韓國컴퓨터情報學會 論文誌
第17卷 第2號, 2012. 2.

2012-17-2-3-6

안전한 웹 애플리케이션 개발을 위한 취약점 분석 및 위협 완화

문 재 찬*, 조 성 제**

Vulnerability Analysis and Threat Mitigation for Secure Web Application Development

Jae-Chan Moon*, Seong-Je Cho**

요 약

최근에 매쉬업(mashups), 웹 3.0, JavaScript, AJAX (Asynchronous JavaScript XML) 등이 널리 사용되면서, 새로운 취약점들이 발견되고 있어 보안 위협이 더 증대되고 있다. 이러한 웹 애플리케이션 취약점과 보안 위협을 효율적으로 완화하기 위해, 그 취약점들을 위험도 기준으로 순서화하여 웹 애플리케이션의 개발 생명주기의 해당 단계에서 우선적으로 고려해야 한다. 본 논문에서는 미국 NVD(National Vulnerability Database)의 웹 애플리케이션 취약점에 대한 데이터를 분석하여, OWASP Top 10 취약점들의 위험도 산정 방법이 타당한 지를 검증하였다. 그 다음, 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. 이를 통해 효율적으로 웹 보안 위협과 취약점을 예방하거나 완화할 수 있다.

▸ Keyword : 웹 애플리케이션, OWASP Top 10, 취약점 분석, 위협 완화, 소프트웨어 개발 생명주기

Abstract

Recently, as modern Internet uses mashups, Web 3.0, JavaScript/AJAX widely, the rate at which new vulnerabilities are being discovered is increasing rapidly. It can subsequently introduce big security threats. In order to efficiently mitigate these web application vulnerabilities and security threats, it is needed to rank vulnerabilities based on severity and consider the severe vulnerabilities during a specific phase of software development lifecycle (SDLC) for web

• 제1저자 : 문재찬 • 교신저자 : 조성제
• 투고일 : 2011. 10. 22, 심사일 : 2011. 12. 09, 게재확정일 : 2012. 01. 11.
* 단국대학교 컴퓨터과학(Dept. of Computer Science, Dankook University)
** 단국대학교 소프트웨어학과(Dept. of Software Science, Dankook University)
※ 이 연구는 2010학년도 단국대학교 대학연구비 지원으로 연구되었음.

applications. In this paper, we have first verified whether the risk rating methodology of OWASP Top 10 vulnerabilities is a reasonable one or not by analyzing the vulnerability data of web applications in the US National Vulnerability Database (NVD). Then, by inspecting the vulnerability information of web applications based on OWASP Top-10 2010 list and CWE (Common Weakness Enumeration) directory, we have mapped the web-related entries of CWE onto the entries of OWASP Top-10 2010 and prioritized them. We have also presented which phase of SDLC is associated with each vulnerability entry. Using this approach, we can prevent or mitigate web application vulnerabilities and security threats efficiently.

▸ Keyword : Web application, OWASP Top 10, Vulnerability analysis, Threat mitigation, Software development lifecycle (SDLC)

I. 서 론

Web 2.0과 Web 3.0이 활용되면서 E-Learning 서비스, 트위터, Facebook과 같은 소셜 네트워킹 서비스(SNS), 온라인 게임 서비스, YouTube와 같은 엔터테인먼트 서비스, u-Healthcare 서비스, 온라인 뱅킹 서비스 등의 웹 기반 서비스가 일반화 되었다.

웹 기반 서비스는 개인 신상 정보 및 주소·위치 정보, 병력 정보 등의 민감한 프라이버시 관련 정보나 신용 정보를 취급하고 있다. 따라서 보안 침해사고가 발생하게 되면 정보유출로 인해 막대한 피해가 발생할 수 있다. 실제로, 2011년 4월 국내 대형 금융사가 웹 사이트의 취약점(vulnerability)을 악용한 악성코드 감염 공격으로 고객 개인정보 유출로 큰 피해를 입었고, 2008년 2월 국내 인터넷 쇼핑몰에서도 유사한 공격을 받아 1863만 여명의 개인정보가 유출 되었다[1].

취약점(vulnerability)은 시스템 보안 정책의 침해나 위반으로 유도될 수 있는 보안상의 결점(flaw)이나 허점(weakness)으로, 모든 보안 사고의 원인이라고 볼 수 있다. 소프트웨어 취약점의 대표적인 예는 SQL-인젝션, DLL 인젝션, XSS, 버퍼 오버플로 등이다.

그림 1은 미국의 보안 업체인 WhiteHat Security Inc가 WhiteHat Sentinel 솔루션을 통해 400여개 회사들의 3,000 개 이상의 웹 사이트들에 대해 보안 평가한 결과를 집계한 것으로, 여러 분야의 웹 사이트들이 보안 취약점을 포함하고 있는 일수의 분포를 나타낸다. 이 표는 웹 사이트들의 44% 정도가 심각한 취약점을 포함하고 있으며, 은행 사이트를 제외한 모든 분야의 웹 사이트들이 1년 중 271일 이상 보안 취약점을 포함한다는 것을 보여준다[2].

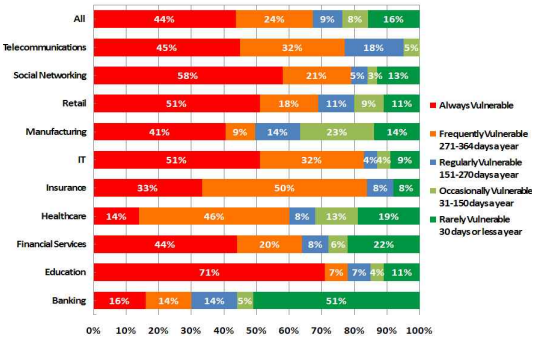

그림 1. 웹사이트들의 취약점 포함일 분포('10)
Fig. 1. Distribution of the number of Vulnerable days of Industry Websites

이처럼 웹 애플리케이션들이 많은 취약점을 내포하고 있기 때문에, 시간과 비용 면에서 효율적으로 이들 취약점들을 예방하기 위한 연구가 필요하다. 한 방안으로는, 취약점 종류 및 발생 추이가 계속 변화하고 있기 때문에 파급효과 및 심각도에 따라 취약점들의 우선순위를 매겨 주요 취약점들을 효과적으로 관리하는 것이다. 이러한 노력 중의 하나가 웹 애플리케이션들의 대표 취약점들의 목록인 OWASP Top-10이다. 다음으로는, 소프트웨어 개발 생명주기(Software Development LifeCycle: SDLC) 각 단계에서 위험도가 높은 순으로 웹 관련 취약점들을 미리 제거 또는 완화하는 것이다.

본 논문에서는, 먼저 미국 NVD (National Vulnerability Database)[3]의 웹 애플리케이션들에 대한 취약점 데이터를 분석하여, OWASP Top 10의 취약점 우선순위에 대한 타당성을 조사한다. 그리고 OWASP Top 10 2010을 중심으로 CWE(Common Weaknesses Enumeration)[4,5]와의 대응 관계를 검증하고 웹 애플리케이션 취약점 정보를 분석하여, OWASP 분류와 CWE 분류의 웹 취약점들을 사상(mapping)시켜 우선 고려할 취약점 항목들을 제안한다. 동

시에 주요 취약점들이 웹 애플리케이션 '소프트웨어 개발 생명주기'(SDLC)의 어느 단계와 연관이 있는지도 보인다. 최종적으로는 소프트웨어 개발 생명주기의 특정 단계에서 심각한 주요 취약점을 미리 고려하는 것이 웹 애플리케이션 취약점들을 제거하거나 완화하는데 시간과 비용 면에서 효율적임을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 취약점 관리체계 등 관련 연구에 대해 기술한다. 3장에서는 웹 취약점 항목들의 순위 변화 및 웹 관련 취약점들의 데이터를 분석한다. 4장에서는 웹 관련 취약점들 분류 항목들 간에 사상(mapping)시키는 과정을 제시하고 소프트웨어 개발 생명주기의 단계들과 연관시킨다. 5장에서는 본 논문에서 제안한 취약점들 항목 및 개발 단계 사상 방법이 유효함을 보이고, 6장에서 결론을 맺는다.

II. 관련 연구

1. 학술 분야

모든 보안 취약점들을 고려하는 것은 현실적으로 불가능하므로, 많은 연구자들이 특정 보안 취약점을 효율적으로 제거하거나 완화하기 위해 노력하였다. 취약점 분류체계(taxonomy)는 소프트웨어 개발자 및 보안 실무자들로 하여금 보안에 영향을 주는 코딩 오류들을 이해하고 바로잡는데 도움을 준다[4,5,6,7].

Martin 등은 [4]에서, 보안 평가도구 및 서비스들의 역량과 커버리지를 정의하는 명명법(nomenclature), 분류체계(taxonomy), 표준 등이 없음을 지적하고, 소프트웨어 보안 취약점들의 표준 분류체계에 대한 연구를 수행하였다. 즉, 널리 알려진 CVE (Common Vulnerabilities and Exposures)[1][4,5,6,8] 보안 취약점 리스트를 확장하여 새로운 코드 보안 평가 기법을 개발하고 관련 도구 및 서비스 사용을 활성화 시키려고 하였다. 이 연구의 결과로 Martin 등은 여러 연구진들과 협력하여 CWE (Common WIFF Enumeration → 후에 Common Weaknesses Enumeration로 변경)[2] [4,5,6,8,11] 리스트의 구성요소들을 정의하였다[4]. 또한 CWE 리스트 개발로 얻을 수 있는 효과와 가치 등을 기술하였다[5].

1) MITRE사가 개발한 취약점에 대한 표준 식별체계

2) WIFF는 Weaknesses, Idiosyncrasies, Faults, Flaw 의 약어. 현재 CWE는 Common Weaknesses Enumeration로, 소프트웨어 약점들에 대한 분류체계를 나타냄. 한 CWE에 여러 CVE가 대응될 수 있음.

Tsipenyuk 등은 소프트웨어 보안 오류들의 분류 방법으로 7가지 치명적인 계(界)를 제안하였다[7]. 또한, 가장 잘 알려지고 유용한 두 가지 리스트인 "19개의 치명적인 오류들(sins)"[12]과 "OWASP[3] Top 10"[7, 9, 10]과 그들이 제안한 "7가지 계"를 사상시켰다.

최근에 Tripathi 등은 소프트웨어 도구 및 서비스들의 보안 평가를 위한 취약점 분류체계의 표준화에 대해 연구하였다[6]. 이를 위해 기존 취약점 분류 방법들과 CVE 디렉토리의 범주화에 대해 분석하여 초기 단계에 머물고 있는, 보안성을 개선하고 위험을 평가하는 연구가 지속되어야 함을 지적하였다.

Wagner 등은 웹 시스템에서 소프트웨어 취약점들로 인한 파급효과를 줄이기 위해서 개발단계의 초기에 보안 요구사항을 도입하는 연구를 수행하였다[13]. 그들은 고수준 요구사항을 구체적이고 평가할 수 있는 수준의 요구사항으로 정제하는 연구와 요구사항의 재사용성을 개선하는 연구를 수행하였다.

Wang 등은 소프트웨어 시스템의 신뢰성(trustworthiness) 정도를 정량적으로 측정하기 위해, 소프트웨어 보안 메트릭(metrics, 측정지표)을 구체적으로 정의하는 실질적인 방법을 제안하였다[8]. 그들은 CVE와 CWE, CVSS (Common Vulnerability Scoring System)[4][14] 등을 주요 소스로 하여 소프트웨어 보안 메트릭을 정의하였다.

김유경 등은 소스코드들에서 고려할 수 있는 보안취약점 데이터베이스를 구축하였다[15]. 그들은 CWE, CVE, CVSS, CAPEC(Common Attack Pattern Enumeration and Classification)[13] 등으로 소프트웨어 취약점들을 재분류하고, 취약점 중요도/위험도를 평가할 수 있는 기준을 제시하였다.

CVSS 산정의 측정지표로는 기본 특성(시간에 무관한 특성), 시간성(시간에 따라 변하는 특성), 환경요인 등 3가지 범주가 활용된다[13]. 위험도는 산정 방식에 있어 객관성이 유지되어야 하고 실제 적용이 용이해야 한다. 이를 위해 CVSS 산정 시에는 관련 업계나 전문가들의 투표하는 기법을 도입하기도 하나, 이 역시 투표에 참여하는 기관이나 전문가의 주관적인 요소가 많이 포함될 수 있다. 최근 미국에서는 CWSS (Common Weakness Scoring System)[5]에 대한

3) The Open Web Application Security Project(OWASP)는 웹 애플리케이션들의 취약성을 다루는 대표적인 조직으로, 웹 애플리케이션 보안을 위한 도구와 표준, 대표 취약점 Top-10 리스트 등의 정보를 제공

4) 소프트웨어 취약점들의 등급 산출을 위해 공개된 표준 방법을 제공하는 취약점 (심각도) 점수화 체계

5) 약점들의 (심각도) 순서화를 위한 점수 산정 체계

연구를 시작하고는 있지만, 이 역시 유사한 문제를 겪고 있다.

다른 한 방법으로는 위험이 높은 취약점을 우선적으로 다루기 위해 Top-N 리스트를 사용하기도 한다. 그 대표적인 예가 OWASP Top-10[10], Seven pernicious kingdoms[7], CWE/SANS Top 25 Most dangerous programming errors[11] 등이다. 이러한 Top-N 리스트가 효과를 발휘하기 위해서는 최신 취약점 정보를 반영해야 한다.

2. 산업 표준 분야

미국 NVD(National Vulnerability Database)는 미 정부 산업 표준 기관인 NIST가 운영하는 취약점 DB이다[3]. 서로 다른 벤더나 전문가들이 하나의 취약점을 다르게 지칭할 수 있는 혼란을 막기 위해 CVE라는 취약점 식별체계를 통해 NVD가 구축되었다. CVE는 OVAL(Open Vulnerability and Assessment Language), CPE(Common Platform Enumeration)와 더불어 미국 정부가 추진 중인 정보보안을 위협하는 기술을 자동화 및 표준화하기 위한 SCAP Security Content Automation Protocol)의 구성 요소이다. SCAP는 자동으로 취약점을 관리하고 평가할 수 있게 하는 표준을 위한 하나의 방법이다[3].

OWASP에서는 2004년 이후로 매 3년마다 웹 애플리케이션들에서 가장 위험한 취약점 10가지를 선정하여 OWASP Top 10으로 제공하고 있다[7,10]. OWASP Top 10의 각 취약점 항목은 순위가 높을수록 위험성이 더 높은 것을 나타낸다. 2010년의 가장 상위 항목인 “A1:Injection(인젝션)”은 SQL 인젝션, XML 인젝션, OS 명령어 인젝션 등을 모두 포함하는 포괄적인 개념으로, 10 개의 각 취약점(위험 요소) 항목들은 크고 추상적인 범주로 분류되어 있다[10].

CWE는 소프트웨어 취약점 분류 체계로, 약 40개 이상의 업체와 연구기관이 협력하여 발전시켜 나가고 있다[4,5,11]. CWE는 다양한 소프트웨어의 취약점을 식별하기 위해 각 취약점 유형에 ID를 할당하였으며, 취약점들 간의 관계를 계층 구조로 체계화 하였다. CWE의 계층 구조에서 가장 상위 엔트리는 뷰(View)로, 공통적인 특성을 갖는 취약점들이나 카테고리들을 하나로 묶어 CWE-ID를 부여한다. 미국 NVD에서 사용 중인 CWE들을 위한 CWE-635 뷰, OWASP Top 10 2010에 해당하는 CWE들을 위한 CWE-809 뷰 등이 대표적이다.

CWE의 장점은 그림 2와 같이 해당 CWE 취약점이 소프트웨어 개발 생명주기(SDLC) 프로세스 단계 중 어느 단계와 관련되는 지를 보여준다는 점이다. 또한 취약점 관련 보안 위험을 완화하는 기법도 제공한다.

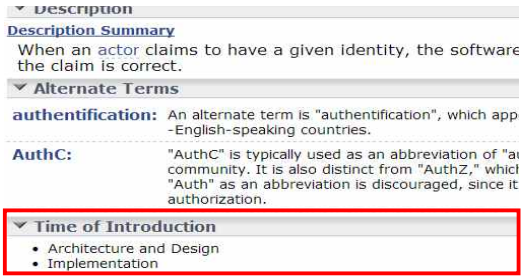

그림 2. CWE 분류에서 취약점 발생 단계
Fig. 2. Time of Introduction of a Vulnerability in CWE Information

MITRE는 SANS와 함께 취약점으로 연결될 수 있는 가장 위험한 소프트웨어 에러들의 순위인 CWE/SANS Top 25를 제공하고 있다. 웹 애플리케이션의 취약점들만을 순서화하는 OWASP Top 10와 다르게, CWE/SANS Top 25는 웹 애플리케이션을 포함하여 전반적인 소프트웨어의 위험한 오류들을 순서화 하였다.

본 논문은 최근 이슈가 되고 있는 웹 애플리케이션 취약점들 항목들 간의 대응 관계를 고려하여 사상시키며, 각 취약점이 소프트웨어 개발 생명주기(SDLC)에서 어느 단계와 관련이 있는지를 다룬다. 이를 위해 기존의 연구 결과들의 유효성을 분석하여 개선한다.

III. 웹 애플리케이션 취약점 분석

1. 웹 애플리케이션 취약점 항목의 변화

2004년부터 2010년 사이에 OWASP Top 10 취약점들 랭킹은 다음 표 1과 같이 변화되어 왔다. 어떤 취약점 항목들의 랭킹 순서가 변경되었고, 어떤 항목들은 랭킹에서 제외되었다. 취약점이 랭킹에서 제외된 예로는 2007년의 “A3:Malicious File Execution”이 있다. 이 취약점은 2007년에 특히 많이 퍼졌던 php 애플리케이션에서 발생했던 취약점이었는데 php의 보안이 개선됨에 따라 해당 취약점의 확산이 줄어 랭킹에서 삭제되었다. 순위가 변경된 대표적인 예는 2004년의 “A4:XSS”로 2007년 A1로, 2010년 A2로 랭킹 순위가 변경되었다. 2007년 “A5:CSRF”는 이전 목록에는 없었던 취약점 항목으로 점점 널리 위험하다고 알려지면서 2007년 랭킹에 새로 추가 되었다.

이처럼 취약점들은 순위가 상승 혹은 하강되기도 하며, 새로 추가되거나 삭제되기도 한다. 취약점이 랭킹에서 삭제되거나 순위가 내려가는 이유는 그 취약점에 대해 충분히 패치 되

었거나 방어기법이 적용되고 있다는 의미이다. 또한 순위가 상승되거나 새로 추가되는 경우는 취약점이 새로 발견되었거나 새로운 기술관련 취약점, 혹은 해당 취약점의 사용 빈도가 늘거나 위험도를 높게 산정한 것들이다. 이를 통해, 취약점들에 대한 정보를 주기적으로 갱신할 필요가 있음을 알 수 있다.

표 1. OWASP Top 10 취약점 항목 랭킹 변화
Table 1. Ranking change of OWASP Top 10 vulnerabilities from 2004 to 2010

순위	2004	2007	2010
A1	Unvalidated Input	XSS	Injection
A2	Broken Access Control	Injection Flaws	XSS
A3	Broken Authentication and Session Management	Malicious File Execution	Broken Authentication and Session Management
A4	XSS	Insecure Direct Object Reference	Insecure Direct Object Reference
A5	Buffer Overflow	CSRF	CSRF
A6	Injection Flaws	Information leakage and Improper Error Handling	Security Misconfiguration
A7	Inproper Error Handling	Broken Authentication and Session Management	Insecure Cryptographic Storage
A8	Insecure Storage	Insecure Cryptographic Storage	Failure to Restrict URL Access
A9	Application Denial of Service	Insecure Communications	Insufficient Transport Layer Protection
A10	Insecure Configuration Management	Failure to Restrict URL Access	Unvalidated Redirects and Forwards

2. 웹 취약점 데이터 분석

2.1 OWASP Top 10의 위험도 산정 타당성 분석

OWASP Top 10 2010의 취약점 위험도를 산정하는 스키마가 표 2에 나타나 있다. 위험도 산정에는 각 취약점의 공격 용이도(Exploitability), 알려진 정도(Prevalence), 탐지 용이도(Detectability)와 영향(Impact) 등이 이용된다. 각 항목은 여러 기관들로부터 제공받은 수치를 분석하여 단계를 나누어 1점부터 3점까지의 점수를 부여한다. A2:XSS의 '알려진 정도'가 유일하게 0 점의 점수를 갖는다.

표 2. OWASP Top 10 2010의 위험도 산정 스키마
Table 2. Risk Rating Schema of OWASP Top 10 2010

Threat Agent	공격 용이도	알려진 정도	탐지 용이도	기술적 영향	Business Impact
.	Easy (3)	Wide-spread(1)	Easy (1)	Severe (3)	.
	Average (2)	Common (2)	Average (2)	Moderate (2)	
	Difficult (1)	Uncommon (3)	Difficult (3)	Minor (1)	

그림 3은 표 2의 스키마를 이용하여 "A2:XSS"의 위험도를 산정하는 과정으로 공격 용이도(2), 알려진 정도(0), 탐지 용이도(1)의 평균에 영향(2)을 곱하여 최종 위험도(2)를 구하게 된다. 이때 알려진 정도가 클수록, 즉 해당 취약점이 많이 발견될수록 점수가 낮아지게 되어 위험도의 점수가 낮아지게 된다.

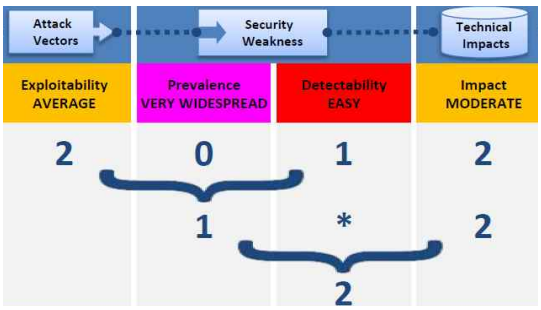

그림 3. OWASP Top 10 2010 A2:XSS의 위험도 산정 과정
Fig. 3. Calculation of the Risk for A2:XSS in OWASP Top 10 2010

본 논문에서는 OWASP Top 10 목록이 타당한지를 확인하기 위해, OWASP Top 10의 2010 버전에 나타난 취약점들의 비율을 미국 NVD에 등록된 취약점 자료를 분석하여 표 3에 나타내었다. 이 결과는 OWASP Top 10의 각 취약점이 10개의 취약점들 중 차지하는 비율을 NVD에 등록된 취약점 수를 기준으로 집계한 것이다. 이는 해당하는 취약점이 NVD에 얼마나 많이 등록되었는지를 각 취약점들에 대해 상대적으로 나타는 것으로 OWASP Top 10 2010의 위험도 산정에 사용되는 항목 중 '악용 용이성'과 '알려진 정도'에 해당한다.

분석 결과를 보면 2003년부터 OWASP Top 10 2010의

“A1:Injection”과 “A2:XSS”가 대상 10개 취약점들 중 약 33~90%에 달하는 큰 비율을 차지하고 있는 것을 알 수 있다. 그러나 항목 A5나 A7의 경우, 항목 A6이나 A8에 비해 많이 등록되지 않았다. 이는 많이 발견되는 취약점일수록 위험도 점수가 낮아지는 OWASP Top 10 2010의 위험도 산정 방법[10]과 반대되는 측면이 있다. 실제로 NVD의 웹 취약점 자료를 분석한 결과 가장 위험도가 높은 취약점이 가장 많이 발견되고 있지는 않다. 즉, NVD에 등록된 해당 취약점 비율과 OWASP의 위험도 산정 방법 사이에 일관성이 결여되는 부분이 있음을 알 수 있었다.

표 3. 2001~2011년 사이 NVD 내의 OWASP Top 10 2010 각 취약점들의 비율 (%)
Table 3. Ratio (%) of Each Vulnerability of OWASP Top 10 2010 in NVD from 2001 to 2011

	~ '02	'03	'04	'05	'06	'07	'08	'09	'10	'11
A1	12.0	18.8	30.8	42.4	44.2	38.8	39.9	33.3	30.4	14.9
A2	21.7	36.8	48.1	47.0	47.8	31.5	21.3	25.6	22.6	21.4
A3	8.2	7.2	4.4	0.6	0.9	4.0	6.2	7.3	5.2	9.4
A4	4.3	5.2	2.1	1.1	0.8	6.1	9.3	8.3	11.3	5.7
A5	0.3	0.0	0.8	0.8	0.7	2.8	2.7	3.0	3.2	6.0
A6	6.2	11.3	1.8	1.3	1.1	4.7	5.5	6.4	8.1	14.9
A7	21.0	6.1	4.4	2.7	1.9	1.1	1.8	3.9	3.6	6.8
A8	13.1	11.0	5.2	2.5	1.7	9.6	12.2	11.4	14.0	18.8
A9	9.5	2.3	1.1	1.1	0.4	0.7	0.4	0.5	0.5	1.3
A10	3.6	1.2	1.3	0.6	0.5	0.6	0.7	0.4	1.1	0.8

2.2. OWASP Top 10 2010 취약점 분석

2.1절의 분석 결과에 의하면, 웹에서 “A1:Injection”과 “A2:XSS” 취약점 비율이 상당 부분을 차지하고 있다. 두 취약점의 비율이 2003년부터 2006년까지 점차 증가하였지만 2007년부터 점차 줄어들고 있으며, 나머지 취약점들의 비율이 늘어나고 있는 추세이다.

이러한 이유는 상위 취약점들은 이전부터 발생비율이 높았기 때문에 그에 대한 방어기술이 많이 개발되었고 하위 취약점들에 대한 예방책은 상대적으로 미흡했기 때문이다. 때문에 공격자들은 하위 취약점들을 악용하려고 시도할 것이다. 실제로 2011년 들어 A1, A2의 비율이 급격히 줄고 나머지 취약점들의 발견 비율이 증가하였다. 이러한 사실을 반영하여 웹 개발 시 상위 취약점에만 중점을 두는 것이 아닌 OWASP Top 10에 등록된 하위 취약점들도 고려해야 함을 알 수 있다. 또한, Top-10 리스트 작성의 개발 주기를 줄여서 최신 취약점 정보를 공표하는 것이 필요하다.

IV. 웹 애플리케이션 취약점항목 사상 및 도입단계 대응

1. 웹 애플리케이션 취약점들 간의 사상

OWASP Top 10은 웹 애플리케이션 개발 및 운영 단계에서 주로 참조되어 왔다. OWASP Top-10의 문제점은, 취약점 항목들이 너무 포괄적으로 분류되어 있어 취약점 예방을 위한 정보를 세부적으로 표현하는 것이 어렵다는 것이다. 반면, CWE는 취약점들을 자세히 분류하고 있지만 너무 많은 항목들을 가지고 있고 웹 보안 취약점에 특화된 것이 아니어서, 웹 관련 정보를 쉽게 참조하기 어렵다는 단점이 있다.

또한, OWASP에서 제공하는 OWASP Top-10과 CWE 대응관계[10], MITRE에서 제공하는 OWASP Top-10과 CWE의 대응관계[11], CWE-809 뷰가 제공하는 OWASP Top-10과 CWE 대응관계[11]가 서로 다르다(표 4 참조). 즉, OWASP와 MITRE의 CWE는 각자 별도로 OWASP Top 10의 각 항목에 해당하는 CWE 항목을 제공하고 있다. II장 2절에서 언급했듯이 OWASP Top 10 2010에 해당하는 CWE 뷰인 CWE-809에도 열 개의 하위 항목들이 CWE-810부터 CWE-819까지 존재하며, 각 항목들은 자신의 내용에 포함되는 CWE 멤버들을 갖고 있다.

결과적으로 OWASP Top-10에 대하여 CWE에서 제공하는 CWE 항목들과 CWE-809를 활용하여 대응시킬 수 있는 CWE 항목들이 서로 달라, 개발자나 운용자가 보안 취약점들을 고려할 때 혼선을 초래할 수 있다.

이를 해결하기 위해 본 논문에서는, OWASP Top 10의 각 항목에 대응되는 CWE 항목들을 분석하여 대응관계를 재정립하여 웹 취약점 목록을 개선하였다. 즉 표 4의 세 종류의 OWASP-CWE 대응에서 음영 처리한 항목인 중복되는 항목들을 분석하고 중복된 CWE 항목들을 선별하여 OWASP Top-10 항목들에 대응하는 통합된 CWE 리스트를 구축하였다 (표 6 참조).

표 4. OWASP와 CWE, CWE-809 뷰들 사이의 취약점 대응
Table 4. Correspondence of Vulnerabilities of OWASP, CWE, and CWE-809 View

OWASP Top 10	OWASP의 OWASP-CWE 대응	MITRE의 OWASP-CWE 대응	CWE-809의 OWASP-CWE 대응
A1 (CWE-810)	CWE-77	CWE-78	CWE-78
			CWE-88
			CWE-89
	CWE-89	CWE-89	CWE-90
			CWE-91
A2 (CWE-811)	CWE-79	CWE-79	CWE-79
A3 (CWE-812)	CWE-287	CWE-306	CWE-287
		CWE-307	
		CWE-798	
A4 (CWE-813)	CWE-22	CWE-285	CWE-22
	CWE-639		CWE-639
A5 (CWE-814)	CWE-352	CWE-352	CWE-352
A6 (CWE-815)	CWE-2	CWE-209	CWE-209
			CWE-219
			CWE-538
			CWE-552
			CWE-732
A7 (CWE-816)	CWE-310	CWE-311	CWE-312
	CWE-312	CWE-327	CWE-326
	CWE-326		CWE-327
A8 (CWE-817)	CWE-285	CWE-285	CWE-285
A9 (CWE-818)	CWE-319	CWE-311	CWE-319
A10 (CWE-819)	CWE-601	CWE-601	CWE-601

그리고 서로 다른 리스트의 항목들이 부모-자식 관계가 성립된다면, CWE의 계층구조를 고려하여 부모-자식 항목으로 리스트로 구축하여 관련 취약점들을 구체적으로 다룰 수 있게 하였다. 표 5는 표 4의 각 OWASP Top 10 항목에 대한 CWE 항목들의 부모-자식 관계를 보여준다. 표 4에서 A1의 CWE-809의 OWASP-CWE 대응에만 존재하는 CWE-90은 표 5에서 알 수 있듯이 CWE-77과 부모-자식 관계를 가지고 있다. 때문에 CWE-809의 OWASP-CWE 대응과 OWASP의 OWASP-CWE 대응에서 중복되므로 최종 통합 CWE 리스트에 포함시켰다.(표 6 참조)

표 4의 세 대응관계 중에서 부모-자식 관계를 포함하여 최소 두 대응관계에서 중복되는 항목들은 표 6의 통합 리스트에 포함하였다. 표 4와 5의 A1에서 CWE-91은 OWASP와 MITRE 대응관계에서는 존재하지 않아 통합목록에서 제외하였다. 같은 이유로, A6에서 CWE-219, CWE-538, CWE-732도 통합 리스트에서 제외되었다. CWE-552의 경우 CWE-2의 자식관계이므로 CWE-552는 통합 리스트에 포함하였다.

표 5. 각 OWASP Top 10 항목별 CWE 항목들의 부모 자식 관계
Table 5. Parent-Child Relationship of CWE Entries in Each OWASP Top 10 Rank

OWASP Top 10	부모 항목	자식 항목
A1	CWE-77	CWE-78
		CWE-88
		CWE-89
		CWE-90
	CWE-91	
A2	CWE-79	
A3	CWE-287	CWE-306
		CWE-307
		CWE-798
A4	CWE-22	
	CWE-285	
	CWE-639	
A5	CWE-352	
A6	CWE-2	CWE-552
	CWE-209	
	CWE-219	
	CWE-538	
	CWE-732	
A7	CWE-310	CWE-311
		CWE-327
		CWE-326
	CWE-312	
A8	CWE-285	
A9	CWE-311	CWE-319
A10	CWE-601	

표 5의 A4 “Insecure Direct Object Reference”와 A8 “Failure to Restrict URL Access”를 보면 공통적으로

CWE-285가 존재함을 알 수 있다. 이러한 포함 관계에 비추어 볼 때, 웹 어플리케이션을 개발할 때 A4에 대한 취약점을 제거하여 개발할 경우 A8에 대한 취약점이 제거될 수 있다. 때문에 표 4의 A4에서 CWE-285는 공통 항목이 아니지만 A8과 중복되므로 이를 유지하였다. 이와 유사한 관계가 A7과 A9에도 존재한다. A9의 CWE-319가 A7의 CWE-311의 자식 항목으로 부모-자식 관계를 이루고 있다. A7에서 CWE-311의 자식 항목들을 모두 고려한다면 A9의 CWE-319에 관련된 취약점들이 A7에서 제거될 수 있다. 때문에 표 4의 A7에서 CWE-311의 자식 항목인 CWE-319를 표 6에 포함시켰다.

2. 통합 취약점 목록과 SDLC 단계 대응

웹 애플리케이션 취약점을 '소프트웨어 개발 생명주기'(SDLC) 프로세스의 초기 단계부터 고려한다면, 보다 안전하고 고 품질의 응용을 개발할 수 있다. 즉, 특정 SDLC 단계에서 관련 주요 취약점들을 미리 고려하여 설계하거나 구현한다면, 배포 및 운영 단계에서 취약점을 패치하는 방법보다 비용과 노력면에서 효율적이다.

본 논문에서는 통합된 OWASP Top-10 와 CWE 대응 목록을 구축하고, 소프트웨어 개발 생명주기(SDLC)에 취약점의 발생 여부 정보를 추가하였다 (표 6 참조). 표 6의 취약점들은 포괄적인 OWASP Top-10 순위를 계승하였으며, 그 바탕위에 CWE 분류를 분석하여 웹 애플리케이션 취약점들을 세부적으로 더 구체화되어 표현하면서 SDLC의 관련 단계들과 사상시켰다는 점에서 의미가 있다. 표 6을 보면, OWASP "A6:Security Misconfiguration"에 대응하는 CWE-209 "Information Exposure Through an Error Message" 취약점의 경우, 소프트웨어 설계단계, 구현단계, 운영단계, 환경설정(설치) 단계 각각에서 발생될 수 있어 각 단계에서 그 취약점을 고려해야 함을 알 수 있다.

'통합된 OWASP Top 10 - CWE 리스트'에서 SDLC중 취약점 도입 단계는 설계단계가 17개로 가장 많으며, 구현단계가 11개, 운영단계가 6개, 시스템 설정 단계가 2개 순으로 나타났다. 웹 애플리케이션 취약점들이 설계단계에서 도입될 가능성이 높기 때문에 설계단계에서부터 관련 취약점들을 고려하는 것이 필요하다.

표 6은 웹 애플리케이션 개발 생명주기에서 우선 고려해야 할 취약점들의 랭킹 및 도입 단계를 보여 준다. 개발자나 테스터가 시간이나 비용 면에서 모든 취약점들을 고려할 수 없을 때, 표를 활용하여 효율적으로 해당 개발단계에서 높은 우선순위 취약점들을 미리 제거하여 웹 애플리케이션 품질도 보증하고 개발 비용을 줄여 주는 것이 본 논문의 주요 기여 중의 하나이다. 또한, 분류 기준이 매우 포괄적인 OWASP Top-10분류에 구체적인 CWE 항목들을 사상하여 웹 애플리케이션 취약점 항목들을 세분화한 것도 주요 기여이다.

표 6. 취약점 도입 단계가 추가된 통합된 OWASP Top 10 - CWE 리스트
Table 6. An Integrated List of OWASP Top 10 and CWE including Time of Introduction

OWASP Top 10	통합된 OWASP-CWE 리스트	취약점 발생 가능한 단계 (SDLC 단계 포함)
A1	CWE-78	설계/구현 단계
	CWE-88	설계/구현 단계
	CWE-89	설계/구현/운영 단계
	CWE-90	설계/구현 단계
A2	CWE-79	설계/구현 단계
A3	CWE-287	설계/구현 단계
	CWE-306	설계 단계
	CWE-307	설계 단계
	CWE-798	설계 단계
A4	CWE-22	설계/구현 단계
	CWE-285(A8)	설계/구현/운영 단계
	CWE-639	설계 단계
A5	CWE-352	설계 단계
A6	CWE-209	설계/구현/운영 단계, 환경설정
	CWE-552	구현/운영 단계
A7	CWE-311	설계/운영 단계
	CWE-319(A9)	설계/운영 단계, 환경설정
	CWE-312	설계 단계
	CWE-326	설계 단계
	CWE-327	설계 단계
A8	CWE-285	설계/구현/운영 단계
A9	CWE-319	설계/운영 단계, 환경설정
A10	CWE-601	설계/구현 단계

V. 연구결과의 적용 및 검증 예

이 장에서는 제안한 OWASP Top-10과 CWE의 대응관계를 활용하는 방법에 대해 기술한다. 그림 4는 XSS 취약점이 존재하는 asp 소스코드를 보인다. Login.asp의 경우 접속한 사용자에게 ID와 PW를 입력받아 Check.asp가 처리할 수 있도록 전송하는 역할을 하고, Check.asp는 ID와 PW를 이용하여 사용자가 로그인 할 수 있는지를 검사하고, 로그인 할 수 없을 경우 ErrorMessage를 login.asp에 다시 전송해 준다.

그림 4의 errorMessage 변수에 그림 5와 같은 URL 인

자를 주고, 그 URL에 접근한 사용자가 ID와 PW를 입력하고 login 버튼을 누르면, Check.asp 가 아닌 "http://www.h4cker.co.kr/stealPW.asp"가 수행되며, stealPW.asp에 미리 정의해 둔 악성 행위를 수행할 수 있게 된다. 이와 같이 Login.asp와 Check.asp에는 XSS 취약점이 존재한다.

```
Login.asp

<form method="POST" action="Check.asp">
<%=request.querystring("ErrorMessage") %>
ID : <input type="text" name="ID" /><br/>
PW : <input type="password" name="PW" /><br/>
<input type="submit" name="log_in" value="log in!" />
</form>
```

```
pseudo code of Check.asp

If rs.eof then
not valid
Response.Redirect("Login.asp?errorMessage=Invalid+ID+or+P
W")
Else
valid
End If
```

그림 4. XSS 취약점이 있는 코드
Fig. 4. Code with a XSS vulnerability

```
http://www.somesite.com/Login.asp?errorMessage=</form><f
orm method="POST"
action="www.h4cker.co.kr/stealPW.asp">
```

그림 5. XSS를 일으키는 악성 URL
Fig. 5. Malicious URL for XSS

이를 취약점을 제거하기 위해서는 표 6의 A2에 해당하는 CWE-79의 취약점 완화 방법을 적용하여야 한다. SDLC 단계에서 현재 소스코드를 작성하는 구현 단계라고 가정한다면, 다음 5가지의 완화 방법을 적용할 수 있다[16].

- 입력받은 데이터가 다른 웹페이지로 나가는 경우, 이를 인코딩해야 함
- 웹페이지가 ISO-8859-1이나 UTF-8 등의 특정 방식으로 인코딩되도록 해야 함
- bean의 필터 속성이 설정되어 있는 form bean으로부터의 데이터로 웹 페이지를 작성해야 함
- 세션 쿠키 헤더에 HttpOnly 플래그를 설정해야 함
- 입력 값을 검증해야 하며 특히 '<' 와 같은 특수기호가 직접 입력 값으로 사용되지 않게 해야 함

Check.asp의 3번째 줄을 보면 입력받은 ID 와 PW가 인코딩 되지 않았음을 알 수 있다. 또한 입력 값에 대한 검증 없이 '<' 등의 문자들이 걸러지지 않고 그대로 사용된다. 그리고 해당 웹 페이지가 지정된 인코딩 방식으로 제공되어야 하므로 웹 페이지의 헤더에 인코딩 방식이 명시되어야 한다. 이들을 적용하여 Check.asp와 Login.asp를 다시 작성하면 그림 6과 같다. 입력 데이터의 특수 기호 등이 다른 대응 문자로 대체되도록 인코딩하여 처리하고, 입력으로부터 생성된 출력을 또다시 HtmlEncode를 통해 인코딩하여 제공함으로써 기존에 존재하던 XSS에 관련된 취약점을 배포 단계 전에 미리 제거하여 취약점으로 인한 위험을 완화하였다.

```
Login.asp

<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />

<form method="POST" action="Check.asp">
<%=Server.HtmlEncode(request.querystring("ErrorMessage"))
%>
ID : <input type="text" name="ID" /><br/>
PW : <input type="password" name="PW" /><br/>
<input type="submit" name="log_in" value="log in!" />
</form>
```

```
pseudo code of Check.asp

If rs.eof then
not valid
E_ID = Encoding(ID) // ex) replace '<' to '&lt;'
E_PW = Encoding(PW) // ex) replace '<' to '&lt;'
Response.Redirect(
            "Login.asp?errorMessage=Invalid+ID+or+PW")
Else
valid
End If
```

그림 6. XSS 취약점이 제거된 코드
Fig. 6. Code free from a XSS vulnerability

다른 취약점들도 이와 유사하게 고려함으로써 관련 보안 위협을 완화할 수 있다. 본 논문에서 제안한 기법을 널리 적용된다면 웹 애플리케이션 취약점들의 순위도 변화될 것이다. 즉, 3장에서 기술했듯이 주요 취약점에 대해 보안기법이 고려되면 해당 취약점의 확산이 줄고, 다른 취약점들이 더 많이 악용될 수 있다. 취약점 순위 변화를 미리 예측하는 것은 매우 어려우며, OWASP 그룹과 같이 수많은 전문가들이 취약점 발생 추이와 데이터를 분석하여 체계적으로 순위예측을 하는 것이 한 방법이다. 본 논문의 초점은 웹 애플리케이션 취약점 주요 목록을 제시하고 관련 취약점들이 어떤 SDLC 단계와 대응되는지를 제안하여, 효율적으로 안전한 웹 애플리케이션을 개발하자는 것이다.

Ⅵ. 결론

웹 애플리케이션 설계나 개발, 운영 단계 또는 환경설정 시에 중점을 두고 고려해야 할 주요 보안 취약점 정보를 제공할 수 있다면, 웹 보안 위협 및 취약점을 사전에 효율적으로 예방하여 애플리케이션 개발 비용을 절감하고 소프트웨어 품질을 개선할 수 있다.

본 논문에서는 안전하고 효율적으로 웹 애플리케이션을 개발할 수 있도록, 주요 취약점 항목들에 대해 분석하여 대응관계를 도출하고, 주요 취약점 항목들이 어떤 SDLC 단계에서 도입되는지를 제시하였다.

OWASP Top 10 취약점들 랭킹 변화를 살펴보고 이 취약점들의 실제 발생 비율을 미국 NVD를 이용하여 분석하여, OWASP Top 10 취약점들 랭킹의 타당성을 검증하였다. 다음으로, 2010년도 OWASP Top 10 리스트를 중심으로 CWE들과의 관련성을 분석하여 더 구체적이고 세분화된 취약점 항목들의 랭킹을 제안하였다. 또한, 각 취약점 항목들이 애플리케이션 개발 단계 중, 어느 단계에서 도입되는 지를 나타내었다. 최종적으로 제안한 방법의 실제 적용 예를 보였다.

향후, 웹 애플리케이션 통합 취약점 항목들 각각을 개발단계에서 취약점을 제거하거나 완화하는 기법을 더 효율적으로 적용하는 연구를 수행할 계획이다.

참고 문헌

[1] Kukinews, “[Financial hacking is an Emergency] Hacking Method Viewed by Experts”, Apr. 11, 2011. Available Online at *http://news.kukinews.com/article/view.asp?page=1&gCode=kmi&arcid=0004844041&cp=du* Accessed in Oct. 2011

[2] WhiteHat Security, Inc., "Measuring Website Security: Windows of Exposure", WhiteHat Website Security Statistics Report, 11th Edition, Winter 2011, *http://img.en25.com/Web/WhiteHatSecurityInc/WPstats_winter11_11th.pdf*

[3] National Institute of Standards and Technology. National Vulnerability Database (NVD). Available at: *http://nvd.nist.gov*, 2011.

[4] R. A. Martin, S. M. Christey and J. Jarzombek, "The Case for Common Flaw Enumeration", NIST Workshop on Software Security Assurance Tools, Techniques and Metrics, November, 2005.

[5] R. A. Martin and S. Barnum, "A Status Update: The Common Weaknesses Enumeration", Proc. of the Static Analysis Summit (NIST Special Publication 500-262), pp. 62-64, July 2006.

[6] A. Tripathi and U.K. Singh, “Towards Standardization of Vulnerability Taxonomy", Proc. of the 2nd International Conference on Computer Technology and Development (ICCTD), pp. 379-384, Nov. 2010.

[7] K. Tsipenyuk, B. Chess and G. McGraw, "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors", IEEE Security & Privacy, pp. 81-84, Nov./Dec. 2005.

[8] J. A. Wang, H. Wang, M. Guo and M. Xia, "Security metrics for software systems", Proc. of the 47th Annual Southeast Regional Conference (ACM-SE-47), 2009.

[9] A. Wiesmann, A. van der Stock, M. Curphey, R. Stirbei, A Guide to Building Secure Web Applications and Web Services, OWASP, 2005.

[10] The Open Web Application Security Project (OWASP), Available Online at *http://www.owasp.org*. Accessed in Sep. 2011

[11] Homeland Security: Common Weakness Enumeration (CWE), Available Online at http://cwe.mitre.org. Accessed in Sep. 2011

[12] M. Howard, D. LeBlanc, and J. Viega, 19 Deadly Sins of Software Security - Programming Flaws and How to Fix Them, McGraw-Hill, 2005

[13] S. Wagner, D. M. Fernandez, S. Islam, and K. Lochmann, "A Security Requirements Approach for Web Systems", Proc. of Quality Assessment in Web (QAW 2009), CEUR, 2009.

[14] P. Mell, K. Scarfone and S. Romanosky, "Common Vulnerability Scoring System", IEEE Security & Privacy, pp. 85-89, Nov./Dec. 2006.

[15] Y. Kim, S. Shin, J. Ahn, O. Lee, E. Lee and H. Han, “Analysis and Documentation of Korean Common Weakness Enumeration for Software Security”,

Communications of the Korean Institute of Information Scientists and Engineers, Vol. 28, No. 2, pp. 20-31, Feb. 2010.

[16] CWE-79 Improper Neutralization of Input During Web Page Generation('Cross-site Scripting'), Available Online at *http://cwe.mitre.org/data/definitions/79.html*, Accessed in Oct. 2011

저 자 소 개

문 재 찬

2011 : 단국대학교 컴퓨터과학과 이학사.

2011~현재 : 단국대학교 컴퓨터학과 공학석사과정.

관심분야 : 컴퓨터 보안, 웹 보안. 스마트폰 보안 등

Email : answocks@dankook.ac.kr

조 성 제

1989 : 서울대학교 컴퓨터공학과 공학사.

1991 : 서울대학교 컴퓨터공학과 공학석사.

1996 : 서울대학교 컴퓨터공학과 공학박사.

2001 : 미국 University of California, Irvine 객원 연구원

2009 : 미국 University of Cincinnati 객원 연구원

1997~현재 : 단국대학교 소프트웨어학과 교수

관심분야 : 컴퓨터보안, 소프트웨어 보증, 시스템소프트웨어, 실시간스케줄링, 임베디드 소프트웨어 등

Email : sjcho@dku.edu