정보시스템에서 시스템연동에 따른 취약점 검증방법

정재훈*, 최명길**,
*중앙대학교 경영대학
**중앙대학교 사회과학대학
e-mail:selpine@naver.com

System Interconnection to Verification for Security Vulnerability based on Information System

Jae-Hun Jeong*, Myung-Gil Choi**
*College of Business Adminstration, Chung-Ang University
**College of Social Sciences, Chung-Ang University

요 약

본 논문에서는 정보시스템에서 시스템이 연동될 때 일어나는 잔여 취약점을 분석하고, 그 검증방법에 대해 연구하였다. 광범위하고 정밀한 정보 침해 공격은 정보시스템으로 하여금 시스템끼리 연동하게 하였고, 미리 취약점 검증을 받은 시스템이라 할지라도 서로 다른 시스템이 연동되게 되면 잔여 취약점이 발생할 수 있다. 따라서 안전한 정보자산의 사용을 위해서는 정보보호제품과 연동된 정보시스템에 대한 보안성을 평가할 필요할 필요가 있다. 이와 같은 보안성을 평가하기 위해, 시스템 연동 시 어떠한 일이 발생하는지 분석하고 예상되는 잔여 취약점을 추출한다. 그리고 그것을 검증하는 방법에 대해 알아보기로 한다.

1. 서론

광범위한 정보자산의 침해로 인해서 조직은 다양한 형태의 정보보호기능이 결합된 형태의 정보시스템을 도입하고 있다. 더욱이 정보 자산에 대한 정교한 공격이 증가함에 따라 조직은 일반 정보시스템과 정보보호제품을 연동된 형태로 정보시스템을 사용하고 있다.

그러나 각종 정보보호제품의 보안성 검토 및 보안성 평가를 거친 정보보호제품이 기존의 정보시스템과 연동이 되면 기존에 존재하지 않았던 새로운 위험과 취약성이 발생될 수 있다. 최근에 급속히 증가하고 있는 정보보호시스템과 정보시스템간의 연동은 새로운 형태의 위험과 취약성을 발생시켜 조직의 안전한 정보자산의 운영을 위험하고 있는 실정이다. 따라서 조직의 안전한 운영과 정보자산의 보호를 위해서는 정보시스템과 정보보호제품의 연동에 따른 취약성을 검증해야 한다. 시스템 연동은 2개 이상의 정보시스템이 데이터와 정보자원을 공유하기 위한 연동을 의미한다.

정보보호제품과 정보시스템의 연동에 따른 취약성 및 위험 평가에 필요한 요소 기술은 다음과 같다. 첫째 상이한 보안환경을 가정하고 개발된 정보시스템의 취약성을 평가할 수 있는 취약성 평가 기술, 둘째, 정보시스템과 정보보호제품의 연동 환경에서 발생하는 위험 및 취약성을 시험할 수 있는 연동 환경 평가 기술, 셋째, 연동에 내재된 위험인 잔존 위험을 평가할 수 있는 보안 관리 기술 등이다.

2. 연동 시스템의 잔존 취약성 평가

연동 환경을 분류하고, 연동 환경에 따른 취약성을 점검할 수 있는 점검리스트를 개발하고, 점검리스트를 연동 시스템의 취약성을 시험해야 한다. 따라서 본 장은 연동 환경에 따라 취약성 및 위험을 검토하는 방안을 제안한다.

2.1. 연동 시스템 환경 분석

해당 시스템에 대한 일반적인 보안요구사항의 정리는 연동 환경 분석에 유익하다. 보안 계획 수립시 먼저 완화해야 하는 위협을 정의해야 한다. 시스템에 대하여 검사를 하여, 특정 위협을 식별하고 기존 보안 대책의 효율성을 결정하는 위험을 평가해야 한다.

연동 환경을 분석은 취약성 및 위험 평가를 위해서 필요하다. 본 연구는 연동 환경을 독립환경

(standalone), 관리환경(managed environment), 제한된 환경(customized environment) 등으로 구분한다.

2.1.1 독립 환경의 취약성 분석 및 위험관리

독립 환경은 소규모의 환경이다. 동 환경은 다양한 규모의 환경과 컴퓨팅 자원을 포함한다. 독립 환경에서 연동시스템의 사용자는 보안과 관련된 지식이 가장 낮은 사용자이며, 시스템의 보안을 침해할 수 있다. 독립 환경은 대체로 가장 작은 보안 등급을 지닌다. 때문에 보안 지식이 낮은 개인 사용자를 위해 보다 보안 기능에 집중할 필요가 있다. 따라서 독립 환경에서 발생할 수 있는 취약성을 점검할 수 있는 시험 방법은 상당히 간단해야 한다. 아직은 많은 독립 환경 사용자들이 보안 정책을 인식하지 못하고 있지만, 네트워크를 통해 다른 시스템을 위협할 수 있는 만큼, 적절한 보안 정책을 가져야 한다.

2.1.2 관리 환경의 취약성 분석 및 위험관리

관리 환경은 전형적으로 대체로 중앙에서 집중 관리되는 대규모의 조직 환경이다. 관리 환경은 일반적으로 대규모의 조직이기 때문에, 대부분의 제어가 중앙에서 이루어지고 보안 시스템 또한 중앙집중식이다. 이 때문에 관리 네트워크에 대한 보안 위협은 관리 환경에 상당한 영향을 미칠 수 있다. 관리 환경은 대규모임으로 외부의 위협뿐 아니라 내부의 위협도 존재할 가능성이 독립 환경에 비해 크다.

2.1.3 독립 환경의 취약성 분석 및 위험관리

제한된 보안환경은 매우 제한적이고, 보안성이 높은 수정된 환경을 의미한다. 제한된 보안환경은 매우 위험한 위협과 파급효과가 높은 시스템이 사용되는 환경을 의미한다. 제한된 보안환경은 민감하고 핵심정인 정보를 다고 있지만 외부의 요청에 응답해야 하므로 시스템자체가 위협될 수 있고 내부의 사용자가 위협될 수도 있다. 전형적인 제한된 보안환경은 외부의 요청에 응답해야 하는 웹시스템, 전자메일, DNS 서버, 등이며, 민감한 정보를 내포한 시스템이나 조직의 핵심적인 정보를 제공하는 시스템 등을 포함한다.

2.2. 취약성 식별 기술

취약성 분석은 취약성을 평가할 수 있도록 취약성을 분류하고, 취약성의 위협 수준을 척도를 사용하여 분류한다. 취약성 평가는 분류된 취약성을 위협 수준에 따라서 취약성 수준을 정하고, 측정하는 과정이다. 취약성분석 및 평가 연구는 정보보호를 수행하는 조직의 특성별로 매우 활발하나, 자산, 위협, 취약성을 취약성의 구성요소로 보고, 각 구성 요소의 수준을 측정하고, 각 구성요소의 상관분석하여 수준화하여, 취약성을 평가한다.

기존 취약성 분석 및 평가 방법은 취약성의 구성요소를 정의하는 방법과 취약성 평가에 대한 정성적 및 정량적 방법 등으로 분류할 수 있다.

[표 1] 위험분석 프로세스의 분류

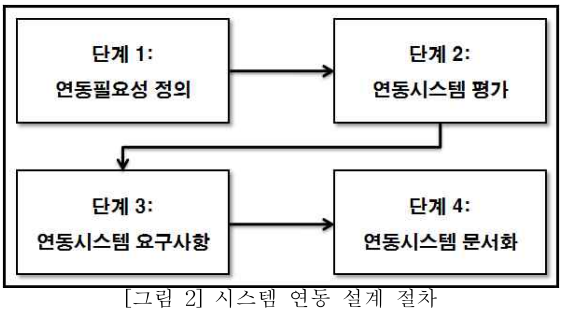
모델형태	방법
자산-취약성 기반	ISO/IEC TR 13335-3
자산-취약성-위협 기반	BDSS, HWAK
자산-위협-취약성 기반	OCTAVE, NIST SP-800-30, CRAMM, BDSS, CISSP
자산-위협 기반	CSE
위협-취약성 기반	SSE-CMM, CAO1, GAO3, 에너지성-SRAG
위협 기반	GAO, 법무성-SRAG
기타	ISO/IEC TR 13335-1, Open Framework

[표 1]은 위험 구성 요소를 자산, 취약성, 위협으로 보고, 위험 구성 요소에 대한 구성 방법과 평가 방법에 따라 위험분석 및 평가방법을 분류하고 있다.

3. 시스템 연동 기술 연구

3.1. 시스템 연동 설계

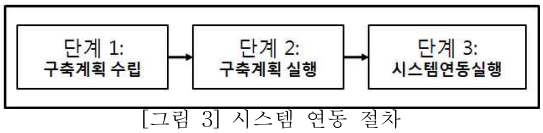
불완전한 시스템 연동 설계는 연결된 시스템과 보관된 자료 등이 위험에 노출될 수 있다. 연동된 시스템 중 일부 시스템의 위험 노출은 전체 시스템의 위험 노출로 이어진다. 그러므로 계획 단계에서 모든 위험을 고려한 설계가 필요하다. [그림 2]는 시스템 연동 설계 절차를 나타낸다. 정보보호제품과 정보시스템의 연동은 연동전의 정보보호제품과 정보시스템에 많은 변화를 일으킨다. 따라서 연동 전에 수행한 보안 평가를 재실시 해야 한다. 시스템 연동 설계는 연동 필요성 정의 단계, 연동시스템 평가 단계, 연동시스템 요구사항 수립 단계, 연동시스템 문서화 단계 등 4단계로 구성된다.



[그림 2] 시스템 연동 설계 절차

3.2. 시스템 연동 구축 및 유지

[그림3]은 상이한 보안환경을 가정한 시스템 연동 계획을 실행하는 절차이다. 시스템 연동 계획팀은 정보보호제품과 정보시스템의 연동 및 연동된 시스템의 보안성 확보를 위해서 시스템 연동 구축계획을 수립한다. 구축 계획이 수립된 뒤에는 개발팀은 구축계획을 검토하고 수행한다. 구축계획서는 시스템 연동의 상세한 절차를 설명해야 한다. 그리고 보안대책의 부재나 부적절한 구성은 해당 시스템을 무단 접근에 노출시킬 수 있다. 따라서 적절한 보안 대책을 구성해야 한다.



[그림 3] 시스템 연동 절차

4. 시스템 연동에 따른 API 취약성분석

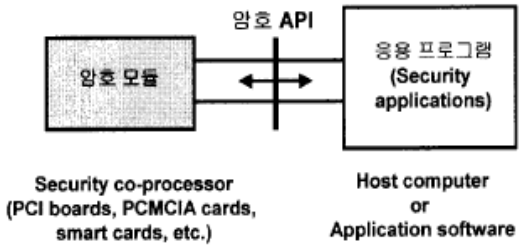
본 절은 시스템 연동의 핵심 요소인 API 구현상을 취약성을 분석하여, API를 통해 연동되는 정보시스템의 연동 시스템을 분석한다.

4.1. 잠재적인 안전성 위협요소

암호 모듈의 구현상의 안전성에 대한 고려사항은 알고리즘 수준의 구현 안전성과 API 수준의 구현 안전성으로 나눌 수 있다. 후자는 알고리즘 자체보다는 알고리즘 사용자가 API 설계 및 구현 API의 불법적인 사용이나 비정상적인 동작과 관련된 안전성이다. 암호 알고리즘은 컴퓨터에서 소프트웨어로 구현하는 경우, 임베디드 프로세스나 암호칩을 이용한 하드웨어 모듈로 구현하는 경우 구현 안전성에 대한 고려사항이 상이하다.

4.1.1 하드웨어 구현시의 취약성

HMAC을 안전한 하드웨어로 구현하는 경우 암호 연산이 안전한 암호칩이나 암호 코프로세서(crypto co-processor)에서 이루어진다. [그림 4]는 비밀키를 관리하고, 이용하는 암호 연산이 이루어지는 하드웨어 모듈을 블랙박스로 간주하고, 공격을 하는 경우이다. Cryptoki나 CryptoAPI, CDISA 등과 같은 범용 상위수준의 암호API는 하위 수준의 암호 라이브러리를 다양한 종류의 소프트웨어 모듈이나 하드웨어 모듈을 붙여서 사용할 수 있는 구조로 되어 있다. 따라서 암호API를 이용하는 어플리케이션 개발자는 하부 모듈의 구현 방식을 고려하지 않아도 된다.



[그림 4] 하드웨어 암호 모듈과 API

하드웨어 암호모듈의 API 수준의 공격이 존재함으로 HMAC의 구현 안전성을 위해 다른 암호 알고리즘에 보다 고려해야 할 부분이 존재한다. 대부분의 MAC 알고리즘은 메시지를 부분적으로 처리할 수 있도록 Init-Update-Final 구조의 API 함수로 구현된다. HMAC API은 HMAC_Init(), HMAC_Update(), HMAC_Final()과 같은 형태의 API 함수로 구현된다. API 함수구조는 연속적인 HMAC API 함수를 호출할 때, 각종 중간 계산값이 들어있는 파라미터를 외부와 주고 받는다. 특히 파라미터 중 연쇄변수는 비밀키와 동일한 정도의 안전성을 보장해야 한다.

4.1.2 소프트웨어 구현시의 취약성

소프트웨어로 구현하는 경우 비밀키나 관련 중간값을 안전하게 보관하는 것은 근본적으로 불가능하다. 그러나 가능한 취약성을 줄이고 공격의 난이도를 높일 수 있는 다양한 방법들이 제안되고 있다.

암호 알고리즘의 API 구현시 가능한 비밀키와 관련된 데이터 구조는 API 함수에서 내부적으로 안전하게 관리해 주는 것이 바람직하다. 저수준의 암호 라이브러리는 오브젝트 관리 함수를 지원하지 않는 경우가 많아 사용자 대부분이 안전한 관리를 책임져야 함으로 주의해서 사용해야 한다. 암호학적 기반이 없는 일반 응용 프로그래머는 가능한 상위 수준의 API를 사용하는 것이 유용하다.

API 수준의 취약성은 노드 운영체제 커널내에 실행 환경의 일부분이 삽입되는 경우 다른 많은 예외 상황을 발생한다. 대부분 알려진 취약점은 시스템 설정 재구성(reconfigure), 갱신(update), 보안 패치(patch) 등을 통해서 제거될 수 있지만, 알려지지 않은 취약점은 주기적인 취약성 검사나 비정상적인(abnormal) 행위를 접근금지 함으로 제거될 수 있다.

[그림 5]는 취약성 분석을 위한 AA 캡슐에서 사용가능한 API의 대표적인 코드를 나타낸다.

```
public byte[]rt_eeVer()
{
Flow f= Flow.currentFlow();
Assert.assert(this == f.getObject(), "node method being
called in correct flow");
String eeVer = "2.0.2";
return eeVer.getBytes();
}
public byte[]       rt_osVer()
{
Flow f = Flow.currentFlow();
Assert.assert(this == f.getObject(), "node method being
called in correct flow");

String str = System.getProperty("os.version",".");
return str.getBytes();
}

public byte[] rt_javaVer()
{
Flow f = Flow.currentFlow();
Assert.assert(this == f.getObject(), "node method being
called in correct flow");

String str = System.getProperty("java.version",".");
return str.getBytes();
}
```

[그림 5] 취약성 분석을 위한 API

5. 결론

기존 정보보호시스템의 보안성 및 안전성을 강화하기 위해서 정보보호시스템에 대한 다양한 평가제도 및 평가기술이 존재하고 있다. 향후 정보자산에 대한 침해는 더욱 심해질 것으로 보인다. 따라서 현존하는 정보시스템은 정보보호제품과 결합되어 나타날 것이다. 특히 현재의 정보시스템과 정보보호시스템 간의 구분이 모호해지고 있다. 정보보호시스템과 정보시스템과의 구분이 모호해지고 있다는 의미는 기존의 정보보호제품이 정보시스템의 기능의 일부분으로 흡수되고 있다는 증거이다. 정보보호제품의 정보시스템 연동에 따른 취약성 및 위험평가는 국내외적으로 연구가 거의 이루어지고 있지 않은 단계이다. 특히 선진국은 정보보호평가 등과 관련된 기술은 국외에 공개를 거의 하지 않고 있다. 그러나 우리나라의 경우 정보보호제품기술이 발전되고 있는 추세이며, 선진국과 기술적이 차이가 매우 작은 실정이다.

본 연구는 정보보호제품과 정보시스템 연동에 대한 보안 평가·인증에 대한 체계화를 위해 필요한 개념, 기술 개발 접근법, 보안 평가·인증 체계, 방법 등을 살펴보았다. 본 연구에서 제기된 내용을 참조하여 향후 시스템 연동에 대한 평가·인증 연구 및 평가·인증 시행에 기여할 수 있을 것이다.

참고문헌

[1] Baskerville, R., "Information System Security Design Method: Implications for Information Systems Development", ACM Computing Surveys, Vol. 25, No.4, December 1993.

[2] CC, “Common Criteria for Information Technology Security Evaluation”, Part1~Part3, Version 2.1, CCIMB-99-031, August 1999.

[3] Christopher King. “Extranet Access Control Issues,” in Harold F. Tipton and Micki Krause, ed., Information Security Management Handbook. Vol. 2, New York: Auerbach, 2000

[4] CSE, Guide to Certification and Accreditation of Information Technology Systems, Government of Canada, Communications Security Establishment, 1996a.

[5] CSE, Guide to Security Risk Management for IT Systems, Government of Canada, Communications Security Establishment, 1996b.

[6] Garry Stoneburner, CS2-Protection Profile Guidance for Near-Term COTS, NIST, July 1998.

[7] Gilbert, I. A., "Risk Analysis: Concepts and tools", Datapro Reports on Information Security, Risk Analysis, September 1991.

[8] Lemieux, J., "The Evaluation of Secure Information Systems," 1996,

(http://csrc.lse.ac.uk/Academic_Papers/Evaluation_Secure_IS.htm)

[9] NCSC, Introduction to Certification and Accreditation, NCSC-TG-029, National Computer Security Center, Jan. 1994.

[10] NIST, Federal Information Processing Standards (FIPS) 186-2, Digital Signature Standard (DSS). January 2000.

[11] NIST, "Guide for the Security Certification and Accreditation of Federal Information Systems", NIST Special Publication 800-37, May 2004.

[12] NIST, "Minimum Security Controls for Federal Information Technology Systems", NIST Special Publication 800-53, December 2006.

[13] NIST, "Techniques and Procedures for the Verification of Security Controls in Federal Information Technology Systems", NIST Special Publication 800-53A, December 2006.

[14] NIST, "U.S. Department of Justice Simplified Risk Analysis Guidelines", NISTIR 4387, August 1990

[15] Otwell, K. and Aldridge, B., "The Role of Vulnerability in Risk Management", Computer Security Journal, Vol.7, No.1, 1989