구현된 장비에 대한 UPnP 프로토콜 Fault 테스트

김민식[O*] 최경희[*] 정기현[*] 김상중[+]
아주대학교[*] 계명문화대학교[+]
nayakms@rpa.re.kr[O], {khchoi, khchung }@ajou.ac.kr, sjkim@km-c.ac.kr

Fault test for UPnP protocol on implemented device

Minsik Kim[O*] , K.H. Choi[*], K. H. Chung[*], S. J. Kim[+]
Ajou university, Korea[*], Keimyung college, korea[+]

요 약

홈 네트워크에 참가하는 많은 장비들이 UPnP기술을 이용하여 네트워크에 연결하고 있다. 우리는 UPnP를 구현한 장비들을 선택하여 fault가 포함된 packet에 대하여 어떻게 동작하는지 테스트하였다. Fault는 발생시키는 개념에 따라서 syntactic fault와 semantic fault로 구분하여 생성하였다. 특히, syntactic fault는 11가지 type으로 구분하여 각각 fault packet을 생성하고, 이를 이용하여 장비를 테스트하였다. Fault type을 구분하여 테스트를 진행함으로써, 장비의 취약점을 체계적으로 찾을 수 있었다. 동시에, 현재 시판되고 있는 장비도 fault packet에 대해 많은 취약점이 존재한다는 것을 확인하였다.

1. 서 론

UPnP(Universal Plug and Play) 기술은 집이나 사무실 등에서 여러 전자부품을 네트워크에 쉽고 유연하게 연결하는 Architecture와 표준 프로토콜을 제공한다[1]. UPnP Architecture는 서비스를 제공하는 장비(Device)와 장비를 관리하기 위한 control point로 이루어져 있다. Control point는 장비 관리를 위하여 상태를 확인하거나 장비에 서비스를 요청할 때 UPnP 프로토콜을 준하는 메시지를 주고 받는다. 따라서, 장비가 UPnP 프로토콜 Specification을 준수하게 구현 되었는지의 여부는 매우 중요하다. 한편, UPnP 장비가 가정이나 사무실 등의 다양한 환경에서 네트워크 접근 및 장비를 관리하기 위한 목적으로 만들었기 때문에, 악의적인 접근이나 운용미숙으로 예기치 않은 조작이 일어날 위험에 많이 노출되어 있다. 이러한 이유로 인하여 장비는 fault가 포함된 입력으로 인하여 오작동이 일어나지 않도록 구현되어야 한다.

여러 업체들이 참가하여 UPnP에 대한 연구 및 보급에 힘쓰고 있는 UPnP Forum[2]에서는 UPnP 장비들에 대한 테스트를 위해 UIC(UPnP Implementers Corporation)[3]라는 내부적인 테스트 기구를 만들어서 운용하고 있다. UIC는 UPnP 장비에 대한 UPnP conformance 테스트를 진행하며, UIC 테스트를 통과한 제품만이 UPnP™ 마크를 사용할 수 있게 한다. 한편, UIC는 프로토콜상 유효하지 않은 값에 대해 테스트를 하지 않기 때문에 예기치 않은 입력에 대한 장비의 동작 여부를 보장할 수 없다. UIC의 테스트는 올바른 값에 대하여 올바르게 작동하는 것은 물론, fault가 포함된 입력에 대하여 타당하게 동작할 수 있어야 한다는 관점에서 약점을 지닌다. 이러한 fault 입력에 대한 테스트의 중요성을 [7, 8, 9, 10, 11]의 연구에서 지적하고 있다.

Fault 입력에 대한 테스트는 여러 연구에서 소개되어 있다. A. Vasan, and A. M. Memon은 [7]에서 HTTP와 SMTP를 구현한 소프트웨어에 대한 fault 입력 테스트에 대하여 연구하였다. [7]에서는 fault를 Syntax fault와 Semantic fault로 분류하고 각 fault를 포함하는 fault PDU(Protocol Data Unit)를 생성한 뒤, HTTP 혹은 SMTP를 구현한 소프트웨어에 fault PDU들을 이용하여 테스트하였다. Lijun Shan and Hong Zhu는 [12]의 연구에서는 Modeling에 mutation을 발생시키는 mutation operations을 24가지로 분류하였다. Mutation operations은 주입대상(Node or Edge)와 주입방법(Add, Delete,...)에 따라 분류되었다.

본 연구에서는 UPnP 프로토콜을 구현한 제품이 fault 입력에 대해 어떠한 동작을 하는지 테스트하였다. Fault 입력을 생성하기 위해서 본 연구는 [7]에서와 마찬가지로 fault를 syntactic fault와 semantic fault로 분류하였다. 이 중 syntactic fault는 [12]의 연구를 참고하여 총 11가지 type으로 분류하여 작성하였다. 이와 같이 분류된 fault들이 포함된 input을 가지고 장비를 테스트 하였다.

시중에서 판매되고 있는 장비들 중 UIC 테스트를 통과한 장비를 선택하여 테스트하였다. 테스트 결과, 장비는 fault가 포함된 입력에 대하여 취약점을 지니는 것을 확인 할 수 있었으며, UPnP 서비스가 마비되는 심각한 오류도 발견할 수 있었다. 이런 점은 UPnP 장비를 테스트함에 있어서 정상적인 입력에 대한 테스트뿐만 아니라 fault 입력을 이용한 테스트도 병행되어야 한다는 것을 시사한다.

다음 2장에서는 UPnP 프로토콜에 대하여 설명하고, 3장에서는 입력 값에 주입될 fault를 분류한 방법을 소개한다. 그리고 4장에서는 fault가 포함된 입력을 가지고 테스트한 결과를 소개한다. 마지막 장에서는 본 연구의 테스트 결과에 대하여 이야기 한다.

2. UPnP 프로토콜 설명

UPnP 프로토콜은 *Addressing, Discovery, Description, Control, Eventing*, 그리고 *Presentation* 등 6 단계로 이루어져 있다.

Addressing 단계는 장비나 control point가 네트워크에 참가 하기 위해 IP address를 할당 받는 단계이다. *Discovery* 단계는 장비나 control point가 네트워크에 참여 하였음을 알리는 단계이다. 장비가 제공하는 서비스 정보가 기술된 파일의 위치도 이 단계에서 네트워크에 알리게 된다. *Description* 단계는 control point가 장비로부터 장비나 장비가 제공하는 서비스의 상세정보들을 가져오는 단계이다. *Control* 단계는 control point가 장비에게 어떠한 행동을 하도록 명령(*action*)하거나 장비의 상태 정보를 가져오라는 요청(*Query*)을 수행하는 단계이다. *Eventing* 단계는 control point가 장비에서 발생하는 이벤트에 대하여 전송 요청, 전송 요청 시간 갱신 및 전송 요청 중단 등의 명령을 수행하는 단계이다. 장비는 control point가 요청한 시간 동안 발생한 이벤트를 control point로 전송하는 작업도 이 단계에서 수행된다. *Presentation* 단계는 사용자가 장비를 조작하거나 사용자에게 장비의 상태를 보여주기 위한 User Interface를 제공한다.

UPnP 프로토콜은 기본적으로 HTTP 프로토콜을 이용하여 메시지를 전달한다. 추가로, SSDP(Simple Service Discovery Protocol)[4], SOAP(Simple Object Access Protocol)[5], 그리고 GENA(General Event Notification Architecture)[6] 프로토콜 등이 사용된다. 장비와 control point 사이에 주고 받는 메시지는 method, field들, 그리고 body로 구성되어 있다. Method는 메시지의 처음 부분에 위치하며 메시지가 어떠한 동작에 관계되는지를 기술한다. [그림1] 메시지의 "POST /uuid:0014-bf92-7b9d020099dc/WANIPConnection:1 HTTP/1.1"(1)이 method이다. 이 한 줄만 가지고도 [1]에 의해 이 메시지가 장비가 제공하는 서비스들 중 하나인 *WANIPConnection* 에 Action을 요청하는 것임을 알 수 있다. 두 번째 구성요소인 field들은 method 다음에 위치한다. 각 field는 field-name과 field-value로 구분되며, 처음 나타나는 ':'문자로 구분한다. [그림1]의 (2)에서 "SOAPAction" 혹은 "Content-Length" 등은 field-name이며, ':' 문자 다음에 나타나는 값이 field-value가 된다. Body는 field들 다음에 위치한다. Body는 데이터를 전송하거나, Action이나 query를 수행하는데 있어서 필요한 인자들을 전송할 때 사용된다. [그림1]의 body(3)에는 *AddPortMapping*이란 이름의 Action을 수행하는데 사용되는 인자들이 SOAP 메시지의 형태로 기술되어 있다.

```
POST /uuid:0014-bf92-7b9d020099dc/WANIPConnection:1 HTTP/1.1                    (1)
SOAPAction: "urn:schemas-upnp-org:service:WANIPConnection:1#AddPortMapping"     (2)
:                                                                               :
Content-Length: 702                                                             (2)
<?xml version="1.0" encoding="utf-8"?>                                          (3)
:                                                                               :
  < NewRemoteHost>192.168.1.11< /NewRemoteHost>
:
</s:Envelope>
```

그림 1. UPnP 프로토콜 메시지의 예

3. Fault 분류

장비에 전달되는 메시지에는 다양한 형태의 fault가 주입될 수 있다. 본 논문에서는 생성할 수 있는 fault를 syntactic fault와 semantic fault로 구분하였다. Syntactic fault는 메시지의 구조나 기술 방법, 값의 형태나 범위상에 fault가 발생된 것을 가리킨다. 즉 specification에서 허용되지 않는 값의 사용으로 기인한 fault를 생성한다. 반면

에 semantic fault는 메시지의 형태나 값은 specification에 준하지만, 장비의 현재 상태 상 처리되는 것이 유효하지 않은 메시지를 전송하여 발생된 fault를 가리킨다. 즉, semantic fault는 syntactic fault와는 달리 메시지상에 fault를 포함하는 것이 아니라, UPnP 프로토콜의 flow 관점에서 옳지 않은 순서대로 메시지들을 전송시켜 fault를 발생시키게 된다.

3.1 Syntactic Fault

UPnP 프로토콜에서 전송되는 각 메시지는 일정한 규칙과 구조를 지니고 있으며, 이는 specification에 명시되어 있다. 따라서 specification상 규칙과 구조를 벗어나는 다양한 syntactic fault를 생성할 수 있다. 단, 메시지의 Method가 변경되면 이 메시지 본래의 목적이 변경되어 버리므로, method에는 fault를 발생시키지 않았다. 반면, field들과 body 부분에는 자유롭게 syntactic fault를 발생시켰다. 예를 들어, [그림1]의 method 부분(1)에 fault가 발생하여 어느 부분이라도 변경이 생긴다면, 이 메시지는 더 이상 *WANIPConnection* 서비스에 Action을 수행하는 메시지가 아니게 된다. 그러므로, method부분에는 fault가 발생되면 안 된다. 반면, field들이나 body에는 구조나 값을 변경하여 syntactic fault를 발생시켰다. 이 두 부분들에 fault가 발생되어 생성된 메시지를 syntactic fault가 포함된 fault packet이라 한다.

Syntactic fault가 포함된 fault packet을 체계적으로 생성하기 위해, [12]의 연구에서 mutation operations을 분류된 방법을 참고하였다. fault의 유형들을 fault가 적용될 대상(TARGET)과 fault를 생성하는 방법(OPERATION)을 기준으로 분류하였다. TARGET은 field들을 구분했던 field-name, field-value와 field 자체로 구분 하였다. 반면, OPERATION은 fault를 발생시키는 간단한 7가지 방법으로 구분하였다. OPERATION은 "*INSERT*", "*APPEND*", "*MODIFY*", "*DELETE*", "*BALNK*", "*DUPLICATE*", 그리고 "*REMOVE*"가 있다. TARGET과 OPERATION의 조합에 따라 총 11가지 fault type으로 분류하였으며, 각 fault type은 [표1]과 같다. [표1]은 "Callback: <http://192.168.1.3:1650>" field와 body의 한 부분인 "<NewProtocol>TCP</NewProtocol>"에 각 syntactic fault type을 주입한 예를 보여준다. F_1~F_5 fault type은 field-value에 fault를 발생시켜 생성된 syntactic fault들이다. 반면, F_6~F_9 fault type은 field-name에 fault가 주입되어 생성된 syntactic fault 들이다. F_{10}, F_{11} fault type은 field 자체에 fault를 발생시켜 생성한 syntactic fault들이다.

표 1. Syntactic fault types

TARGET	OPERATION	Fault ID	Comment	Examples
Field-value	INSERT	F_1	Field-value 에 Fault 문자열이 삽입됨	Callback: <htflgtp://192.168.1.3:1650> <NewProtocol>TCflgP</NewProtocol>
	APPEND	F_2	Field-value 에 Fault 문자열을 이어 붙임	Callback: <http://192.168.1.3:1650> flg <NewProtocol>TCP flg</NewProtocol>
	MODIFY	F_3	Field-value 의 값을 일부 변경	Callback: <http://192.1flel.3:1650> <NewProtocol>Tfg</NewProtocol>
	DELETE	F_4	Field-value 의 값을 일부 삭제	Callback: <http://10> <NewProtocol>P</NewProtocol>
	BLANK	F_5	Field-value 의 값을 모두 삭제	Callback: <NewProtocol> </NewProtocol>
Field-name	INSERT	F_6	Field-name 에 Fault 문자열을 삽입	Cafiglback: <http://192.168.1.3:1650> <NeflgwProtocol>TCP</NeflgwProtocol>
	APPEND	F_7	Field-name 에 Fault 문자열을 이어 붙임	Callback flj: <http://192.168.1.3:1650> <NewProtocol flj>TCP</NewProtocol flj>
	MODIFY	F_8	Field-name 의 값을 일부 변경	Cafigck: <http://192.168.1.3:1650> <Neflgotocol>TCP</Neflgotocol>
	DELETE	F_9	Field-name 의 값을 일부 삭제	Caback: <http://192.168.1.3:1650> <Neol>TCP</Neol>
Field	DUPLICATE	F_{10}	Field 중복	Callback: <http://192.168.1.3:1650> Callback: <http://192.168.1.3:1650> <NewProtocol>TCP</NewProtocol> <NewProtocol>TCP</NewProtocol>
	REMOVE	F_{11}	Field 제거	(Field 자체를 지움)

3.2 Semantic Fault

Semantic fault를 생성하기 위해서는 프로토콜의 State에 대한 이해가 선행되어야 한다. UPnP 프로토콜에서는 *Eventing* 단계를 제외한 다른 단계에서는 state를 유지하지 않는다. *Eventing* 단계 중에서도 3가지 주요 메시지와 1가지 이벤트에 의해서 state가 관리 된다. Control point가 장비에서 발생되는 이벤트를 전송 받기를 요구하는 subscribe 메시지, Control point가 장비로부터 발생된 이벤트를 전송 받는 시간의 갱신을 요청하는 renew 메시지, 그리고 control point가 장비로부터 발생된 이벤트를 전송 받는 것을 중단할 것을 요구하는 unsubscribe 메시지와 장비에서 Control point가 요청한 시간이 경과하는 Time-expire 이벤트가 그것이다. 이들에 의한 state의 변화를 [그림2]에서 state diagram으로 표현 하였다. Subscribe, Renew, Unsubscribe 메시지들과 Time-expire 이벤트를 제외한 메시지들이나 이벤트들은 어느 상태에서도 전송 및 발생될 수 있으며, 장비상 State변화에 영향을 끼치지

않는다. 그러므로 [그림2]에 표시하지 않았다. Time-expire 이벤트는 Subscribe 메시지를 받은 이후, control point가 요청한 시간이 흐르면 장비에서 발생되는 보이지 않는 이벤트(아무런 메시지를 발생시키지 않는다)이다. 그러므로 S_1 state에서만 발생할 수 있으며, S_0 state에서는 발생될 수 없다.

[그림2]의 'S_0', 'S_1'는 장비의 State를 나타내며, 화살표는 장비가 control point에서 전송한 메시지를 수신하였을 때의 상태변화를 가리킨다. 예를 들면, 장비가 S_0 상태에서 subscribe 메시지를 수신하면 S_1상태로 변경되며, S_1 상태에서 State변화에 영향을 줄 수 있는 사건은 renew, unsubscribe 메시지를 수신하거나, Time-expire 이벤트가 발생하는 것이다. S_1 State에서 subscribe 메시지를 수신하는 부분이 점선으로 표시되어 있다. 이는 S_1 state에서 subscribe 메시지를 수신하면, 장비에서 발생되는 이벤트 처리를 위한 또 다른 session이 생성된다는 것을 가리킨다. 새로운 session이 생성되었다는 것은 [그림 2]와 동일한 state diagram이 중복으로 생성된다는 것을 의미한다. 새로운 session은 이전에 존재하는 state와는 개별적으로 동작하기 때문에, 중복되는 session을 생성시키는 subscribe 메시지는 semantic fault를 생성하는 고려 대상에서 제외하였다.

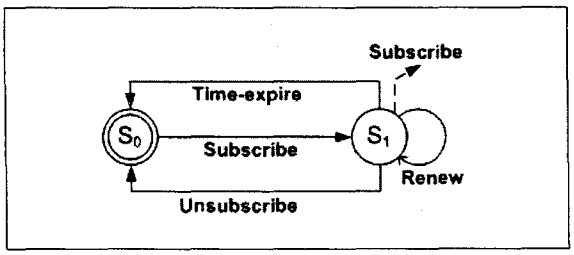

그림 2. Eventing step에서의 state diagram

[그림2] state diagram의 각 state에는 subscribe, renew, unsubscribe 메시지들이 화살표로 모두 표시되어 있지 않다. 세 메시지의 경우, 화살표로 표기 되지 않은 방향으로 메시지를 전송하는 경우가 프로토콜 flow상에서 위배되는 것이 된다. 즉, semantic fault를 발생시킨다. [그림2]를 바탕으로 다음과 같은 2가지 semantic fault를 포함하는 fault 메시지를 생성할 수 있다.

1) 장비가 'S_0' State에서 renew 메시지를 수신하였을 때

2) 장비가 'S_0' State에서 unsubscribe 메시지를 수신하였을 때

4. 테스트

4.1 테스트 환경

테스트를 수행하는 환경은 [그림3]와 같이 구성하였다. 장비들은 UIC 테스트를 통과한 제품으로 시중에서 구입할 수 있는 L사의 A Model이다.

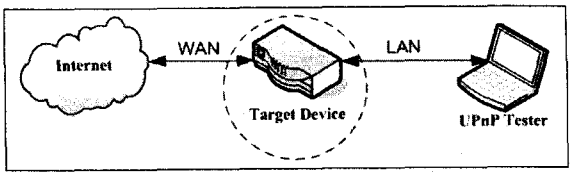

그림 3. 테스트 수행 환경

본 연구에서는 UPnP 의 총 6가지 단계 중 4개인 Discovery, Description, Control, Eventing 단계들에 대해서만 테스트를 수행하였으며, Addressing과 Presentation 단계는 테스트하지 않았다. Addressing 부분은 DHCP와 AutoIP등의 프로토콜을 사용하기 때문에, UPnP 테스트와는 직접적으로 연관을 가지지 않는다. 그러므로, 테스트는 Addressing 단계가 완료되어, 장비가 IP Address를 할당 받은 이후부터 수행된다. Presentation 단계는 User Interface에 대한 부분이므로 이 부분 역시 테스트하지 않았다. 각 단계별로 사용되는 메시지들을 작성하고, 장비가 specification에 따라서 동작을 하는지 확인 하였다. 또한 syntactic fault와 semantic fault를 포함하는 fault packet을 생성하고, 이것들을 이용하여 fault 테스트를 수행하였다. 테스트에 사용하기 위해 선택된 기본적인 메시지들은 [표2]와 같다.

표 2. UPnP Step별 테스트 항목 리스트

메시지 ID	UPnP Step	메시지	Comment
M_1	Discovery	M-SEARCH	네트워크에 참가했음을 알림
M_2	Description	Device Description	장비 정보 가져오기
M_3		Service Description	서비스 정보 가져오기
M_4	Eventing	Subscribe	이벤트 subscription 을 요청
M_5		Renew	이벤트 subscription 시간 갱신 요청
M_6		Unsubscribe	이벤트 subscription 종료 요청
M_7	Control	Control	서비스가 제공하는 Action 수행

4.2 테스트 packet 생성

기본적인 메시지들을 "M_1 M_2 M_3 M_4 M_5 M_6 M_7"과 같은 순서로 전송하도록 하였다.

Syntactic fault를 포함하는 fault packet은 [표2]의 모든 메시지들의 각 field들마다 fault를 발생시켰다. 모든 field들에 11개의 type으로 분류된 syntactic fault를 발생시켜 syntactic fault를 포함하는 fault packet을 생성하였다. 테스트에서 fault를 포함한 packet에 의한 장비의 오작동 여부를 판단하기 위해, fault packet을 전송 이후에 전혀 fault가 포함되지 않은 메시지를 이어서 전송하였다. 만약 장비상에 문제가 발생하였다면, 장비는 뒤이어 전송되는 메시지들을 specification에 준하게 처리할 수 없을 것이다. 이를 통해서 테스트 장비가 fault packet에 의해 문제가 발생하는지 판단하게 된다.

Semantic fault가 발생할 수 있는 상황을 앞 3절의 마지막에서 기술하였다. *'1)'*과 *'2)'*의 2가지 경우를 각각 "M_4 M_6 M_5"과 "M_4 M_6 M_6"의 순서로 메시지를 전송하게 함으로써, semantic fault를 발생시킬 수 있다.

두 테스트에서 앞에 "M_4 M_6"을 넣어주는 이유는 state를 S_0로 만들기 위함과 Eventing step에서 메시지를 전송하는데 사용되는 서비스의 고유 ID값인 SID값을 얻기 위해서다.

4.3 정상 Packet에 대한 반응

메시지들을 "M_1 M_2 M_3 M_4 M_5 M_6 M_7"와 같은 순서대로 전송한 결과 장비는 UPnP specification에 준하는 응답을 전송하였다. 이는 Conformance 테스트인 UIC 테스트를 통과한 장비들인 만큼, 예상된 결과였다.

4.4 Syntactic fault를 포함한 fault packet에 대한 반응

메시지의 각 field마다 syntactic fault를 발생시킨 fault packet들로 테스트를 수행하였을 때, 장비의 오작동 여부를 테스트하였다. [표3]과 [표4]는 각 장비에 대하여 syntactic fault를 포함하는 fault packet으로 장비를 테스트한 결과이다. [표3]과 [표4]는 메시지의 ID별, 각 field별로 지정된 type의 fault를 발생시켜 테스트한 결과를 정리 한 것이다. 'Warring'이라 표시된 부분은 비록 specification 상에는 어긋나지만 장비가 지속적으로 정상적인 서비스를 제공할 수 있었다는 것을 가리킨다. 반면, 'Failure'는 테스트 장비가 더 이상 UPnP 서비스를 제공할 수 없는 심각한 오류가 발생하였음을 의미한다. 'Warring'과 'Failure'란을 채우고 있는 기호는 기입된 field에 발생시킨 fault type의 ID이다.

테스트 결과, fault packet을 사용하여 테스트를 하였음에도 장비가 이를 인지하고 에러 메시지를 전송하거나 뒤이어 전송되는 메시지를 specification에 준하게 처리하는 등의 동작을 보이는 경우도 있었다. 이와 같은 경우 장비가 fault 입력에 대하여 타당하게 처리한 것으로 판단하고, [표3]에서 생략하였다.

표 3. Fault PDU에 대한 테스트 장비의 결과

메시지 ID	Field	Warring	Failure
M_1	ST	F_{10}	
	MX	F_{10}	
	MAN	F_{10}	
M_4	NT	F_1, F_2, F_3, F_4, F_5, F_{10}	
	Callback	F_1, F_2, F_3, F_4, F_{10}	F_3, F_4
	Timeout	F_1, F_2, F_3, F_4, F_5	
M_5	TIMEOUT	F_1, F_2, F_3, F_4, F_5, F_{11}	
M_7	SOAPAction	F_1, F_{10}	
	CONTENT-TYPE	F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9, F_{10}, F_{11}	
	Content-Length	F_2, F_{10}	
	SOAP.NewRemoteHost	F_1, F_2, F_3, F_4, F_5, F_7, F_{10}	
	SOAP.NewExternalPort	F_7, F_{10}	
	SOAP.NewProtocol	F_1, F_2, F_3, F_4, F_5, F_7, F_{10}	
	SOAP.NewInternalPort	F_7, F_{10}	
	SOAP.NewInternalClient	F_1, F_2, F_3, F_4, F_5, F_7, F_{10}	
	SOAP.NewEnabled	F_7, F_{10}	
	SOAP.NewPortMappingDescription	F_1, F_2, F_3, F_4, F_5, F_7, F_{10}	
	SOAP.NewLeaseDuration	F_7, F_{10}	

L사에서 구현한 A 장비에 대하여 Syntactic fault 테스트를 수행한 결과이다. 테스트에 사용된 fault packet에 대하여 테스트 장비는 타당하게 처리하지 못하는 상황이 존재하였다. 가령 예를 들어, M_4 의 'NT' field의 경우 specification상에서는 일정한 문자열을 가져야 한다고 지정하고 있다. 그러나 F_1, F_2, F_3, F_4, F_5와 같이 문자열을 변경하는 fault packet에 대하여, 테스트 장비는 fault를 인지 하지 못하고 fault가 없는 메시지처럼 처리하였다. 그러나, 정말 심각한 문제는 M_4의 'Callback' field에 F_3, F_4 fault type의 fault을 발생시킨 fault

packet을 가지고 테스트하는 경우에 발생하였다. 이 fault packet을 이용하여 장비를 테스트한 경우, 더 이상 UPnP 서비스를 제공하지 못하는 심각한 문제점을 야기 시켰다.

4.5 Semantic fault를 포함한 fault packet에 대한 반응

Semantic fault를 위하여 4.2절에서 언급한 2가지 순서로 메시지들을 전송하고, 장비의 응답을 확인 하였다. [표4] 에서 밑줄로 표시된 부분이 Semantic fault를 포함하는 fault packet들이다. 이들을 장비에 전송하여 테스트 하였다. 테스트 결과, 테스트 장비는 semantic fault를 인지하고 에러 메시지로 반응한다는 것을 확인할 수 있었다.

표 4. Semantic Fault 테스트 결과 [Model A]

메시지 순서	오류 메시지
M_4	
M_6	
$\underline{M_5}$	HTTP/1.1 412 Precondition Failed
M_4	
M_6	
$\underline{M_6}$	HTTP/1.1 412 Precondition Failed

5. 결 론

본 논문에서는 syntactic fault는 물론 semantic fault를 발생시켜 생성한 fault packet을 가지고 UPnP 장비를 테스트 하였다. UIC인증을 취득한 장비를 대상으로 테스트를 수행한 결과, specification 에 일치하는 입력에 대하여는 정확한 반응을 보인다는 것을 확인하였다. 그러나 syntactic fault 가 주입된 일부 입력에 대하여는 오작동을 한다는 것을 확인하였다. 테스트 장비로 전송하는 메시지 중 'Callback' field에 fault를 발생시켜 생성한 fault packet을 이용하여 테스트한 이후, 장비는 UPnP 기능이 마비되는 것을 확인하였다.

Syntactic fault를 테스트함에 있어서 fault type을 구분하여 fault를 발생시키는 방법을 사용함으로써, 장비와 주고 받는 메시지 중 어떤 메시지가 취약점을 지니고, 어떤 field들이 취약점을 지니며, 어떤 형태의 fault에 취약점을 가지는 체계적으로 파악 할 수 있었다.

Semantic fault를 테스트함에 있어서 테스트 대상 장비 모두 발생된 semantic fault를 인지하고, 타당하게 처리하는 것을 확인 할 수 있었다.

참 조

[1] UPnP™ Device Architecture Version 1.0, http://www.upnp.org.

[2] UPnP™ Forum, http://upnp.org/

[3] UPnP Implementers Corporation, http://upnp-ic.org/

[4] Simple Service Discovery Protocol/1.0, http://quimby.gnus.org/internet-drafts/draft-cai-ssdp-v1-03.txt

[5] Simple Object Access Protocol Specification, http://www.w3.org/TR/soap/

[6] General Event Notification Architecture Base: Client to Arbiter, http://www.upnp.org/download/draft-cohen-gena-client-01.txt

[7] A. Vasan, and A. M. Memon., "ASPIRE: Atutomated Systematic Protocol Implementation Robustness Evaluation"., In Proceedings of Software Engineering Conference.

[8] Clark, J.A.; Pradhan, D.K., "Fault injection: a method for validating computer-system dependability"., Computer, Volume 28, Issue 6, June 1995.

[9] Carreira, J.V.; Costa, D.; Silva, J.G., "Fault injection spot-checks computer system dependability"., Spectrum, IEEE, Volume 36, Issue 8, Aug. 1999.

[10] Benso, A.; Rebaudengo, M.; Impagliazzo, L.; Marmo, P., "Fault-list collapsing for fault-injection experiments"., Reliability and Maintainability Symposium, 1998. Proceedings., Annual19-22 Jan. 1998.

[11] National Institute of Standards & Technology, "The Economic Impacts of Inadequate Infrastructure for Software Testing", Planning Report 02-3, May 2002.

[12] Lijun Shan, Hong Zhu., "Test ase generation: Testing software modelling tools using data mutation"., Proceedings of the 2006 international workshop on Automation of software test AST '06, May 2006