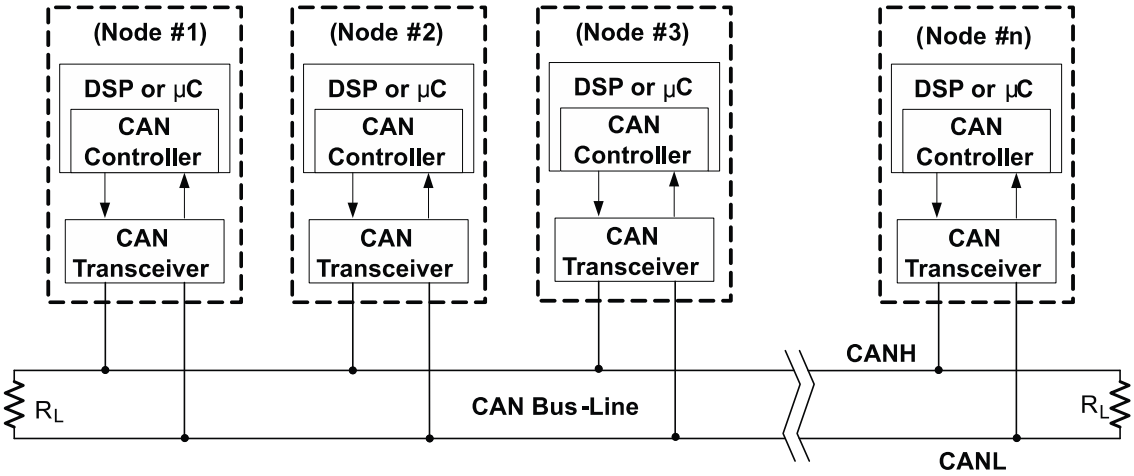
차량 이상 징후 탐지 TUTORIAL

1 ABSTRACT

본 튜토리얼에서는 차량에서 발생할 수 있는 공격 상황을 탐지하는 알고리즘을 소개합니다. CAN (Controller Area Network)은 차량 내부 네트워크에 사용되는 핵심적인 프로토콜로서 ECU (Electronic Control Unit) 간 신뢰성 있고 효율적인 통신 메커니즘을 지원합니다. CAN 프로토콜은 전송되는 패킷 상 시작점과 도착점에 대한 주소 정보를 포함하지 않고 우선순위를 기반으로 통신하는 특징이 있습니다. 더불어 모든 ECU 는 수신하는 메시지를 검증하는 과정이 존재하지 않기 때문에 메시지 주입 공격에 취약합니다. 공격자는 이러한 프로토콜의 특징을 이용하여 다양한 공격 방법을 재현할 수 있으며, 이는 운전자에게 심각한 위협을 초래합니다. 본 튜토리얼은 CAN 프로토콜의 원격 프레임을 이용하여 각 메시지의 응답 성능을 기준으로 침입의 여부를 판단하는 방법론을 사용하였습니다.

2 CONTROLLER AREA NETWORK

2.1 CONTROLLER AREA NETWORK

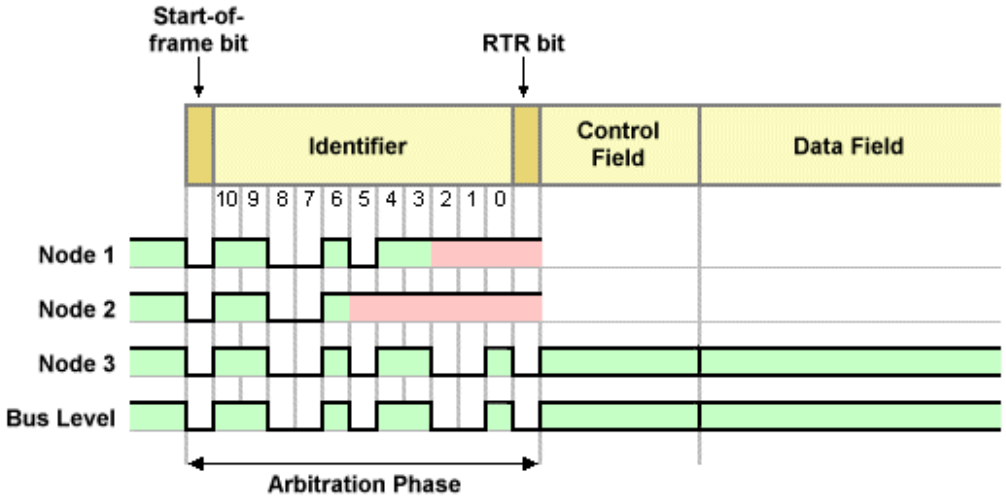


[그림 1] Controller Area Network Topology

CAN 프로토콜은 단일 버스에서 피어 투 피어 통신을 지원하며, 각 노드는 패킷을 전송하기 위해 프로토콜의 규약을 지키며 운영됩니다. CAN 패킷은 시작점과 도착점에 대한 주소가 없기 때문에 브로드 캐스트 방식을 통해 전달되며, 모든 노드는 해당 메시지를 수신 시 수용 여부를 결정하게 됩니다. 또한 주소 기반이 아닌 우선순위 기반으로 통신이 이루어지며, 우선순위가 낮은 패킷은 CSMA/CA (Carrier Sense Multiple Acees with Collision Avoidance)와 AMP (Arbitration on Message Priority) 정책에 따라 다음 순서에 전송되도록 설계되었습니다.

2.2 ARBITRATION

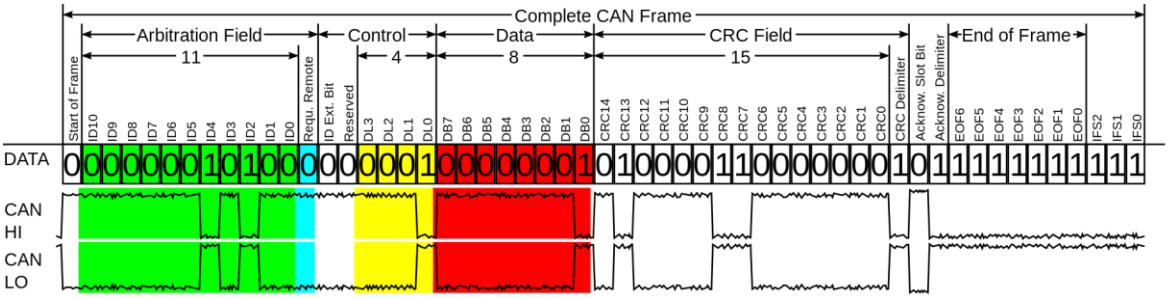
CAN 프로토콜은 데이터 충돌 시 동작되는 중재 규약을 갖고 있습니다. 모든 비트는 '0'일 때 우성 상태, '1'일 때 열성 상태를 나타냅니다. 왜냐하면 CAN 버스가 비트의 충돌시 AND 게이트에 따라 연산되기 때문입니다. 따라서 여러 노드가 동시에 메시지를 전송할 때 중재 식별자의 우성 여부를 판단하며 전송 될 메시지를 최종적으로 선택합니다.



[그림 2] Arbitration Process

[그림 2]는 세 개의 노드가 동시에 메시지를 전송하는 상황을 보여줍니다. 10 에서 6 번 비트까지는 모든 노드가 같은 비트 값을 갖기 때문에 CAN 버스는 동기화가 가능합니다. 하지만 5 번 비트에서 노드 2 는 열성 비트를 나타내므로 AND 연산에 따라 전송이 중시되고, 노드 1 과 3 의 우성 비트로 CAN 버스가 동기화 됩니다. 그리고 2 번 비트에서 다시 중재가 일어나 우성 비트를 포함한 노드 3 의 메시지가 최종적으로 선택됩니다. 따라서 작은 값의 식별자가 중재 과정에서 CAN 버스를 점유할 수 있으며 이는 다른 메시지보다 높은 우선순위를 나타냅니다.

2.3 STRUCTURE OF CAN FRAME



[그림 3] Standard CAN Frame

2.3.1 Start of frame

하나의 우성 비트로 구성된 필드로서 버스의 모든 노드에게 전송을 알리는 역할을 합니다. 모든 메시지는 버스의 전송 가능 여부를 확인 후 시작됩니다.

2.3.2 Arbitration Field

열한 개의 중재 식별자와 한개의 RTR 비트로 구성된 필드로서 전송되는 메시지의 성격을 알려주는 역할을 합니다. 중재 식별자는 여러 메시지가 동시에 전송될 때 중재할 수 있는 우선순위 정보를 나타내며, RTR 비트는 해당 메시지의 프레임 타입 대한 정보를 나타냅니다. RTR 비트가 우성 비트일 시 데이터 프레임이며 열성 비트일 시 원격 프레임입니다.

2.3.3 Control Field

두 개의 예약된 비트와 네 개의 DLC (Data Length Code)로 구성된 필드로서 메시지에 포함 된 데이터가 몇 바이트인지에 대한 정보를 담고 있습니다. 데이터 프레임은 DLC 에 명시한 만큼 데이터 필드의 크기를 구성하게 됩니다.

2.3.4 Data Field

다른 노드로 전송하기 위한 실질적인 데이터를 담고 있으며, 0 부터 8 바이트까지 구성될 수 있습니다.

2.3.5 Cyclic Redundancy Code Field

전송되는 메시지의 정보가 정상적인지 판단할 수 있도록 하는 필드로서 모든 노드는 해당 메시지의 변질 여부를 확인할 수 있습니다. 메시지가 오염되었다고 판단될 때 에러 프레임을 전송하여 해당 메시지를 다시 요청하는 기능을 합니다.

2.3.6 Acknowledge Field

메시지가 정상적으로 도착하였을 때 전송한 노드에게 알리는 역할을 하는 필드입니다. 정상적으로 도착하여 수신했다면 두 개의 비트 중 첫 번째 비트를 우성 비트로 변환합니다.

2.3.7 End of Frame

메시지의 끝을 알리는 프레임으로서 7 개의 열성 비트로 구성되어 있는 필드입니다.

2.4 TYPES OF CAN FRAME

2.4.1 Data Frame

데이터 프레임은 CAN 프로토콜의 핵심적인 프레임으로서 정보 및 명령 전달을 목적으로 사용됩니다. 따라서 데이터 프레임의 Data Field 의 값이 전달하고자 하는 정보를 담고 있습니다. Arbitration Field 의 RTR 비트가 우성(0) 비트일 때, 데이터 프레임으로 구성될 수 있습니다.

2.4.2 Remote Frame

원격 프레임은 목적지 노드로 부터 데이터 프레임을 요청할 때 사용됩니다. 특정 식별자를 갖는 원격 프레임이 브로드 캐스트 되면, 해당 식별자를 전송하는 목적지 노드는 응답으로 식별자가 일치하는 데이터 프레임을 전송합니다. Arbitration Field 의 RTR 비트가 열성(1) 비트일 때, 원격 프레임으로 구성될 수 있습니다.

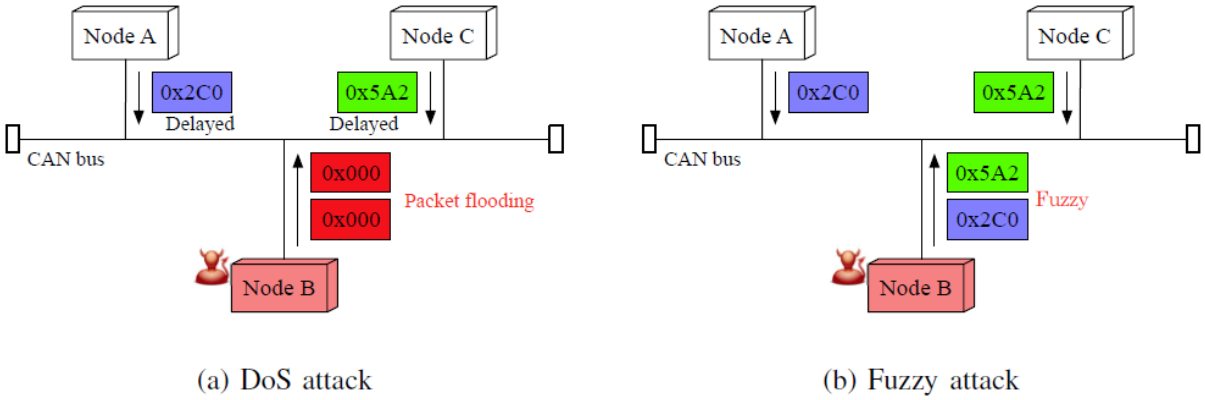
2.4.3 Error Frame

에러 프레임은 현재 전송되는 메시지가 변질되거나 CRC 검증 중 이상이 발견될 때 전송되어 해당 메시지의 이상 여부를 알리는 역할을 합니다. 에러 프레임이 브로드 캐스트되면 이상 여부를 모든 노드가 알 수 있으며, 해당 메시지를 전송한 노드는 재전송하도록 설계되어 있습니다.

2.4.4 Overload Frame

오버로드 프레임은 어떤 메시지의 처리가 완전히 이루어지지 않은 상태에서 다음 메시지를 수신할 때 전송되는 프레임입니다. 모든 노드는 수신 된 메시지가 완전히 처리된 후 다음 메시지를 수신할 수 있기 때문에 오버로드 프레임을 이용하여 순차적인 처리가 이루어질 수 있도록 합니다.

3 DATASET



[그림 4] Denial-of-Service & Fuzzy Attack

본 데이터 챌린지에서 제공되는 데이터셋은 도스, 퍼지, 재사용의 세 가지 공격 상황에 대한 탐지가 이루어져야 합니다. 총 1 시간 20 분으로 이루어진 CAN 데이터 세트는 임의의 시간에 공격 상황이 존재하며, 제출되는 알고리즘은 공격이 시작되는 시간과 끝나는 시간을 탐지할 수 있어야 합니다. *(평가 방법은 tutorial 의 5.Evaluation Standard 를참고하시기 바랍니다.)*

3.1 DENIAL-OF-SERVICE ATTACK

도스 공격은 상대적으로 우선순위가 높은 메시지를 짧은 주기로 CAN 버스에 전송하는 공격입니다. CAN 프로토콜은 우선순위 기반 프로토콜이기 때문에 메시지의 우선순위가 높은 메시지를 빠르게 주입하면 정상 메시지는 자연스럽게 지연되거나 전송이 실패하게 됩니다. 차량은 실시간으로 운전자의 명령에 따라 동작해야하기 때문에 메시지의 전송이 지연되거나 실패할 경우 심각한 위협을 초래할 수 있습니다.

3.2 FUZZY ATTACK

퍼지 공격은 랜덤하게 식별자를 선택하여 CAN 버스에 주입하는 공격입니다. 랜덤하게 식별자가 선택되기 때문에 차량에서 실제 역할이 있는 식별자가 주입될 수 있습니다. 실제 차량에 퍼지 공격을 재현하게 된다면 차량의 핸들이 갑자기 움직이거나 계기판이 비정상적으로 작동할 수 있습니다.

3.3 REPLAY ATTACK

재사용 공격은 특정 시간 동안 CAN 버스의 데이터를 추출하여 해당 데이터 묶음을 똑같이 주입하는 공격을 나타냅니다. 퍼지 공격과 같이 실제 역할이 있는 메시지가 삽입되기 때문에 운전자가 의도하지 않은 대로 차량이 동작할 위험이 있습니다.

4 METHODOLOGY

이번 섹션은 앞서 제시한 메시지 주입 공격을 탐지하는 알고리즘의 방법론에 대해 설명합니다. 우리는 원격 프레임을 이용하여 각 메시지의 응답 성능을 기준으로 침입을 탐지하는 알고리즘을 제시합니다. 탐지 노드는 여러 메시지를 선택하여 주기적으로 원격 프레임을 전송하도록 설계되어 있습니다. 따라서 원격 프레임을 수신 노드는 응답을 전송하기 때문에, 우리는 시간 간격과 요청과 응답 사이의 순서를 이용하여 침입을 탐지하도록 구성했습니다. 모든 노드는 메시지 수신 시 우선순위에 따라 순차적으로 수신하기 때문에 우리는 요청과 응답 사이의 시간 간격과 순서를 측정할 수 있었습니다. 우리의 탐지 알고리즘은 다음과 같은 피쳐를 선정하여 응답 성능을 평가합니다.

4.1 즉시 응답 비율

[그림 5] 즉시 응답

[그림 **5**]는 요청에 대한 즉시 응답을 보이는 데이터입니다. 3 번 인덱스의 '100'은 원격 프레임을 의미하는 플래그입니다. 즉시 응답은 탐지 노드가 특정 식별자를 포함한 원격 프레임을 전송했을 때, 응답으로 보내지는 메시지가 즉시 수신되는 경우를 나타냅니다. 따라서 메시지를 순서에 따라 출력하면 원격 프레임과 동일한 식별자를 갖는 데이터 프레임이 순차적으로 보이며, 이 사이에는 다른 메시지가 포함되어 있지 않습니다. 우리는 각 식별자 별로 1000 개의 큐 (Queue)를 생성하여 즉시 응답의 비율을 측정하였습니다. [그림 5]에서 볼 수 있듯이 공격이 없는 상황에서 모든 식별자는 응답 성능이 일정한 것을 알 수 있습니다. 하지만 메시지 주입 공격이 시작되는 순간 즉시 응답 비율이 줄어드는 것을 확인할 수 있습니다.

4.2 손실 응답 비율

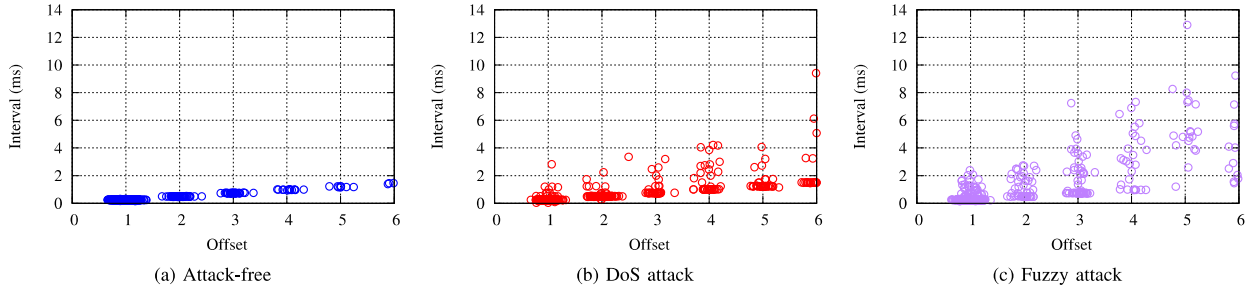
Remote Frame

```
Timestamp: 0.254877   ID: 01f1   100   DLC: 0
Timestamp: 0.255148   ID: 0080   000   DLC: 8    00 17 c8 09 19 11 19 8d
Timestamp: 0.255230   ID: 0000   000   DLC: 0
Timestamp: 0.255479   ID: 0081   000   DLC: 8    7f 84 8e 00 00 00 00 bd
Timestamp: 0.255576   ID: 0000   000   DLC: 0
Timestamp: 0.255824   ID: 018f   000   DLC: 8    00 36 19 00 00 3f 00 00
Timestamp: 0.255922   ID: 0000   000   DLC: 0
```

[그림 6] 손실 응답

손실 응답은 탐지 노드가 특정 식별자를 포함한 원격 프레임을 전송했을 때, 응답 메시지가 도착하지 않은 경우를 나타냅니다. 우리는 손실 응답의 기준을 원격 프레임 전송 이후 여섯번째 메시지까지 응답 메시지가 도착하지 않았을 때를 기준으로 정하였습니다. 공격이 없을 때 손실 응답의 비율이 1% 미만이기 때문에 여섯 번째 메시지를 기준으로 정할 수 있었습니다. 손실 응답의 비율은 즉시 응답 비율과 마찬가지로 1000 개의 큐를 생성하여 측정하였습니다. [그림 6]에서 볼 수 있듯이 공격이 없을 때 손실 응답 비율은 0~1%사이에서 유지되는 것을 알 수 있지만 공격이 재현되는 동시에 급격히 증가하는 것을 알 수 있습니다.

4.3 시간 간격과 응답 순서의 상관 관계



[그림 7] 각 상황 별 시간 간격과 응답 순서의 산점도 그래프

우리는 요청과 응답 메시지 사이의 시간 간격과 응답 순서에 대한 상관 관계를 사용하여 침입을 탐지할 수 있었습니다. [그림 7]에서 볼 수 있듯이 공격이 없을 때 응답 순서에 따라 시간 간격이 일정한 비율에 따라 증가하는 것을 알 수 있습니다. 하지만 주입 공격이 이루어진다면 응답 순서와 시간 간격 사이의 상관 관계가 작아지는 것을 확인할 수 있었습니다.

4.4 알고리즘

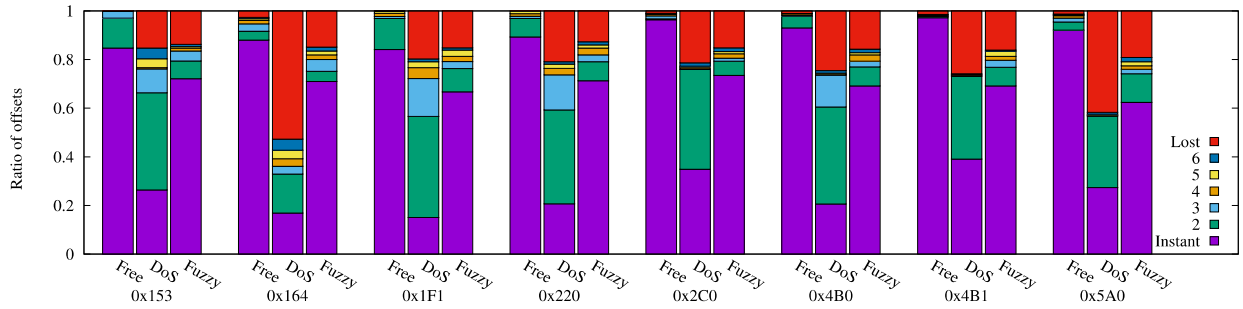
Algorithm 1 Offset and time interval measurement

```
Initialize:
    Queue_k = []
    Instant_k = 0,  Lost_k = 0
while message arrives do
    time, id ← timestamp, identifier of arrival message
    if arrival message is remote frame then
        remote_flag ← True
        target ← id
        offset ← 0
        request_time ← time
        continue
    if remote_flag then
        offset ← offset + 1
        if offset > 6 then                                  ▷ Lost reply
            increment Lost_target
            remote_flag ← False
            continue
        if id == target then
            interval ← time − request_time
            enqueue (offset, interval) to Queue_target
            if offset == 1 then                             ▷ Instant reply
                increment Instant_target
            remote_flag ← False
            continue
```

[그림 8] 요청과 응답 사이의 시간 간격과 응답 순서를 구하는 알고리즘

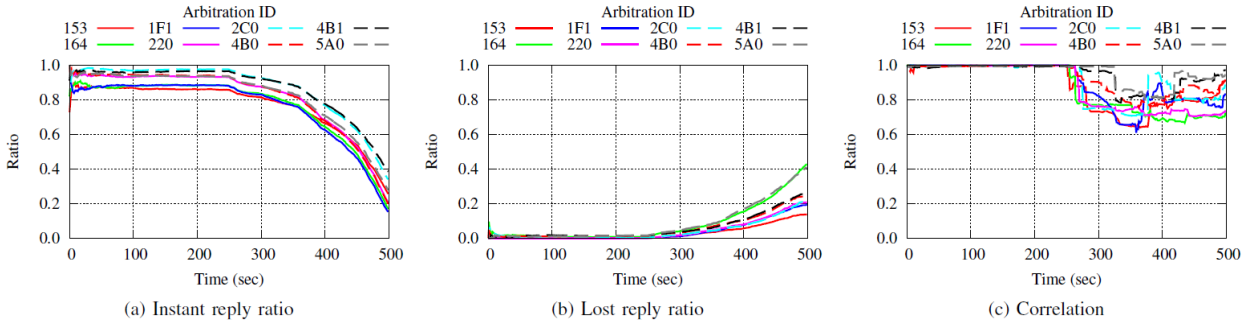
각 식별자의 큐에 요청과 응답 사이의 시간 간격과 순서를 측정하는 알고리즘입니다. 요청에 대한 응답 순서가 1 일 시 즉시 응답의 개수를 증가시키며, 응답 순서가 6 이상일 시 손실 응답의 개수를 증가시킵니다. 시스템 구현 시 모든 큐는 FIFO (First-in-First-out)으로 생성되며, 최신 1000 개에 대한 응답 비율을 계산하여 이상 범위를 초과하는지 탐지합니다.

4.5 EVALUATION



[그림 9] 각 상황 별 식별자의 응답 비율 히스토그램

각 상황별 식별자의 요청에 대한 응답 비율을 측정한 결과는 [그림 9]과 같습니다. 각 상황에 대해서 모두 1000 개의 원격 프레임 (요청)을 전송하고 응답으로 수신한 순서를 히스토그램으로 표현하였습니다. 식별자 별 차이는 있지만 모든 경우에 대해서 도스 공격과 퍼지 공격시 즉시 응답이 줄어들고 손실 응답이 증가하는 것을 확인할 수 있습니다.



[그림 10] 공격 상황 별 즉시 응답, 손실 응답, 상관 관계에 대한 시계열 그래프

본 방법론의 성능을 평가하기 위해 우리는 [그림 10]과 같이 500 초의 데이터 세트를 생성하였습니다. 초반 250 초는 공격이 없는 정상 상황이며 이후 250 초는 주입 공격이 이루어지는 데이터 세트입니다. 위의 그림과 같이 250 초를 기준으로 즉시 응답 비율이 감소하며, 손실 응답 비율이 증가합니다. 또한, 시간 간격과 응답 순서의 상관 관계 역시 250 초를 기준으로 급격히 변동되는 것을 확인할 수 있습니다.

5 EVALUATION STANDARD

제출된 알고리즘은 데이터셋의 공격이 이루어지는 상황을 탐지해야합니다. 제출된 알고리즘이 공격 상황의 시작 시간과 종료 시간을 산출하면 실제 공격의 시작 시간과 시간 차이를 구하여 탐지 응답 속도를 측정하도록 합니다. 또한 공격 상황을 탐지한 횟수에 따라 정확도를 측정합니다. 정확도 측정을 위해 *F1-Score* 식이 사용되며 다음과 같은 수식으로 평가됩니다.

Category		**실제 정답**	
		Positivie	Negative
실험 결과	Positive	*True Positive*	*False Positive*
	Negative	*False Negative*	*True Negative*

True Positive 는 실제 *True* 를 *True* 로 정확하게 예측한 상황을 의미하며, ***False Positive*** 는 실제 *False* 를 *True* 로 예측한 상황을 의미합니다. ***False Negative*** 는 실제 *True* 를 *False* 로 예측한 상황이며, ***True Negative*** 는 실제 *False* 를 *False* 로 예측한 상황을 의미합니다. 다음은 *Precision* 과 *Recall* 을 구하는 수식에 대한 설명입니다.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive},\ Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Precision 은 정확률로서 제안한 알고리즘이 공격으로 판단했을 때 실제 공격이 이루어진 비율을 나타내며, ***Recall*** 은 재현율로서 실제 공격이 이루어졌을 때 제안된 알고리즘이 공격으로 판단하는 비율을 나타냅니다.

$$F1 - Score = 2 * \frac{Precision \times Recall}{Precision + Recall}$$

F1-Score 은 Precision 과 Recall 에 따라 위의 수식을 사용하여 계산됩니다.

6 REFERENCE

본 tutorial 에서 소개되는 모델은 참고문헌 중 하나인 OTIDS 의 작동 원리를 기반으로 하고 있습니다. CAN 플레임 타입, 공격 유형 등 상세한 내용은 아래 논문에서 참고하시기 바랍니다. 논문 파일은 Data Analysis Challenge 사이트 Challenge - Tutorial 메뉴에서 확인하실 수 있습니다.

- Hyunsung Lee, Seong Hoon Jeong and Huy Kang Kim, “OTIDS: A Novel Intrusion Detection System for In-vehicle Network by using Remote Frame”, PST (Privacy, Security and Trust) 2017
- Song, Hyun Min, Ha Rang Kim, and Huy Kang Kim. "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network." Information Networking (ICOIN), 2016 International Conference on. IEEE, 2016.

- 본 데이터 분석 챌린지에서 제공하는 데이터셋을 기반으로 논문을 작성할 경우, 해당 논문을 인용해야합니다.