실행시간 의존성 측정을 통한 SOA 취약성 평가

SOA Vulnerability Evaluation using Run-Time Dependency Measurement

김유경(Yukyong Kim)*, 도경구(Kyung-Goo Doh)**

초 록

현재까지 서비스지향 아키텍처(SOA) 보안은 개별 표준들과 솔루션에 의한 대응에 집중해 있을 뿐 전사적인 위험관리 차원의 통합적인 방법론은 부족한 실정이다. 따라서 SOA 보안 취약점 식별을 통한 다양한 보안 위협들에 대한 사전 분석과 대책 수립이 필요하다. 이를 위해 본 논문에서는 SOA 취약성을 정량적으로 분석하기 위한 SOA 동적 특성을 이용한 메트릭 기반의 취약성 평가 방법을 제안한다. SOA를 구성하는 서비스들 사이의 실행시간 종속성을 측정하여 서비스와 아키텍처 수준의 취약성을 평가한다. 서비스 사이의 실행시간 종속성은 한 서비스가 취약할 때 다른 서비스들이 얼마나 영향을 받게 되는지를 분석하기 위해 사용되는 중요한 특징이다. 한 서비스가 공격에 노출될 때 그 서비스에 종속된 서비스들도 역시 공격가능성이 높아진다. 따라서 실행시간 종속성은 SOA 아키텍처 수준의 취약성에 대한 지표로 활용할 수 있다.

ABSTRACT

Traditionally research in Service Oriented Architecture(SOA) security has focused primarily on exploiting standards and solutions separately. There exists no unified methodology for SOA security to manage risks at the enterprise level. It needs to analyze preliminarily security threats and to manage enterprise risks by identifying vulnerabilities of SOA. In this paper, we propose a metric-based vulnerability assessment method using dynamic properties of services in SOA. The method is to assess vulnerability at the architecture level as well as the service level by measuring run-time dependency between services. The run-time dependency between services is an important characteristic to understand which services are affected by a vulnerable service. All services which directly or indirectly depend on the vulnerable service are exposed to the risk. Thus run-time dependency is a good indicator of vulnerability of SOA.

키워드 : 서비스지향 아키텍처, 소프트웨어 보안취약성, 실행시간 종속성 측정
SOA(Service-Oriented Architecture), Software Vulnerability, Run-Time Dependency Measurement

---

본 연구는 교육과학기술부/한국연구재단 우수연구센터 육성사업의 지원으로 수행되었음(과제번호 2011-0000974).

* 한양대학교 ERICA캠퍼스 컴퓨터공학과 연구교수
** 교신저자, 한양대학교 ERICA캠퍼스 컴퓨터공학과 교수

2011년 04월 15일 접수, 2011년 05월 08일 심사완료 후 2011년 05월 18일 게재확정.

1. 서 론

비즈니스 요구를 실시간으로 충족시키기 위한 유연성과 외부 시스템과 다양한 수준의 협업을 가능하게 해주는 IT서비스의 개방성이 필수적으로 요구되면서, 기업들은 IT 인프라의 복잡성 및 유지비용을 최소화하고, 기업의 생산성과 유연성을 극대화 할 수 있는 새로운 IT전략을 필요로 하고 있다. SOA(Service-Oriented Architecture)는 소프트웨어 재활용과 단위 기능들 간의 상호호환성에 중심을 둔 소프트웨어 설계 방식으로 기업의 생산성과 유연성을 보장함으로써 비즈니스 목표를 효과적으로 달성하게 해주는 최적의 대안으로 기대를 모으고 있다.

SOA는 명확한 인터페이스 정의와 느슨한 연결(loosely coupled)을 통해 유연성을 증가시키며, 사용자 애플리케이션의 기능을 서비스 형식으로 전달하는 개방적 구조를 가지고 있다. 초기 SOA 개념이 도입될 때에는 효율성, 가용성, 재사용 및 비용효과성과 같은 기능적인 가치에 우위를 두었으며, 정보보호에 대한 고려가 거의 없었다. 암호화와 같은 기본적인 정보보호 기능도 제공되지 않아 보안에 매우 취약했으나, OASIS(Organization for the Advancement of Structured Information Standards)와 W3C(The World Wide Web Consortium) 등의 국제표준기구에 의해 지난 수년 동안 SOA 보안에 관련한 다양한 표준이 마련되었다. 대부분의 SOA 보안 표준은 XML 프레임워크 내에서 정의되어, 현재는 인증, 권한부여 등 기본적인 웹 서비스 보안기능을 제공하는 XML 기반 웹 서비스 보안 표준이 존재한다[1].

SOA 보안 표준 및 각종 웹 서비스 보안 솔루션들과 같은 노력들이 있었지만, 서비스 개발 및 배치(deployment)에 필요한 보안 가이드라인 부재로 인해, 구현상의 오류와 웹 취약성을 이용한 웹서비스에 대한 다양한 공격과 같은 보안 위험들이 상존하고 있다[2]. 현재까지 SOA 보안은 개별 표준들과 솔루션에 의한 대응에 집중해 있을 뿐 전사적인 위험관리 차원의 통합적인 방법론은 부족한 실정으로 SOA 보안취약성 식별을 통한 다양한 보안 위협들에 대한 사전 분석과 대책 수립이 필요하다. 이를 위해서는 SOA 취약성을 정량적으로 분석할 수 있는 취약성 평가 방안에 대한 연구가 선행되어야 한다. 본 논문에서는 SOA 취약성을 정량적으로 분석하기 위해 SOA 동적인 특성을 반영한 메트릭 기반의 취약성 평가 방법을 제안한다.

SOA는 약결합의 재사용 가능한 서비스들로 구성된다. 이는 상황 변화와 고객의 요구사항에 보다 쉽게 적응할 수 있는 매우 유동적이고 동적인 서비스 환경을 가능하게 한다. 본 논문의 목적은 유연성을 강조하고 잦은 변경을 허용하는 복잡한 SOA 시스템의 취약성을 효과적으로 분석하여 정량화 할 수 있는 메커니즘을 통해, 비즈니스의 위험 요인을 인식하고 관리할 수 있도록 하는 것이다.

본 논문에서는 SOA를 구성하는 서비스들 사이의 실행시간 종속성(즉, 동적 종속성)을 측정하여 서비스와 아키텍처 수준의 보안취약성을 평가한다. 서비스 사이의 실행시간 종속성은 한 서비스가 취약할 때 다른 서비스들이 얼마나 영향을 받게 되는지를 분석하기 위해 사용되는 중요한 특징이다. 한 서비스가 공격에 노출될 때 그 서비스의 공격가능성이

다른 서비스에 얼마나 영향을 주게 될지를 분석함으로써, 아키텍처 수준의 취약성을 평가할 수 있다. 이를 통해 취약성이 비즈니스에 미치는 영향을 예측하고 공격가능성이 높은 서비스들의 노출로 인한 성능 저하와 같은 위험 요인들을 관리 할 수 있을 것으로 기대된다.

본 논문의 구성은 다음과 같다. 먼저 제 2장에서는 SOA 취약성과 관련된 기존의 연구를 살펴보고, 제 3장에서 SOA 서비스의 동적 특성에 대해 기술한다. 취약성을 평가하기 위한 메트릭의 설계 원칙과 평가 방법이 제 4장에서 제시되며, 제 5장에서 제시된 방법에 대한 유효성 검증을 위한 실험 및 평가 결과를 제시한다. 마지막으로 6장에서 결론을 맺는다.

2. 관련 연구

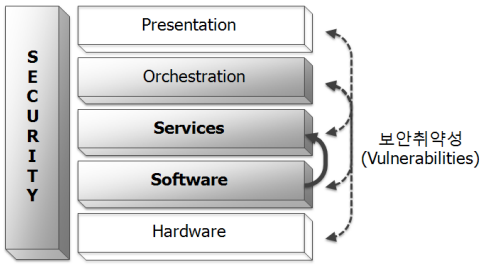
2.1 SOA 취약성

일반적으로 소프트웨어 시스템의 취약성은 시스템에 저장 또는 전송되는 정보의 기밀성(confidentiality), 무결성(integrity), 그리고 가용성(availability)을 위협할 수 있는 정보시스템의 설계 또는 구현상의 오류로 정의된다[3].

SOA는 보안에 대한 충분한 고려나 사전 보안성 평가가 이루어지지 않은 기술로, 유연성과 민첩성을 확보하기 위한 SOA 고유의 특성과 구성기술들의 특성으로 인한 다양한 보안 취약점들이 존재한다. OASIS에서 정의한 SOA 참조 모델과 아키텍처에 따른 실행환경은 가시성(visibility) 및 상호작용(interaction)과 같은 주요 개념을 가지며, SOA 취약성은 이런 환경적 요인에 기인할 수 있다. 약결합(loosely coupling) 메커니즘이나 XML, SOAP, 공개 표준, 공개 서비스, 분산 아키텍처 및 웹 기반 기술에 기인하는 취약성이 대표적이라고 할 수 있다. 이와 함께 감사(audit), 모니터링, 권한 분리 및 접근 통제가 부적절하게 이루어질 경우의 보안 취약성을 고려할 수 있다.

<그림 1>과 같이 SOA 취약성은 서비스 계층의 어디에도 존재할 수 있다[4, 5]. 이런 취약성들은 일단 공격이 발생하면, 범위를 포착하기 힘들면서 다양한 영향을 주게 되어 실질적으로 예상하는 것보다 훨씬 더 엄청난 결과를 가져올 수도 있다.

SOA에서 비즈니스 프로세스를 구성하는 행위(activities)는 더 이상 웹 애플리케이션 스크립트에 존재하는 단편적인 코드 섹션이 아니라 WSDL(Web Services Description Language) 명세와 SOAP(Simple Object Access Protocol) 인터페이스로 구현되는 서비스들이다. 따라서 소프트웨어 애플리케이션의 취약성에 기반을 둔 새로운 취약성이 발생하게 된다.



〈그림 1〉 SOA 계층과 보안취약성

SOA 서비스 취약성은 공격자가 비즈니스 프로세스의 활동이나 순서를 수정할 수 있게 만든다. 예를 들면, 공격자가 사전에 정의된 행위(action)의 내용과 다르게 실행되도록 SOAP

메시지를 변경하는 SOAP 액션 스프핑(action spoofing)이 있다[6]. 서비스들은 WSDL 명세를 제공하고 SOAP을 통해 의사소통하며, 하나의 서비스 자체가 완전히 구현된 함수(self-contained functions)이다. 소프트웨어 시스템의 계층 구조에 따라 점차적으로 잘 정의되고 관리되는 코드로 구현됨으로써 취약성들이 줄어든다고 해도, 취약성들은 여전히 기존 소프트웨어(legacy software)에 존재하게 된다.

본 논문에서는 SOA 서비스와 아키텍처 수준에서, 서비스들 사이의 동적 의존성을 분석하고 측정함으로써, 한 서비스의 공격가능성이 다른 서비스에 영향을 줄 수 있는 범위를 분석하고 위험 요인을 예측해 보는 서비스 수준에서의 취약성 평가 방법을 제안한다.

2.2 SOA 취약성 평가 연구

SOA 취약성은 SOA 개념이 보안보다는 서비스 가용성 및 효율성에 중점을 두고 발전이 되어왔기 때문에 보안 개념이나 취약성 평가와 관련된 연구는 전무한 실정이다. 최근들어 SOA 환경에서의 보안 표준과 보안 솔루션들이 많이 등장하고 있는 상황이지만, 실질적 구현 기술인 웹 서비스에 다양한 취약점들과 공격들이 존재하고 SOA 보안 표준들 간의 비호환성, 적절하지 않은 구현과 개방적 분산 환경에서 일관된 ID 정책의 부재로 인한 신뢰 관계의 어려움이 존재한다.

복잡도, 결합도 그리고 응집도(cohesion)와 같은 메트릭을 이용한 결함 예측은 소프트웨어 공학의 연구주제로 많이 다루어져 왔다. 그러나 SOA 서비스의 취약성을 설계, 구현 또는 환경적 오류에 기인하는 소프트웨어 결함(defect)으로 고려하고, 복잡도 및 결합도와 같은 메트릭을 이용하여 SOA 수준의 보안취약성을 평가하는 것은 매우 새로운 영역이다. 최근 코드 수준의 복잡도 메트릭을 이용하여 취약한 함수들(vulnerability-prone functions)을 찾아내려는 연구 결과가 발표되었다[7, 8]. 또한 [9, 10]에서 메트릭을 통해 취약성을 예측하고, 취약한 시스템 컴포넌트들을 식별하려는 시도가 있었고, [11]의 연구에서는 실험을 통해 서비스 결합도가 서비스 거부(DoS, Denial of Service) 공격에 영향을 준다는 가설을 실험적으로 입증하였다. [12]에서는 결합도와 소프트웨어 취약성을 평가하고, 실험을 통해 결합도가 일반적인 취약성에 의미 있는 평가 지표임을 보였다.

3. SOA 서비스 특성

소프트웨어 시스템이 취약하면, 무결성, 가용성, 기밀성에 영향을 주는 손상을 야기하는 공격에 노출된다. 시스템이 취약해질수록 보안성은 점점 더 감소하게 된다. 즉, 보안은 취약성에 반비례한다. 따라서 소프트웨어 보안을 소프트웨어에 존재하는 취약성을 통해 정량적으로 측정할 수 있는 외부 속성으로 볼 수 있다.

시스템을 구성하는 서비스들은 비즈니스 프로세스 작업을 수행하기 위해 결합되지만, 전통적인 시스템과 비교해볼 때, SOA는 약결합의 특징을 갖는다. 이것은 다른 요소들에 대한 영향 범위를 제한하게 만드는 것이다. 한 서비스에 의해 영향을 받는 서비스들 사이의 관계는 종속성을 의미한다. 만약 한 서비스가

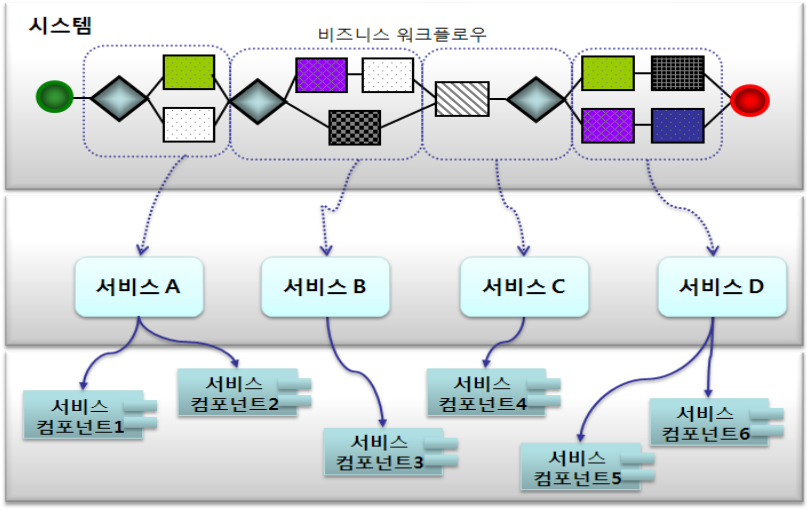
많은 다른 서비스들과 관계를 갖게 된다면, 다른 서비스들에게 종속되어 있다고 볼 수 있으며, 역으로 그 서비스에 종속된 서비스들이 많다는 의미도 성립한다. 따라서 서비스가 종속적인 관계를 많이 가질수록 서비스와 다른 서비스 사이의 결합은 좀 더 강해진다.

SOA가 갖는 약결합의 속성은 이해용이성, 유지보수성, 신뢰성, 테스트용이성 및 재사용성에 매우 큰 영향을 준다. 따라서 약결합의 특징을 잘 반영한 SOA 서비스의 동적 특성을 고려하여, 아키텍처 수준의 취약성 평가 메트릭을 설계해야 한다.

SOA 서비스의 실행시간 특성은 서비스의 상태비저장(stateless)이나 상태보존(stateful)의 특징, 서비스의 요구 및 제공 인터페이스 구성, 단방향 또는 양방향 상호작용 및 독립특성(self-containment)이 있다. 독립특성은 서비스가 자립적(stand-alone)이거나 간접적 in-direct dependent)임을 나타낸다. 만약 서비스 컴포넌트가 완전하게 독립적(stateless and self-contained)이라면, 이런 서비스 컴포넌트들은 매우 약한 결합의 형태를 갖는다. 이론적으로 SOA의 모든 서비스 컴포넌트는 독립적인 단위(stand-alone unit)이지만, 사실상 실제 많은 서비스 컴포넌트들은 다른 서비스 컴포넌트에 서비스를 요청하거나 다른 서비스 컴포넌트들과 정보를 공유하거나 또는 서비스 그 자체가 상태를 가지는 등 의존적이거나 종속되어 있다.

간접적으로 상태를 공유하고, 서비스 컴포넌트 자체가 상태를 유지함으로써 발생하는 서비스 종속성이 서비스들 사이의 결합의 원인이 된다. 또한 서비스가 자신만의 서비스를 제공하기 위해 다른 서비스가 제공하는 기능을 요청하는 것도 결합의 한 원인이 될 수 있다.

서비스는 비즈니스 프로세스 분석으로부터 유도되는 비즈니스 기능에 대응되며, 비즈니스 프로세스에 따라 서비스는 다른 서비스들과 조합되기도 한다. 따라서 비즈니스 프로세스에 따라 하나의 서비스는 다양한 서비스



〈그림 2〉 SOA 서비스 구조

컴포넌트로 구현될 수 있다.

<그림 2>와 같이 SOA 기반 소프트웨어 시스템은 서비스의 집합으로 구성된다. 시스템을 구성하는 각 서비스는 대응되는 기능을 수행하기 위한 지원 서비스들(supporting services)인 서비스 컴포넌트로 구현된다. 서비스들 사이의 이런 계층 구조에서, 최하위 계층의 단말들은 서비스 컴포넌트들이다. 이들 서비스 컴포넌트들은 자신을 지원하는 서비스 컴포넌트를 갖지 않는 단위 서비스(atomic service)가 된다.

취약성을 내포한 단위 서비스들의 조합으로 구현되는 서비스는 결과적으로 취약할 수밖에 없다. 이런 서비스 구조에서 서비스 및 아키텍처 수준의 취약성을 판단하기 위한 단위 서비스 및 조합 서비스들 사이의 종속성의 여부와 결합의 정도에 대한 정량적인 분석 방법이 필요하다.

소프트웨어 메트릭(metric)은 신뢰성, 성능 및 유지보수성(maintainability) 등을 포함하는 광범위한 품질 속성을 평가할 수 있도록 성공적으로 개발되어 왔지만, 소프트웨어 보안 분야에서는 여전히 미흡하다. 본 논문에서는 SOA 취약성 평가 메트릭을 정의하기 위해, 서비스의 동적 특징(properties) 및 보안 시스템을 설계하고 구현하기 위해 사용되는 보안 설계 원칙을 고려한다. 보안 설계 원칙은 단순(simplicity), 분리(separation), 그리고 제한(restriction)이라는 특성을 갖는다[10]. 본 논문에서는 분리와 제한이라는 보안 설계 원칙에 초점을 두고, 독립적인 단위로서 서비스가 다른 서비스와 의존성을 평가하여, 서비스 취약성을 평가하고자 한다.

4. 보안취약성 평가 메트릭의 설계

일반적으로 소프트웨어 메트릭은 소프트웨어 일부 또는 소프트웨어와 관련된 산출물의 특성(property)에 대한 측정(measurement)으로, 취약성 평가를 위한 보안 메트릭은 서비스의 정적 및 동적 상태를 이해하고 서비스의 종속성을 식별하기 위해 필요한 모든 정보들로부터 정의되어야 한다.

4.1 실행시간 서비스의 취약성

SOA와 웹 서비스 기술은 많은 이점을 제공하며 소프트웨어가 상호작용하는 방법에 큰 변화를 가져왔다. 서비스 요청자(service requester)와 서비스 제공자(service provider) 및 서비스 디렉토리(service directory)로 구성된 구조적 특성으로 인해 서비스들의 실행은 다양한 플랫폼의 다양한 시스템들 사이에서 메시지 기반의 상호작용으로 이루어진다. 특히 SOA가 웹 서비스로 구현될 때, 서비스들은 SOAP 메시지를 통해서 소통하게 된다.

SOA 실행 환경에서 애플리케이션과 서비스 컴포넌트는 일대일로 대응될 수 있으므로, 취약한 애플리케이션이 존재한다면, 대응되는 서비스 컴포넌트는 취약할 수밖에 없다. 또한 이 서비스 컴포넌트가 생성한 메시지를 통해 소통하는 서비스들은 전파(propagation) 효과를 통해 영향을 받게 된다. 따라서 메시지는 서비스들을 서로 연결하기 때문에 메시지가 갖는 영향력은 반드시 고려되어야 하는 특징이다. 특히 [13]의 SOA와 웹 서비스 취약

성 분류를 통해 알 수 있듯이, 삽입취약점(injection flaws)과 불안전한 통신(insecure communications)과 같은 메시지 기반 구조의 특징으로 인한 취약성 문제는 간과할 수 없다.

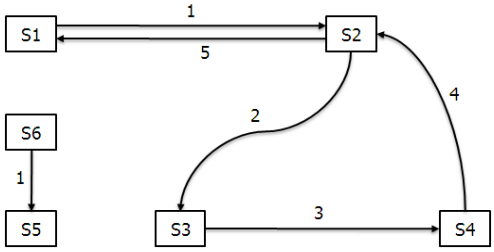
SOA의 서비스 계층에서 서비스들 사이의 영향력 전파는 한 서비스와 모든 종속된 서비스들 사이에서 발생하기 때문에, 취약성의 영향력도 이와 같다고 볼 수 있다. 특히 발신 메시지(outgoing messages)의 취약성으로 인한 영향력은 서비스의 무결성(integrity)이나 가용성(availability) 측면에 더욱 크게 작용하게 된다[14]. 따라서 취약한 서비스에 직접적으로 또는 간접적으로 종속된 모든 서비스들은 영향을 받게 되므로 반드시 식별되어 위험요인으로 관리되어야 한다. 이런 관점에서 SOA 서비스의 취약성을 평가하기 위해 서비스의 종속성을 판단하여, 영향 범위 내에 존재하는 서비스들을 파악하고자 한다.

4.2 서비스 취약성 평가

서비스 실행시간 종속성에 대한 평가는 서비스들 사이의 관계에 대한 정량적인 표현이다. 따라서 서비스와 서비스들 사이의 관계를 표현하는 그래프 기반으로 메트릭을 정의한다.

정의 1(서비스 관계 그래프) : SOA 시스템의 서비스와 서비스들 사이의 실행 시간 의존성을 표현하는 동적 관계 그래프 G = (V, E)는 서비스를 나타내는 그래프 노드들의 집합 V와 서비스들 사이의 실행 시간 의존성 관계를 나타내는 방향을 갖는 간선들의 집합 E로 구성된다.

- 간선의 방향은 서비스 요청자에서 서비스 제공자로 향하도록 한다.
- 모든 간선은 서비스 요청자가 제공자에게 서비스를 요청한 횟수를 나타내는 정수를 가중치로 갖는다.
- 정적 관계 그래프로 사용하는 경우, 간선의 가중치는 1로 일정하게 표현한다.



〈그림 3〉 서비스 관계 그래프

정의 2(의존성 메트릭) : 서비스들 사이의 의존관계를 평가하기 위한 서비스 의존성 메트릭 SD(Service Dependency)는 식 (1)과 같다.

$$SD = \frac{(W_e \times N_e) - \sum_{u,v \in V} P(u, v)}{(W_e - 1) \times N_e} \qquad (1)$$

SD는 서비스 관계 그래프에서 각 노드 쌍에 대해 얼마나 쉽게 도달할 수 있는지를 계산한다. 식 (1)에서 W_e는 모든 간선의 가중치의 합이며, N_e는 그래프에서 노드들 사이의 모든 가능한 관계를 의미하며, n개 노드를 가진 완전 그래프의 양방향 간선의 개수로 정의한다. 즉, $N_e = n \times (n-1)$이다. $P(u, v)$는 두 노드 u에서 v까지 가장 빠른 경로의 길이를 나타낸다. 만약 두 노드 사이에 경로가 존재하지 않을 경우, W_e로 계산한다.

<그림 3>의 그래프에 대해 SD의 계산은 다음과 같다. <그림 4>(a)는 <그림 3>에 대

	S1	S2	S3	S4	S5	S6
S1	0	1	3	6	16	16
S2	5	0	2	5	16	16
S3	12	7	0	3	16	16
S4	9	4	6	0	16	16
S5	16	16	16	16	0	16
S6	16	16	16	16	1	0

(a) 인접 행렬

$$n = 6,\ W_e = 16,\ N_e = 30$$
$$\sum_{i,j=1}^{6} P(S_i,\ S_j) = 336$$
$$SD = \frac{(16 \times 30) - 336}{(16-1) \times 30} = 0.32$$

(b) 메트릭 계산 결과

〈그림 4〉 메트릭 계산 예

한 인접행렬 표현으로 인접한 두 노드의 간선 가중치를 더한 값을 엔트리로 갖는다. 식 (1)에서 주어진 그래프에 존재하는 모든 두 노드 쌍 사이의 가장 빠른 경로의 길이를 나타내는 $\sum P(u, v)$는 인접행렬의 모든 원소의 총합으로 계산한다.

식 (1)에서 $0 < SD < 1$의 범위를 갖게 되며, 서비스 사이의 의존성이 높아질수록 SD 값은 1에 가까운 값을 갖게 되고 의존성이 낮을수록 SD 값은 0에 가까운 값을 갖게 된다.

정의 3(취약성 평가) : 서비스 사이의 의존성이 높아지면, 취약성도 높아지며, 보안은 약해지는 관계를 가진다. 다시 말하면, 의존성과 취약성은 보안에 반비례한다. 이 관점으로부터, 취약성 메트릭 DV(Degree of Vulnerability)는 다음과 같이 정의한다.

$$DV = 1 - \left(\frac{1}{SD}\right) \quad (2)$$

식 (2)에서 $0 < DV < 1$이고, DV의 값이 0에 가까울수록 낮은 의존성을 가지고 있으며 따라서 덜 취약할 수 있다는 의미이고, DV의 값이 1이면 높은 의존성을 가지므로 결과적으로 매우 취약할 수 있음을 의미한다. 취약성 평가 결과가 0과 1사이의 범위를 가지므로 좀더 이해하기 쉬워진다.

5. 실험 및 평가

5.1 안전성(Soundness) 증명

측정 이론에 기반을 둔 특성 기반 소프트웨어 공학 측정 프레임워크(property-based software engineering measurement framework)는 크기, 길이, 복잡도, 응집도 및 결합도와 같은 구조적 속성에 대한 중요한 측정 개념 및 속성에 대한 수학적 특성을 정의하고 있다[15]. 이들 특성은 일반적으로 메트릭의 안전성 검증을 위해 사용된다. 본 논문에서 제안한 서비스 의존성 메트릭의 안전성에 대한 이론적 검증을 위해 위의 5가지 결합도 특성을 만족함을 증명한다.

[Property 1] Non-negativity

Proof) 한 시스템이 n개 서비스로 구성되어 있고, 서비스들 사이의 의존관계가 하나 이상

존재한다면, $(W_e \times N_e)$는 모든 서비스 쌍들 사이에 의존관계가 존재하는 경우를 나타내므로, 항상 $(W_e \times N_e) > \sum_{u,v \in V} P(u, v)$이 된다. 항상 $(W_e \times N_e) - \sum_{u,v \in V} P(u, v) > 0$이 성립하게 되므로, $SD > 0$이다. 또한 모든 서비스가 완전하게 독립적인 단위로서, 의존관계가 존재하지 않는다면 $SD = 0$이다. 그러므로 $SD \geq 0$이 항상 성립한다.

[Property 2] Null value

Proof) 한 시스템이 n개 서비스로 구성되어 있고, 모든 서비스가 완전하게 독립적인 단위로서, 의존관계가 존재하지 않는다면 $SD = 0$이다. 그러므로 메트릭 값은 0을 허용한다.

[Property 3] Monotonicity : 임의의 두 시스템 S, S'에 대해, 시스템 S가 S'보다 서비스들 사이의 의존성이 낮다면, $SD(S) < SD(S')$이어야 한다.

Proof) 두 시스템 S, S'이 개 서비스로 구성되어 있고, 시스템 S가 S'보다 서비스들 사이의 의존성이 낮다면, $W_e(S) < W_e(S')$이고, $\sum_{u,v \in V(S)} P(u, v) < \sum_{u',v' \in V(S')} P(u', v')$이다. 그러면 부등식의 성질에 따라 $W_e(S) - \sum_{u',v' \in V(S')} P(u', v') < W_e(S') - \sum_{u',v' \in V(S')} P(u', v')$이므로, $W_e(S) - \sum_{u,v \in V(S)} P(u, v) < W_e(S) - - \sum_{u',v' \in V(S')} P(u', v') < W_e(S') - \sum_{u',v' \in V(S')} P(u', v')$이다. $N_e(S) = N_e(S')$이므로 $W_e(S) \times N_e(S) < W_e(S') \times N_e(S')$이고, 따라서 $SD(S) < SD(S')$가 성립한다.

[Property 4] Merging of Modules : 임의의 두 시스템 S1, S2을 결합한 시스템 S의 의존성은 S1와 S2보다 높지 않다. 즉, $SD(S) \leq SD(S1) + SD(S2)$이다.

Proof) 두 시스템 $S1 = \{s_i | 1 \leq i \leq n_1\}$, $S2 = \{s_j | 1 \leq j \leq n_2\}$에 대해, $S = S1 \cup S2$이라고 하자. 그러면, $n \leq n_1 + n_2$이고 $W_e(S) \leq W_e(S1) + W_e(S2)$이다.

$$
\begin{aligned}
SD(S) &= \frac{(W_e(S) \times N_e(S)) - \sum_{u,v \in V(S)} P(u,v)}{(W_e(S) - 1) \times N_e(S)} \\
&= \frac{(W_e(S) \times n \times (n-1)) - \sum_{u,v \in V(S)} P(u,v)}{(W_e(S) - 1) \times n \times (n-1)} \\
&\leq \frac{(W_e(S1) \times n_1 \times (n_1 - 1)) - \sum_{u,v \in V(S)} P(u,v)}{(W_e(S1) - 1) \times n_1 \times (n_1 - 1)} \\
&+ \frac{(W_e(S2) \times n_2 \times (n_2 - 1)) - \sum_{u,v \in V(S)} P(u,v)}{(W_e(S2) - 1) \times n_2 \times (n_2 - 1)} \\
&= \frac{(W_e(S1) \times N_e(S1)) - \sum_{u,v \in V(S)} P(u,v)}{(W_e(S1) - 1) \times N_e(S1)} \\
&+ \frac{(W_e(S2) \times N_e(S2)) - \sum_{u,v \in V(S)} P(u,v)}{(W_e(S2) - 1) \times N_e(S2)}
\end{aligned}
$$

$S = S1 \cup S2$이므로, S1과 S사이에 $\sum_{u',v' \in V(S1)} P(u', v') < \sum_{u,v \in V(S)} P(u, v)$가 성립한다. 또한 $\sum_{u',v' \in V(S2)} P(u', v') < \sum_{u,v \in V(S)} P(u, v)$도 성립한다. 따라서 임의의 양의 값 X에 대하여 $X - \sum_{u',v' \in V(S2)} P(u', v') > X - \sum_{u,v \in V(S)} P(u, v)$가 되므로, 다음이 성립한다.

$$
\begin{aligned}
SD(S) &\leq \frac{(W_e(S1) \times N_e(S1)) - \sum_{u,v \in V(S1)} P(u,v)}{(W_e(S1) - 1) \times N_e(S1)} \\
&+ \frac{(W_e(S2) \times N_e(S2)) - \sum_{u,v \in V(S2)} P(u,v)}{(W_e(S2) - 1) \times N_e(S2)} \\
&= SD(S1) + SD(S2)
\end{aligned}
$$

따라서 $S = S1 \cup S2$에 대해 $SD(S) \leq SD(S1) + SD(S2)$이다.

[Property 5] Disjoint Module Additivity : 임의의 두 시스템 S1, S2가 공통의 서비스를 갖지 않는 경우, 두 시스템을 결합한 시스템 S의 의존성은 S1, S2와 같아진다. 즉 $S1 \cap S2 = \varnothing$ 이고 $S = S1 \cup S2$에 대해 $SD(S) = SD(S1) + SD(S2)$이다.

Proof) 두 시스템 $S1 = \{s_i | 1 \le i \le n_1\}$, $S2 = \{s_j | 1 \le j \le n_2\}$에 대해, $S = S1 \cup S2$이라고 하자. 그러면, $S1 \cap S2 = \varnothing$이므로 $n = n_1 + n_2$이고 $W_e(S) = W_e(S1) + W_e(S2)$이다. 또한 최단경로의 합에 대해서도 $\sum_{u,v \in V(S)} P(u,v) = \sum_{u',v' \in V(S1)} P(u',v') + \sum_{u',v' \in V(S2)} P(u',v')$가 성립한다. 따라서 [Property 4]의 증명과 유사한 방법으로 $SD(S) = SD(S1) + SD(S2)$임을 알 수 있다.

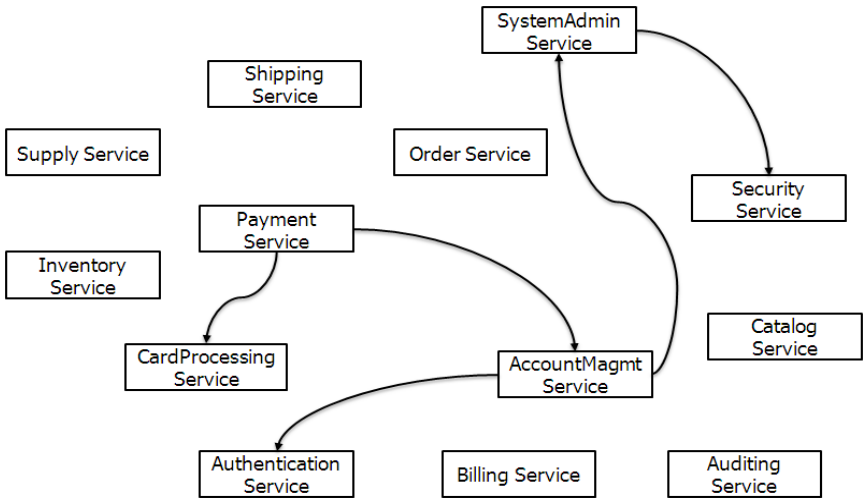
5.2 사례 연구

본 논문에서 제안한 취약성 메트릭을 온라인 쇼핑몰 애플리케이션을 구성하는 13개 서비스에 적용해 보았다. <그림 5>는 애플리케이션을 구성하는 13개 서비스의 구조를 단순화하여 보여주고 있다. 각 서비스들은 지원서비스들로 구현된 조합서비스들로서, <그림 6>에서와 같이 각 서비스를 구성하는 서비스 컴포넌트들로 구현된다.

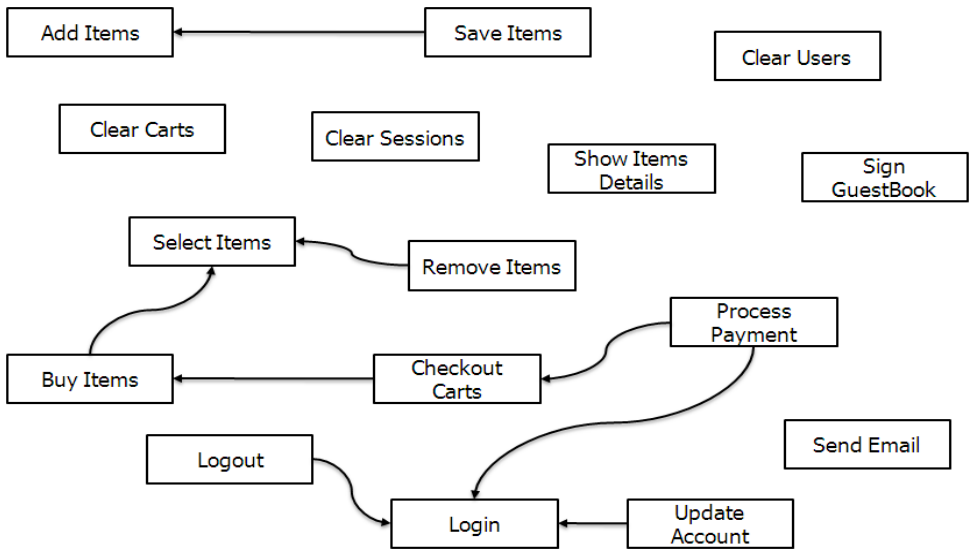
위의 서비스들에 대해, 제안된 서비스 의존성 메트릭 SD 값을 계산한다. 계산된 결과 값은 <표 1>와 같다.

URL 점핑 공격이나 DOS 공격 가능성을 내포한 취약성과 서비스 의존성 사이에는 60~80% 범위의 상관계수 값을 갖는 높은 상관관계를 보인다고 알려져 있다[12]. 따라서 메트릭 값이 1에 가까운 서비스들은 의존성이 높고 결과적으로 취약하여 공격가능성을 갖는 서비스들이므로 주의 깊은 관리가 필요하게 된다.

<표 1>에서 이들 서비스의 SD 값의 평균은 0.37이며, 따라서 SD 값들이 평균을 상회하는 서비스들은 다른 서비스에 영향을 많이 받게 되므로 위험성을 내포하고 있다는 의미를



〈그림 5〉 온라인 쇼핑몰의 서비스 구성



〈그림 6〉 Order Service와 Payment Service를 구현하는 서비스 컴포넌트

갖는다. SD 값에 따라 위험 관리 대상에 대한 우선순위를 정할 수 있으며, 이를 통해 서비스 취약성에 대한 영향 범위를 파악할 수 있다.

〈표 1〉 온라인 쇼핑몰 서비스의 SD 계산 결과

서비스 이름	SD 값
AccountMagmt Service	0.71
Auditing Service	0.15
Authentication Service	0.39
Billing Service	0.2
CardProcessing Service	0.47
Catalog Service	0.2
Inventory Service	0.15
Order Service	0.4
Payment Service	0.67
Security Service	0.4
Shipping Service	0.16
Supply Service	0.16
SystemAdmin Service	0.69

이들 메트릭에 의한 측정 결과는 SOA 소프트웨어 아키텍트와 개발자들 모두에게 매우 큰 의미가 있다. 서비스 설계와 아키텍처 수준에서 보안 위험을 최소화하기 위해 주의를 기울여야 하는 대상에 대한 우선순위를 제공하기 때문이다.

6. 결론 및 향후 연구과제

본 논문에서는 SOA 서비스와 아키텍처 수준에서, 서비스들 사이의 동적 의존성을 분석하고 측정함으로써, 한 서비스의 공격가능성이 다른 서비스에 얼마나 영향을 미치는지에 대한 관점으로 SOA의 취약성을 평가하였다. SOA 보안 표준 및 각종 웹 서비스 보안 솔루션들이 존재하고 있지만, 서비스 개발 단계 전반에 걸친 보안 가이드라인의 부재로 인해, 서비스 및 서비스 아키텍처에 대한 다양한

보안 위험들이 있다.

이에 따라 SOA 취약점 식별을 통한 다양한 보안 위협들에 대한 사전 분석과 대책 수립에 대한 정량적인 접근방법이 필요하다. 이를 위해 본 논문에서는 SOA 취약성을 정량적으로 분석하기 위해 SOA 동적 특성을 이용한 메트릭 기반의 취약성 평가 방법을 제안하였다. 제안된 메트릭에 대한 이론적 검증과 함께 소규모 온라인 쇼핑몰 애플리케이션을 대상으로 메트릭을 적용해 보았다.

한 서비스가 공격에 노출될 때 그 서비스의 공격가능성이 다른 서비스에 얼마나 영향을 주게 될지를 분석함으로써, 아키텍처 수준의 취약성을 평가할 수 있다. 이를 통해 취약성이 비즈니스에 미치는 영향을 예측하고 공격가능성이 높은 서비스들의 노출로 인한 성능 저하와 같은 위험 요인들을 관리 할 수 있을 것으로 기대된다.

SOA와 웹 서비스에 대한 취약성은 매우 다양하기 때문에 실질적으로 광범위하게 적용할 수 있는 취약성 평가가 이루어지기는 어렵다. 따라서 제안된 메트릭을 수정 보완하여 SOA 취약성 항목에 따라 보안 특성을 반영한 구체적인 평가 방법으로 확장해야 할 것이다.

참 고 문 헌

[1] Periorellis, P., Securing Web Services : Practical usage of standards and specifications, IgiGlobal, 2007.

[2] Lim, J. I., “Security in SOA,” www.webkoreaforum.or.kr, 2007.

[3] Turner, D. et al., “Symantec global internet security threat report : Trends for july to december 2007,” Symantec, Tech. Rep., 2008.

[4] Krafzig, D., Banke, K., and Slama, D., Enterprise SOA, Prentice-Hall, 2004.

[5] Arsanjani, A., Zhang, L. J., Ellis, M., Allam, A. and Channabasavaiah, K., “IBM Developer Works : Design an SOA solution using a Reference Architecture,” IBM Developer Works, 2007.

[6] Gruschka, N., Jensen, M., Herkenhöner, R., and Luttenberger, N., “SOA and Web Services : New technologies, new standards-new attacks,” Proceedings of the 5th IEEE European Conference on Web Services(ECOWS), pp. 35-44, 2007.

[7] Yu, W. D., Aravind, D. and Supthaweesuk, P., “Software vulnerability analysis for web services software systems,” Proceedings of the 11th IEEE Symposium on Computers and Communications(ISCC), pp. 740-748, 2006.

[8] Shin, Y. and Williams, L., “An Empirical Model to Predict Security Vulnerabilities Using Code Complexity Metrics,” Proceedings of the 2nd ACM-IEEE International Symposiumon Empirical Software Engineering and Measurement, pp. 315-317, 2008.

[9] Jones, J. R., “Estimating Software Vul-

nerability," IEEE Security and Privacy, Vol. 5, No. 4, pp. 28-32, 2007.

[10] Neuhaus, S., Zimmermann, T., Holler, C. and Zeller, A., "Predicting Vulnerable Software Components," Proceedings of ACM conference on CCCS '07, pp. 529-540, 2007.

[11] Liu, M. Y. and Traore, I., "Empirical Relationships between attack-ability and coupling : case study for DOS," Proceedings of ACM SIGPLAN Workshop on PLAS '06, pp. 57-64, 2006.

[12] Ayanam, V. S., "Software Security Vulnerability vs Software Coupling : A Study with Empirical Evidence," M. S. Thesis, The School of Computing and Software Engineering, Southern Polytechnic State University, 2009.

[13] SNAC white paper, "Service Oriented Architecture Security Vulnerabilities-Web Services," http://www.nsa.gov/snac.

[14] Lowis, L., "Towards Automated Risk Identification in Service-Oriented Architectures," Proceedings of Multikonferenz Wirtschaftsinformatik, (MKWI '08), pp. 1149-1158, 2008.

[15] Briand, L. C., Morasca, S. and Basili, V. R., "Property Based Software Engineering Measurement," IEEE Transactions on Software Engineering, Vol. 22, No. 1, pp. 68-86, 1996.

저 자 소 개

김유경	(E-mail : yukyong@hanyang.ac.kr)
1991년	숙명여자대학교 수학과 (학사)
1994년	숙명여자대학교 전산학과 (석사)
2001년	숙명여자대학교 컴퓨터과학과 (박사)
2001년~2005년	숙명여자대학교 정보과학부 컴퓨터과학전공 초빙교수
2005년~2006년	University of California Davis, 박사후연구원
2006년~현재	한양대학교 ERICA 캠퍼스 컴퓨터공학과 연구교수
관심분야	소프트웨어 품질평가, 웹 서비스 신뢰성 평가, SOA 모델링

도경구	(E-mail : doh@hanyang.ac.kr)
1980년	한양대학교 산업공학과 (학사)
1987년	미국 아이오와주립대학교 전산학과 (석사)
1992년	미국 캔사스주립대학교 전산학과 (박사)
1993년~1995년	일본 Aizu 대학, 교수
2000년~2001년	스마트카드연구소 대표이사
2005년~2006년	미국 캘리포니아대학 데이비스캠퍼스 방문교수
1995년~현재	한양대학교 ERICA 캠퍼스 컴퓨터공학과 교수
관심분야	프로그래밍언어, 프로그램 분석, SW 보안, 프로그래밍언어 의미표기법