RFID 시스템의 미들웨어를 위한 접근제어[1)]

김태성 김호원
한국전자통신연구원
{taesung, khw}@etri.re.kr

Access Control Policy for Middleware in RFID Systems

Kim, Taesung Kim, Howon
Electronics and Telecommunications Research Institute(ETRI)

요약

RFID의 무선을 통해 자동 인식 기술은 산업, 상업 그리고 의료 분야에 혁신적인 개선을 가져다 줄 것으로 기대되고 있다. RFID 기술의 핵심은 물리적인 물체에 관련한 정보를 쉽게 수집할 수 있다는 것이다. 미들웨어는 RFID 리더와 응용 사이에 위치하면서 RFID 정보를 실시간으로 수집하고 기업응용에 전달해주는 중요한 역할을 담당한다. 따라서 네트워크로부터의 공격이나 잘못된 명령으로부터 미들웨어를 보호하기 위해 적절한 접근제어가 필요하다.

본 논문은 RFID 미들웨어에 적용가능한 접근제어 정책을 제안한다. 이 정책은 제공가능한 태그 범위, 접근가능한 리더, 명령, 보고가 배달되어질 주소등을 제한할 수 있도록 고안되었다. 또한, 접근제어 정책을 수행하는 프로세스의 구현 알고리즘을 제시한다.

1. 서론

RFID 시스템은 자동적으로 물체를 인식하는 특성으로 인해 많은 분야에 적용되어 생산성을 높일 수 있을 것으로 기대되는 새로운 기술이다. RFID 시스템에서 미들웨어는 리더와 기업응용의 사이에 위치하면서 리더로부터 받은 태그 데이터를 수집하고, 이를 정제, 분류, 카운팅등의 처리 후, 응용에게 전달 하는 역할을 수행한다.

미들웨어가 처리하는 태그 데이터는 개인의 프라이버시에 관련되거나, 기업의 중요한 가치를 가진 정보 일수 있다. 따라서, 미들웨어가 제공하는 태그 데이터의 접근은 인가된 사용자만이 접근할 수 있도록 제어되어야 한다. 미들웨어는 리더로부터의 태그 정보를 기업응용에게 실시간으로 처리, 전달하기 때문에 미들웨어의 가용성은 RFID 네트워크의 성능에 많은 영향을 미친다. 그러므로, 요청할 수 있는 명령, 접근할 수 있는 태그 데이터의 범위 등을 제한하여 미들웨어의 성능을 저하시킬 수 있는 요소를 사전에 차단하여야 한다.

본 논문은 RFID 시스템의 미들웨어에 적용 가능한 접근제어 정책을 제안하였다. 이 접근제어 정책은 제공 받을 수 있는 태그 데이터, 접근할 수 있는 리더, 호출 할 수 있는 명령, 그리고, 보고를 받는 공지 주소에 대한 제어를 표현할 수 있다. 이러한 접근제어 매커니즘이 적용된 미들웨어는 불법적이거나 잘못된 요청으로부터 보호 받을 수 있고, 공격자가 클라이언트를 공격해 미들웨어에 요청을 보낸다 할지라도 해당클라이언트가 가진 권한만을 행사하게 되므로 피해를 최소화할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 RFID 기술에 대한 소개를 하고 3장에서는 RFID 표준인 EPCglobal의 네트워크 구조를 설명한다. 4장에서 미들웨어의 접근제어 정책을 설명하고 5장에서 이 접근제어 정책의 구현 방법에 관하여 논의하고 6장에서 결론을 맺는다.

2. RFID 기술

RFID는 전파를 이용해 무선으로 물체나 사람의 ID를 전송하는 시스템을 지칭하는 일반적인 용어이다. RFID 태그는 이 ID를 일련의 번호 형태로 저장하는 것으로서, 회로 기판위에 전파 안테나를 부착한 형태의 마이크로칩이다. 태그 안에 들어 있는 ID를 추출 하기 위해서는 RFID 리더가 필요하다. RFID 리더는 하나 이상의 안테나를 가진 장치로서 태그에게 전파를 발생해서 태그로부터 되돌아 오는 전파를 수신하여 ID를 추출해낸다.

태그가 리더 주변에 위치해 있을 때 자동적으로 태그의 ID를 추출하고, 이 정보는 실시간으로 컴퓨터 시스템으로 전송된다. RFID의 이러한 특성으로 인해 사람의 간섭이 필요 없고 에러율이 낮아지며 업무의 처리속도가 증가하여 전체적으로 비용 절감 효과가 극대화 된다. RFID 응용으로는 기업자산관리, 제조공정관리, 공급망관리, 소매점관리, 지불시스템, 보안접근통제 등 여러 분야에 적용 될 수 있다.

3 EPCglobal RFID 네트워크 구조

- *A) EPCglobal 네트워크 구조*

1) 본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술 개발사업의 일환으로 수행하였음. [2005-S-088-03, 안전한 RFID/USN을 위한 정보보호 기술]

EPCglobal 네트워크는 공급망 체인내의 물품을 자동적이고 즉각적으로 인식하고 관련 정보의 공유가 가능하도록 하기 위한 국제 기술 표준이다. 이러한 표준들은 UHF(Ultra High Frequency) 태그에 집중하고 물품의 고유 인식을 위한 시스템을 공급하고 이러한 데이터의 저장과 전송 방법에 관하여 정의한다. EPCglobal은 다섯 가지의 기초적 요소로 구성되어 있다. ID 시스템(태그 및 리더), EPC 코드, 미들웨어, 정보서버, ONS(Object Name Service)이다. 태그내에 들어가는 EPC 코드는 공급망 체인에서 개별 물품을 고유하게 인식하기 위해 설계된 숫자 배열이다. EPC는 리더로부터 읽혀져서 ONS에 의해 인터넷 주소로 변경된다. 이 인터넷 주소는 EPC와 관련한 물품의 정보를 저장하고 있는 시스템을 가리킨다. 교환되는 방대한 정보를 조종하기 위해 RFID 미들웨어는 네트워 트래픽을 감소시키고 소프트웨어 인터페이스를 제공하는 방법으로 이 데이터를 관리한다. ALE 서비스는 EPC 리더 혹은 리더 네트워크와 정보 시스템이 데이타를 교환 할 수 있도록 해준다. EPCIS는 사용자가 EPCglobal 네트워크를 통해 교역 파트너와 EPC 관련한 정보를 교환이 가능하도록 한다.

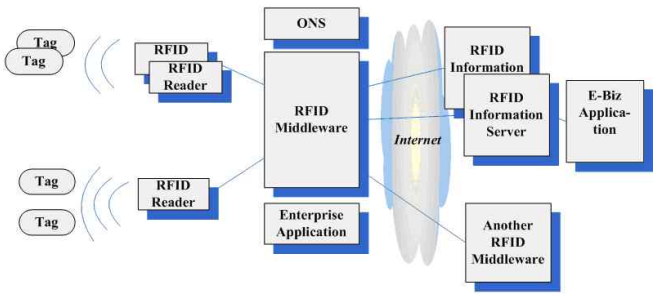

그림 1) EPCglobal 네트워크 구조

각 요소의 자세한 설명은 다음과 같다.

1) EPC: 공급망 체인에서 물품을 인식하기 위한 고유의 연속된 숫자이다. EPC는 물품이 공급망의 어디에 위치하든 물품의 정보에 관한 질의가 가능하도록 한다.

2) ID 시스템: ID 시스템은 태그와 리더로 구성된다. 태그는 회로 기판 위에 마이크로칩과 안테나로 구성된 RFID 장치이다. EPC 코드는 이 태그안에 저장되고, 태그는 상자, 팔레트, 또는 물품에 직접 부착된다. 태그는 무선 주파수 인식을 이용해 EPC 코드를 리더와 통신한다.

3) 미들웨어: 미들웨어는 실시간 리더 이벤트와 정보를 관리하고 경고를 제공하고 기업의 다른 시스템뿐만 아니라 EPC 정보 서비스에 기본적인 태그 정보의 통신을 관리한다.

4) EPCIS: EPC 정보 서비스는 EPCglobal 네트워크를 통해 교역 파트너와 EPC 관련 정보를 교환이 가능하게 한다.

5) 검색 서비스: 특정 EPC 코드에 관련한 데이타를 찾고 접근을 요청할 수 있도록 하는 서비스가 검색 서비스이다. ONS(Object Naming Service)는 검색 서비스 중의 일부분이다.

- *B. ALE(APPLICATION LEVEL EVENTS)*

EPC 네트워크 구조는 여러 소스로부터 여과되고 정제된 EPC 데이타를 클라이언트에게 제공하는 인터페이스를 정의하고 있다. 이 인터페이스와 기능은 ALE(Application Level Events)로 불린다. ALE는 클라이언트가 어떠한 EPC 데이타를 원하는지를 상위 레벨, 서술적인 방법으로 지칭할 수 있는 수단을 제공한다. ALE는 EPC 데이타의 출처 또는 처리 방법에 독립적으로 축적되고 여과된 EPC 데이타를 보고하는 표준화된 형식을 제공한다. ALE는 EPC 데이타의 출처(예를 들어 리더, 바코드 스캐너)를 "location"이라는 상위수준의 개념으로 추상화한다. 이에 따라 특정 위치로부터 EPC 데이타를 수집하기 위해 사용된 물리적 리더의 자세한 정보를 클라이언트에게 감출수 있다. 읽기 사이클은 ALE 계층이 리더와 통신하는 최소의 단위이다. 이벤트 사이클은 하나 이상의 리더로 부터 그리고 하나 이상의 읽기 사이클로 구성된다. 이벤트 사이클은 클라이언트 관점에서 최소 단위로 인식된다. ALE 계층의 처리는 독립적인 소프트웨어인 미들웨어, 기능을 갖춘 리더, 혹은 이 두 조합에서 수행된다. ALE 계층은 SOAP 프로토콜을 통해 기능이 제공된다. ECSpec은 이벤트 사이클이 어떻게 계산되어지는지를 정의한 복잡한 형식이다. 이벤트 사이클이 시작되도록 하는 방법은 두가지이다. 독립적인 ECSpec이 define 함수에 의해 만들어질 수 있다. 이어서, 하나 이상의 클라이언트가 subscribe 함수를 이용해 ECSpec에 가입할 수 있다. ECSpec은 최소 하나의 가입자가 있을 때까지 이벤트 사이클을 작동시킨다. Poll 함수는 ECSpec에 가입한 후 하나의 이벤트 사이클이 만들어진 이후 즉각적으로 가입해지하는것과 같다. ECSpec를 만드는 두번째 방법은 immediate 함수를 호출하여 직접 실행 시키는 것이다. 이것은 ECSpec을 define하고 하나의 poll을 수행한 후 바로 undefine하는 것과 같다.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <aac:ALEAccessControl xmlns:aac="urn:etri:smw:aac:xsd:1">
  - <filter permitAllFilter="false">
    - <includePatterns>
        <includePattern>urn:epc:pat:gid-96:20.[0-20].*</includePattern>
      </includePatterns>
    </filter>
  - <permittedReaders permitALL="false">
      <logicalReader>FirstGate</logicalReader>
      <logicalReader>SecondGate</logicalReader>
    </permittedReaders>
  - <notificationURIs permitAnyURIs="false">
      <notificationURI>http://*.foo.com/AleService/*</notificationURI>
      <notificationURI>tcp://129.254.*.*:1234</notificationURI>
    </notificationURIs>
  - <permittedAPIs mayDefine="true" mayImmediate="false"
      mayGetECSpecNames="true">
      <mayUndefine permitAll="true" />
    - <maySubscribe permitAll="false" permitSelf="true">
        <entities>md.foo.com</entities>
        <entities>boo.foo.com</entities>
      </maySubscribe>
    </permittedAPIs>
</aac:ALEAccessControl>
```

그림 2) 접근제어 정책의 예

4. 접근제어 정책

본 논문에서 제안된 접근제어 정책은 XML 스키마로 정의하였다. 그림 2는 XML 문서로 표현한 접근제어 정책의 예제이다. 이 정책은 filter, permittedReaders, notificationURIs, permittedAPIs의 엘리먼트로 구성된다. 다음은 각 엘리먼트의 스키마와 자세한 설명이다.

A) filter

이 엘리먼트는 요청자가 접근할 수 있는 태그 데이터의 범위를 지정한다. permitAllFilter 속성은 true일 경우 모든 태그 데이터접근이 가능하다. 태그 데이터 범위의 표현은 TDS의 패턴 URI를 따른다[2]. *는 해당 필드의 모든 값이 허용되고, [lo-hi] 일 경우는 lo부터 hi까지의 범위의 값이 허용됨을 의미한다. 이 엘리먼트의 접근제어 판단은 두가지 형태로 구현될 수 있다. 클라이언트에게 태그 정보를 전달하기 전에 이 범위에 들어가지 않은 태그가 있을 경우 이를 제거한 후 전송하는 방법과, 클라이언트가 define 또는 immediate를 이용해 태그정보를 요청할 때 이벤트사이클내의 FilterSpec과 비교하는 방법이다. 전자는 구현이 쉬운 반면에 매 보고시마다 모든 태그 데이터를 검사해야 하므

로 미들웨어의 성능에 영향을 미칠 가능성이 있다.

```
- <xs:complexType name="ALEFilterSpec">
  - <xs:sequence>
    - <xs:element name="includePatterns" minOccurs="0">
      - <xs:complexType>
        - <xs:sequence>
            <xs:element name="includePattern" type="xs:string"
              minOccurs="0" maxOccurs="unbounded" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="permitAllFilter" type="xs:boolean"
      use="optional" />
  </xs:complexType>
```

그림 3) 필터 엘리먼트의 스키마

B) permittedReaders

이 엘리먼트는 요청자가 접근할 수 있는 리더를 지정한다. permitALL 속성이 true일 경우 모든 리더에 접근이 가능하다. 리더는 리스트로 표현한다.

```
- <xs:complexType name="ALELogicalReaders">
  - <xs:sequence>
      <xs:element name="logicalReader" type="xs:string" minOccurs="0"
        maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="permitALL" type="xs:boolean" use="optional" />
  </xs:complexType>
```

그림 4) permittedReader 엘리먼트의 스키마

C) notificationURIs

클라이언트는 subscribe 명령을 통해 태그 데이터 보고를 주기적으로 받는 주소를 미들웨어에게 알려준다. 이 주소를 공지주소라고 한다. 공지주소는 http와 tcp 중 하나의 형태로 표현한다. 이 엘리먼트는 이 공지주소의 범위를 지정한다. permitAnyURIs 속성이 true일 경우 모든 공지주소가 허용된다. 그림 1의 예제처럼 *는 해당 필드에 모든 값이 허용됨을 의미한다. 예를 들어, http://*.foo.com/AleService/*는 foo.com 도메인의 모든 호스트의 AleService라는 웹 어플리케이션에게 http 방식으로 태그 데이터 보고를 허용함을 의미한다. 또, tcp://129.254.254.*:1234는 C 클래스에 해당하는 호스트 중에서 1234 포트로 tcp 방식으로 태그 데이터 보고를 허용함을 의미한다.

```
- <xs:complexType name="NotificationURIs">
  - <xs:sequence>
      <xs:element name="notificationURI" type="xs:string" minOccurs="0"
        maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="permitAnyURIs" type="xs:boolean"
      use="optional" />
  </xs:complexType>
```

그림 5) notificationURI 엘리먼트의 스키마

D) permittedAPIs

클라이언트가 요청할 수 있는 명령을 제한하는 엘리먼트이다. mayDefine, mayImmediate, mayGetECSpecNames는 각각 해당 명령의 호출 가능 여부를 결정한다. 자식 엘리먼트인 mayUndefine은 undefined 명령의 호출 여부를 결정하고, maySubscribe는 subscribe, unsubscribe, poll 의 호출 여부를 결정한다. mayUndefine과 maySubscribe는 EntityList의 type이다. permitAll은 항상 해당 명령을 호출 할수 있다는 의미이며, permitSelf는 자신이 정의한 이벤트사이클에 해당 명령을 호출 할 수 있음을 의미한다. entities는 클라이언트의 아이디를 리스트 하는데 이 리스트에 속한 클라이언트가 정의한 이벤트사이클에 해당 명령을 호출 할 수 있다.

```
- <xs:complexType name="ALEAPIs">
  - <xs:sequence>
      <xs:element name="mayUndefine" type="aac:EntityList"
        minOccurs="0" />
      <xs:element name="maySubscribe" type="aac:EntityList"
        minOccurs="0" />
    </xs:sequence>
    <xs:attribute name="mayDefine" type="xs:boolean" use="optional" />
    <xs:attribute name="mayImmediate" type="xs:boolean"
      use="optional" />
    <xs:attribute name="mayGetECSpecNames" type="xs:boolean"
      use="optional" />
  </xs:complexType>
- <xs:complexType name="EntityList">
  - <xs:sequence>
      <xs:element name="entities" type="xs:string" minOccurs="0"
        maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="permitAll" type="xs:boolean" use="optional" />
    <xs:attribute name="permitSelf" type="xs:boolean" use="optional" />
  </xs:complexType>
```

그림 6) ALE API 엘리먼트의 스키마

5. 접근제어의 구현

이번 장에서는 접근제어 정책의 구현에 관하여 기술한다. permittedReader와 permittedAPIs정책을 판단하는 엔진의 구현은 간단하다. 엔진은 요청하는 클라이언트가 정책과 비교해 해당 권한을 가지고 있는지만 판단하면 된다. notificationURI의 경우에는 클라이언트가 요청하는 주소가 정책에 기술된 리스트 중에 속하는지 살펴보면 된다. 반면에 filter 정책은 이보다 복잡한 경우를 점검하여야 한다. 클라이언트는 ECSpec에 includePattern과 excludePattern을 기술할 것이다. 미들웨어는 excludePattern에 속하지 않고, includePattern에 속하는 태그만을 보고하도록 되어 있다. 따라서, 접근제어 엔진은 includePattern에서 excludePattern을 뺀 태그 값들이 filter 정책에 포함되는지를 확인하면 된다. includePattern을 I, excludePattern을 e, filter 정책을 I라고 하자. 접근제어 엔진의 판단을 다시 표현하면

I - e ⊂ I 이면 요청은 허락된다. 그러나, 실제에서는 태그 집합의 여집합이나 차집합은 전체 태그 집합을 계산 할 수 없기 때문에 구현할 수 없다. 따라서, 차집합을 가지지 않는 표현으로 변환이 필요하다. i ⊂I ∪e는 같은 의미를 지니는 표현이다. 다음은 위 표현식 변환의 증명이다.

Proposition: $i \subset I \cup e$ if and only if $i - e \subset I$

Proof: If $i \subset I \cup e$, then $i \cup e \subset I \cup e$ and so $(i \cup e) \cap e^c \subset (I \cup e) \cap e^c$. Therefore, $i - e \subset I \cap e^c \subset I$.

Suppose $i - e \subset I$, Then $(i \cap e^c) \cup e \subset I \cup e$. Therefore $i \subset i \cup e \subset I \cup e$

패턴 리스트에 속하는지 비교하기 위해 패턴은 비교 패턴에 맞게 잘게 쪼개져야 한다. 이러한 구현을 위한 의사 코드는 그림 7에 있다.

6 결론

RFID 시스템에서 미들웨어는 리더와 응용 사이에 위치하면서 실시간으로 태그 이벤트를 전달하는 중요한 역할을 수행한다. 불법적인 태그데이터의 유출 방지, 미들웨어 시스템의 가용성 보장 등을 위해 클라이언트의요청은 정책에 근거하여 제어될 필요가 있다.

본 논문은 RFID 미들웨어의 접근제어 정책을 제안하였다. 이 정책은 클라이언트가 제공 받을 수 있는 태그 데이터의 정보, 접근할 수 있는 리더, 호출할 수 있는 함수 그리고, 보고를 받는 공지주소에 대한 접근제어를 나타낼 수 있다.

```
i = List of includePattern in ECSpec
e = List of excludePattern in ECSpec
I = List of access control policy

IF e is not null THEN IUe = Union of I and e
ELSE IUe = I END IF

spliting = i
splited = null

WHILE spliting is not empty
   FOR each of spliting and I∪e
      CALL SPLIT with one pattern in spliting and one pattern in
IUe
      IF return of SPLIT is not null
         replace the pattern with return patterns in spliting
      ELSE
       add the pattern to splited
      ENDIF
   ENDFOR
ENDWHILE

IF every pattern of splited match at leaset one of pattern of I∪e
  the client is allowed
ELSE
  the client is not allowed
ENDIF

Procedure SPLIT(member, set) : return splited_patterns{
  IF member and set is range // In this case is [lo~hi]
    mlo = lo of member
    mhi = hi of member
    slo = lo of set
    shi = hi of set

    IF two pattern is disjoint  RETURN null  ENDIF

    IF mhi belong to [slo~shi]
      split member to [mlo~(slo-1)], [slo~mhi]
    ENDIF
    IF mlo belong to [slo~shi]
      split member to [mlo~shi], [(shi+1)~mhi]
    ENDIF
  ENDIF
    RETURN splited list of member pattern
  }
```

그림 7) 패턴 Split 의사 코드

[참고문헌]

[1] RFID journal Web site, www.rfidjournal.com

[2] EPCGlobal Web site, www.epcglobalinc.org

[3] EPCGlobal Inc : EPCGlobal Object Name Service (ONS)1.0, Technical report,April (2004)

[4] EPCGlobal Inc : The Application Level Events(ALE) Specification, Version 1.0,September (2005)

[5] EPCGlobal Inc, EPC Tag Data Standard Version 1.1, May (2005)

[6] EPCGlobal Inc: Filtering and Collection Threat Analysis. Technical report, July (2004)