사물인터넷융복합논문지 Vol. 11, No. 2, pp. 185-191, 2025 https://doi.org/10.20465/KIOTS.2025.11.2.185

토큰 트리(token tree)를 이용한 사이버물리시스템 대상 퍼징 테스트(fuzzing test data) 데이터 생성 기법

신인철*
국립부경대학교 컴퓨터·인공지능공학부 교수

Fuzzing Test Data Generation Technique to Cyber-Physical System by Exploring Token Tree

Incheol Shin*
Associate Professor, Department of Computer Engineering and Artificial Intelligence, Pukyong National University

요 약 특히, 사이버시스템과 물리시스템이 결합된 사이버물리시스템(CPS, Cyber Physical System)으로 구성된 사물인터넷(IoTs, Internet of Things) 기기는 공개된 장소에서 동작하는 특성으로 인해 잠재된 취약점으로 인한 파급력이 타 IT(Information Technologies) 시스템에 비해 영향력이 더 크다고 할 수 있다. 따라서, 이 같은 사물인터넷 기기에 존재하는 취약점을 식별하고 이를 제거함으로써 보안성을 향상하기 위한 도구로 퍼징 테스가 유용하지만, 기존의 IT 기반 퍼징 테스트는 사물인터넷의 보안 특성상 활용도가 부족하다. 본 연구에서는 이 같은 단점을 극복하기 위해 토큰 트리(Token Tree) 기반 퍼징 테스트 데이터 변이 기술을 제안함으로써, 사물인터넷 동작 및 보안 특성을 고려한 취약점 식별 기술을 제안한다.

주제어 : 사물인터넷, 사이버물리시스템, 퍼징 테스트, 사이버물리취약점

Abstract The Korea Internet of Things Society. In particular, Internet of Things (IoTs) devices, which are composed of Cyber Physical Systems (CPS), which combine cyber systems and physical systems, have a greater impact from potential vulnerabilities than other Information Technologies(IT) systems due to their characteristics of operating in public places. Therefore, fuzzing tests are useful as a tool for improving security by identifying vulnerabilities in such IoT devices and removing them, but existing IT-based fuzzing tests are not very useful due to the security characteristics of the IoT. In order to overcome this shortcoming, this study proposes a vulnerability identification technique that considers the operation and security characteristics of the IoT by proposing a Token Tree-based fuzzing test data mutation technique.

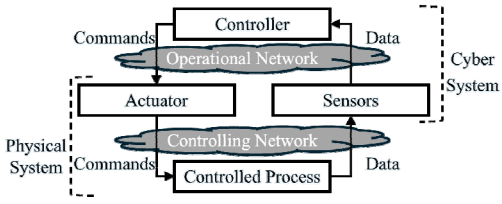
Key Words : Internet of Things, Cyber-Physical Systems, Fuzzing Test, Cyber-Physical Vulnerability

이 논문은 국립부경대학교 자율창의학술연구비(2023년)에 의하여 연구되었음
*교신저자 : 신인철(icshin@pknu.ac.kr)
접수일: 2025년 01월 20일 수정일: 2025년 02월 10일 심사완료일: 2025년 02월 25일

1. 서론

1.1 사물인터넷 기기 동작 및 보안 특성

사이버물리시스템 구조로 구성된 사물인터넷 기기는 그림 1과 같이 실세계에서 사용자가 원하는 제어를 수행하는 물리시스템과 이 같은 물리시스템과 연계된 센서(sensor)를 통해 데이터를 수집하고 이를 분석한 후, 제어 명령(control commands)을 생성하는 사이버시스템으로 구성된다.



[Fig. 1] Structure of Cyber Physical Systems

사물인터넷은 사이버시스템을 통해 물리시스템을 제어할 수 있으므로 사람이 작업하기 어렵거나 위험한 환경이나 원격에서 특정 환경을 관리할 수 있다. [1][2]

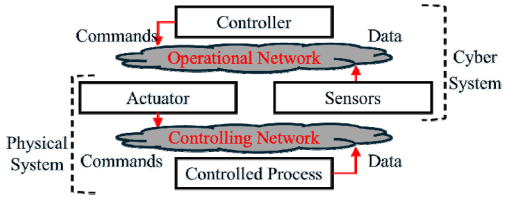
이 같은 사이버물리시스템 기반 사물인터넷 기기는 기존 정보기술 시스템과 비교하여 그 구성이나 운용 측면에서 새로운 형태의 보안 특성이 있다. 사이버물리시스템은 정보만을 다루는 정보기술 시스템과 달리 정보와 함께 물리시스템을 제어하기 위한 명령어를 함께 처리한다. 표 1과 같이 사이버물리시스템을 정보기술 시스템과 비교할 때, 정보와 제어 명령을 함께 다루는 특성으로 인해 정보보호의 3대 요소의 우선순위가 AIC(Availability, Integrity, Confidentiality)인 모델을 통해 가용성을 극대화해야 함을 알 수 있다.[3,4,5,6]

〈Table 1〉 Comparison between Information Tehcnology Systems and Industrial Control Systems

		Information Technology Systems	Cyber Physical Systems
Users/ Operators		Information System Users	Physical System Experts
Operation Period		About 5 Years	More Than 10 Years
Communication Data		Information	Information + Control Commands
Security Triad	High	Confidentiality	Availability
	Medium	Integrity	Integrity
	Low	Availability	Confidentiality

1.2 퍼징 테스트 기반 사물인터넷 기기 취약점 식별 접근법

이 같은 사이버물리시스템을 대상으로 기존의 정보기술 시스템에 적용한 차단(isolation) 기반 보안기술을 적용하면 그림 2와 같이 제어 명령이나 정보를 교환할 수 없는 상황이 발생하여 가용성을 보장할 수 없게 된다.



[Fig. 2] Isolation based security approach for Cyber Physical Systems

따라서, 사이버물리시스템 기반의 사물인터넷 기기의 가용성을 보장하기 위해서는 기존 정보기술 시스템을 대상으로 취약점을 탐지하기 위한 기법이 아닌 정상 상태에서 동작하는 정보를 변이(mutation)하여 사이버물리시스템의 오동작을 유발할 수 있는 퍼징 테스트 기법이 유용하다. 따라서, 본 논문에서는 미공개된 프로토콜을 사용하는 사물인터넷 기기에 대한 취약점을 식별하기 위해 기존의 LZFuzz[7]의 토큰(token) 정보를 효과적으로 관리하여 퍼징 테스트 데이터를 생성하는 TTFuzz[8] 기법에 소프트웨어공학에서 사용하는 조건 커버리지(condition coverage) 테스트를 활용한 토큰 변이 기법을 제안한다.

이를 위해 1장에서는 사이버물리시스템 기반 사물인터넷 기기의 특징과 보안 요구사항을 소개하였으며, 2장에서는 퍼징 테스트 관련 기술에 관한 기존 연구 조사 내용을 설명한다. 사이버물리시스템의 일종인 산업제어시스템 대상 퍼징 테스트 관련 기술에 대한 문제점과 본 연구에서 제시하는 개선 기술 내용을 3장에서 설명한다. 기술 구현에 따른 실험 내용과 결론을 각각 4장과 5장에서 소개한 후 본 논문을 마무리한다.

2. 퍼징 테스트 기반 취약점 식별 관련연구

Radamsa는 변이 기반 퍼징 도구로, 퍼징 설정을 학습하거나 조정하지 않고 항상 동일한 방식으로 주어진 입력 데이터를 랜덤 변이하여 테스트 입력 데이터로 생성한다. 정상 입력 데이터를 무작위로 변이만 하는 경우

유효한 테스트 데이터를 생성할 확률이 낮다. 특히 구조화된 데이터를 다루는 경우, 랜덤 변이만 가하는 방식은 프로토콜에 맞지 않는 입력을 생성하기 쉬워 테스트의 대부분이 유효하지 않을 수 있다.[9]

Peach라는 변이 기반 퍼저는 입력 데이터 모델을 정의하고 이를 기반으로 테스트 데이터를 생성한다. 변이 전략은 랜덤 변이, 순차 변이, 랜덤 결정적 변이를 사용한다. 랜덤 변이는 규칙 없이 무작위 변형을 가하고, 순차 변이는 정해진 순서에 따라 변형하며, 랜덤 결정적 변이는 무작위로 변이를 생성하지만 입력 데이터 모델을 통해 특정 조건들을 준수하며 변이를 발생시킨다. 입력 데이터 모델(PIT 파일)을 사용자가 직접 정의해야 하며 이 과정에서 시간이 많이 소요되며 데이터 형식, 데이터 요소 간의 상호 연관성 그리고 프로토콜에 대한 깊은 이해가 필요하다. 만약 잘못된 규칙을 정의하게 되면 테스트 결과에 대한 신뢰성과 데이터 필드의 상호 연관성이 떨어져 효과적이지 못하고 비효율적인 테스트 케이스를 생성하게 된다. [10]

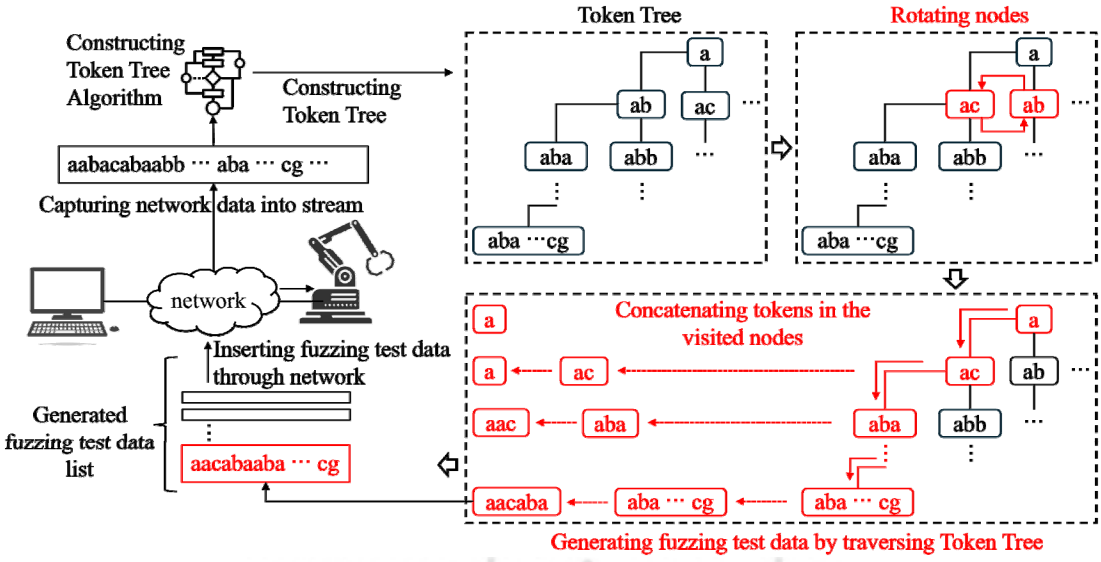
Boofuzz는 동적 프로토콜 정의 구문을 제공하여, 취약점 점검 대상 시스템의 프로토콜을 상세하게 정의한 후 프로토콜 정의에 따라 각 필드에 유효한 변이 값을 생성해 퍼징을 진행한다. 하지만 Peach와 마찬가지로 프로토콜 세부 사항에 대한 의존성이 크기 때문에 점검 대상 시스템 혹은 대상 프로토콜에 대한 세부적인 사전 지식이 필요하다. 그뿐만 아니라 체크섬, 시퀀스 번호 등과 같은 동적 필드들의 업데이트를 처리하는데 추가적인 연산 시간이 소요되어 퍼징 속도에도 영향을 미친다. 이에 따라 복잡한 동적 요소가 있는 프로토콜에서는 퍼징 속도가 느려질 수 있다. [11]

이 외에도 PULSAR[12], AFLNET[13], Fw-fuzz[14], Logos[15]등 다양한 퍼징 데이터 생성 기법이 연구 및 개발되었지만, 이 같은 기술들은 정보기술 시스템 대상 취약점을 식별하기 위한 기술로서 앞서 설명한 바와 같이 사이버물리시스템 기반의 사물인터넷 기기의 취약점을 식별하는 기법으로 적합하지 않다.

3. 토큰 트리를 이용한 사물인터넷 기기 대상 퍼징 테스트 데이터 생성 기법

3.1 토큰 트리(token tree) 개요

LZFuzz 알고리즘을 통해 네트워크 트래픽으로부터 식별한 토큰에 대한 선후 관계를 구축하기 위한 RTT(Relational Token Tree)와 토큰의 존재여부를 신속하게 확인할 수 있는 TT(Token Tree) 두 가지의 자료구조 구성 기법이 이미 연구된 바 있다. [8] 관계형 토큰 트리(relational token tree)는 네트워크를 통해 수집된 트래픽의 필드로 예측한 토큰의 선후 관계를 표현하기 위한 트리로써, 토큰 식별이 완료된 후 이들 관계를 조합하여 퍼징 테스트 데이터를 생성하기 위한 자료 구조이



[Fig. 3] Token tree based data mutation, fuzzing test data generation and test

며, 토큰 트리(token tree)는 기존의 LZFuzz에서 토큰 후보 문자열이 LZ 테이블에 존재하는지를 확인하는 과정에서 걸리는 시간을 단축하기 위한 검색 트리이다.

3.2 토큰 트리 기반 퍼징 테스트 수행 구조

본 논문에서는 기존 TTFuzz[8]에서 연구한 토큰 트리 구성 기법을 이용하며, 그림 3에서는 이 같은 기법을 통해 구성된 토큰 트리 노드에 대한 회전(rotating)을 적용한 변이 과정을 거친 후, 해당 트리의 노드별 방문에 따른 각 노드의 토큰들을 결합(concatenate)하는 방식으로 퍼징 테스트 데이터를 생성한다. 이같이 생성된 테스트 데이터는 취약점 점검 대상 시스템으로 전송하여 테스트를 수행한다.

perform_fuzzing_test procedure	
Input	*TT_root* (Root node of the Token Tree) *target* (Target to perform vulnerable test)
Output	*fuzzing_data (Data for fuzzing test)*

fuzzing_data_list ← ∅;
/* Initializes the fuzzing test data list. */

vulnerabilities ← ∅;
/* Initializes the variables that store the list of vulnerabilities (both fuzzing data and response */

p_root ← *TT_root*;
/* The root node of the token tree (TT) passed as an argument is stored in the variable p_root. */

fuzzing_data_list ← mutate_tokens(*p_root*);
/* The fuzzing test data generated as a result of performing mutation work using a token tree is stored in fuzzing_data_list. */

WHILE *fuzzing_data_list* ≠ ∅ **DO**
  fuzzing_data ← retrieve(*fuzzing_data_list*);
  /* Extract test data one by one from the fuzzing data list and save it in fuzzing_data. */

  response ← send_to_target(*fuzzing_data*, *target*);
  /* Fuzzing test data is sent to the vulnerability inspection target system through the network and the response is stored in the response variable.*/

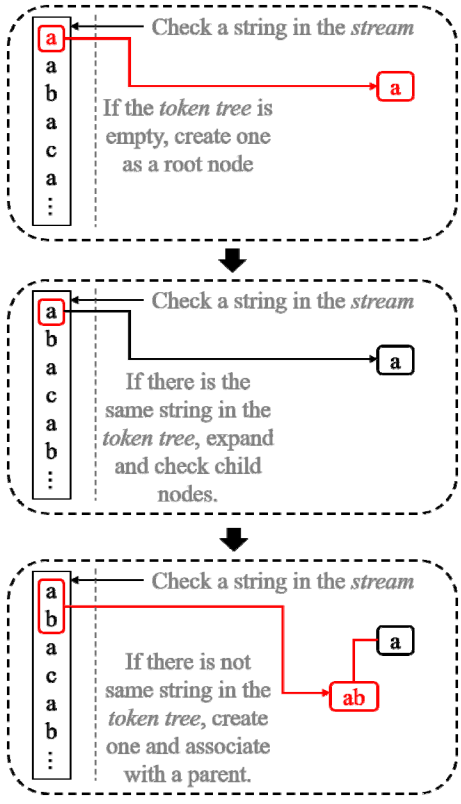
  IF is_normal(*response*) ≠ TRUE **THEN**
    add_to(*vulnerabilities*, *fuzzing_data*, *response*);
  ENDIF
  /* If the response to the fuzzing test data is not normal, it is stored in vulnerabilities by associating the input and response together. */
ENDWHILE

RETURN *vulnerabilities*;
/* Returns vulnerabilities containing the fuzzing test data that caused abnormal states in the target of the vulnerability scan and the response to it. */

[Algo. 1] Algorithm to perform fuzzing test to target systems.

위 알고리즘 1은 그림 3에서 제시한 퍼징 테스트를 수행하는 절차에 관한 내용으로, 네트워크로부터 수집한 스트림을 통해 생성한 토큰 트리를 입력받은 후 다음에 설명할 변이(mutation) 및 결합(concatenation) 알고리즘들을 통해 퍼징 테스트 데이터를 생성한다. 이후 점검 대상 시스템으로 전송된 퍼징 테스트 데이터에 대한 반응(response)을 통해 비정상 상태를 유발할 때 그 테스트 입력과 응답을 함께 취약점 목록에 저장하여 반환한다.

다음 그림 4에서는 사물인터넷 시스템으로 구성된 네트워크에서 수집한 스트림을 통해 실시간으로 토큰 트리를 구성하는 과정을 설명한다.[8]



[Fig. 4] Example of constructing token tree from network stream data

3.3 토큰 트리 기반 퍼징 테스트 데이터 생성 기법

3.3.1 조건 커버리지(condition coverage) 테스트 데이터 생성 기법

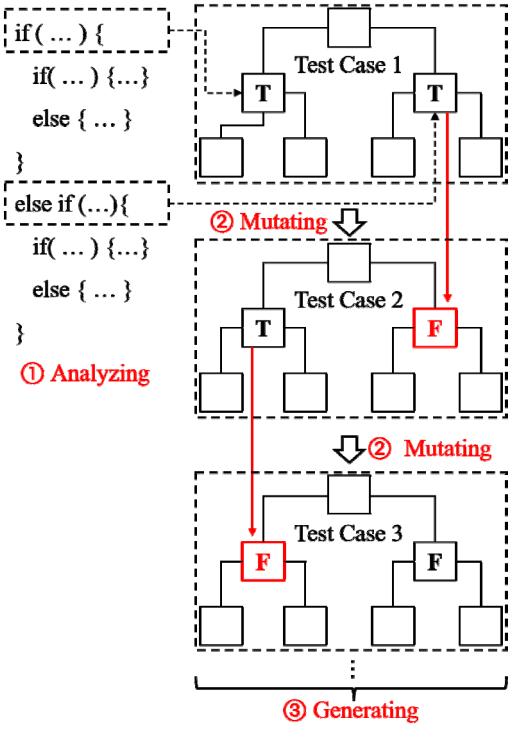
소프트웨어 공학에서 사용하는 소프트웨어 테스팅 기법으로 우선 브랜치 커버리지(branch coverage는 각

조건분기문의 불리언 표기식(boolean expression)이 참(T, True)/거짓(F, False)으로 모두 실행되었는지를 검사하기 때문에, 프로그램의 조건 분기문에 있는 전체 불리언 조건에 대해 참 혹은 거짓을 조합하여 프로그램 내 명령문들이 모두 실행되도록 테스트 데이터를 생성한다.

명령문 커버리지(statement coverage)가 가지고 있는 단점을 보완하는 커버리지 중 브랜치 커버리지는 검사 데이터 생성량이 비교적 적은 편이지만, 하지만 앞서 언급한 바와 같이 전체 불리언 표기를 참 혹은 거짓 중 하나의 논리값으로 검사하기 때문에 조건문에 대한 조합을 통해 모든 경로를 확인하지 않는 단점이 존재한다.

이를 보완하기 위해 조건 커버리지 테스트 기법이 개발되었으며, 조건 분기문에 있는 모든 표현에 대한 참 혹은 거짓을 다양하게 조합하여 검사하기 위한 데이터를 생성하며, 이를 통해 모든 분기 경로에 존재하는 참 혹은 거짓에 대한 조합을 검사할 수 있다.

그림 5에서는 각 조건문에 대해 참과 거짓을 조합하여 데이터를 입력함으로써 모든 경로에 존재하는 명령어들에 대한 실행을 통해 시스템의 취약점을 찾기 위한 퍼징 테스트 데이터를 생성하는 방법을 설명한다.



[Fig. 5] Structure of condition coverage test

3.3.2 토큰 트리 기반 퍼징 데이터 생성 기법

앞서 언급한 바와 같이 관계형 토큰 트리는 취약점 분석 대상 네트워크로부터 수집한 트래픽을 여러 개의 토큰으로 분리한 후, 이들 간 선행 및 후행 관계를 표현하기 위한 자료 구조이다. 이 같은 토큰들에 대한 선행 및 후행 관계는 소프트웨어 공학에서 검증된 조건 커버리지 기반의 테스팅 기법을 기반으로, 토큰 트리의 서브 트리(sub tree)에 대한 회전(rotation)을 통한 변이 과정을 수행한다. 다음 알고리즘 2에 설명한 바에 의하면, 이 같은 변이 과정의 중간에는 깊이 우선 탐색 기법을 활용한 순회-결합 기법을 이용하여 각 노드에 저장된 토큰들을 조합하여 퍼징 테스트 데이터를 생성한다. 토큰 변이 알고리즘은 모든 자식에 대한 위치 순회 시마다 모든 노드

mutate_tokens procedure	
Input	*TT_root* (Root node of the Token Tree), *cur_node* (Current node to swap children)
Output	*fuzzing_data (Data for fuzzing test)*

```
fuzzing_data ← ∅;
/* Initializes the fuzzing test data list.*/

p_root ← TT_root;
/* The root node of the token tree (TT) passed as an
argument is stored in the variable p_root. */

n = num_of_children(cur_node)
/* The number of child nodes of the current node (cur_node)
passed as an argument is stored in the variable n. */

FOR i = 1, ..., n DO
/* Repeats the number of child nodes of the current node
(cur_node) passed as an argument. */

    tmp ← cur_node.child1;
    FOR j = 1, ..., n-1 DO
        cur_node.childj ← cur_node.childj+1;
    cur_node.childn ← tmp;
    /* Rotates left using the pointer to the child nodes of the
current node (cur_node) passed as an argument.*/
    ENDFOR

    fuzzing_test_data ← fuzzing_test_data ∪
    concatenate_tokens_by_traversing(p_root, S)
    /* It traverses the Token Tree and combines the tokens
stored in each node to generate fuzzing test data. */

    FOR j = 1, ..., n DO
        mutate_tokens(TT_root, cur_node.childj);
        /* Recursively calls a function transforming the TT and
generates mutant fuzzing data by rotating the child nodes of
cur_node. */
    ENDFOR
ENDFOR

RETURN fuzzing_data;
```

[Algo. 2] Algorithm to Mutate Tokens in Token Tree in order to Generate Fuzzing Data for Industrial Control Systems

에 대한 순회가 일어나기 때문에 $O(n^3)$의 수행 시간 복잡도를 가지며 이를 구현한 실험 결과를 4장에서 설명한다.

알고리즘 3에 따르면, 관계형 토큰 트리 내 서브 트리 교환을 통해 관계형 토큰 트리의 구조 변형 후, 해당 트리를 깊이 우선 탐색(DFS, Depth First Search) 기반 순회를 통해 각 노드를 방문하고 첫 번째 단말 노드(leaf node) 방문 후부터 수행하는 순회 과정에서 방문하는 노드 내 저장된 토큰들을 하나의 문자열에 결합하는 방식으로 퍼징 테스트 데이터를 생성한다.

concatenate_tokens_by_traversing procedure	
Input	*vertex* (A node of the Token Tree), S (Statck to store tokens in Token Tree)
Output	*fuzzing_data (Data for fuzzing test)*

```
fuzzing_data ← ∅;
/* Initialize variables to store fuzzing test data. */

IF v = LEAF THEN
   fuzzing_data ← convert_to_data(S);

   RETURN fuzzing_test_data;
/* If the currently visited node is a leaf node, all tokens
stored in the stack are concatenated and returned as fuzzing
test data. */
ENDIF

p_root.isVisited ← TRUE

FOR u ∈ p_root.children DO
   IF u.isVisited ≠ TRUE THEN
      S.push(u);

      concatenate_tokens_by_traversing(u, S);

      S.pop()
   ENDIF
ENDFOR

RETURN fuzzing_data
```

[Algo. 3] Algorithm to Concatenate Tokens in Token Tree by Traversing

이 같은 방식을 통해 관계형 트리에 존재하는 다양한 경로(다양한 토큰 간 선후 관계 조합)를 식별할 수 있으며, 퍼징 테스트 데이터를 생성하기 위한 체계적인 변이 과정을 지원한다.

4. 성능분석

앞서 연구한 사이버물리시스템 기반 사물인터넷 기기 대상 취약점 식별을 위한 퍼징 테스트 데이터 생성 기법에 대한 실효성을 검증하기 위한 구현에 따른 성능 결과는 표 2와 같다. 사이버물리시스템의 일종으로 다양한 사물인터넷 시스템이 적용된 산업제어시스템에 활용되고 있는 DNP(Distributed Network Protocol) 3.0 프로토콜을 구현한 opendnp3를 통해 재현한 제어네트워크를 대상으로 실험을 진행하였으며, 비교군으로써, LZFuzz와 그 성능을 비교 분석하였다. 실험은 24 core Apple M2 Ultra 칩과 192GB 메모리를 탑재한 시스템에서 구동하였으며, 실험결과는 다음과 같다.

〈Table 2〉 Performance Comparison on Generating Fuzzing Test Data between LZFuzz and TTFuzz with Mutation

	TTFuzz based Mutation		LZFuzz	
	Time(ms)	Increment(%)	Time(ms)	Increment(%)
10KB	2889.7	N/A	59389.2	N/A
15KB	3789.2	31.12	82989.8	39.74
20KB	4423.9	43.14	129283.7	55.78
25KB	6928.8	27.74	212482.3	64.35
30KB	8438.3	21.78	382918.3	80.21
35KB	12283.6	45.56	759342.8	98.30

LZFuzz 기법을 통해 퍼징 데이터를 생성하는 경우, 실험 환경 산업제어네트워크로부터 수집한 스트림 형태의 데이터를 토큰화하여 무작위로 가능한 많은 테스트 케이스를 생성하기 때문에 TTFuzz에 비해 더 많은 데이터를 생성하는 것으로 보인다. 본 연구를 통해 제시한 TTFuzz 기반 변이 기법은 토큰 간 관계를 조합하는 방식을 통해 변이 작업을 수행하기 때문에, 테스트 데이터 생성 시간이 많이 줄어든 것으로 판단된다. 이 같은 성능을 통해 실시간으로 수집된 정상 데이터를 통해 토큰 트리를 생성하고 변이가 가능할 것이다. 본 연구에서는 사이버물리시스템의 일종인 산업제어시스템 대상 퍼징 데이터 생성과 관련한 성능을 비교 분석한 연구로써 향후 이 같은 기술들을 실제 시스템을 대상으로 취약점에 활용한 결과를 추가적인 연구를 통해 제시할 계획이다.

5. 결론

독점 프로토콜을 사용하는 사이버물리시스템 취약점 식별 방법으로 검증된 토큰 기반 퍼징 테스트 방식을 활용하되 실시간 적응형 퍼저를 개발하기 위한 데이터 변이 기법을 제시하였다. 이를 위해, 식별된 토큰의 선행 관계를 트리 구조로 구성한 관계형 토큰 트리는 기존에

소프트웨어 테스트 기법에서 검증한 조건 커버리지 기반의 테스트 데이터 생성 기법을 이용하는 변이에 활용함으로써, 효과적인 테스트 데이터 생성이 수행되도록 개선하였다. 이를 통해 다양한 사물인터넷 기기에서 발생할 수 있는 물리적 취약점을 조기에 식별하여, 가용성을 극대화할 수 있을 것이다.

REFERENCES

[1] E.J.Colbert and A.Kott, "Cyber-security of SCADA and other Industrial Control Systems," Springer, Vol.66, 2016.

[2] S.Ali, T.A.Balush, Z.Nadir and O.K.Hussain, "ICS/SCADA System Security for CPS," Cyber Security for Cyber Physical Systems, pp.89-113, 2018.

[3] A.Ara, "Security in Supervisory Control and Data Acquisition (SCADA) based Industrial Control Systems: Challenges and Solutions," IOP Conference Series: Earth and Environmental Science, International Conference on Sustainability: Developments and Innovations (ICSDI-2022), Vol.1026, 2022.

[4] Y.H.Lee, J.H.Ryu and J.H.Park, "Research Trends and Considerations of Security Technology of Industrial Control System," Annual Conference of Korea Information Processing Society Conference, pp.149-152, 2018.

[5] T.H.Lee, "Control of Cyber-Physical Systems Under Cyber-Attacks," Institute of Embedded Engineering of Korea, Vol.14, No.5, pp.269-275, 2019.

[6] T.H.Lee. "Control of Cyber-Physical Systems Under Cyber-Attacks," IEMEK Journal of Embedded Systems and Applications, Vol.14, No.5, pp.269-275, 2019.

[7] R.Shapiro, S.Bratus, E.Rogers and S.Smith. "Identifying Vulnerabilities in SCADA Systems via Fuzz-Testing," Critical Infrastructure Protection V, Springer Berlin Heidelberg, pp.57-72, 2011.

[8] I.Shin, "Token Tree based Fuzzing Technique for Physical Vulnerability Assessment for Industrial Control Systems," Journal of the Korea Institute of Information & Communication Engineering; Vol.27, pp.397-404, 2023.

[9] V.J.M.Manès et al., "The Art, Science, and Engineering of Fuzzing: A Survey," in IEEE Transactions on Software Engineering, Vol.47, No.11, pp.2312-2331, 2021.

[10] L.Zhang, J.Liang, L.Liu, Z.Jiang and J.Liu, "Improvement of the sample mutation strategy based on fuzzing framework peach," 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, China, pp.33-37. 2018.

[11] P.Cousineau and B.Lachine, "Enhancing Boofuzz Process Monitoring for Closed-Source SCADA System Fuzzing," 2023 IEEE International Systems Conference (SysCon), pp.1-8, 2023.

[12] H.Gascon, C.Wressnegger, F.Yamaguchi, D.Arp, K.Rieck. "Pulsar: Stateful Black-Box Fuzzing of Proprietary Network Protocols," SecureComm, Vol.164. pp.330-347, 2015.

[13] V.T.Pham, M. Böhme and A. Roychoudhury, "AFLNET: A Greybox Fuzzer for Network Protocols," 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), Porto, Portugal, pp.460-465, 2020.

[14] Z.Wang, Y.Zhang, X.Li, and J.Chen, "Fw-fuzz: A Code Coverage-Guided Fuzzing Framework for Network Protocols on Firmware", Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS), pp.1982-1995, 2022.

[15] F.Wu, Z.Luo, Y.Zhao, Q.Du, J.Yu, R.Peng, H.Shi, and Y.Jiang, "Logos: Log Guided Fuzzing for Protocol Implementations," Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), pp.1720-1732, 2024.

신 인 철(Incheol Shin) [정회원]

- 2010년 8월 : 플로리다대학교 컴퓨터공학 (공학박사)
- 2010년 6월 ~ 2014년 2월 : 국가보안기술연구소 선임연구원
- 2014년 3월 ~ 2021년 2월 : 국립목포대학교 정보보호학과 조교수
- 2021년 3월 ~ 현재 : 국립부경대학교 컴퓨터인공지능공학부 부교수

〈관심분야〉

근사알고리즘, 컴퓨터이론, 사물인터넷, 사이버물리시스템 보안