Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.27, NO.4, Aug. 2017 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2017.27.4.885

금융기관 공개용 홈페이지 취약점 분석평가에 대한 효율적인 대처방안

박 현 진,† 김 인 석‡
고려대학교 정보보호대학원

Effective Countermeasures against Vulnerability Assessment for the Public Website of Financial Institution

Hyun-jin Park,† In-seok Kim‡
Korea University

요 약

급변하는 IT환경만큼 다양한 형태의 외부 침입에 의한 보안이슈가 발생하고 있다. 웹 어플리케이션에 대한 취약점을 이용한 공격은 갈수록 늘어나고 있고, 각 기업은 취약점 발생의 원인을 찾아서 조치하여 외부 침입을 막고 자사의 시스템 및 주요 정보를 보호하기 위해 노력하고 있다. 특별히 감독규정에 의하여 각 금융기관 및 전자금융업자는 6개월에 한번 씩 공개용 홈페이지에 대한 취약점 분석평가를 실시하고 그 결과를 금융위원회에 보고하도록 되어있다. 본 연구에서는 감독규정에서 정의하는 웹 취약점 항목을 기준으로 실제 금융기관의 점검 사례를 바탕으로 발생빈도가 높은 항목을 분석하고 이에 대한 효율적인 대처방안과 발생을 미리 예방할 수 있는 방안을 제시하고자 한다.

ABSTRACT

Security issues arise due to various types of external intrusions as much as the rapidly changing IT environment. Attacks using vulnerabilities in web applications are increasing, and companies are trying to find the cause of the vulnerability, prevent external intrusion, and protect their systems and important information. Especially, according to the Supervision Regulation, each financial institution and electronic financial service provider shall perform vulnerability analysis evaluation for the website for disclosure once every six months and report the result to the Financial Services Commission. In this study, based on the Web vulnerability items defined in the Supervision Regulation, based on the inspection cases of actual financial institution, we analyze the most frequently occurring items and propose effective countermeasures against them and ways to prevent them from occurring in advance.

Keywords: Vulnerability Assessment, Web application, Secure coding

I. 서 론

인터넷 기반의 정보통신 기술 성장으로 금융, 통신, 전자상거래 등 다양한 분야에서 중요한 수단으로 진화해 왔고, 이에 따라 새로운 위협도 급속도로 증가했다. 특히 2016년은 대규모 보안사고의 해로 기록될 만큼 많은 보안 사고가 일어났다. 공개된 취약점 수도 2010년 이래 처음으로 1만 건을 넘었으며, 이중 웹 애플리케이션 취약점이 22%를 차지했다. 대다수가 취약한 시스템 공격에 쓰일 수 있는 Cross-Site Scripting, SQL Injection 취약점이다[1].

Received(06. 01. 2017), Modified(07. 19. 2017), Accepted(07. 27. 2017)
† 주저자, hyunjin.park@kbfg.com
‡ 교신저자, iskim11@korea.ac.kr(Corresponding author)

웹 서비스는 일반적으로 인터넷을 이용하면 어디서나 접속할 수 있으며, 취약점이 노출되어 공격당할 경우 중요 정보가 노출되거나 시스템이 비정상적으로 작동하게 됨에 따라 각 기업에서는 취약점을 찾아 보완하는 것이 중요한 과제로 볼 수 있다. 이에 기업들은 자사의 보안성을 유지하기 위하여 많은 비용을 들여 취약점을 찾아 조치를 취하고 있으며, 특히 금융기관들은 관련 법규에 따라 주기적으로 취약점 분석평가를 실시한 뒤 보고하고 있다.

이렇게 발견된 취약점들은 한꺼번에 모두 조치 하는데는 많은 시간과 리스크가 수반되기 때문에 가장 위험하다고 판단되는 취약점부터 우선 단계적으로 보완하는 것이 바람직한 방향일 것이다[2].

본 연구에서는 금융기관에서 실시하고 있는 공개용 홈페이지 취약점 분석평가의 현황을 분석해 보고 발생한 취약점에 대한 대처방안과 이러한 취약점이 발생하지 않도록 사전에 예방할 수 있는 방안을 제시하고자 한다.

II. 관련 연구

웹 취약점에 대한 연구사례와 금융회사 및 전자금융업자가 정기적으로 실시하고 있는 공개용 홈페이지 취약점 분석평가에 대하여 관련 법규, 취약점 항목 및 관련연구에 대하여 알아본다.

2.1 관련 연구

변으뜸은 취약점 항목에 대한 위험을 분석하기 위해 웹 취약성 분석 평가를 위한 스코어링 방법을 제시하였고[2], 김정숙은 부가적인 보안시스템을 통해 사고를 방어하는 것보다, 안전한 소프트웨어 개발을 위한 시큐어 코딩에 대한 중요성을 인식하고 시큐어 코딩 가이드 및 보안 진단방법에 대하여 제시하였다[4]. 김영직은 홈페이지 변조사고의 주요한 원인과 공격도구, 그리고 해커조직에 대하여 연구하였다[5].

문재찬은 OWASP TOP-10 2010과 CWE(Common Weakness Enumeration) 데이터를 중심으로 웹 애플리케이션 취약점 정보를 분석하여 웹 취약점들을 사상시켜 순서화하고, 그 취약점들이 어떤 개발 생명주기 단계와 관련이 있는지를 제시하였다[6].

2.2 공개용 홈페이지 취약점 분석평가 관련 법규

금융회사 및 전자금융업자는 관련 법규에 따라 주기적으로(6개월에 한번) 공개용 홈페이지에 대한 취약점 분석평가를 한 후 금융위원회에 그 결과를 보고하여야 한다[3].

다음의 Table 1.은 금융위원회에서 권고하는 웹 서비스에 대한 취약성 점검항목 28개에 대한 목록이다.

○ 관련 법규
- 「전자금융거래법」 제21조의 3(전자금융기반시설의 취약점 분석평가)
- 「전자금융감독규정」 제37조의 2(전자금융기반시설의 취약점 분석·평가 주기 내용 등)
- 「전자금융감독규정 시행세칙」 제7조의 2(전자금융기반시설의 취약점 분석·평가의 내용)

Table 1. Financial Services Commission Web Service Vulnerability

No.	Vulnerability
1	Buffer Overflow
2	Format String
3	LDAP Injection
4	Excuting Operating System Command
5	SQL Injection
6	SSI Injection
7	XPath Injection
8	Directory Indexing
9	Information leak
10	Malicious Content
11	Cross-Site Scripting
12	Weak String Strength
13	Insufficient Authentication
14	Recovering Weak Passwords
15	Cross-Site Request Forgery (CSRF)
16	Session Prediction
17	Insufficient Authorization
18	Insufficient Session Expiration
19	Fixed Session
20	Automation Attack
21	Missing Process Verification
22	File Upload
23	File Download
24	Admin Page Exposure
25	Path Tracking
26	Disclose Location
27	Data Plain Text Transmission
28	Cookie Forgery

○ 시행 주기
- 홈페이지에 대해서는 6개월에 1회 이상 실시

○ 점검 항목
- 금융분야 취약점 분석평가 기준 가이드라인 내에 '금융분야 취약점 분석평가 기준'(금융위원회, 2012.6.26.)중 '웹 서비스' 평가항목 28개

2.3 OWASP TOP 10 2017 과 비교

2.3.1 OWASP TOP 10 2017 항목

OWASP(Open Web Application Security Project)는 전 세계 웹 보안 전문가들이 모여서 체계적으로 웹 어플리케이션을 보호하기 위해 연구하는 프로젝트로, 웹 애플리케이션 취약점 중에서 빈도가 많이 발생하고 보안상 영향을 크게 줄 수 있는 10가지를 선정하여 발표한다.

가장 위험한 상위의 취약점 중 Injection 및 Cross-Site Scriting 공격 등은 웹 애플리케이션에 내재된 취약점을 이용해 발생하며, 이러한 취약점은 웹 방화벽 등 보안장비만으로는 차단에 한계가 있어 취약점 조치가 근본적인 대책이 될 수 있다.

OWASP TOP 10 2013 과 비교하여 A4, A7, A10의 3가지 항목이 변경되었다.

Table 2. OWASP Top 10 2017[7]

Ranking	Vulnerabilities
A1	Injection
A2	Broken Authentication and Session Management
A3	Cross-Site Scripting
A4	Broken Access Control
A5	Security Misconfiguration
A6	Sensitive Data Exposure
A7	Insufficient Attack Protection
A8	Cross-Site Request Forgery (CSRF)
A9	Using Known Vunlnerable Components
A10	Underprotected APIs

2.3.2 OWASP TOP 10 2017 과 금융위원회 웹 서비스 취약점 항목과의 비교

OWASP TOP 10 항목은 웹 애플리케이션 개발 시 많이 참조하는 항목이지만 너무 포괄적인 내용으로 분류되어 예방을 위한 정보를 세부적으로 표현하는 것이 어려웠다. 이로 인하여 개발자들이 취약점을 조치하려해도 실제 어떻게 구현해야하는지 어려워했다. 이에 Table 3. 에서는 OWASP TOP 10 2017 항목을 금융위원회 웹 서비스 항목과 비교해 취약점에 대한 이해를 쉽게 할 수 있도록 했다.

A10 항목은 어떤 API가 어떤 용도로 어디에 적

Table 3. OWASP Top 10 2017 and Financial Services Commission Web Service Vulnerability Mapping

OWASP Ranking	Financial Services Commision
A1	- LDAP Injection - SQL Injection - SSI Injection - XPath Injection
A2	- Weak String Strength - Insufficient Authentication - Recovering Weak Passwords - Session Prediction - Insufficient Session Expiration - Fixed Session - Cookie Forgery
A3	- Cross-Site Scripting
A4	- Missing Process Verification - Insufficient Authentication - Insufficient Authorization - Excuting Operating System Command - File Download - Path Tracking
A5	- Directory Indexing - Automation Attack - Admin Page Exposure - Disclose Location
A6	- Information leak - Data Plain Text Transmission
A7	- Automation Attack
A8	- Cross-Site Request Forgery (CSRF)
A9	- File Upload - File Download
A10	

용되는지에 따라 다양한 취약점이 노출 될 수 있으므로 금융위원회 웹 서비스 항목 중 특정 항목으로 지정하기가 어렵다.

III. 공개용 홈페이지 취약점 분석평가 실태

3.1 취약점 분석평가 수행 절차

Fig. 1. 은 금융기관에 대한 공개용 홈페이지 점검 시 일반적인 수행 절차이다. 보유자산에 대한 현황을 분석하고 취약성 점검을 실시 한 후, 결과에 대한 분석평가 및 수준 평가를 실시한다. 도출된 취약점에 대하여 개선방안을 제시하여 취약점을 제거한 후 이행점검을 통하여 확인하여 결과보고서를 제출한다.

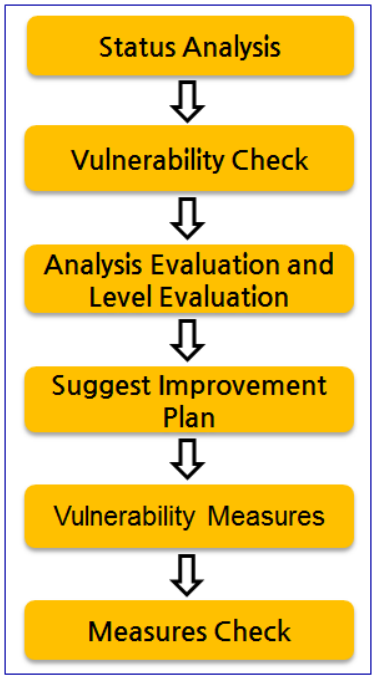

Fig. 1. Procedure of Vulnerability Analysis

3.2 최근 취약점 발생 빈도 분석

Table 4.는 A금융기관의 최근 3년간 6회에 걸쳐 공개용 홈페이지 점검을 한 결과이며, 결과를 보면 정보누출이 가장 많이 발생 했고, 크로스사이트스크립팅과 악성 콘텐츠 취약점이 그 뒤를 따르고 있다.

키보드 보안등이 작동되지 않아 해킹 툴에 의해 정보가 노출되는 경우가 많았고, 화면 상 에서는 마스킹 처리가 되어 있으나 웹 브라우저에 전송되는 데이터에는 적용되지 않는 경우도 다수 차지했다.

Table 4. Frequency by Vulnerability(A Financial Institution Recent 3-Year Check Results)

Rank of occurrence	Vulnerability
1	Information leak
2	Cross-Site Scripting
3	Malicious Content
4	Data Plain Text Transmission
5	Insufficient Authentication
6	Insufficient Authorization
7	Missing Process Verification
8	Fixed Session
9	SQL Injection

크로스사이트스크립팅과 악성 콘텐츠의 경우는 동시에 발생하는 경우가 많은데, 사용자의 클라이언트에서 입력 값으로 〈, 〉, &, ‘, “ 등의 특수기호를 사용하여 해킹을 할 경우 각 단위 프로그램 별 검증을 하지 않아 취약점을 차단하지 못했다.

그 외 로그인 거래 시 사용자의 권한이나 본인인증을 체크하지 않아 발생하는 불충분한 인가, 불충분한 인증, 프로세스 검증 누락과 세션을 복사하여 다른 클라이언트에서도 세션이 유지되는 취약점등이 발생했다. 그러나 앞서 대다수 공격에 사용된 방법 중 하나인 SQL Injection은 거의 발견되지 않았다.

Fig. 2.는 OWASP 코리아 챕터에 자료를 제공한 Softtek 의 자료이며, 이에 따르면 웹 어플리케이션 취약점으로 Cross-Site Scripting 이 27%, SQL Injection 이 17%로 상위권에 있다[8]. 국내

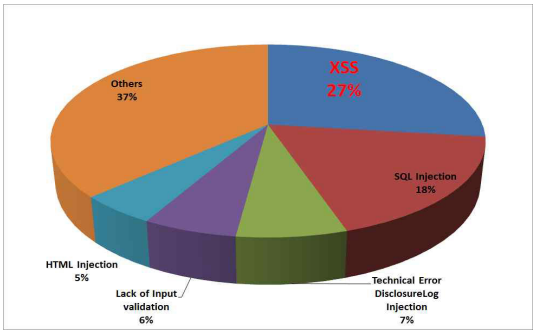

Fig. 2. Softtek Vulnerability Statistics

금융기관의 점검 결과와 비교하여 SQL Injection이 많이 발생하는 것으로 나타나는데, 다른 기관과 비교하여 금융기관에서는 SQL Injection에 대한 대비가 철저한 것으로 분석된다.

3.3 취약점 별 위협 및 조치 방안

아래의 조치 방안은 3년간 A금융기관의 공개용 홈페이지 점검 및 조치 사례를 바탕으로 작성하였다.

3.3.1 Buffer Overflow

애플리케이션 입력 값의 크기에 대한 적절성이 검증되지 않을 경우 개발 시에 할당된 저장 공간보다 더 큰 값의 입력이 가능하고 이로 인한 오류 발생으로 공격자가 악의적인 코드 삽입으로 프로그램을 통제할 수 있는 권한 획득이 가능한 취약점이다.

이 취약점은 웹 서버의 제품 버전을 안전한 버전으로 유지하고, 보안 패치를 항상 최신으로 유지하거나, 웹 애플리케이션에 전달되는 인수값을 필요한 크기만큼만 받을 수 있도록 변경한다. 또한 동적 메모리 할당을 위해 크기를 사용하는 경우 그 값이 음수가 아닌지 검사해야 한다.

3.3.2 Format String

외부로부터 입력된 값에 대한 검증이 없으면 공격자가 포맷 스트링(format string) 문자("%f", "%p", "%n")를 사용하여 메모리 값을 노출 할 수 있으며 악의적인 코드를 실행 가능한 취약점이다.

이 취약점은 문자열 입력 포맷과 실제 입력 값의 일치 여부를 검사하고 사용자 입력 값에 대한 타당성을 검사하여 취약점을 해소한다.

3.3.3 LDAP Injection

사용자 입력 값에 대한 적절한 필터링 및 유효성 검증을 하지 않아 공격자가 LDAP 쿼리를 주입함으로써 개인 정보 등의 내용이 유출될 수 있는 취약점이다.

이 취약점은 사용자 입력 값을 White List로 지정하여 영문 (a-z, A-Z)과 숫자(0-9)만을 허용하며, DN과 필터에 사용되는 사용자 입력 값에는 특수문자가 포함되지 않도록 특수문자를 제거하고, 특수문자 사용이 필요한 경우에는(특히, "(", ")", "; ", "*", "|", "&"등) 실행 명령이 아닌 일반문자로 인식되도록 처리한다.

3.3.4 Excuting Operating System Command

웹 사이트의 인터페이스를 통해 웹 서버를 운영하는 운영체제 명령을 실행할 수 있는 취약점이다.

이 취약점은 애플리케이션이 운영체제로부터 명령어를 직접적으로 호출하지 않도록 구현하고, 명령어 직접 호출이 필요한 경우 명령어를 해석하기 전에 입력 값을 검사함으로써 취약점을 해소한다.

3.3.5 SQL Injection

데이터베이스와 연동된 웹 애플리케이션에서 공격자의 비정상적인 입력 값에 의해 SQL 쿼리가 완성되어 DBMS 및 데이터를 열람하거나 조작할 수 있는 취약점으로 특수문자를 필터링하여 문자열의 유효성을 검사하고 동적 SQL 사용을 지양하여 취약점을 해소한다.

PreparedStatement 방식을 사용하여 SQL 쿼리를 입력 받았을 때 정의된 SQL 구문을 바인딩 처리하여 사용자가 입력한 SQL 구문이 실행되지 않게 조치하며, MyBatis를 이용하여 SQL 구문 작성 시 매개변수 매핑할 때 ${}을 사용하지 말고 #{}을 사용하여 바인딩 처리되도록 한다. 특히 오류에 대한 에러 메시지 출력을 방지하여 예외처리하고 Stored Procedure를 사용한다.

최근 정보유출 사고가 발생한 숙박업체 P2P 사이트도 이 취약점을 이용하여 발생하였으니 철저한 점검이 필요한 취약점 중 하나이다.

3.3.6 SSI(server side include) Injection

HTML 문서 내 입력 받은 변수 값을 서버 측에서 처리할 때 부적절한 명령문이 포함 및 실행되어 서버의 데이터가 유출되는 취약점으로 GET 질의 문자열, POST 데이터, 쿠키, URL 등 웹 서버가 주고받는 모든 데이터를 포함하여 사용자 입력으로 사용 가능한 문자들을 White List 방식으로 정해놓고 필터링 처리하여 취약점을 해소한다.

3.3.7 XPATH(XML path) Injection

조작된 XPATH 쿼리를 전달하여 인가되지 않은 데이터를 열람할 수 있는 취약점으로 (,), =, ‘, [,], :, *, / 등 XPATH 쿼리를 파괴하는 특수문자 입력을 방지하여 취약점을 해소한다.

3.3.8 Directory Indexing

URL 접근 시 웹 서버에 파일이 존재하지 않을 경우, 자동적으로 디렉터리 리스트를 출력하며, 이를 통해 공격자는 웹 서버 내의 디렉터리 구조를 파악하고 주요 설정 파일 획득이 가능한 취약점으로 웹 서버의 환경설정에서 Directory Indexing 기능을 제거하여 취약점을 해소한다.

3.3.9 Information Leak

웹 사이트 데이터가 노출되는 것으로 개발 과정의 코멘트나 오류 메시지 등에서 중요한 정보가 노출되어 공격자에게 2차 공격을 하기 위한 정보를 제공할 수 있는 취약점으로 HTML 소스에서 중요 정보 주석 처리 금지하고 민감한 정보에 대하여 hidden 값으로 기록을 금지한다. 마스킹 처리는 클라이언트가 아닌 서버에서 처리해야하며, 또한 오류 메시지 발생 시, 별도의 에러 페이지로 Redirect 하거나 에러 처리 루틴 추가하여 공격자에게 불필요한 정보가 노출되는 것을 방지한다.

3.3.10 Malicious Content

웹 애플리케이션에 정상적인 콘텐츠 대신에 악성 콘텐츠를 삽입하여 피싱 사이트로 유도하는 등 사용자에게 악의적인 영향을 미치는 취약점으로 사용자 입력 문자(〈, 〉, &, /, \, ′, ″ 등)에 대한 필터링을 서버에서 수행하여 차단하며, 페이지 이동 기능이 존재할 경우 사용가능한 URL을 White List로 관리하여 검증되지 않은 페이지로 이동을 방지한다.

3.3.11 Cross Site Scripting

사용자가 입력한 내용을 기반으로 페이지를 동적으로 생성하는 웹 애플리케이션에 악의적인 스크립트를 포함시켜 사용자에게 전송하여 사용자 측에서 스크립트가 실행되게 하는 취약점으로 사용자 입력 문자(〈, 〉, &, /, \, ′, ″ 등)에 대한 필터링을 서버에서 수행하여 차단하며 게시물에 HTML이나 자바 스크립트에 해당되는 태그 사용을 사전에 제한한다.

이 취약점은 모든 어플리케이션에서 검증할 수 있도록 공통모듈로 사용하는 방안이 필요하며, 공격에 사용하는 입력문자나 공격방법이 다양하게 변하고 있으니 계속적인 추가 및 변경이 필요하며, 공통모듈에 적용 시 사용자의 입력에 대한 오류가 발생하지 않도록 철저한 테스트가 필요한 항목이다. 이 취약점은 악성 콘텐츠 취약점과 동시에 나타나는 취약점으로 해소 시 두 취약점을 동시에 보완할 수 있다.

Fig. 3.을 보면 HTML entity를 이용하여 태그 문자 〈, 〉 를 필터링하여 “&1t”와 “&gt”로 변환하여 브라우저가 일반문자로 인식하게 하여 script가 실행되지 못하도록 예방했다.

```
Function Filter(input As String)

    Dim output As String = input.Replace("<","&lt")
    output = output.Replace(">","&gt")

    Return output

End Function
```

Fig. 3. Cross-Site Scripting Prevention Example

3.3.12 Weak String Strength

유추가 용이한 계정 및 패스워드를 사용하는 경우 반복적으로 값을 입력하여 해당 값을 발견하여 타 사용자 도용 및 타 사용자의 중요정보 열람 또는 수정이 가능한 취약점으로 영문 대소문자, 숫자, 특수문자가 포함된 패스워드 설정, 적절한 패스워드 변경 주기 설정 등을 통한 패스워드 관리를 철저히 하며, 테스트를 위한 계정은 테스트 완료 후 반드시 삭제 처리한다.

3.3.13 Insufficient Authentication

민감한 데이터에 접근 가능한 경로에 대한 인증 절차가 불충분할 경우 권한 외의 페이지에 접근하여 정보를 유출하거나 변조할 수 있는 취약점으로 중요 정

보를 표시하는 페이지는 본인 인증을 재확인하는 처리를 추가하고 인증 과정을 처리하는 부분은 클라이언트 단이 아닌 서버 단에서 수행하여 취약점을 해소한다.

3.3.14 Recovering Weak Password

취약한 패스워드 복구 로직(패스워드 찾기 등)으로 인하여 공격자가 불법적으로 다른 사용자의 패스워드를 획득하여 변경할 수 있는 취약점으로 패스워드 설정 시 사용자의 개인정보(주민등록번호, 연락처 등)로 패스워드 생성을 금지하고, 난수를 이용한 불규칙적이고 최소 길이 8자 이상의 패턴이 없는 패스워드를 사용하며 비밀번호 찾기 시 사용자 정보와 입력한 정보의 일치여부를 확인 및 이메일 또는 SMS로 전송한다.

3.3.15 Cross Site Request Forgery

인증된 정보를 기반으로 하는 웹 애플리케이션에서 사용자의 요청을 변조함으로써 해당 사용자의 권한으로 악의적인 공격을 수행할 수 있는 취약점으로 정상 요청과 비정상 요청 구분을 위해 Form/URL에 임의 토큰을 추가하여 검증을 수행하고, 사용자 입력하는 모든 파라미터 값에 대한 필터링을 서버에서 구현한다.

3.3.16 Session Prediction

Session ID가 단순한 방법으로 생성되는 경우, 공격자가 이를 추측하여 세션을 가로채어 타 사용자의 권한 도용이 가능한 취약점으로 Session ID 생성 범위 값을 사용자 수에 대비하여 충분히 큰 값으로 설정하고, 추측이 가능한 단순 조합보다는 상용 웹 서버나 웹 애플리케이션 플랫폼에서 제공하는 세션 ID를 사용하며 특히 중복 로그인에 대한 검증을 추가로 한다.

3.3.17 Insufficient Authorization

사용자 별 기능에 대한 접근 또는 사용 제한이 없는 경우, 공격자에 의해 관리자 또는 타 사용자의 권한으로 실행 가능한 취약점으로 민감한 중요 데이터에 대한 접근 페이지 인증을 실시하고, 페이지 별 권한 매트릭스를 작성하여 모든 페이지에 대하여 부여된 권한의 타당성을 검증한다.

3.3.18 Insufficient Session Expiration

세션의 만료 기간을 정하지 않거나, 만료기한을 너무 길게 설정하여 공격자가 만료되지 않은 세션을 활용하여 타 사용자의 권한 도용이 가능한 취약점으로 세션 타임아웃 시간을 10분 이내로 설정한다.

3.3.19 Fixed Session

사용자 로그인 시 항상 일정하게 고정된 세션 ID 값을 사용하는 경우, 세션 ID를 도용한 비 인가자의 접근 및 권한 우회가 가능한 취약점으로 세션 체크 시 기기의 맥주소나 IP 정보를 보유하여 타당성 검증을 한다. 특히 휴대전화 등 이동이 빈번한 기기의 사용에 대비하여 유동 IP에 대하여 세션이 끊어지지 않도록 별도의 대책을 강구하여야 한다.

3.3.20 Automation Attack

데이터 등록 또는 메일 발송 기능 등을 가진 웹 애플리케이션에 자동화된 공격을 수행하여 많은 수의 프로세스를 실행시켜 맞는 정보를 찾아내는 취약점으로 데이터 등록 또는 메일 발송 기능 등에 CAPTCHA와 같은 일회성 확인 방안을 추가해 취약점을 해소한다.

Fig. 4. CAPTCHA Application Example

3.3.21 Missing Process Verification

공격자가 웹 애플리케이션에서 계획된 플로우 통제우회가 가능한 취약점으로 페이지 별 권한 매트릭스를 작성하여 모든 페이지에 부여된 권한의 타당성을 검증한다. 유효 세션의 검증 및 페이지 접근 권한 확인은 스크립트가 아닌 서버 단에서 수행하여

검증한다.

3.3.22 File Upload

업로드 되는 파일의 확장자에 대한 적절성 여부를 검증하지 않아 공격자가 시스템 명령어를 실행할 수 있는 Server Side Script 파일 업로드가 가능한 취약점으로 업로드 파일을 위한 디렉터리를 별도로 두고 업로드 파일의 크기를 제한한다. 또한 업로드 파일의 실행 권한을 제거하고 첨부 파일 확장자 필터링 기능을 서버에서 수행하여 취약점을 해소한다.

3.3.23 File Download

다운로드 스크립트에서 파라미터 값을 검증하지 않아 공격자가 임의의 문자를 통해 웹 서버의 주요 파일을 다운로드 할 수 있는 취약점으로 절대/상대경로를 통한 다운로드가 아닌 시퀀스 넘버를 통한 다운로드를 구현하고 다운받을 수 있는 디렉터리를 특정하고 하위 경로 탐색을 제한하도록 설정한다.

3.3.24 Admin Page Exposure

일반적으로 추측하기 쉬운 URL을 관리자 페이지로 사용하는 경우, 공격자가 이를 추측하여 관리자 페이지에 접근이 가능한 취약점으로 관리자 페이지는 주소를 유추하기 어려운 이름으로 설정하고 WAS(web application server)의 기본 관리 포트번호를 변경한다.

WAS	Admin Page URL
Tomcat	[Domain-name]/manager/html [Domain-name]:8080/manager/html
Weblogic	[Domain-name]:7001/console
Websphere	[Domain-name]:7090,9090,9043/admin
Jeus	[Domain-name]/webadmin

Fig. 5. Admin Page of WAS

3.3.25 Path Tracking

웹 서버 또는 웹 애플리케이션의 파일 또는 디렉터리 접근이 통제되지 않아 서버 또는 웹 애플리케이션의 중요한 파일 또는 데이터에 접근을 허용하는 취약점으로 웹 서버 설정에서 웹 사이트의 최상위 폴더를 웹 사이트 Root 디렉터리로 제한하고, "../" 같은 경로이동 문자 등의 사용자 입력 값(특수문자)에 대한 필터링을 처리한다.

3.3.26 Disclose Location

폴더나 파일명의 위치가 예측 가능하고 쉽게 노출되어 공격자가 이를 악용하여 대상에 대한 정보를 획득하고 민감한 데이터에 접근 가능하게 하는 취약점으로 웹 서버 상에는 운영에 필요한 최소한의 파일만을 유지하고 작업 중 생성된 백업 및 테스트 파일을 모두 삭제 처리한다. 특히 웹 서버 설정을 통해 Apache, IIS, Tomcat 등의 웹 서버 디폴트 페이지와 디폴트 디렉터리, Banner 도 삭제 처리한다.

3.3.27 Data Plain Text Transmission

서버와 클라이언트 간 통신 시 데이터 암호화 프로세스가 구현되지 않아 중요 정보 등이 평문으로 전송되는 취약점으로 중요 정보는 전송 구간 암호화 통신(SSL, HTTPS, VPN 등)을 적용하고, 암호화 통신 적용 시 안전한 알고리즘을 사용하도록 적용한다. 또한 주요 개인정보에 대하여는 화면에서 볼 때 마스킹 처리가 되어 있더라도 데이터 송·수신 시 평문이 전송되는지 확인하여 데이터 송·수신 시 에도 암호화하여 처리하도록 조치한다.

3.3.28 Cookie Forgery

적절하게 보호되지 않은 쿠키를 사용하여 쿠키 인젝션 등과 같은 쿠키 값 변조를 통한 타 사용자 권한 도용 등을 가능하게 하는 취약점으로 쿠키 대신 보안성이 강한 Server Side Session을 사용하고 쿠키에 중요정보(비밀번호, 주민등록번호, 계좌번호 등) 기록을 금지하며, 어쩔 수 없이 중요정보를 사용해야 할 경우에는 암호화하여 사용한다.

3.4 취약점 예방을 위한 제안

3.4.1 시큐어코딩 교육

가트너의 보고서에 의하면 Fig. 6.처럼 보안 침해 사고의 75%는 취약점을 내포하는 응용프로그램에서

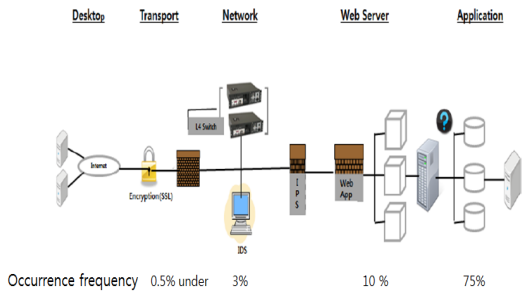

Fig. 6. Frequency of security accident

발생되었다고 알려져 있다. 이미 개발된 프로그램에서 취약점을 찾고 조치하기 위해서는 더 많은 비용과 리스크를 수반하게 된다. 이에 개발자들에 대한 주기적인 시큐어코딩 교육으로 개발 초기단계부터 취약점이 발생하지 않도록 관리하는 것이 필요하다.

3.4.2 정적 취약점 분석도구를 이용한 자동 점검

정적 분석이란 어플리케이션에 내재되어 있는 취약점을 분석하기 위하여 취약점으로 발전 가능성을 가진 소스 코드들의 형태를 분석하는 기법으로, 대부분의 금융기관들은 하나이상의 소스코드 취약점 분석 도구들을 도입하여 사용하고 있다[9]. 이러한 소스코드 취약점 분석 도구를 프로그램 변경관리시스템과 연동하여 프로그램 적용 전에 취약점을 조치한 후 적용 할 수 있도록 자동화한다. 그러나 이러한 정적인 방법은 취약점을 모두 해소하기에는 부족하다.

3.4.3 상시 취약성 점검 체계 구축

정적인 분석 방법에는 분명 한계가 있기 때문에 업무 프로세스에 의한 시나리오 별 취약성 점검이 필요하다. 이에 동적 분석 방법을 사용하며, 동적 분석 방법이란 대상 프로그램을 실제로 실행 시켜가면서 분석하는 방법이다[10].

동적 분석 방법은 프로그램에 입력 값을 주면서 실행시키기도 하고 프로그램의 기능을 차례차례 혹은 임의적으로 실행시켜가면서 분석한다. 동적 분석 방법은 개발자의 미숙함이나 보안의식 부족 등으로 발생되는 알려진 취약점 외에 예상치 못한 취약점을 발견할 수도 있다. 이 때문에 정적 분석 방법을 거친 후 동적 분석 방법을 수행하는 것이 일반적이다.

따라서 분석도구에 의한 점검으로는 찾아낼 수 없는 취약점을 자체 조직 또는 전문 컨설턴트를 고용하여 프로그램 변경 시마다 점검하여 선 조치 후 적용할 수 있도록 하는 프로세스가 필요하다. 이를 위해서는 테스트 시스템에서의 점검 환경 구축이 선행되어야 할 것이다.

3.4.4 시나리오 기반의 취약성 점검 수행

시나리오 기반의 취약성 점검은 사고사례와 기술문서로 1차 시나리오를 작성하고 현업 담당자, 보안담당자, 보안 컨설턴트 등이 모여 최종 시나리오를 작성하여 점검을 수행한다. 이러한 방법은 일반적인 취약성 분석에서는 할 수 없는 각 기업의 특화된 위협을 찾아 안전하게 시스템을 운영할 수 있도록 해줄 것이다[11].

IV. 결 론

최근 인터넷 은행이 인가를 받고 영업을 시작했고 또한 시행을 기다리고 있다. 이제 금융기관이 오프라인이 아닌 온라인으로 영업을 확대하고 있으며, 이에 따라 웹 취약점에 대한 이슈가 점점 많아질 것이다. 많은 금융기관에서 수행하고 있는 공개용 홈페이지에 대한 취약점 분석평가에 대하여 항목 별 위협과 조치방안을 살펴보았다. 또한 보안사고 중 응용프로그램에 의한 사고 빈도가 가장 높음을 알 수 있었고 이에 취약점을 예방하기 위해 가장 먼저 선행되어야 할 것으로 개발자들에 대한 시큐어 코딩 교육이라는 것도 알 수 있었다.

금융기관들이 많이 발생하고 조치를 취하는 취약점에 대하여는 공개를 하지 않아 그에 대한 자료는 충분하지 않으나 대량의 중요정보들을 취급하는 국내외 금융기관들에서 일어나는 사고 유형에 미루어 크로스사이트스크립팅, SQL Injection 등이 매우 위협적인 취약점이라는 사실은 유추 할 수 있다.

향후 각 금융기관의 보안업무 담당자들은 금융위원회의 웹 서비스 취약점뿐만 아니라 보안사고의 트랜드에 따라 한국인터넷진흥원, 금융보안원 등에서 발표하는 취약점과 대처방안에 대하여도 점검 및 조치해야하며, 상시 점검 할 수 있는 프로세스를 갖추어 안전한 웹 애플리케이션이 개발될 수 있기를 바란다.

References

[1] I NEWS24 Broadcast, "By 2016, a massive security incident" http://news.inews24.com/php/news_view.php?g_serial=1017593&g_menu=020200, Apr, 2017

[2] Autumn Byeon and Jong In Lim and Kyong-Ho Lee, "A Study On Advanced Model of Web Vulnerability Scoring Technique," Journal of The Korea Institute of information Security & Cryptology, 25(5), Oct. 2015.

[3] Keun-dug Park and Heung-youl Youm, "Improvements of Information Security Level in Electronic Financial Infrastructure(By Analyzing Information Security Management Level)," Journal of The Korea Institute of information Security & Cryptology, 26(6), Dec. 2016.

[4] Jung-Sook Kim, "Secure Coding for Software Security," The Korea Contents Association 11(4), pp. 56-60, Dec. 2013.

[5] Young-Jik Kim and Bong-Nam Noh, "Study on the trend Homepage Defacement Incident and Defacement Hacker group," Proceedings of Symposium of the Korean Institute of communications and Information Sciences, pp. 695-701, Nov. 2008.

[6] Jae-Chan Moon and Seong-Je Cho, "Vulnerability Analysis and Threat Mitigation for Secure Web Application Development," Journal of the Korea Society of Computer and Information, 17(2), pp. 127-137, Feb 2012.

[7] https://www.owasp.org/index.php/Top_10_2017-Top_10.

[8] "Survey and Analysis of Web Application Vulnerability and Inspection," Industry-Academia Collaboration Foundation of Seoul Women's University, KISA, pp. 157-168, Dec. 2014.

[9] Yeon-Soo Choo, "A Study on Intelligent Method of Vulnerability Analysis for Secure Application," Department of Computer Science and Engineering Graduate School of Soongsil University, pp. 31-33, Jun 2016.

[10] Lo R., Kerchen P., Crawford R., Ho W., Crossley J., Fink G., Levitt K., Olsson R. and Archer M., "Towards a Testbed for malicious code detection," COMPCON Spring '91. Digest of Paper. San Fransisco, CA, pp.160-166, Feb-Mar. 1991.

[11] Min-Seong Seo, "A Study on Threat Scenario-based Security Vulnerability Analysis," Dept. of Information Security The Graduate School of Information & Communications SungKyunKwan University, pp. 38-48, Jun. 2011.

〈저자소개〉

박 현 진(Hyun-Jin Park) 정회원
1993년 2월: 동국대학교 전자계산학과 졸업(학사)
2003년 8월: 동국대학교 광고홍보학과 졸업(석사)
2015년~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 취약점 진단, 정보보호 정책, 디지털포렌식 등

김 인 석 (In-Seok Kim) 정회원
1973년 2월: 홍익대학교 전자계산학과 졸업(학사)
2003년 2월: 동국대학교 정보보호학과 졸업(석사)
2008년 2월: 고려대학교 정보경영공학과 졸업(박사)
2009년~현재: 고려대학교 정보보호대학원 교수
〈관심분야〉 전자금융보안, IT감사, 전자금융법규