Fig. 6. Structure of compare frame.
그림 6. 비교 프레임의 구조

(3) s2: Feature Index가 지정하는 Compare Frame의 Feature Data를 Node RAM에 저장된 Threshold와 비교하여 Compared Result 신호를 생성한다. 이때 Feature Data가 Threshold보다 크거나 같으면 Compared Result 신호는 1(참), 그렇지 않으면 0(거짓)이 된다.

(4) s3: s2에서의 비교 결과를 토대로 피드백 여부를 판단한다. 이는 Node SRAM에 저장된 Determined 신호로 판단할 수 있으며 Compared Result 신호가 참이면 Determined 신호의 MSB의 값에 따르고 거짓일 경우 LSB의 값에 따른다. Determined에서 각 Bit 값은 1이 Class 분류 완료, 0이 그렇지 않은 것을 의미한다. 예를 들어 Determined가 “10”이라면 비교 연산 결과가 참일 경우 Class가 결정되고 거짓이라면 추가적인 판단이 필요하므로 피드백을 하게 된다. 만약 Class가 결정된다면 State가 s5로 되어 결과가 출력되고 그렇지 않다면 State가 s4로 된다.

(5) s4: 피드백이 결정된 경우이며, Node RAM으로부터 다음 노드 번호를 지정하는 Nextnode를 읽어와서 Node SRAM의 Current Node에 저장하게 된다. 이때 Compared Result 신호가 참이면 Nextnode_T를, 거짓이면 Nextnode_F를 사용하게 된다. 이 Current Node 값은 s1에서 다시 Node RAM을 액세스하는 주소로 사용된다. 이와 같은 방식으로 피드백이 진행되며 Class가 결정날 때까지 반복하게 된다.

(6) s5: Decision Tree에서 해당 CAN 메시지에 대한 Class가 결정된 경우이다. CAN 메시지에 대한 Class는 비교 연산 결과와 그림 6에 저장된 Class 값에 따라 결정되며 1은 Attack, 0은 Normal을 의미한다. 이후 Compare Frame에 저장되어 있던 CAN ID와 Class를 그림 4의 Voter로 출력하고 모든 Tree의 결정 프로세스가 끝날 때까지 값을 유지한다.

모든 Tree의 결정 프로세스가 끝나면 Voter에서는 최종적으로 Class를 결정하게 된다. Class가 Attack이라

Nextnode_T (8bit)	Nextnode_F (8bit)	Class_T (1bit)	Class_F (1bit)	Determined (2bit)	Index (3bit)	Threshold (38bit)

Fig. 5. Data structure stored in node RAM.
그림 5. Node RAM에 저장되는 데이터 구조

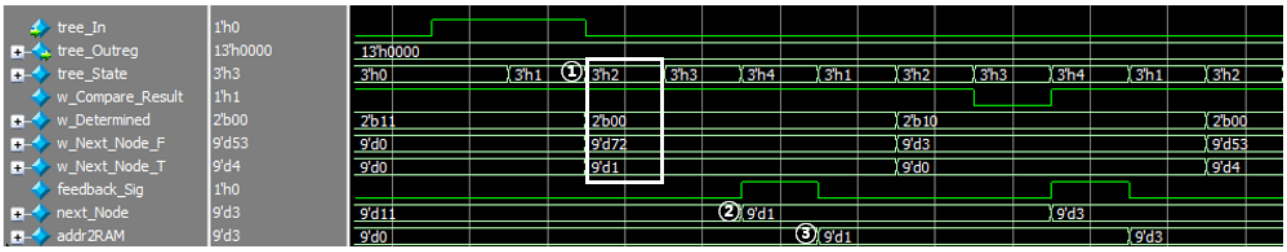

Fig. 7. Simulation waveforms when feedback is determined.
그림 7. 피드백이 결정되었을 때의 시뮬레이션 파형

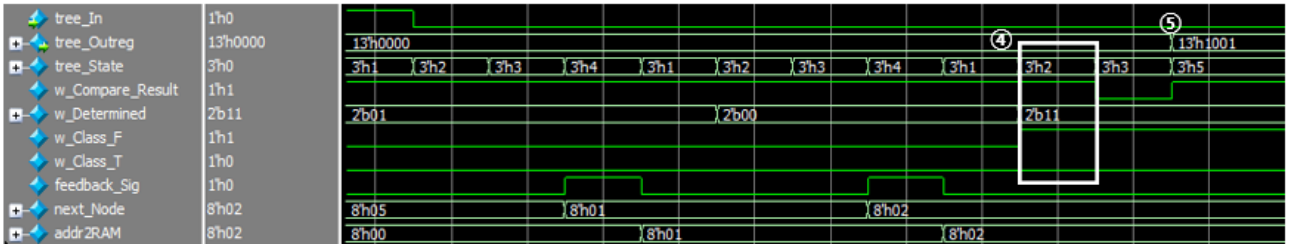

Fig. 8. Simulation waveforms when class is determined.
그림 8. Class가 결정되었을 때의 시뮬레이션 파형

면 값이 1이기 때문에 모든 Class를 더한 값이 Tree 수의 과반수를 넘으면 최종적으로 Attack, 아니면 Normal로 결정하게 된다.

2. 시뮬레이션

RIDS의 동작 및 정확도를 시험하기 위해 Siemens Modelsim을 이용하여 시뮬레이션을 진행했다. 그림 7은 Tree에서 Decision Process가 진행되던 중 피드백이 결정된 경우의 신호 파형을 나타낸다. ①에서 Tree의 State에 따라 Node RAM의 값이 출력되는 것을 확인할 수 있다. 이 경우 Determined가 "00"이기 때문에 비교 연산 결과가 어느 것이든 피드백이 된다는 것을 의미하며 참일 경우 노드 1번을, 거짓일 경우 노드 72번을 Current Node로 저장된다는 것을 알 수 있다. 이후 ②에서 판단 결과가 참이었기 때문에 노드 1번이 Current Node에 저장이 되는 것을 알 수있다. 그 다음에는 피드백되어 ③에서 노드 1번으로 Node RAM에 액세스 하는 동작을 확인할 수 있다.

그림 8은 Class가 결정되어 CAN 메시지에 대한 Class가 출력되는 경우의 신호 파형을 나타낸다. ④에서 Determined가 "11"이기 때문에 비교 연산결과가 어느 것이든 Class로 결정이 된다는 것을 의미한다. 또한, Class_F 신호와 Class_T 신호를 통해 어떤 Class가 결정되는지 알 수 있으며 ④에서는 비교 연산결과가 참일 경우 Normal, 거짓일 경우 Attack으로 결정된다. 이후 ⑤에서 CAN ID와 Class가 함께 출력되는 동작을 확인할 수 있다.

3. 트리의 개수와 깊이 최적화

RIDS는 트리의 개수와 깊이만 조절하면 CAN 버스에서 DoS 공격, Fuzzing 공격, Spoofing 공격을 모두 탐지할 수 있도록 확장이 용이한 구조로 설계되었다. 각각의 공격 중 하나의 유형만을 탐지할 수 있는 RIDS는 크기가 작아지고 모든 유형을 탐지할 수 있는 RIDS는 크기가 커진다. 트리의 개수와 깊이는 gridsearchCV[11]를 통해 최적화하였으며, 각각의 공격 유형에 대해 트리의 최적 개수 및 깊이는 표 1과 같다. RIDS의 하드웨어 크기는 트리의 개수와 깊이에 대체적으로 비례하는데 표 1에서 보면 Fuzzing 공격을 탐지하기 위해 하드웨어가 커지고 이 때문에 복합 유형을 탐지하기 위한 트리의 개수와 깊이도 커지는 것으로 보인다. 따라서 추후 RIDS의 개선을 통해 보다 적은 트리 개수와 깊이로 Fuzzing 공격을 탐지할 수 있는 하드웨어 아키텍쳐를 개발해야 할 것으로 보인다.

4. 성능 평가

해킹 탐지 성능을 평가하기 위해서 다음과 같은 과정을 수행하였다. 먼저 실제 차량에서 측정한 데이터 세트[14][15]에서 Scikit-Learn으로 일부 데이터 세트에 대해 학습을 수행하였다. 이후 Siemens Modelsim에서 RIDS를 수행하여 나머지 데이터 세트를 Attack과 Normal로 분류한 Class 출력값을 텍스트 파일로 추출하였다. 이때

단일 유형 탐지는 40만 개로 학습하고 10만 개로 검증하였으며 복합 유형 탐지는 120만 개로 학습하고 30만 개로 검증하였다. 마지막으로 RIDS의 출력값을 실제 해킹 여부와 비교하여 표 2와 같이 정확도, 민감도, 정밀도, F1 점수를 얻었다.

표 2를 보면 DoS 공격과 Fuzzing 공격과 같이 CAN 버스를 점유하기 위한 공격에 효과적으로 대처하는 것을 알 수 있으며 차량용 IDS로 사용하기에 적합하다. 그러나 Spoofing 공격에 대해서는 비교적 낮은 점수를 가지는 것을 알 수 있다. 따라서 추후 RIDS의 개선을 통해 Spoofing 공격에 대한 탐지 성능을 높이는 하드웨어 아키텍쳐를 개발해야 할 것으로 보인다.

Table 1. Optimized number and depth of trees in RIDS.
표 1. RIDS에서 최적화된 트리의 개수 및 깊이

Attack Type	Number of Trees	Depth of Trees
DoS Attack Only	16	8
Fuzzing Attack Only	39	9
Spoofing Attack Only	9	8
All Combined	46	9

Table 2. Performance evaluation of RIDS.
표 2. RIDS의 성능 평가

Attack Type	Accuracy	Sensitivity	Precision	F1 Score
DoS Attack Only	0.9999	1.0000	0.9999	0.9999
Fuzzing Attack Only	0.9900	0.9910	0.9800	0.9830
Spoofing Attack Only	0.9878	0.9644	0.9710	0.9677
All Combined	0.9859	0.9387	0.9850	0.9613

Ⅵ. 결론

본 논문에서는 자동차 내부 네트워크의 보안 취약점을 개선하기 위해 랜덤 포레스트 기반 침입 감지 시스템인 RIDS를 제안하였다. RIDS는 CAN 버스의 트래픽을 분석해 추출한 특징점을 바탕으로 훈련되었으며 Verilog HDL로 기술되었다. 제안된 RIDS를 검증하기 위해 Siemens Modelsim을 사용하여 로직 및 파형을 확인하였고 실제 차량에서 측정한 데이터 세트에 대해 정확도 및 F1 점수를 계산했다. 시뮬레이션 결과 RIDS는 DoS 공격과 Fuzzing 공격에 대해 높은 탐지 성능을 보이지만Spoofing 공격에 대해 다소 낮은 탐지 성능을 보인다. 따라서 RIDS의 성능을 높이기 위해 Spoofing 공격을 효과적으로 탐지하기 위한 추가적인 연구가 필요하다. 또한, 트리의 개수와 깊이는 하드웨어 크기에 직접 연관되어 있으며 제한적인 하드웨어 리소스를 효율적으로 사용하기 위해 개선된 하드웨어 아키텍처 개발이 필요하다.

References

[1] C. Kim, “A Study on the Standard Development Trend for Automotive Security Threats,” *Review of KIISC,* vol.29, no.1, pp.20-25, 2019.

[2] ISO 21434:2021, “Road vehicles - Cybersecurity engineering,” https://www.iso.org/standard/70918.html

[3] ISO 11898-1:2015, “Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical signalling,” https://www.iso.org/standard/63648.html

[4] S. Jeong, Y. Kim, and S. Lee, “Vehicle ECU Design Incorporating LIN/CAN Vehicle Interface with Kalman Filter Function,” *J.inst.Korean.electr.elctron.eng.*, vol.25, no.4, pp.762-765, 2021. DOI: 10.7471/ikeee.2021.25.4.762

[5] UNECE WP.29, “Proposal for a new UN Regulation on uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system,” http://www.unec.org/DAM/trans/doc/2020/wp29grva/ECE-TRANS-WP29-2020-079-Revised.pdf

[6] L. Breiman, “Random Forests,” *Machine Learning*, vol.45, pp.5-32, 2001.

[7] S. Mehedi, A. Anwar, Z. Rahman, and K. Ahmed, “Deep Transfer Learning Based Intrusion Detection System for Electric Vehicular Networks,” Sensors, vol.21, no.14, pp.4736, 2021.

[8] ISO 15031-1:2010, “Road vehicles – Communication between vehicle and external equipment for emissions-related diagnostics - Part 1: General information and use case definition,” https://www.iso.org/standard/51828.html

[9] C. Miller and C. Valasek, “Adventures in Automotive Networks and Control Units,” https://ioactive.com/pdfs/IOActive_Adventures_in_Auto

motive_Networks_and_Control_Units.pdf
[10] H. Park, Self-Studying Machine Learning + Deep Learning, Hanbit Media, 2020.
[11] https://www.python.org
[12] https://scikit-learn.org/stable
[13] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[14] H. Lee, S. Jeong and H. Kim, "OTIDS: A Novel Intrusion Detection System for In-vehicle Network by using Remote Frame," *Proceedings of Annual Conference on Privacy, Security and Trust,* pp.57-5709, 2017. DOI: 10.1109/PST.2017.00017
[15] E. Seo, H. Song, and H. Kim, "GIDS: GAN-Based Intrusion Detection System for In-Vehicle Network," *Proceedings of Annual Conference on Privacy, Security and Trust*, pp.1-6, 2018. DOI: 10.1109/PST.2018.8514157
[16] H. Song, H. Kim and H. Kim, "Intrusion Detection System-Based on the Analysis of Time Intervals of Messages for In-Vehicle Network," *Proceedings of International Conference on Information Networking*, pp.63-68, 2016. DOI: 10.1109/ICOIN.2016.7427089
[17] R. Hamming, "Error detecting and error correcting codes," *Bell Labs Technical Journal,* vol.29, no.2, pp.147-160, 1960. DOI: 10.1002/j.1538-7305.1950.tb00463.x
[18] D. Stabil, M. Marchetti, and M. Colajanni, "Detecting Attacks to Internal Vehicle Networks through Hamming Distance," *Proceedings of AEIT International Annual Conference,* pp.1-6, 2017. DOI: 10.23919/AEIT.2017.8240550
[19] X. Lin, R. Blanton, and D. Thomas, "Random Forest Architectures on FPGA for Multiple Applications," *Proceedings of GLS-VLSI,* pp.415-418, 2017. DOI: 10.1145/3060403.3060416

BIOGRAPHY

Daegi Lee (Member)

2022 : BS degree in Electronic Engineering, Soongsil University.
2022~ : Candidate for MS degree in Electronic Engineering, Soongsil University.
〈Main Interest〉 Automotive Electronics, Automotive SoC, Sensor Signal Processing

Changseon Han (Member)

2020 : BS degree in Electronic Engineering, Soongsil University.
2022 : MS degree in Electronic Engineering, Soongsil University.
〈Main Interest〉 Automotive Electronics, Automotive SoC, Sensor Signal Processing

Seongsoo Lee (Life Member)

1991 : BS degree in Electronic Engineering, Seoul National University.
1993 : MS degree in Electronic Engineering, Seoul National University.
1998 : PhD degree in Electrical Engineering, Seoul National University.
1998~2000 : Research Associate, University of Tokyo
2000~2002 : Research Professor, Ewha Womans University
2002~Now : Professor in School of Electronic Engineering, Soongsil University