JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 23, No. 9: 1160~1172, Sep. 2019

국내 자바 웹 응용을 위한 SAML 소프트웨어의 개발

조진용[1*] · 채영훈[2] · 공정욱[3]

Development of SAML Software for JAVA Web Applications in Korea

Jinyong Jo[1*] · Yeonghun Chae[2] · JongUk Kong[3]

[1*]Senior Researcher, Advanced KREONET Center, KISTI, Daejeon, 34141, Korea
[2]Researcher, Advanced KREONET Center, KISTI, Daejeon, 34141, Korea
[3]Principal Researcher, Advanced KREONET Center, KISTI, Daejeon, 34141, Korea

요 약

연합인증은 다수의 보안도메인 간에 적용되는 사용자 인증 및 인가체계이다. 연구 및 교육 분야에서 활용되고 있는 다수의 국외 웹 응용서비스들은 표준화된 사용자 인증방식으로 SAML(Security Assertion Markup Language) 기반의 연합인증을 채택하고 있다. 하지만 국내는 공개 SAML 소프트웨어를 이용하기 힘든 특정 웹 서버나 웹 응용 서버의 시장 점유율이 높고 전자정부 표준프레임워크 기반의 Java 웹 응용이 많기 때문에 연합인증 기술을 적용하기 어려운 상황이다. 본 논문은 Java 기반의 웹 응용개발 환경에서 연합인증 기술을 쉽고 안전하게 활용케 할 목적으로 개발된 SAML4J 소프트웨어를 소개한다. SAML4J는 개발 프레임워크에 독립적인 세션 저장소를 지원하고 API를 통해 Web SSO 플로우를 처리케 함으로써 개발자 친화적인 장점이 있다. 네트워킹 테스트베드를 구성하고 개발한 소프트웨어의 기능과 성능, 확장성 및 보안성에 대해서 검증함으로써 SAML4J의 높은 활용가능성을 확인한다.

ABSTRACT

Federated authentication is a user authentication and authorization infrastructure that spans multiple security domains. Many overseas Web applications have been adopting SAML-based federated authentication. However, in Korea, it is difficult to apply the authentication because of the high market share of a specific Web (application) server, which is hard to use open-source SAML software and the high adoption of Java-based standard framework which is not easy to integrate with SAML library. This paper proposes the SAML4J, which is developed in order to have Web applications easily and safely integrated with the Java-based framework. SAML4J has a developer-friendly advantage of using a session storage independent of the framework and processing Web SSO flows through simple API. We evaluate the functionality, performance, and security of the SAML4J to demonstrate the high feasibility of it.

키워드 : SAML, SOAP, 자바, 스프링 프레임워크, 계정연합

Keywords : SAML, SOAP, Java, Spring framework, Identity Federation

Received 4 July 2019, **Revised** 15 July 2019, **Accepted** 25 July 2019

*** Corresponding Author** Jinyong Jo(E-mail:jiny92@kisti.re.kr, Tel:+82-42-869-0585)
Senior Researcher, Advanced KREONET Center, KISTI, Daejeon, 34141, Korea

Open Access http://doi.org/10.6109/jkiice.2019.23.9.1160 print ISSN: 2234-4772 online ISSN: 2288-4165

ⓒThis is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(http://creativecommons.org/li-censes/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Copyright ⓒ The Korea Institute of Information and Communication Engineering.

Ⅰ. 서 론

연합인증(Federated authentication)은 다수의 보안도메인 사이에서 신뢰관계를 바탕으로 구성된 사용자 인증(authentication) 및 인가(authorization) 체계이다. 연구분야와 교육분야(이하, R&E 분야)에서는 국제표준인 OASIS SAML(Security Assertion Markup Language)을 표준규격으로 채택해 연합인증에 활용하고 있다. 하나의 계정연합(Identity federation)에 속한 다수의 개체(서비스제공자나 식별정보제공자)들이 동일한 연합인증 정책과 기술프로파일을 공유하기 때문에 개체들이 서로 다른 보안도메인에 속해 있어도 웹 통합인증(Web Single Sign-On)이 가능하다.

SAML 기반의 웹 통합인증은 사용자 편의성을 제공할 뿐만 아니라 일반적인 클라우드 보안 메커니즘으로도 인정받고 있다[1]. 연합인증은 계정관리(Identity management)과 접근관리(Access management)를 위한 제어점을 단일화함으로써 정보시스템의 보안을 강화하고 효과적인 법규준수를 가능케 한다. 또한, 비밀번호의 복구와 같은 사용자 응대 등 지원비용을 절감할 수 있으며 서비스제공자의 빠른 시장진입을 가능케 한다. 2013년에 수행된 조사결과에 따르면 약 67%의 글로벌 SaaS(Software as a Service) 제공자들이 연합인증을 활용할 수 있도록 SAML 기능을 지원하고 있다[2].

R&E 분야에서는 2000년 초에 SAML 기반의 연합인증이 최초로 설계[3]되었으며 현재는 전 세계 64개 국가에서 총 68개의 계정연합을 형성하고 있다[4]. 60여 국가의 계정연합은 상호 간에 호환이 가능한 기술 체계를 갖추고 있으며 국가 간 계정연합 서비스인 eduGAIN[5]을 통해 서로 연동되고 있다. 즉, 계정연합에 참여 중인 국내 기관의 구성원들은 별도의 계정을 생성하지 않아도 국외 계정연합에 연동된 R&E 응용서비스에 로그인 할 수 있다. 우리나라의 계정연합은 국가과학기술연구망에서 운영 중인 KAFE(Korean Access FEderation[6])이다.

CTA(Cherenkov Telescope Array[7]), ELIXIR[8], WLCG(Worldwide LHC Computing Grid[9]), GENI (Global Environment for Network Innovations[10]) 등 다수의 국외 R&E 응용서비스들은 Shibboleth[11]나 simpleSAMLphp[12]와 같은 공개 SAML 소프트웨어를 이용해 연합인증을 구현하고 있다. 하지만 국내에서는 공개 SAML 소프트웨어를 활용하기 힘든 특정 웹 서버 또는 웹 응용서버의 높은 시장점유율, 전자정부 표준프레임워크(eGovernment framework [13])의 활용 확대로 인한 웹 응용개발 환경의 경직화, 낙후된 SAML 기반의 IAM(Identity and Access Management) 산업생태계로 인한 전문인력 부족 등으로 인해 공개 SAML 소프트웨어를 활용하기 힘든 상황이다.

SAML4J(SAML for Java Web applications)는 전자정부 표준프레임워크 등 Java 기반의 웹 응용개발 환경에서 R&E 응용서비스가 연합인증 기술을 쉽게 통합해 활용할 수 있도록 개발되었다. 개발 프레임워크(예, Spring)에 대한 종속성을 최소화시키고 환경설정이 어려운 XML(eXtensible Markup Language) 대신에 API (Application Programming Interface)를 이용해 Web SSO(Single Sign On) 플로우를 처리케 함으로써 개발자 친화적이다. 또한, 계정연합에 속한 다수의 식별정보제공자들과 연동될 수 있도록 연합 메타데이터를 확장성 있게 관리하며 로그인 할 식별정보제공자를 선택할 수 있도록 탐색서비스(Discovery service)를 제공한다. 속성관리기관(Attribute Authority)으로부터 그룹정보나 자격증명정보와 같은 부가속성을 쉽게 획득할 수 있도록 SOAP(Simple Object Access Protocol) 기반의 속성질의(Attribute Query) 기능을 제공함으로써 정교한 접근제어와 권한관리가 가능하도록 설계되었다.

본 논문의 기여 점은 다음과 같다. 첫째, 국내에서 개발된 웹 기반 R&E 응용의 개발환경을 분석하고 Shibboleth와 같은 기존의 공개 SAML 소프트웨어가 국내 웹 서비스 환경에서 활용되기가 어려운 이유를 설명했다. 둘째, Java 기반의 개발 프레임워크에 쉽고 유연하게 적용할 수 있고 국내 연합인증의 표준규격을 준용하며 계정연합 환경에서도 활용 가능한 국내 최초의 SAML 소프트웨어를 개발했다. 마지막으로, 네트워킹 테스트베드를 구성하고 개발된 SAML 소프트웨어의 확장성과 보안성을 평가함으로써 활용 가능성을 확인하였다.

본 논문은 다음과 같이 구성되어 있다. 국내 R&E 분야의 웹 서비스 환경에 대한 분석과 공개소프트웨어를 활용한 SAML 연동의 문제점을 2장에서 기술한다. 제3장에서는 계정연합에서 주로 활용되는 SAML 공개소프트웨어를 소개하고 개발된 SAML4J와 비교한다. SAML4J의 주요 설계목표와 세부적인 구현내용에 대해

서는 각각 4장과 5장에서 설명한다. 제6장에서는 개발된 소프트웨어의 기능과 성능을 분석하고 보안검사를 수행하며 마지막으로 제7장에서 결론을 맺는다.

Ⅱ. 문제 정의

R&E 분야에서 사용되는 웹 서버(Web server, 이하 WS)와 웹 응용서버(Web application server, 이하 WAS)의 종류를 파악하기 위해 국내 10개 연구기관이 운영 중인 15개 과학기술 응용서비스를 조사했다. 일반적으로 WS는 HTTP 요청을 처리하며 WAS는 비즈니스 로직(Business logic)을 처리한다. 조사한 결과, 표 1과 같이 하나의 서버가 HTTP 요청과 비즈니스 로직을 함께 처리하는 경우는 27%였으며 73%의 과학기술 응용서비스가 WS와 WAS를 분리해 사용했다. 약 93%의 응용서비스는 Java 언어를 이용해 구현되어 있었다.

Table. 1 All kinds of surveyed Web servers and Web application servers

Web server			Web application server		
Apache	WebtoB	Resin	Tomcat	JEUS	Resin
45%	45%	10%	27%	63%	10%

Table. 2 Percentage of the Web server and Web application server

(Apache, Tomcat)	(Apache, JEUS)	(WebtoB, JEUS)	(Resin, Resin)
27%	18%	45%	10%

WS와 WAS가 분리되지 않은 4건의 경우에는 1개의 응용서비스만 아파치 HTTPD를 이용했으며 나머지 3개는 JEUS(Java Enterprise User Solution) 또는 Tomcat을 이용했다. WS와 WAS가 분리된 경우에 활용된 (WS, WAS) 쌍은 표 2와 같다. JEUS와 WebtoB는 국내 Tmax사의 WAS와 WS를 각각 의미한다. WS와 WAS를 분리해 사용하는 11개 과학기술 응용서비스 중에 아파치 HTTPD와 Tomcat 쌍을 사용하는 응용서비스는 총 3개(27%)에 불과했다. 약 63%의 응용서비스가 아파치 HTTPD 또는 WebtoB를 WS로 이용하고 JEUS를 WAS로 활용하고 있음을 확인할 수 있었다. JEUS의 국내 WAS 시장 점유율이 43.8%[14]인 것을 고려하면 상대적으로 높은 수치이다.

아파치 HTTPD와 Tomcat 조합의 경우에는 일반적으로 Shibboleth를 이용해 Web SSO 플로우를 처리한다. Shibboleth는 SAML을 구현한 공개 소프트웨어로써 아파치 모듈(mod_shib)로 동작한다. Shibboleth를 이용하면 아파치 HTTPD가, Web SSO 플로우의 처리를 통해, 사용자의 인증정보와 속성정보를 획득한다. Tomcat에서 구동중인 웹 응용은 Web SSO 플로우 처리와 관련된 비즈니스 로직을 간소화시킬 수 있다. 일반적으로 아파치 HTTPD가 Tomcat에 정보를 전달하기 위해서 AJP(Apache Jserv Protocol)를 활용한다. 하지만 WebtoB가 WS로 사용되거나 JEUS가 WAS로 사용되는 경우에는 문제가 발생한다. WebtoB는 Shibboleth를 지원하지 않는다. 또한, JEUS는 AJP 환경 변수 중에서 선택 변수(optional variables)를 지원하지 않기 때문에 AJP를 이용하면 속성정보를 전달받지 못하는 문제가 있다. 필요한 경우, WS는 속성정보를 HTTP 헤더에 삽입해서 JEUS에게 전달할 수 있지만, HTTP 헤더를 이용한 속성정보의 전달은 HTTP 속임(spoofing) 공격이나 밀수(smuggling) 공격[15]에 취약할 수 있으므로 일반적으로 이용을 권장하지 않는다[16].

결과적으로 JEUS를 이용하는 과학기술 응용서비스들은 Web SSO 플로우 웹 응용에 직접 구현하거나 공개된 Java 기반의 SAML 소프트웨어(예, Spring Security SAML extension)를 이용해야 한다. 하지만 국내에서는 SAML을 활용할 수 있는 전문 인력이 부족하기 때문에 Java 웹 응용에 SAML 소프트웨어를 통합하는 것도 쉽지 않은 상황이다. 웹 응용을 개발하고 있는 12개 국내 소프트웨어 산업체를 설문한 결과, SAML과 관련된 기술개발이나 시스템 통합의 경험이 있는 곳은 없었다. 실제로 개발 프레임워크(예, 전자정부 표준프레임워크)를 이용해 SAML을 통합하는 경우, 보안 문맥(Security context)의 설정이나 사용자 세션 관리에 문제가 발생하면 경험부족으로 인해 개발자들이 신속하게 대처하지 못하는 것을 확인할 수 있었다.

표본 수가 적어서 일반화하기는 어렵지만, 조사된 국내 과학기술 응용들의 다수는 Java 언어를 이용해 구현되었으며 JEUS WAS 상에서 구동되고 있다. Java 웹 응용이 연합인증을 지원하기 위해서는 공개 SAML 소프트웨어를 이용해 Web SSO 플로우를 처리케 하는 것이 취약점 회피 등 보안 상 안전할 것으로 판단된다. 하지

만 국내 여건을 고려하면 Java 웹 응용에 공개 SAML 소프트웨어를 연동하기가 쉽지 않으므로 Web SSO 플로우나 연합 메타데이터의 처리 등 개발자 관점에서 연합인증을 쉽게 구현할 수 있는 Java 기반의 SAML 소프트웨어가 필요하다.

Ⅲ. 관련 기술

앞서 2장에서 기술했듯이 국내에서는 다수의 R&E 응용서비스가 Java를 이용해 개발되었고 JEUS WAS의 시장점유율이 매우 높으므로 Shibboleth나 simpleSAMLphp와 같은 공개 SAML 소프트웨어를 활용하기 어려운 문제가 있다. 연합인증을 쉽게 구현할 수 있도록 OneLogin의 Java SAML 라이브러리[17]를 기반으로 SAML4J를 개발했다. Java 웹 응용이 이용하는 개발 프레임워크(예, Spring 또는 전자정부 표준프레임워크)와 독립적으로 동작하기 때문에 오류의 가능성을 줄일 수 있고 다수의 식별정보제공자와 SAML 연동이 가능(Federation-ready)하므로 계정연합에 참여하는 서비스제공자들이 활용할 수 있다는 장점이 있다. 또한, 프로그래밍 인터페이스(API)를 제공하기 때문에 개발자 친화적(Developer-friendly)인 특징이 있다.

일반적으로 Java 언어로 구현된 웹 응용에 SAML을 직접 연동할 때는 OIOSAML[18], Spring Security SAML[19], OneLogin 등이 활용된다. OIOSAML은 덴마크의 공공분야 응용서비스(예, eGovernment 포털)에 SAML을 적용하기 위해 개발되었다. OASIS SAML 2.0 표준을 준수하지만, 세부 기술프로파일의 구현이 타 SAML 소프트웨어와 차이가 있을 수 있다. 예를 들어, OIOSAML은 RelayState 값을 암호화한다. SAML4J와 유사한 기능을 가졌지만, 개발 목표가 SAML4J와 상이하며 세부 기술프로파일의 지원 여부에 차이가 있다.

Spring Security는 Spring 프레임워크에서 인증과 인가를 담당하는 프레임워크이다. SAML에 대한 지원을 강화하기 위해서 Spring Security SAML(이하, Spring SAML)이 개발되었다. 국내 공공분야에서 개발되는 웹 응용은 일반적으로 전자정부 표준프레임워크를 채택하고 있으므로 Spring SAML을 활용할 수 있다. 하지만 오래된 웹 응용의 경우, 전자정부 표준프레임워크와 Spring SAML 간의 버전 차이로 인해 호환성 문제가 발생할 수 있다. 또한, 오래된 웹 응용의 운용환경을 다시 설정하거나 일부 기능(예, handler)을 구현하기 어려운 문제도 있다. SAML4J는 전자정부 표준프레임워크에서 Spring SAML의 이용으로 인해 발생하는 문제점들을 회피할 수 있도록 설계되었다.

OneLogin의 Java SAML 라이브러리는 사용자 세션의 관리를 웹 응용에게 위임(Session-less)하고 사용하기 쉬운 API를 제공함으로써 개발자들이 SAML을 쉽게 연동할 수 있도록 개발되었다. 하나의 서비스제공자는 하나의 식별정보제공자와만 연동될 수 있다. SAML4J는 OneLogin의 SAML core(예, AuthNRequest나 AuthNResponse 등 SAML 메시지의 생성과 처리를 담당)를 이용했으며 식별정보제공자의 탐색, 사용자 부가속성에 대한 질의와 응답메시지의 처리, 연합 메타데이터의 관리와 같이 계정연합에서 필요로 하는 기능들이 확장 개발되었다. 개발 프레임워크 또는 웹 콘테이너와 독립적으로 사용자 세션을 유지한다는 점도 SAML4J와 OneLogin 소프트웨어의 주요 차이점이다.

Table. 3 Comparison between OIOSAML(O), Spring SAML(S), OneLogin(L), and SAML4J(J)

Function	O	S	L	J
Signature validation	○	○	○	○
Encrypted assertion	○	○	○	○
Discovery service	○	○	×	○
ForceAuthn	○	○	○	○
AttributeQuery	○	×	×	○
Session-less	×	×	○	×
Multiple IdPs	○	○	×	○

앞서 기술한 내용을 정리하면 표 3과 같다. ForceAuthn은 서비스제공자가 특정한 식별정보제공자에게 사용자의 로그인 또는 재로그인을 강제하도록 요청하는 옵션이다. AttributeQuery를 통해 서비스제공자는 SAML 2.0 SOAP 엔드포인트(Endpoint 예, 속성관리기관)로부터 로그인에 성공한 사용자의 부가속성을 가져올 수 있다. SAML4J는 OneLogin의 Java SAML을 상속했기 때문에 HTTP-Redirect 바인딩과 HTTP-POST 바인딩만 지원하는 기능적 한계가 있다. 하지만 두 바인딩만으로도 국내 표준규격을 준수할 수 있으므로 SAML4J를 활용하는 데 있어서 문제가 없을 것으로 판단된다. SAML은 전송규약을 포함하지 않으므로 메시

지를 교환하기 위해서는 인터넷 전송규약을 바인딩해야 한다. 행정기관 SSO 연계 표준 규격[20]에 따르면 인증요청(<samlp:AuthnRequest>)과 인증응답(<samlp:Response>)은 각각 HTTP-Redirect 바인딩과 HTTP-POST 바인딩을 사용하며 로그아웃의 요청(<samlp:LogoutRequest>)과 응답(<samlp:LogoutResponse>)은 HTTP-Redirect 바인딩을 사용하게 되어 있다.

Ⅳ. 설계 요구사항

전문 인력의 부족 등 SAML 기반의 IAM 산업생태계가 조성되지 못한 국내 웹 응용서비스 환경을 고려해 다음과 같은 설계 목표를 기준으로 SAML4J가 개발되었다.

- **개발자 친화(Developer-friendly)** SAML4J는 개발 프레임워크(예, Spring)와의 의존성(Dependency)을 제거함으로써 Java 웹 응용에 SAML을 통합하는 과정에서 발생할 수 있는 오류들을 최소화하도록 설계되어야 한다. 예를 들어, 의존성이 높을 경우, 사용자 인증을 위해 웹 응용이 활용했던 기존 문맥(Context, servlet- context 등)에 SAML을 처리하기 위해 새로운 문맥을 추가하는 과정에서 오류가 발생할 수 있다. 특히 개발한 지 오래되었거나 프레임워크에서 사용하는 문맥설정 방식(예, Spring에서 다수의 인증방식을 활용하기 위해서는 AuthenticationProvider 인터페이스를 활용)을 따르지 않고 자체적으로 인증기능을 구현한 경우에는 SAML 통합과정에서 문제가 생길 수 있다. 개발자에게 문맥을 설정케 하는 대신에 API를 제공하고 웹 응용을 대신해 SAML4J가 Web SSO 플로우를 처리하게 함으로써 오류 발생의 가능성을 줄여야 한다.
  웹 응용 또는 개발 프레임워크는 사용자 세션에 대한 관리를 SAML4J에게 위임할 수 있어야 한다. 연합인증 환경에서 세션은 로그인한 사용자 정보와 단일 로그온(Single Logon)이나 단일 로그아웃(SLO, Single Logout)에 필요한 엔드포인트 정보도 함께 저장하고 있다. 세션 관리를 SAML4J에게 위임함으로써 개발자는 속성정보의 활용이나 단일 로그아웃을 API 수준에서 간단하게 처리할 수 있게 된다. 추가로 세션 관리의 유연성을 제공하기 위해서 SAML4J가 자체적으로 세션을 관리하는 기능 이외에 서블릿 컨테이너(예, Tomcat WAS)의 HttpSession 클래스 객체에 세션 정보를 저장하고 개발자의 필요에 따라 활용될 수 있도록 SAML4J가 설계되어야 한다.

- **계정연합 지원(Federation-ready)** SAML4J는 계정연합 환경에서 범용적으로 사용될 수 있도록 설계되어야 한다. 계정연합은 많은 수의 식별정보제공자와 서비스제공자로 구성되기 때문에 서비스제공자는 다수의 메타데이터에 대해서도 검증하고 탐색할 수 있는 등 높은 확장성을 가져야 한다. 참고로 eduGAIN의 연합 메타데이터에는 3천개 이상의 식별정보제공자가 포함되어 있다. SAML4J를 활용하는 서비스제공자는 식별정보제공자의 메타데이터를 다수 포함하고 있는 연합 메타데이터의 XML을 파싱해 메타정보를 저장하거나 읽을 수 있어야 하고 중앙형(Central) 또는 탑재형(Embedded) 탐색서비스를 이용해 사용자를 인증할 식별정보제공자를 선택할 수 있어야 한다. 또한, 국내 계정연합에서 규정한 표준규격(예, 한글 속성명 등)이나 기술프로파일(예, SAML entity category[21])을 지원하고 필요에 따라 속성관리기관으로부터 부가속성을 얻을 수 있게 설계함으로써 기술호환성을 높여야 한다.
  계정연합은 개체들 간의 신뢰를 바탕으로 동작하기 때문에 개별 개체들은 신뢰를 검증하거나 제어할 수 있는 기능적 능력(Functional capabilities)을 갖추도록 설계되어야 한다. 서비스제공자가 사용자 인증과 신원확인에 대해서 높은 보증등급(Level of Assurance, LoA)을 요구하는 경우에 SAML4J는 신뢰도가 낮은 식별정보제공자(예, Guest Identity Provider)에 대해서 접근을 제한할 수 있어야 한다. 또한, 연합인증에서 사용하는 SAML이나 SOAP 메시지에 전자적으로 서명하거나 검증할 수 있는 방법을 제공해 메시지의 무결성(Integrity)을 보장함으로써 보안 신뢰도를 높일 수 있어야 한다.

Ⅴ. 구 현

제4장에서 기술한 설계 요구사항을 바탕으로 SAML4J가 개발되었다. SAML4J는 그림 1과 같이 메타데이터

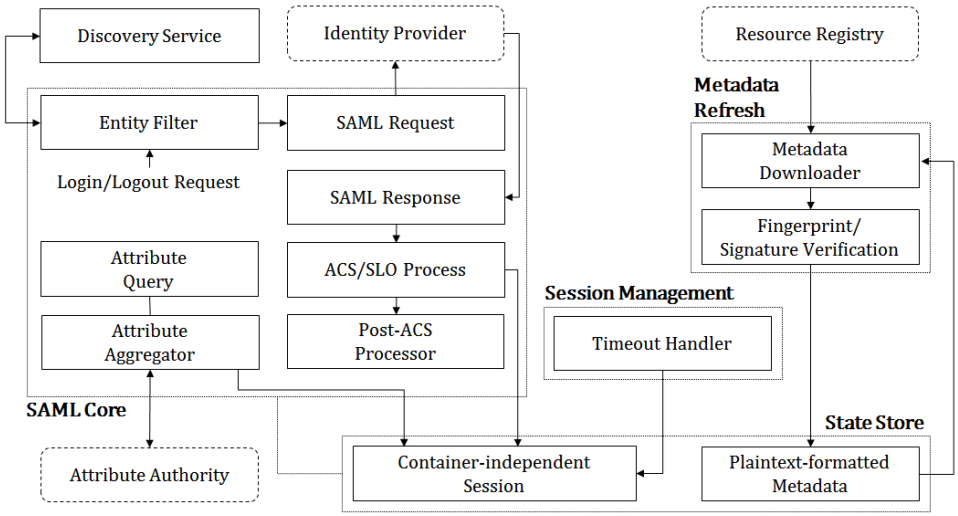

Fig. 1 High-level overview of developed SAML4J

관리(Metadata Refresh), 세션 관리(Session Management), 메시지 처리(SAML Core) 및 상태 저장소(State Store)로 구성된다. 식별정보제공자(Identity Provider)와 메타데이터 등록기(Resource Registry) 및 속성관리기관(Attribute Authority)은 SAML4J와 연동될 수 있는 외부 정보시스템들이다. 메타데이터 등록기는 연합 메타데이터를 등록하고 배포하는 저장소이며 속성관리기관은 사용자의 부가 속성을 제공한다. 일반적으로 SAML4J와 연동되는 외부 정보시스템들은 계정연합의 운영기관 또는 R&E 기관에서 제공한다. 메타데이터 등록기와 속성관리기관은 생략될 수 있다.

5.1. SAML 메타데이터의 갱신

서비스제공자나 식별정보제공자와 같은 SAML 개체들은 메타데이터의 교환을 통해 신뢰체인(Circle of Trust)을 형성하고 기준 상호운용성(Baseline interoperability)을 확보한다. 일반적으로 개별 계정연합의 신뢰할 수 있는 제3자(Trusted 3rd party)는 메타데이터 등록기(Resource Registry)를 통해 SAML 개체들의 개별 메타데이터를 수집한다. 신뢰할 수 있는 제3자는 수집된 개별 메타데이터를 취합해 연합메타데이터를 구성하고 전자적으로 서명한 후에 공개된 URL 주소를 통해 연합 메타데이터를 배포한다. 추가로, 연합 메타데이터의 서명 값을 검증할 수 있는 X.509 인증서 정보와 인증서의 위변조 여부를 검사할 수 있도록 인증서의 해시값(Fingerprint)을 공개한다.

메타데이터 갱신(Metadata Refresh) 블록은 지정된 URL 주소로부터 연합 메타데이터를 주기적으로 내려받고 메타정보의 유효성을 검증하기 위해 메타데이터의 서명 값과 유효기간(ValidUntil), 인증서의 해시값을 검증할 수 있도록 개발했다. 위변조 여부와 유효기간의 검증을 통과하면 연합 메타데이터에 포함된 XML 형태의 개별 메타데이터는 평문(Plaintext) 형태의 메타정보로 변환되어 SAML4J의 클래스 객체(Plaintext-formatted Metadata)에 저장된다. 메시지 처리(SAML Core) 블록에서 식별정보제공자의 엔드포인트 URL 주소 등을 얻기 위해 클래스 객체에 저장한 정보를 이용한다. 유효성 검증을 통과하지 못하면 내려받은 연합 메타데이터는 폐기된다. 공개된 URL 주소에서 자동으로 내려받는 방법 이외에 연합 메타데이터를 파일 형태로 읽거나 쓸 수 있도록 SAML4J를 설계함으로써 메타데이터의 획득방법을 다양화할 수 있다. 메타데이터 갱신 블록은 메시지 처리 블록에 독립적으로 동작하기 때문에 서블릿 문맥 리스너(Servlet context listener)를 사용하여 쓰레드(Thread)로 구현했다.

마지막으로, 메타데이터 갱신 블록은 새로운 메타정보를 상태 저장소(Status Store)에 저장하거나 유효기간이 지난 메타정보를 파기하는 등 저장된 메타정보들이 최신 상태를 유지할 수 있도록 관리함으로써 연합 메타데이터와 상태 저장소에 저장된 메타정보가 일관성

(Consistency)을 가질 수 있도록 설계되었다. 상태 저장소에 저장된 메타정보는 메시지 처리 블록에서 사용한다.

5.2. SAML 메시지의 생성 및 교환

메시지 처리(SAML Core) 블록은 상태 저장소를 검색해 식별정보제공자의 메타정보(엔드포인트 URL 주소 등)를 획득하고 SAML 요청메시지를 생성한다. 사용자가 탐색서비스에서 선택한 식별정보제공자의 개체식별자(EntityID)를 키값으로 저장소를 검색한다. 또한, 식별정보제공자가 전달한 SAML 응답메시지를 수신하고 사용자의 인증정보와 속성정보 등을 상태 저장소의 SAML4J 세션(Container-independent session)에 저장한다. ACS 후처리기(Post-ACS Processor)는 다수의 속성관리기관으로부터 사용자의 부가속성을 쉽게 얻을 수 있도록 설계되었다. 추상클래스로 정의되었기 때문에 개발자가 SAML4J의 API를 이용해 후처리기를 구현함으로써 SOAP 기반의 속성질의(AttributeQuery) 메시지를 속성관리기관에 보내거나 전달받은 응답메시지를 처리할 수 있다. ACS(Assertion Consumer Service)는 SAML 응답메시지를 검증하고 특정 URL 주소로 사용자의 웹 브라우저를 리다이렉트한다.

서비스제공자는 SAML4J의 환경설정을 통해 탑재형 또는 중앙형 탐색서비스를 선택적으로 이용할 수 있다. 개체필터(Entity Filter) 블록은 서비스제공자가 연동을 허락하거나 거부하는 식별정보제공자들에 대해서 접근제어를 실행한다. 예를 들어, 서비스제공자는 개체필터를 이용해 특정 식별정보제공자를 거부할 수 있다. 접근이 거부된 식별정보제공자는 탑재형 탐색서비스에 나타나지 않는다. 탐색서비스는 사용자가 선택한 식별정보제공자의 개체식별자를 URL 매개변수를 이용하여 서비스제공자에게 전달한다. 매개변수가 변조되면 접근이 거부된 식별정보제공자에도 사용자가 로그인할 수 있으므로 보안위협의 가능성이 있다. 즉, 상태 저장소에 존재하는 개별 메타정보의 집합을 M_L, 개체필터가 허용하는 메타데이터의 집합을 $M_F(M_P \subseteq M_L)$, $M_R = M_L - M_F$이고 $M_R \neq \varnothing$ 인 식별정보제공자 $x \in M_R$에 대해서 문제가 발생할 수 있다. 개체필터는 SAML 요청메시지를 보내기 전이나 응답메시지를 수신한 후에 메시지를 교환한 식별정보제공자가 접근 권한을 가지고 있는지 확인함으로써 매개변수의 변조여부를 검증한다.

식별정보제공자에게 SAML 인증요청(AuthnRequest) 메시지를 전달하기 위해 HTTP-Redirect 바인딩을 이용하고 메시지의 무결성을 보장하기 위해서 서비스제공자의 개인키로 서명하도록 설계했다. SAML 응답메시지를 처리하기 위해서 OneLogin의 Java SAML 라이브러리를 이용했다. 해당 SAML 라이브러리는 HTTP-POST 바인딩을 통해 식별정보제공자로부터 응답메시지를 전달받으며 메시지에 포함된 전자서명의 유효성을 검증할 수 있다. ACS 처리(ACS Process)에서는 SAML 어썰션(Assertion)에 포함된 사용자의 인증정보와 속성정보를 파싱하고 세션 정보를 상태 저장소에 저장한다. SLO 처리(SLO Process)는 식별정보제공자로부터 단일 로그아웃에 대한 요청메시지를 수신하면 상태 저장소에서 해당되는 세션 정보를 삭제하고 로그아웃할 수 하도록 구현되었다.

5.3. SOAP 메시지의 생성 및 교환

연합인증의 적용범위를 확대하기 위해서 서비스제공자는 가상조직(Virtual organization)에 대한 그룹속성(예, isMemberOf 속성)이나 SSH 공개키속성(예, sshPublicKey)과 같이 식별정보제공자가 관리하지 않는 사용자의 부가속성들을 속성관리기관로부터 획득할 수 있어야 한다. 사용자는 속성관리기관에 속성값들을 등록하고 속성관리기관은 인증된 사용자의 속성정보를 서비스제공자에게 전달한다. 일반적으로 속성관리기관은 SOAP이나 REST(Representational State Transfer) API를 통해 속성정보를 제공한다.

속성취합(Attribute Aggregate) 블록은 SOAP을 이용해 하나 이상의 속성관리기관으로부터 사용자의 부가속성을 획득할 수 있도록 개발되었다. 개발자는 속성관리기관의 개체식별자와 사용자 검색키로 이용할 속성명(예, eduPersonPrincipalName)을 SAML4J에 정의할 수 있다. 검색키로 이용할 속성값은 SAML 어썰션에 포함되어 있어야 한다. 속성취합 블록은 속성질의(Attribute Query) 블록을 이용해 SAML 2.0 <AttributeQuery> 메시지(표 4의 예시 참조)를 생성하고 속성관리기관의 SOAP 엔드포인트에게 사용자의 부가속성을 요청한다. SOAP 엔드포인트는 속성관리기관의 메타데이터나 연합 메타데이터에 포함된다. SOAP 메시지의 위변조를 막기 위해 속성질의 블록이 메시지를 생성하는 과정에서 전자서명을 삽입할 수 있도록 개발되었다.

Table. 4 An example of the SOAP <AttributeQuery> message

```
<soap-env:Envelope …,>
 <soap-env:Header/>
 <soap-env:Body>
 <samlp:AttributeQuery …, ID="_ecd02e88">
  <saml:Issuer>[entityID of the service provider]
    </saml:Issuer>
  <ds:Signature …,>
   <ds:SignedInfo>
    …,
    <ds:Reference URI="#_ecd02e88">
      …,
     <ds:DigestValue>[Digest value]
     </ds:DigestValue>
    </ds:Reference>
   </ds:SignedInfo>
  <ds:SignatureValue>[Signature]
  </ds:SignatureValue>
  <ds:KeyInfo> …, </ds:KeyInfo>
 </ds:Signature>
  <saml:Subject>
   <saml:NameID ..., >[Search key]
   </saml:NameID>
  </saml:Subject>
 </samlp:AttributeQuery>
</soap-env:Body>
</soap-env:Envelope>
```

속성관리기관로부터 SOAP 응답메시지를 수신하면 속성취합 블록은 상태 저장소에 보관 중인 해당 사용자의 속성정보에 전달받은 부가속성을 추가한다. SOAP 응답메시지의 무결성을 확인하기 위해서 속성관리기관이 메시지에 삽입한 전자서명은 반드시 검증하도록 개발되었다.

5.4. 세션 관리 및 프로그래밍 인터페이스

SAML4J는 전자정부 표준프레임워크 등에서 제공하는 Session 클래스나 보안 문맥을 활용하지 않도록 설계함으로써 개발 프레임워크와의 종속성을 최소화했다. 세션 정보는 SAML4J의 상태 저장소에서 관리되며 JSESSIONID를 키값으로 활용하여 검색할 수 있도록 설계되었다. 웹 응용이 개발 프레임워크로부터 독립적인 상태 저장소에서 인증된 사용자의 세션 정보를 읽고 권한을 부여할 수 있기 때문에 프레임워크에서 활용하는 사용자 세션이나 보안 문맥과의 충돌을 회피할 수 있다.

Table. 5 Binding of an endpoint URL to a class function

URL(/saml/)	Function	Description
discovery	discovery()	Request Discovery Service
login	login()	Request user login
logout	logout()	Request user logout
metadata	metadata()	Get SAML metadata
acs	acs()	Request Assertion Consumer Service (ACS)
handler	handler()	Store attributes to a HTTP Session
slo	sls()	Request Single Log-out (SLO)

엔드포인트 URL로 사용자가 접근하거나 웹 브라우저가 리다이렉트되면 SAML4J는 클래스 함수로 정의된 비즈니스 로직을 실행한다. 예를 들어, SAML4J가 설치된 웹 응용의 도메인 주소가 example.org인 경우, 사용자가 https://example.org/saml/metadata로 접근하면 클래스 함수인 metadata()가 실행되고 서비스제공자의 메타데이터가 화면에 표시된다. 개발자가 API 수준에서 비즈니스 로직을 설계할 수 있도록 표 5와 같은 클래스 함수들이 구현되었다. 표 5는 구현된 클래스 함수의 일부를 보여준다. Controller(예, HomeController) 클래스에 엔드포인트 URL을 정의하고 함수로 구현할 수 있기 때문에 개발자의 필요에 따라 다양하게 활용될 수 있다. SAML4J는 기본적으로 상태 저장소를 이용해 세션 정보를 관리하지만 handler() 함수를 통해 HTTP 세션도 이용할 수 있도록 설계되었다.

표 6의 유사코드는 웹 응용이 SAML4J의 예제 응용을 통해 Web SSO 플로우(사용자 인증 및 속성정보의 획득)를 처리하는 방법을 간략히 보여준다. Auth 클래스의 생성자는 HttpServletRequest 객체와 HttpServletResponse 객체를 인자로 가지며 서비스제공자의 SAML 메타데이터, SAML 메시지의 서명과 검증 여부, 속성관리기관의 개체식별자, 허용되는 식별정보제공자의 개체식별자 등을 클래스 내에 정의할 수 있다. Auth 클래스는 식별정보제공자로부터 사용자가 인증되었는지 확인할 수 있도록 isAuthenticate() 함수를 제공하고 인증에 성공한 사용자의 속성정보를 얻기 위해서 getAttributes() 함수를 이용할 수 있다. 유사코드에서 확인할 수 있듯이 SAML4J는 수 라인의 코드 구현만으로 Java 기반의 웹 응용에서 SAML을 쉽게 활용할 수 있도록 개발되었다.

Table. 6 Pseudo code used in a Web application for user authentication

```
import kr.or.kafe.Auth

Auth auth = new Auth(request, response);
if (auth.isAuthenticated() == true) {
  Map<String, List<String>> attributes =
auth.getAttributes();
  for (String key : attributes.keySet()) {
    attributes.get(key).get(0);
}
```

Ⅵ. 평가 및 결과

본 논문은 SAML4J 예제 응용이 SAML과 SOAP 프로토콜을 이용해 각각 식별정보제공자와 속성관리기관으로 부터 사용자의 속성과 부가속성을 정상적으로 획득할 수 있는지 검증하고 다수의 개별 메타데이터로 구성된 연합 메타데이터를 활용해 SAML4J가 확장성 있게 메타데이터를 처리할 수 있는지 확인한다. 또한, SAML과 SOAP 메시지를 처리하는데 걸리는 시간을 측정함으로써 SAML4J의 성능을 살펴본다. 마지막으로 보안 검사를 통해 XML Signature Wrapping(XSW) 공격을 정상적으로 방어하는지 검증함으로써 Java 기반의 R&E 응용서비스 환경에서 SAML4J가 안전하게 활용될 수 있음을 확인한다.

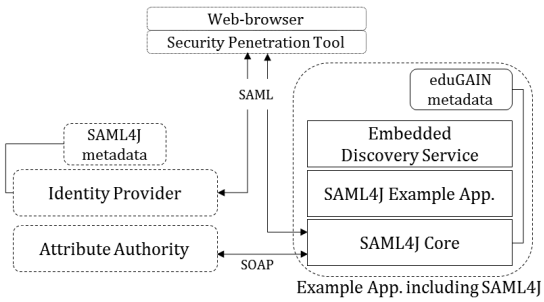

Fig. 2 Networking testbed for a federated authentication

먼저, SAML4J의 기능과 성능을 검증하기 위해서 그림 2와 같이 네트워킹 테스트베드를 구성했다. PyFF[22] 등을 이용하여 eduGAIN의 연합 메타데이터(eduGAIN metadata)를 재구성한 후에 SAML4J에 등록했으며 검증용 식별정보제공자(Identity Provider)와 SAML4J의 개체 메타데이터(SAML4J metadata)는 양자 간에만 교환되게 하였다. SAML4J는 재구성된 연합 메타데이터와 검증용 식별정보제공자의 메타데이터 및 검증용 속성관리기관(Attribute Authority)의 메타데이터를 가지고 있으며 예제 응용을 접속하는 사용자는 검증용 식별정보제공자를 통해서만 인증받을 수 있다. PyFF는 SAML 메타데이터를 수정할 수 있는 소프트웨어이다. 예제 응용은 JVM(Java Virtual Machine) 1.7에서 동작 중인 Tomcat WAS 8.5에서 구동되었으며 물리 서버는 4 GHz Intel i7 CPU와 32 GB 메모리로 구성되었다.

SOAP 기반의 속성질의(Attribute Query) 기능을 검증하기 위해서 검증용 속성관리기관이 테스트베드 상에 구축되었으며 SAML4J와 속성관리기관 간에 개별 메타데이터를 서로 교환했다. 검증용 식별정보제공자와 검증용 속성관리기관은 simpleSAMLphp를 이용해 구현했다. 마지막으로 SAML4J의 보안 취약점을 검증하기 위해서 사용자 컴퓨터에 모의침투도구(Security Penetration Tool)를 설치했다.

6.1. 메시지 처리 및 확장성 분석

먼저, 예제 응용에서 Web SSO 플로우를 처리한 결과를 바탕으로 개발된 SAML4J가 인증된 사용자의 속성정보를 정상적으로 획득할 수 있는지를 확인한다. 그림 3의 상단은 로그인한 사용자에 대해서 검증용 식별정보제공자가 예제 응용으로 전달하는 속성정보를 보여주며 그림의 하단은 예제 응용에서 최종적으로 확보한 속성정보의 목록을 보여준다. 예제 응용은 SAML과 SOAP을 이용해 각각 식별정보제공자와 속성관리기관으로부터 사용자 속성정보를 획득한다. 구성된 네트워킹 테스트베드에서 속성관리기관이 인증된 사용자의 그룹정보(isMemberOf 속성)를 제공하는 것을 확인할 수 있다. isMemberOf 속성의 개체 ID(OID)는 1.3.6.1.4.1.5923.1.5.1.1이다. 가명 ID(Persistent pseudonymous ID)의 정식 속성명은 eduPersonTargetedID이며 개체 ID는 1.3.6.1.4.1.5923.1.1.1.10이다. 가명 ID가 동일한 사용자에 대해서 식별정보제공자가 전달하지 않은 isMemberOf 속성정보를 SAML4J 예제 응용이 획득했음을 알 수 있다. 결과적으로, 개발된 SAML4J가 SAML 메시지와 SOAP 메시지를 정상적으로 처리함을 확인할 수 있다.

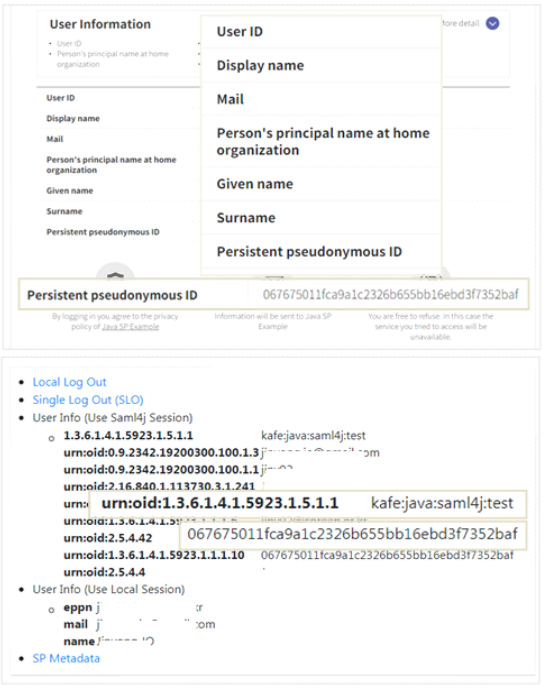

Fig. 3 Attributes sent by an identity provider (up) and received by the SAML4J example app. (down)

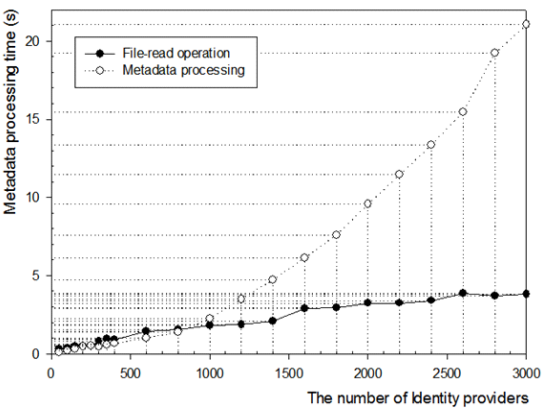

Fig. 4 Processing time to refresh a federation metadata

그림 4는 SAML4J가 연합 메타데이터를 상태 저장소에 등록하는 데 걸리는 시간을 보여준다. 국내 계정연합에서는 최소 4일에 한 번 이상 연합 메타데이터를 갱신하여 배포하고 있다. 서비스제공자는 주기적으로 연합 메타데이터를 내려받고 백그라운드 프로세스로 처리해 상태 저장소에 등록한다. eduGAIN에서 제공하는 연합 메타데이터를 이용했으며 메타데이터에 포함된 식별정보제공자의 수를 줄여가면서 측정했다. 실험에 활용한 연합 메타데이터는 최대 3,000개의 식별정보제공자와 2,500여 개의 서비스제공자가 등록되어 있다.

SAML4J는 특정 URL 주로로부터 연합 메타데이터를 읽거나 지정된 파일로부터 연합 메타데이터를 읽은 후에 메타정보를 상태저장소에 등록한다. 본 논문에서는 URL 주소로부터 연합 메타데이터를 내려받아 스토리지에 파일로 저장한 후에 상태 저장소에 등록하는 순서대로 실험을 진행했다. 측정된 파일 읽기(File-read operation)와 메타데이터 처리(Metadata processing) 결과는 네트워크에서 발생한 전송지연을 포함하지 않는다. 연합 메타데이터에 포함된 식별정보제공자들의 수가 증가할수록 파일의 크기가 커지기 때문에 파일을 읽는데 필요한 시간이 선형적으로 증가하는 것을 알 수 있다.

메타데이터 처리시간은 연합 메타데이터에서 메타정보를 추출해 상태 저장소에 등록하는 데 걸리는 시간이다. 연합 메타데이터에 포함된 식별정보제공자가 600개일 때 약 1초가 소요되었으며 1,000개일 경우에는 약 2.2초가 걸렸다. 식별정보제공자가 증가할수록 처리하는 데 걸리는 시간이 늘어나는 것을 확인할 수 있다. SAML4J는 메타정보를 등록하는 과정에서 상태 저장소를 초기화시키지 않음으로써 메타정보의 일시적 부재로 인한 서비스 중단 가능성을 최소화시킨다. 서비스제공자가 식별정보제공자의 메타정보를 가지고 있지 않으면 Web SSO 플로우를 처리할 수 없다. SAML4J는 메타데이터의 처리를 위해 상태 저장소를 먼저 검색하고 기존에 등록된 메타정보를 필요에 따라 삭제하거나 갱신하며 새로운 메타정보는 추가하는 방식으로 동작한다. 메타데이터의 처리 방식으로 인해 개체 수가 증가하면 처리시간도 길어지게 된다. 하지만 메타데이터의 처리가 백그라운드 프로세스로 동작하고 계정연합(예, 미국의 InCommon)이 일반적으로 1,000개 이하의 식별정보제공자를 관리하기 때문에 처리시간의 증가가 SAML4J의 확장성에 주는 영향은 크지 않을 것으로 판단된다. 참고로 국내 계정연합에서는 eduGAIN에 등록된 개체 중에 600여 개의 식별정보제공자와 500여 개의 서비스제공자만 수용하고 있다.

Table. 7 Processing time in the SAML4J Core (s)

Entity Filter	SAML Response	SOAP Response
0.019	0.029	0.009

마지막으로 SAML 메시지처리 블록(그림 1의 SAML Core)에 속한 구성요소들의 성능을 살펴보기 위해서 개체필터(Entity Filter), SAML 응답메시지(SAML Response) 및 SOAP 응답메시지(SOAP Response, 그림 1의 Attribute Aggregator)의 처리속도를 측정했다. 구현된 개체필터는 총 N개의 개체식별자에 대해서 $O(N)$의 검색 복잡도를 갖는다. 해시맵을 사용했기 때문에 제거해야할 개체식별자의 수(M)가 증가하여도 복잡도에 영향을 주지 않는다. 총 3,000개의 식별정보제공자에 대해서 개체필터의 평균 처리속도는 표 7과 같이 19 ms로 측정되었다.SAML 응답메시지와 SOAP 응답메시지의 처리를 위해서는 평균적으로 각각 29 ms와 9 ms가 걸린 것으로 확인되었다. 메시지 처리시간은 식별정보제공자 또는 속성관리기관으로부터 각 응답메시지를 수신한 시점부터 처리를 완료한 시점까지 걸린 시간으로써 네트워크 지연은 제외했다. 결과적으로 탐색서비스 이용을 위한 사용자 인터렉션 시간과 네트워크 지연을 제외하면 SAML 메시지처리 블록에서 60 ms 이내의 처리지연이 발생한다. 일반적으로 사용자가 인내할 수 있는 웹 응답시간의 최대치가 4초로 여겨지므로[23] SAML 메시지처리 블록에서 발생한 처리지연은 무시할 만한 수준으로 여겨진다.

6.2. 보안 검사

SAML 프레임워크가 XML Signature Wrapping(XSW) 취약점을 가질 수 있다는 것은 잘 알려진 사실이다. XSW 취약점은 SAML 어썰션을 처리하는 특정 소프트웨어가 사용자 속성의 유효성을 정상적으로 검증하지 못하기 때문에 발생하는 문제이다[24]. 공격자가 어썰션에 포함된 XML Element 트리를 변경함으로써 취약점이 있는 SAML 소프트웨어가 서명 검증에 이용된 Element 대신에 공격을 위해 의도적으로 삽입된 Element를 파싱하게 하는 방법이다[25]. XSW 취약점을 가지고 있는 서비스제공자는 식별정보제공자로부터 인증받지 않은 임의의 사용자를 인증 받은 사용자로 잘못 인식할 수 있다.

SAML4J가 OneLogin의 Java SAML 라이브러리를 이용해 SAML 어썰션을 처리하기 때문에 XSW 취약점의 존재여부는 해당 라이브러리로부터 크게 영향을 받는다. 모의침투도구인 BURP Suite[26]를 이용했고 8가지 공격유형[27]에 대해서 SAML4J의 XSW 취약점을 점검했다. 모든 공격유형에 대해서 '어썰션의 발행자(Issuer)가 없거나 다수 존재', '서명검증의 실패', '하나 이상의 어썰션 탐지' 등의 이유로 HTTP Status 500 오류를 발생시켰다. 결론적으로 SAML4J가 XSW 공격에 대해서 높은 보안성을 가지고 있음을 확인할 수 있었다.

SOAP 속성질의와 응답에 대한 XSW 공격을 방어하기 위해서 OneLogin의 Java SAML 라이브러리(SAML Response 함수)를 분석했다. 해당 라이브러리에서 SAML 어썰션 메시지를 처리하는 절차는 다음과 같다. ① XML 최상위(Root) Element가 ID 속성을 가졌는지 여부를 검사하고 최상위 Element가 ID 속성을 가지고 있지 않으면 오류를 발생시킨다. ② SAML 어썰션 메시지에 어썰션이 하나만 포함되어 있는지 검사하고 하나도 없거나 두 개 이상의 어썰션이 메시지에 포함되어 있으면 오류를 발생시킨다. ③ SAML 문법검사를 수행하며 마지막으로 ④ 전자서명을 검증한다. 문법 오류가 있거나 전자서명의 검증에 통과하지 못하면 오류가 발생한다. 위 절차 중에 ①, ②, ④가 XSW 공격을 효과적으로 방어하는 수단으로 활용됨을 확인할 수 있었다. SAML4J는 SOAP 응답메시지에 대해서 OneLogin과 동일한 방법으로 XSW 공격을 방어하기 때문에 해당 공격에 대해 효과적으로 대응할 수 있을 것으로 기대한다.

Ⅶ. 결 론

본 논문은 전자정부 표준프레임워크 등 Java 기반의 웹 응용개발 환경에서 SAML 기반의 Web SSO 플로우를 쉽게 구현할 수 있도록 SAML4J를 개발하고 네트워킹 테스트베드 상에서 기능과 성능 및 보안 고려사항을 검증했다. SAML과 SOAP 규약을 통해 사용자의 속성정보를 정상적으로 획득했으며 수천 개의 메타데이터에 대해서도 메타정보를 빠르게 구조화함으로써 SAML4J 소프트웨어의 확장성을 확인할 수 있었다. 또한, XSW 보안공격에 대해서도 효과적으로 대응할 수 있음을 알 수 있었다. 탐색서비스와 개체필터의 기능을 확장하고 신규로 개발되는 또는 구동중인 Java 기반의 R&E 응용을 대

상으로 SAML4J를 적용해 나갈 계획이다.

ACKNOWLEDGEMENT

The work of this paper is supported by Korea Institute of Science and Technology Information (K-19-L02-C02)

REFERENCES

[1] I. M. Khalil, A. Khreishah, and M. Azeem, “Cloud Computing Security: A Survey,” *Computers*, vol.3, no.1, pp.1-35, 2014.

[2] OneLogin. OneLogin 2014 State of SaaS Identity Management [Internet]. Available: https://resources.onelogin.com/WP-OneLogin-2014-SaaS-Identity-Management.pdf?path=wp-content/images/OneLogin_2014_SaaS_Identity_Management.pdf.

[3] S. Droz, C. Hassenstein, G. Heim, T. Meier, D. Monnard, and H. C. Tschudin, “Concept for an Electronic Academic Community in Switzerland and the creation of a Common Authentication and Authorization Infrastructure (AAI) for the Swiss Higher Education System,” Inter-University Working Group, Oct., 2001.

[4] Metadata Explorer Tool [Internet]. Available: https://met.refeds.org/.

[5] eduGAIN [Internet]. Available: https://www.edugain.org/.

[6] Korean Access Federation [Internet]. Available: https://www.kafe.or.kr/.

[7] A. Costa, M. Pietro, B. Marilena, B. Ugo, K. Mel, P. Costantino, R. Simone, S. Eva, and V. Fabio, “An Innovative Science Gateway for the Cherenkov Telescope Array," *Journal of Grid Computing*, vol.13, no.4, pp.547-559, 2015.

[8] M. Linden, M. Prochazka, I. Lappalainen, D. Ducik, P. Vyskocil, M. Kuba, S. Silen, P. Belmann, A. Sczrba, S. Newhouse, L. Matyska, and T. Nyronen, “Common ELIXIR Service for Researcher Authentication and Authorisation,” *F1000Research 7*, pp.1-15. Aug., 2018.

[9] H. Short, A. Manzi, V. D. Notaris, O. Keeble, A. Kiryanov, H. Mikkonen, P. Tedesco, and R. Wartel, “x509-free Access to WLCG Resources,” *Journal of Physics: Conference Series*, vol.898, no.8, pp.1-7, Oct., 2017.

[10] M. Brinn, “GENI Architecture Foundation,” The GENI Book, Springer, Cham, p.101-116, 2016.

[11] Shibboleth Consortium [Internet]. Available: https://shibboleth.net/.

[12] simpleSAMLphp, [Internet]. Available: https://www.simplesamlphp.org/.

[13] K. Kim and K. Lee, “Visualization of Geo-spatial Data and Public Data Using Mobile Operating Environment in the eGovernment Standard Framework,” *Journal of Korea Spatial Information Society*, vol.23, no.1, pp.9-17, Feb., 2015.

[14] J. Park, Ranked 1st in the WAS Market in 2017, Electronic Times Internet [Internet]. Available: https://news.v.daum.net/v/20180802140303883.

[15] C. Linhart, A. Klein, R. Heled, and S. Orrin, HTTP REQUEST SMUGGLING. (2005) [Internet]. Available: http://www.cgisecurity.com/lib/HTTP-Request-Smuggling.pdf.

[16] Shibboleth Wiki [Internet]. Available: https://wiki.shibboleth.net/confluence/display/SP3/ReleaseNotes.

[17] OneLogin’s SAML Java Toolkit [Internet]. Available: https://github.com/onelogin/java-saml.

[18] Danish Agency for Digitisation, “OIOSAML Web SSO Profile 3.0 ‘Release Candidate’,” 2019.

[19] Spring Security SAML Extension [Internet]. Available: https://docs.spring.io/autorepo/docs/spring-security-saml/1.0.x-SNAPSHOT/reference/htmlsingle/.

[20] Ministry of the Interior and Safety, “Technical Specification of SSO Authentication Gateway,” June 2018.

[21] SAML Entity Category [Internet]. Available: https://refeds.org/specifications.

[22] PyFF - A SAML Metadata Appliance [Internet]. Available: http://pyff.io/.

[23] C. Lorentzen, M. Fiedler, H. Johnson, J. Shaikh, and J. Ivar, “On User Perception of Web Login - a Study on QoE in the Context of Security,” in *Proceedings of the Australasian Telecommunication Networks and Applications Conference, Auckland*, New Zealand, pp.84-89, 2010.

[24] J. Somorovsky, A. Mayer, J. Schwenk, M. Kampmann, and M. Jensen, “On Breaking SAML: Be Whoever You Want to Be,” in *Proceedings of the 21st USENIX Security Symposium*, Bellevue, USA, Aug., 2012.

[25] N. Engelbertz, N. Erinola, D. Herring, J. Somorovsky, V. Mladenov, and J. Schwenk, “Security Analysis of eIDAS - The Cross-Country Authentication Scheme in Europe,” in *Proceedings of the 12th USENIX Workshop on Offensive Technologies*, Baltimore, USA, 2018.

[26] M. Christian, M. Vladislav, G. Tim, and J. Schwenk, “Automatic Recognition, Processing and Attacking of Single Sign-on Protocols with BURP Suite,” *Open Identity Summit*, 2015.
[27] H. Phong, How to use Brup Suite to Verify SAML Signature Wrapping Attack [Internet]. Available: https://blog.ritvn.com/testing/2018/02/16/burp-suite-saml-signature-wrapping-attack.html.

조진용(Jinyong Jo)

2003년 광주과학기술원 정보통신학과 공학석사
2013년 광주과학기술원 정보통신학과 공학박사
2003년- 한국과학기술정보연구원
2016년- eduGAIN interfederation 운영그룹 위원
※관심분야 : Federated Identity management

채영훈(Yeonghun Chae)

2015년 고려대학교 전자 및 정보공학과 학사
2017년 과학기술연합대학원대학교 빅데이터과학 석사
2017년- 한국과학기술정보연구원
※관심분야 : Deep learning, Federated Identity Management

공정욱(JongUk Kong)

1998년 포항공과대학교 석사
2015년 충남대학교 정보통신공학과 박사
2002년- 한국과학기술정보연구원
※관심분야 : 네트워크 자원제어, SDN