REPORT

제 목 : 기말고사 대체 과제 보고서

발표희망일 6월 10일

과 목 명 : 시큐어코딩
담당교수 : 우사무엘 교수님
이　　름 : 조 정 민
학　　번 : 32164420

단국대학교
Dankook University

목차

Ⅰ. 시큐어코딩에 대한 설명

시큐어코딩이란?

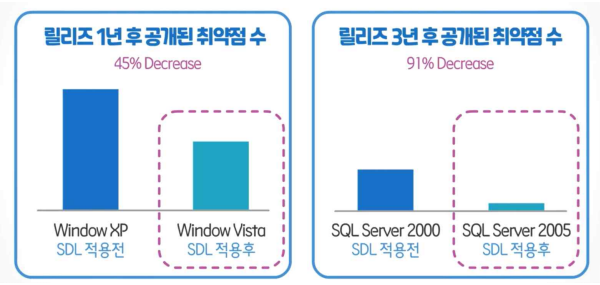
소프트웨어 개발 보안은 크게 광의적 의미와 협의적 의미로 나눌 수 있다. 광의적 의미의 소프트웨어 개발 보안은 소프트웨어 개발 생명주기의 각 단계에서 수행되는 모든 보안 활동을 뜻한다. 협의적 의미의 소프트웨어 개발 보안은 소프트웨어 개발 과정 중 소스코드 구현 단계에서 보안 약점을 배제시키기 위해 수행하는 시큐어코딩 활동이다.

즉, 시큐어코딩을 간단하게 말하자면 안전한 소프트웨어를 개발하기 위한 보안 활동이며 취약점을 발생시키지 않게 하거나 취약점을 최소화시키기 위한 개발 방법이다. 좀 더 자세하게 설명하자면, 안전한 소프트웨어를 개발하기 위해 소스 코드에 존재하는 약점, 취약점 등을 제거하고 보안을 고려하여 기능을 설계, 구현하는 일련의 과정이며, 소프트웨어 개발 과정 중 설계 단계와 구현 단계에 수행된다. 시큐어코딩을 통해 우리가 개발한 소프트웨어나 서비스에 소속된 보안 자산을 외부 위협으로부터 안전하게 지키고, 사이버 공격으로부터 발생할 수 있는 위험도를 낮출 수 있다.

시큐어코딩의 필요성

소프트웨어는 우리가 살아가는 현실 세계에서 많은 영향을 주고 있다. 이러한 상황에서 시큐어코딩은 선택이 아닌 필수가 되었다.

과거 시큐어코딩의 부재로 발생했던 사고 사례로는 웹 서비스, 모바일 및 PC 어플리케이션 뿐만 아니라 자율 주행 자동차 등을 포함한 매우 다양한 사례들이 있다. 쉽게 말해, 소프트웨어가 있는 모든 곳에서 사고 사례가 발생한다고 볼 수 있다. 이 사고 사례들이 내포하고 있는 가장 큰 공통점은 시스템 개발 과정에서 시큐어코딩을 고려하지 않았다는 점이다. 이 때문에, 실제 오류와 약점을 기반으로 위협이 발생했고, 이런 사례로부터 알 수 있는 사실은 보안 자산을 지키기 위해 소프트웨어 개발에서 시큐어코딩이 필수라는 것이다.



실제로, 마이크로소프트에서는 안전한 소프트웨어를 만들기 위해 시큐어코딩이 포함된 소프트웨어 개발 보안 방법론을 개발하여 자사 제품 개발에 사용했다. 그 결과, 제품 출시 후 공개된 취약점 수가 현저히 감소했다. 이러한 결과로만 봐도, 시큐어코딩이 얼마나 필요하고 중요한 지 알 수 있다.

Ⅱ. 사례 기반 정보보호 용어 정의 및 설명

용어 정의 및 설명

보안 자산(Security Asset) 취약점(Vulnerability) 위협원(Threat Agent)

위협행동(Threat Action) 위험(RISK) 보안 조치(Security Measure)

시큐어코딩의 주요 정보보호 용어가 있다. 보안 자산(Security Asset), 취약점(Vulnerability), 위협원(Threat Agent), 위협행동(Threat Action), 위험(RISK), 보안 조치(Security Measure) 등이 있다. 이러한 용어들의 의미와 차이점은 무엇인지 설명하도록 하겠다.

보안 자산(Security Asset)

보안 자산은 우리가 개발한 소프트웨어나 운영하고 있는 서비스에서 가치가 있고 보호해야 하는 중요한 요소이다. 국내 표준에서는 정보(데이터), 소프트웨어(컴퓨터 소프트웨어 등), 물리적 자산(서버 등), 서비스, 인력 등 조직에서 보유하고 있는 가치가 있는 모든 것으로 정의하고 있다. 풀어서 설명하자면, 이는 제품의 이해 관계자에게 가치가 있는 것이라고 볼 수 있고, 유형 자산과 무형 자산으로 나뉘어질 수 있다.

유형 자산으로는 현금, 장비, 하드웨어 등이 될 수 있고, 무형 자산으로는 소프트웨어, 영업권 및 지적재산권이 될 수 있다. 우리가 배우는 시큐어코딩 관점에서의 보안 자산은 소프트웨어이다.

취약점(Vulnearability)

취약점은 시스템을 위협의 영향에 노출시키는 시스템의 약점을 의미한다. 사용자에게 허용된 권한 이상의 동작이나 허용된 범위 이상의 정보 열람을 가능하게 하는 약점이다. 이와 함께 사용자 및 관리자의 부주의나 사회공학 기법에 의한 약점도 취약점에 포함된다. 취약점은 우리가 지켜야하는 보안 자산에 위치하고 있다. 이런 취약점 때문에 보안 자산은 위협에 노출되어 보안사고가 발생할 수 있는 것이다.

취약점과 약점이라는 용어의 차이점은 다음과 같다. 약점은 개발 단계에서 발생하는 보안 관련 오류이며 공격에 활용될 여지가 있는 오류이다. 취약점은 운영 단계에서 발생하는 보안 관련 오류이며 실제로 공격 구현이 가능한 오류를 뜻한다.

위협원(Threat Agent)

위협원은 우리가 지켜야하는 보안 자산을 노리는 공격자 또는 해커 등을 의미한다. 일부 논문에서는 어떤 목적을 가지고 위협을 수행하거나 그 원인이 될 수 있는 활동을 수행하고 지원하는 사람이나 조직 등을 위협원이라 정의하고 있다. 여기서 말하는 위협은 우리가 지켜야하는 보안 자산에 손실을 초래할 수 있는, 원치 않는 사건의 잠재적 원인이나 행위를

뜻한다.

위협 행동(Threat Action)

위에서 말한 위협원이 보안 자산을 공격하는 행위. 즉, 보안 자산에 손실을 초래할 수 있는 행위를 위협 행동이라 한다. 위협 행동은 보안 자산에 피해를 입힐 의도로 목표 대상에 취하는 모든 조치이다. 공격자가 자신의 목표를 달성하기 위한 전략을 포함된 일련의 행위를 위협 행동이라 부를 수 있다. 위협원은 위협 행동을 통해 보안 자산에 내포하고 있는 취약성을 분석하여 사용한다.

위험(RISK)

위험은 보안 자산에 내포된 취약점을 이용하여 위협원이 위협 행동을 수행하면 그에 따른 피해를 뜻한다. 표준에서는 위험을 외부의 위협이 내부의 취약성을 이용하여 보유한 각종 자산에 피해를 입힐 수 있는 잠재적인 가능성이라 정의한다.

※ 위협과 위험의 비교

위협은 위험을 일으킬 소지가 다분한 요소를 의미한다. 일반적으로 공격자의 위협 행위를 원천 차단할 수 없기 때문에 위협은 제어되지 않는다. 위험은 위협 요소가 일으킬 수 있는 피해를 의미한다. 보안 자산에 내포되어 있는 취약성은 제거할 수 있기 때문에 취약점을 미리 찾아서 조치한다면 위험은 줄일 수 있으므로 관리가 가능하다고 볼 수 있다.

보안 조치(Security Measure)는 공격자와 방어자의 관점에 따라 나뉘어질 수 있다. 우선 공격자 관점에서의 보안 조치의 개념이다.

공격자 관점에서의 보안 조치(Security Measure)

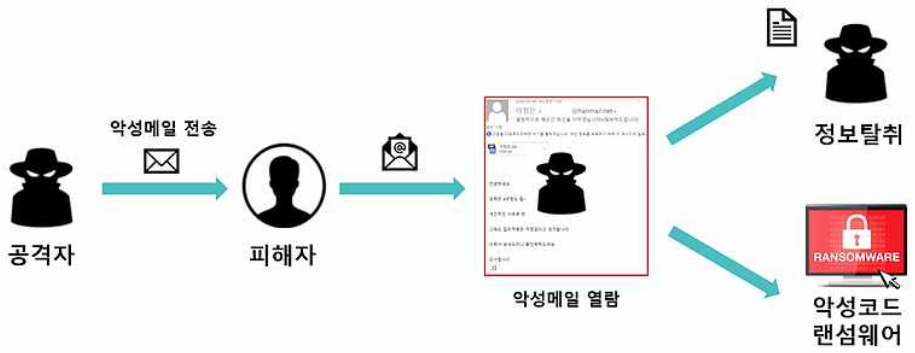
우리는 위험을 줄이기 위해 보안 자산에 내포된 취약점을 찾아 제거해야 한다. 이러한 행위를 보안 조치라고 부른다. 보안 조치를 취함으로써 공격자의 위협 행위로부터 발생하는 보안 위험을 최소화할 수 있다. 보안 조치를 수행할 때에는 경제적인 관점의 보안 균형이 매우 중요하다. 보안 자산을 지키기 위한 보안 조치가 보안 자산의 가치보다 많은 비용이 발생하면 보안 조치를 수행하는 것이 어려울 것이다. 이렇기 때문에 위험과 비용의 균형을 맞춘 보안 조치를 수행하는 것이 매우 중요하다. 경제적 측면을 고려하면서 보안 대책을 정의하려면 체계적인 접근이 필요하며, 보안 자산의 위험을 미리 알아야 한다.

다음은 방어자 관점에서 바라보는 보안 조치의 개념이다.

방어자 관점에서의 보안 조치(Security Measure)

방어자 관점에서의 보안 조치는 기밀성, 무결성 또는 가용성의 손실을 예방하는 기술적 대책이다. 보안 요구사항을 구체적이고 기술적으로 설계하고 개발하는 과정이라고 볼 수 있다.

업무 사칭 악성메일에 의한 정보 유출 & 악성코드 유포 사례



업무 사칭 악성메일로 인한 공격은 공격자가 어려 종류의 업무 관련 메일을 미리 수집하여 메일 발송자를 겨냥해 악성 파일의 압축본을 첨부해 회신하는 과정을 통해 이루어졌다. 공격 사례들의 공통점으로는 업무상 메일에 대한 회신이다. 공격자는 메일 발신자에게 공격 파일을 열어보도록 유도하고 해당 공격 파일을 실행한 발신자는 공격자가 심어놓은 악성코드에 감염된다. 악성코드는 C&C(명령제어) 서버에 접속해 랜섬웨어, 정보유출 악성코드 등을 추가로 실행할 수 있다. 추가적으로, 탈취한 정보들을 이용하여 메일 발신자에게 협박을 하여 금품을 요구받는 등의 피해도 발생했다.

위협원

위 사례에서 위협원은 악성 첨부파일을 발송하고, 금품을 요구한 공격자/해커에 해당한다.

보안자산

메일 발신자가 소속된 회사의 업무용 파일, 파일에 포함된 회사 내 정보, 개인 정보, 개인의 금품 등이 포함된다.

취약점

업데이트 되지 않은 구버전의 OS와, 오프되어 있는 백신, 메일 송수신에 대한 암호화의 부재로 볼 수 있다.

위협

메일 발송자가 보낸 메일을 탈취하는 행위, 보안이 취약한 PC에 악성 코드를 첨부 파일 형식으로 전송해 감염시키는 행위 등이 포함된다.

위험

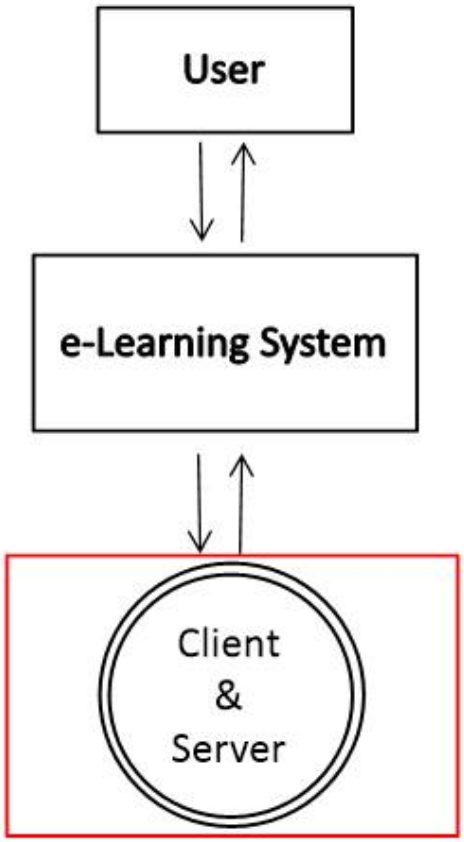
컴퓨터 시스템 감염, 회사 내 주요 정보 유출, 개인 정보 유출, 금품 탈취 등이 포함된다.

보안대책

백신 설치 및 최신화, OS 최신화, 중요 송수신 메일에 대한 암호화, 메일에 대한 경계심 등이 보안대책에 활용될 수 있다.

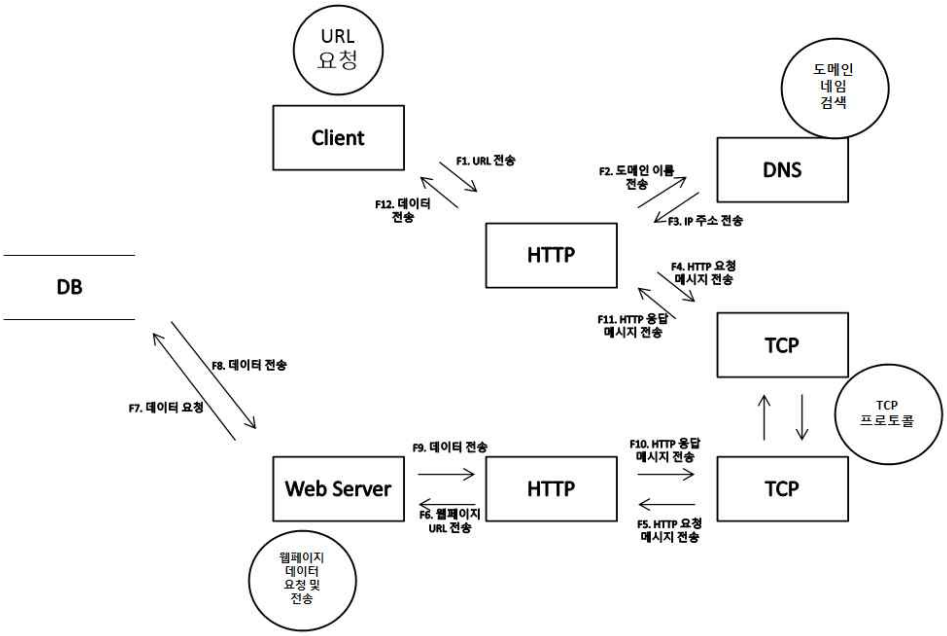
Ⅲ. 이러닝시스템 위협 모델링

DFD 그리기 – LV0



Lv0 Data Flow Diagram이다. e-Learning System이라는 개체가 중심이 된다. User 개체와 연결이 되어있으며 유저가 원하는 정보, 보여지는 데이터들은 Clinet와 Server 다중 프로세스를 통해 제공될 수 있다고 생각했다. 실질적으로 우리 학교 이러닝 시스템에서 주를 이루고 있는 프로세스는 프론트엔드와 백엔드이므로 해당 부분에 대해 Data Flow Diagram을 계속 그려보도록 하겠다.

DFD 그리기 – LV1



Lv1 Data Flow Diagram이다. 얼핏 보면 단순한 웹 애플리케이션의 구조로 보일 수 있지만 이러닝 시스템을 웹으로 이용하는 사용자가 많은만큼 이러닝 시스템의 전반적인 분석을 하기에 제일 적합하다고 생각했다. 모든 사용자는 클라이언트라고 볼 수 있고, 클라이언트에서 요청하는 데이터를 웹서버에서 처리하기 때문에 이 부분에 대한 Data Flow Diagram을 그렸다.

F1부터 F12에 해당하는 데이터 흐름은 다음과 같다.

F1	사용자가 웹 브라우저를 통해 찾고 싶은 웨페이지의 URL 주소를 입력한다.
F2	사용자가 입력한 URL 주소 중에서 도메인 네임 부분을 DNS 서버에서 검색한다.
F3	DNS 서버에서 해당 도메인 네임에 해당하는 IP 주소를 찾아 사용자(교수, 학생)가 입력한 URL 정보와 함께 전달한다.
F4, F5	웹 페이지 URL 저보와 전달받은 IP 주소는 HTTP 프로토콜을 사용하여 HTTP 요청 메시지를 생성한다. 이렇게 생성된 HTTP 요청 메시지는 TCP 프로토콜을 사용하여 인터넷을 거쳐 해당 IP 주소의 컴퓨터로 전송된다.
F6	도착한 HTTP 요청 메시지는 HTTP 프로토콜을 사용하여 웹 페이지 URL 정보로 변환된다.
F7	웹 서버는 도착한 웹 페이지 URL 정보에 해당하는 데이터를 조회한다.
F8 ~ F10	조회된 웹 페이지 데이터를 다시 HTTP 프로토콜을 사용하여 HTTP 응답 메시지를 생성한다. 생성된 HTTP 응답 메시지는 TCP 프로토콜을 사용하여 인터넷을 거쳐 원래 컴퓨터로 전송한다.
F12	도착한 HTTP 응답 메시지는 HTTP프로토콜을 사용하여 웹페이지 데이터로 변환된다.

위협 식별하기

Lv1 Data Flow Diagram에서 식별된 위협은 다음과 같다.

No.	위협
A1	정보 전송 행위의 부인
A2	다른 클라이언트로 위장
A3	정보 요청 행위의 부인
A4	요청 데이터 정보 변경
A5	개인 정보 유출
A6	다른 URL로 위장
A7	도메인 이름 변경
A8	IP 주소 정보 변경
A9	악성 프로그램 배포 및 서비스 마비

프로세스 자체가 클라이언트에서 서버 쪽으로 데이터를 요청하여 받는 과정이기 때문에 우선 정보 전송/요청의 부인, 정보 변경, 정보 유출 등의 위협이 다수 존재했다. URL, 도메인, IP 주소를 변경하여 공격을 당할 수 있는 위협 또한 존재했다.

식별된 위협을 STRIDE에 매칭시키기 & 보안 대책 제시

Data Flow	위협	STRIDE	방어 기법
F1	A1. 정보 전송 행위의 부인	Repudiation	감사로그, 전자서명
	A2. 다른 클라이언트로 위장	Spoofing	인증, 전자서명
F2	A1. 정보 전송 행위의 부인	Repudiation	감사로그, 전자서명
	A7. 도메인 이름 변경	Spoofing	인증, 전자서명
F3	A1. 정보 전송 행위의 부인	Repudiation	감사로그, 전자서명
	A8. IP 주소 정보 변경	Spoofing	인증, 전자서명
F4	A1. 정보 전송 행위의 부인	Repudiation	감사로그, 전자서명
	A3. 정보 요청 행위의 부인	Repudiation	감사로그, 전자서명
F5	A1. 정보 전송 행위의 부인	Repudiation	감사로그, 전자서명
	A3. 정보 요청 행위의 부인	Repudiation	감사로그, 전자서명
F6	A1. 정보 전송 행위의 부인	Repudiation	감사로그, 전자서명
	A6. 다른 URL로 위장	Spoofing	인증, 전자서명
	A9. 악성 프로그램 배포 및 서비스 마비	Dos	트래픽 방지 /방화벽
F7	A3. 정보 요청 행위의 부인	Repudiation	감사로그, 전자서명
F8	A1. 정보 전송 행위의 부인	Repudiation	감사로그, 전자서명
	A4. 요청 데이터 정보 변경	Tampering	메시지 인증, 해시, 전자서명
	A5. 개인 정보 유출	Information disclosure	해시
F9	A1. 정보 전송 행위의 부인	Repudiation	감사로그, 전자서명
	A4. 요청 데이터 정보 변경	Tampering	메시지 인증, 해시, 전자서명
	A5. 개인 정보 유출	Information disclosure	해시
F10	A1. 정보 전송 행위의 부인	Repudiation	감사로그, 전자서명
	A3. 정보 요청 행위의 부인	Repudiation	감사로그, 전자서명
F11	A1. 정보 전송 행위의 부인	Repudiation	감사로그, 전자서명
	A3. 정보 요청 행위의 부인	Repudiation	감사로그, 전자서명
F12	A1. 정보 전송 행위의 부인	Repudiation	감사로그, 전자서명
	A4. 요청 데이터 정보 변경	Tampering	메시지 인증, 해시, 전자서명
	A5. 개인 정보 유출	Information disclosure	해시
	A9. 악성 프로그램 배포 및 서비스 마비	Dos	트래픽 방지 /방화벽

DFD에서 나타낸 데이터 흐름 F1 ~ F12까지의 위협을 각각 선정해주었다. 크게 위장, 부인, 정보변경, 정보유출, 악성 프로그램 배포 및 서비스 마비 등의 위협이 있었고 그에 따른 STRIDE를 위의 표에서 매칭시켰다.

IV. 학생 활동 보고서

1) SQL 삽입(Injection)

2017년 '여기어때' 고객정보 유출 사건

2017년, '여기어때'를 이용하는 고객들의 개인정보가 노출되고 해당 고객들에게 불쾌한 내용의 문자 메시지들이 발송됐다. 해커는 여기어때 마케팅센터 웹페이지에 SQL 인젝션 공격을 통해 DB에 저장된 관리자 세션값을 탈취했다. 탈취한 관리자 세션값으로 외부에 노출된 서비스 관리 웹페이지를 관리자 권한으로 우회접속하여 예약 정보, 제휴점 정보 및 회원 정보를 유출했다. 유출된 개인정보 수는 중복제거를 했음에도 불구하고 약 323만건이며, 제휴점 정보는 약 1100여 건, 회원정보 약 17만 건이다. 특히, 예약정보에는 숙박일수, 제휴점명부터 예약일시, 예약자, 회원번호, 휴대폰 번호, 결제방법과 금액까지 민감하게 여겨질 정보들이 여럿 포함되어 있었다.

해당 사건과 매우 유사한 취약점에 대한 설명을 CVE에서 찾아봤다. 해당 취약점 넘버링은 'CVE-2020-14054'이다. 우선, 공격자는 웹 페이지 로그인 입력 폼에 비정상적인 입력값을 삽입해 인증 우회가 가능한지 그 가능성을 확인할 수 있다. 아이디와 비밀번호 입력 폼에 입력할 문자 조합은 *' or'a'=a* 이다.

입력값	' or'a'='a
의도한 쿼리문	select * from table where id='' or 'a'='a' and password='' or 'a'='a'
설명	아이디와 비밀번호는 빈 입력으로 주어지고 'a'='a'라는 항상 참이되는 값이 반환되게 한다. 이는, 모든 테이블의 정보를 확인하라는 명령어로 변경된다.

SQL Injection 공격에 취약하다는 사실을 확인한 공격자는 2차 공격을 수행한다. 여기서 아이디 입력폼에는 *공격아이디' #*으로 입력하고 비밀번호 입력폼에는 의미없는 데이터를 삽입한다.

아이디 입력값	공격아이디' #
의도한 쿼리문	select * from table where id = '공격아이디' # and password = 'string'
설명	아이디 입력폼의 문자조합에서 #은 주석을 의미하며 #이하 문자열들은 주석처리가 된다. 이로써, 패스워드를 확인하지않고 공격아이디로 로그인할 수 있다.

'여기어때 고객정보 유출사건'의 공격자는 이러한 공격 과정을 거쳐 로그인을 우회하고 제한된 영역에 대한 엑세스 권한을 얻은 것으로 보이며 탈취한 관리자 계정으로 고객 정보를 조회한 것으로 보인다. 해당 취약점에 대한 SQL Injection은 널리 흔하게 알려진 방법이지만 쉽게 사용할 수 있는 공격에 비해 큰 피해를 입힐 수 있다.

`CVE-2020-14054'에서 위 취약점과 공격에 대한 설명을 하고 있다. 앞서 설명한 공격 방법과 매우 비슷한 형태의 공격 과정이다. 추가적으로, sokkia 웹사이트가 해당 취약적점에 영향을 받는 대표적인 웹사이트로 소개가 되고 있음을 확인할 수 있다.

SQL Injection 공격 유형

SQL은 관계형 데이터베이스 관리 시스템의 데이터를 관리하기 위해 설계된 특수 목적의 프로그래밍 언어다. 쉽게 말해, 데이터베이스를 관리하기 위한 프로그래밍 언어의 일종이다. SQL Injection은 데이터베이스를 액세스하기 위해 동적 SQL문을 사용하는 환경에서 사용자로부터 입력된 값에 대한 검증 작업 없이 쿼리문을 생성하는 취약점을 기반으로 동작한다.

다수의 웹 애플리케이션들은 사용자가 입력한 값을 사용하여 데이터베이스로 쿼리를 전송하고 쿼리의 실행결과를 받아서 사용자에게 처리 결과를 응답하는 구조를 가지고 있다. 이때, 사용자 입력값을 사용하여 SQL 명령문을 만들 때 정적 쿼리 방식과 동적 쿼리 방식을 사용한다. 여기서 문제가 되는 방식은 동적 쿼리 방식이다. 외부 입력값에 따라 쿼리문 구조가 바뀔 수 있기 때문에 동적 쿼리를 사용하는 경우 SQL 삽입 취약점을 가질 수 있다.

이러한 SQL Injection을 통해 비정상적인 데이터베이스 조작을 수행해 보안 자산에 심각한 피해를 발생시킬 수 있다. SQL Injection은 모든 종류의 DBMS에 적용 가능하며 현재에도 지속적으로 발전되고 있는 공격 기법이다.

대표적인 SQL Injection의 유형은 다음과 같다.
1) Form Based SQL 삽입
2) UNION SQL 삽입

이 두 가지 공격 유형에 대해 알아보겠다.

1) Form Based SQL 삽입

웹 사이트의 많은 페이지들이 폼을 이용하여 외부로부터 입력값을 받아서 데이터베이스 조회 작업이나 생성, 수정, 삭제 작업을 수행한다. 만약, 웹 페이지에 SQL Injection 취약점이 존재한다면 공격자는 아이디, 패스워드 입력 폼에 SQL 문으로 사용될 수 있는 특수문자를 입력하여 인증 과정을 우회한다.

2) UNION SQL 삽입

웹 애플리케이션이 데이터베이스 조회 결과를 리스트로 출력하는 화면을 가지고 있는 경우, UNION SQL 삽입 취약점은 데이터베이스의 모든 정보를 유출하는 데에 사용될 수 있다. 애플리케이션의 출력 화면에 사용되는 컬럼의 개수와 동일한 셀렉트 문을 UNION으로 연결하여 쿼리문을 생성한 후 실행하면 공격자는 자신이 원하는 정보를 웹 페이지에 표시할 수 있다.

보안 정보 공유 체계(CVE, CWE, NVD)에서 찾아본 SQL Injection

CWE

1350 - Weaknesses in the 2020 CWE Top 25 Most Dangerous Software Weaknesses

- Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') - *(79)*
- Out-of-bounds Write - *(787)*
- Improper Input Validation - *(20)*
- Out-of-bounds Read - *(125)*
- Improper Restriction of Operations within the Bounds of a Memory Buffer - *(119)*
- Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') - *(89)*
- Exposure of Sensitive Information to an Unauthorized Actor - *(200)*
- Use After Free - *(416)*
- Cross-Site Request Forgery (CSRF) - *(352)*
- Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') - *(78)*
- Integer Overflow or Wraparound - *(190)*
- Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') - *(22)*
- NULL Pointer Dereference - *(476)*
- Improper Authentication - *(287)*
- Unrestricted Upload of File with Dangerous Type - *(434)*
- Incorrect Permission Assignment for Critical Resource - *(732)*
- Improper Control of Generation of Code ('Code Injection') - *(94)*
- Insufficiently Protected Credentials - *(522)*
- Improper Restriction of XML External Entity Reference - *(611)*
- Use of Hard-coded Credentials - *(798)*
- Deserialization of Untrusted Data - *(502)*
- Improper Privilege Management - *(269)*
- Uncontrolled Resource Consumption - *(400)*
- Missing Authentication for Critical Function - *(306)*
- Missing Authorization - *(862)*

우선 CWE에서 SQL Injection에 대해 알아봤다. SQL Injection은 2020년 가장 위험한 소프트웨어 약점 Top 25에 포함되어 있음을 확인할 수 있다. 위 리스트는 위험도가 높은 순으로 랭크가 되어있다. SQL Injection은 6위에 랭크됨을 알 수 있다.

CVE

48 CVE-2021-27316	89	Sql	2021-03-24	2021-03-24	5.0	None	Remote	Low	Not required	Partial	None	None
Blind SQL injection in contactus.php in doctor appointment system 1.0 allows an unauthenticated attacker to insert malicious SQL queries via lastname parameter.												
49 CVE-2021-27315	89	Sql	2021-03-24	2021-03-24	5.0	None	Remote	Low	Not required	Partial	None	None
Blind SQL injection in contactus.php in Doctor Appointment System 1.0 allows an unauthenticated attacker to insert malicious SQL queries via the comment parameter.												
50 CVE-2021-27314	89	Sql	2021-03-05	2021-03-05	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
SQL injection in admin.php in doctor appointment system 1.0 allows an unauthenticated attacker to insert malicious SQL queries via username parameter at login page.												

Total number of vulnerabilities : **8688** Page : 1 (This Page) 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174

SQL Injection과 관련된 취약점이 8688개가 있었다. 그 중에서도 위에서 소개한 **'여기어때 고객정보 유출 사건'**에서 해커가 사용한 관리자 권한 우회 접속과 관련된 취약점을 CVE에서 찾아봤다.

Vulnerability Details : CVE-2020-14054

SOKKIA GNR5 Vanguard WEB version 1.2 (build: 91f2b2c3a04d203d79862f87e2440cb7cefc3cd3) and hardware version 212 allows remote attackers to bypass admin authentication via a SQL injection attack that uses the User Name or Password field on the login page.

Publish Date : 2020-06-15 Last Update Date : 2020-06-23

Collapse All Expand All Select Select&Copy Scroll To Comments External Links
Search Twitter Search YouTube Search Google

– CVSS Scores & Vulnerability Types

CVSS Score	7.5
Confidentiality Impact	Partial (There is considerable informational disclosure.)
Integrity Impact	Partial (Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.)
Availability Impact	Partial (There is reduced performance or interruptions in resource availability.)
Access Complexity	Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit.)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Sql Injection Bypass a restriction or similar
CWE ID	89

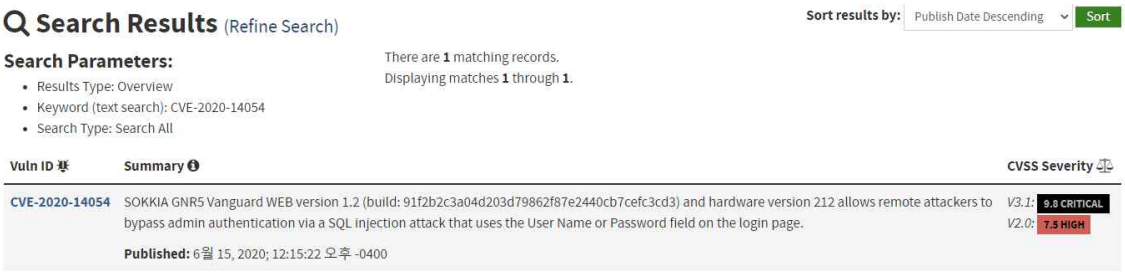
해당 취약점은 2020년 14054번을 부여받았다. 로그인 우회는 가장 널리 사용되는 SQL Injection 기술 중 하나이다. 공격자가 로그인 페이지의 아이디 또는 패스워드 입력 폼에 SQL Injection 공격을 하여 관리자 인증을 우회할 수 있음이 정의되어 있다.

CVSS 점수는 취약점의 위험도를 다음과 같이 나타낸다.

1) 저위험군: 0.1 ~ 3.9점
2) 중위험군: 4.0 ~ 6.9점
3) 고위험군: 7.0 ~ 8.9점
4) 치명적 위험군: 9.0 ~ 10.0점

해당 취약점의 위험도는 7.5점으로 고위험군에 속한다.

NVD



Search Results (Refine Search)

Sort results by: Publish Date Descending Sort

Search Parameters:

- Results Type: Overview
- Keyword (text search): CVE-2020-14054
- Search Type: Search All

There are **1** matching records.
Displaying matches **1** through **1**.

Vuln ID	Summary	CVSS Severity
CVE-2020-14054	SOKKIA GNR5 Vanguard WEB version 1.2 (build: 91f2b2c3a04d203d79862f87e2440cb7cefc3cd3) and hardware version 212 allows remote attackers to bypass admin authentication via a SQL injection attack that uses the User Name or Password field on the login page. **Published:** 6월 15, 2020; 12:15:22 오후 -0400	*V3.1:* 9.8 CRITICAL *V2.0:* 7.5 HIGH

NVD에서 위 취약점 ID를 검색한 결과이다. 해당 취약점의 요약글과 게시 날짜를 확인할 수 있다. 추가적으로, CVSS 위험도까지 확인이 가능하다.

CVE-2020-14054 Detail

Current Description

SOKKIA GNR5 Vanguard WEB version 1.2 (build: 91f2b2c3a04d203d79862f87e2440cb7cefc3cd3) and hardware version 212 allows remote attackers to bypass admin authentication via a SQL injection attack that uses the User Name or Password field on the login page.

+View Analysis Description

Severity CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 9.8 CRITICAL **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

검색한 해당 취약점의 세부 정보이다. 취약점에 대한 설명과 심각성 등이 포함되어 있다.

SQL Injection 공격 유형 별 방어 대책

대표적인 방어 방법으로는 정적 쿼리를 사용하는 것이다. 외부 입력값 삽입 시 쿼리의 구조가 바뀌지 않는 정적 쿼리를 사용하는 것이 가장 기본적인 대응 방법이다. 정적 쿼리는 외부 입력값 삽입 전, SQL 명령문을 생성하고 입력값을 자료형에 따라 바인딩하는 방식으로 쿼리문을 생성하는 구조를 가진다. 외부 입력 데이터를 쿼리문에 설정하여 쿼리 구조가 변조되는 것을 방지한다.

외부 입력값에 따라 쿼리의 구조가 바뀌는 동적 쿼리를 사용해야 하는 경우가 발생할 수도 있다. 이처럼, 정적 쿼리 형식 사용이 어려운 경우에는 특수문자 및 SQL 명령문으로 사용되는 키워드를 입력값에서 철저히 제거한 후 동적 쿼리에 삽입해 사용해야 한다. 일반적으로, 화이트리스트 기반의 입력값 검증 방식이나 ESAPI와 같은 보안 라이브러리를 사용하는 방법을 권장하고 있다.

다음은 SQL Injection 취약점 제거를 위해 필터링이 요구되는 특수문자와 구문들의 일부다.

필터링 대상				
'	"	--	#	=
(	)	*/	/*	+
<	>	&&	||	%
union	select	insert	from	where
update	drop	if	join	decalre
and	or	column_name	table_name	openrowset
substr	substring	xp_	sysobjects	syscolumns

2) 크로스사이트 스크립팅(Cross Site Scripting, XSS)

2010년 '트위터' XSS 공격 사건

2010년 유명 소셜네트워크 서비스인 트위터(Twitter)가 악성공격자에 의한 해킹 공격을 받아 포르노 사이트나 악성코드가 삽입된 사이트로 접속되는 등의 피해 사례가 발생했다. 트위터에서 나타난 공격은 XSS(Cross-Site Scripting) 공격이었다. 해당 공격은 사용자로부터 입력받은 값을 검증하지 않은 채 바로 실행해서 나오는 취약점을 악용해 사용된다. 공격자는 웹페이지에 XSS 공격 코드를 삽입해 악성코드 감염이나 다른 사이트로 이동시키는 등의 다양한 공격을 수행할 수 있다.

트위터에 나타난 XSS공격은 공격코드가 삽입된 글을 클릭 하지 않고 글에 마우스를 올려놓기만 해도 악성코드나 포르노와 같은 유해정보가 담겨있는 사이트로 연결되는 현상이 나타났다. 당시 트위터의 XSS 공격 취약점은 보안전문가들에 의해 거론되기도 했다. 해당 취약점은 개발자들이 사용하는 애플리케이션 등록 페이지의 '애플리케이션 이름' 필드값에 입력값 검증을 하지 않아 발생하는 것으로 나타났다. 해당 사례에서 공격의 영향을 받은 사람은 최소 10만명 이상이었으며 당시 백악관 대변인, 영국 전 총리 부인 등의 유명인사들도 피해를 입었다.

해당 공격은 공격자가 URL 내 특정 섹션에 'onmouseover'와 같은 자바스크립트 이벤트를 통해 임의의 웹 스크립트 또는 HTML을 삽입하면서 이루어진 공격으로 보인다. 해커가 사전에 사용할 자바스크립트 이벤트를 사용자 화면에서 작동하도록 삽입하고, 사용자는 의도치 않게 해당 공격 이벤트를 실행하게 되면서 피해가 발생하게 된다. 소개한 사례에서는 공격자가 이벤트를 삽입한 특정 섹션에 사용자가 마우스를 올리면 특정 웹사이트로 이동하는 피해가 발생했다.

XSS 공격 유형

XSS는 외부 입력 값이 충분한 검증 없이 동적으로 생성되는 응답 페이지에 사용되는 경우에 발생한다. 대표적인 XSS 공격 기법은 Reflective XSS, Stored XSS 두 가지다.

1) Reflective XSS(반사 공격)

공격자가 악성 스크립트가 포함된 URL을 클라이언트에 노출시켜 클릭을 유도하고 악성 행위를 수행하는 공격 방법이다. 공격자는 공격 대상 브라우저가 신뢰하고 있는 웹 서버를 이용한다. 공격자는 웹사이트의 XSS에 대한 취약점을 이용하여 악성 스크립트가 포함된 URL을 클라이언트에 노출시켜 클릭을 유도하고 쿠키 정보를 탈취하거나 피싱 사이트, 불법/광고 사이트로 이동하게 만들 수 있다.

2) Stored XSS

악성 스크립트를 데이터베이스에 저장해서 해당 데이터베이스 정보를 이용하는 애플리케이션을 통해 시스템을 사용하는 모든 사용자들이 해당 스크립트를 실행하여 악성 행위를 수행하게 하는 악성 공격 방법이다. 악성 스크립트를 데이터베이스에 저장한 후 해당 데이터베이스 정보를 이용하는 애플리케이션을 통해 시스템을 사용하는 모든 사용자들이 해당 스크립트를 실행하게하므로써 사용자의 쿠키 정보를 탈출하거나 악성 사이트로 이동하게 하는 공격이다. 많은 개발자들이 데이터베이스에 저장된 값을 읽어서 응답에 사용하는 경우, 그 값에 대한 XSS 취약점 검증을 철저하게 수행하지 않는다는 특징을 이용하여 공격에 사용한다.

3) DOM Based XSS

피해자의 웹 브라우저에서 응답 페이지에 포함된 정상적인 스크립트가 동작하면서 DOM 객체를 실행할 때 URL 등에 포함된 악성 스크립트가 동작하도록 하는 공격 방법이다. 응답 페이지에 관계없이 웹 브라우저에서 발생한다.

보안 정보 공유 체계(CVE, CWE, NVD)에서 찾아본 XSS

CWE

1350 - Weaknesses in the 2020 CWE Top 25 Most Dangerous Software Weaknesses

- Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') - *(79)*
- Out-of-bounds Write - *(787)*
- Improper Input Validation - *(20)*
- Out-of-bounds Read - *(125)*
- Improper Restriction of Operations within the Bounds of a Memory Buffer - *(119)*
- Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') - *(89)*
- Exposure of Sensitive Information to an Unauthorized Actor - *(200)*
- Use After Free - *(416)*
- Cross-Site Request Forgery (CSRF) - *(352)*
- Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') - *(78)*
- Integer Overflow or Wraparound - *(190)*
- Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') - *(22)*
- NULL Pointer Dereference - *(476)*
- Improper Authentication - *(287)*
- Unrestricted Upload of File with Dangerous Type - *(434)*
- Incorrect Permission Assignment for Critical Resource - *(732)*
- Improper Control of Generation of Code ('Code Injection') - *(94)*
- Insufficiently Protected Credentials - *(522)*
- Improper Restriction of XML External Entity Reference - *(611)*
- Use of Hard-coded Credentials - *(798)*
- Deserialization of Untrusted Data - *(502)*
- Improper Privilege Management - *(269)*
- Uncontrolled Resource Consumption - *(400)*
- Missing Authentication for Critical Function - *(306)*
- Missing Authorization - *(862)*

우선 CWE에서 XSS에 대해 알아봤다. XSS는 SQL Injection과 함께 2020년 가장 위험한 소프트웨어 약점 Top 25에 포함되어 있음을 확인할 수 있으며 SQL Injection(6위)보다 높은 랭크인 1위에 랭크되어 있다.

CVE

47	CVE-2021-31329	79	XSS	2021-04-21	2021-04-22	3.5	None	Remote	Medium	???	None	Partial	None
Cross Site Scripting (XSS) in Remote Clinic v2.0 via the "Chat" and "Personal Address" field on staff/register.php													
48	CVE-2021-31327	79	XSS	2021-04-21	2021-04-22	3.5	None	Remote	Medium	???	None	Partial	None
Stored XSS in Remote Clinic v2.0 in /medicines due to Medicine Name Field.													
49	CVE-2021-31250		XSS	2021-06-04	2021-06-04	0.0	None	???	???	???	???	???	???
Multiple storage XSS vulnerabilities were discovered on BF-430, BF-431 and BF-450M TCP/IP Converter devices from CHIYU Technology Inc due to a lack of sanitization of the input on the components man.cgi, if.cgi, dhcpc.cgi, ppp.cgi.													
50	CVE-2021-30637	79	XSS	2021-04-13	2021-04-16	3.5	None	Remote	Medium	???	None	Partial	None
htmly 2.8.0 allows stored XSS via the blog title, Tagline, or Description to config.html.php.													

Total number of vulnerabilities : **19227** Page : 1 (This Page) 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385

XSS와 관련된 취약점이 SQL Injection의 취약점보다 약 2.5배 많은 19227개가 있었다. 그 중에서도 위에서 소개한 **'트위터 XSS 공격 사건'**에서 해커가 사용한 'onmouseover' 이벤트 실행에 관련된 취약점을 CVE에서 찾아봤다.

Vulnerability Details : CVE-2006-3383

Cross-site scripting (XSS) vulnerability in index.php in mAds 1.0 allows remote attackers to inject arbitrary web script or HTML via Javascript events such as onmouseover within a URL. NOTE: the provenance of this information is unknown; the details are obtained solely from third party reports.

Publish Date : 2006-07-06 Last Update Date : 2017-07-20

Collapse All Expand All Select Select&Copy Scroll To Comments External Links

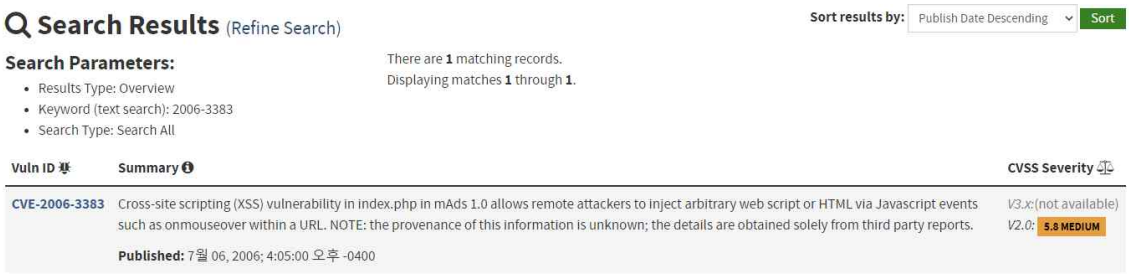
Search Twitter Search YouTube Search Google

– CVSS Scores & Vulnerability Types

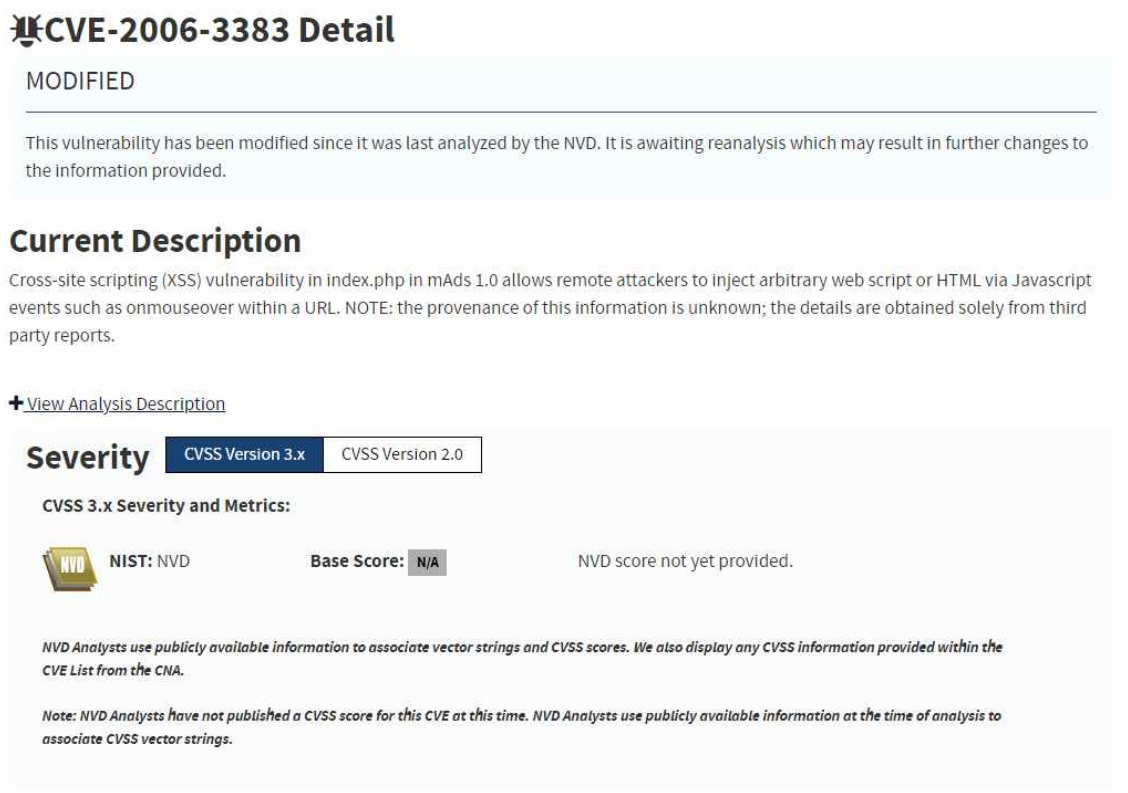
CVSS Score	5.8
Confidentiality Impact	Partial (There is considerable informational disclosure.)
Integrity Impact	Partial (Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.)
Availability Impact	None (There is no impact to the availability of the system.)
Access Complexity	Medium (The access conditions are somewhat specialized. Some preconditions must be satistified to exploit)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Cross Site Scripting
CWE ID	CWE id is not defined for this vulnerability

해당 취약점은 2006년 3383번을 부여받았다. 공격자가 해당 취약점을 이용해 URL 내의 onmouseover와 같은 자바스크립트 이벤트를 통해 임의의 웹 스크립트 또는 HTML을 삽입할 수 있음이 정의되어 있다. 공격 과정은 어느정도 전문적인 과정이며 시스템 가용성에는 영향을 미치지 않는다고 나타낸다. 추가적으로, 해당 취약점의 위험도는 5.8점으로 중위험군에 속한다.

NVD



SQL Injection과 마찬가지로 해당 취약점을 NVD에서 취약점 ID를 통해 검색했다. 해당 취약점의 요약글과 게시 날짜를 확인할 수 있으며, CVSS 위험도까지 확인이 가능하다.



검색한 해당 취약점의 세부 정보이다. 취약점에 대한 설명과 심각성 등이 포함되어 있다.

XSS 공격 유형 별 방어 대책

1) 입출력 값에 필터링 적용

XXS 취약점은 결론적으로 보면, 스크립트 코드가 사용자의 브라우저로 전달되어 실행되는 경우에 침해 사고로 이어지게 된다. 즉, Stored XSS 공격과 Reflective XSS 공격을 동시에 제거하는 확실한 방법은 출력값에 대해 철저하게 필터링하는 것이다.

이미 개발이 끝나서 서비스를 실행하고 있는 시스템의 경우에는 출력값에 대해 필터링을 적용하는 소스를 추가하거나 수정하는 것은 쉬운 일이 아니다. 이럴 때에는 차선책으로 입력값에 대해 철저하게 필터링을 하여 안전한 값만 애플리케이션이 처리할 수 있도록 정책을 설정하여 보안 기능을 강화할 수 있다.

입력값에 대한 필터링을 적용하는 경우에는 HTTP 헤더, 쿠키, 쿼리 문자열, 폼필드, 히든 필드 등 서버에 전달되어 사용될 수 있는 모든 인자들에 대해 정규식을 사용하여 허용된 유형의 데이터만을 받아들이도록 입력값 검증을 실시해야 한다.

2) 오픈소스 라이브러리 활용 XSS Filter

다음 방법으로는 오픈소스 라이브러리를 활용하여 입력값에 대한 XSS Filter를 적용하는 것이다. 필터를 우회하기 위해 각종 방식으로 인코딩되어 전달되는 데이터를 필터링하려면 인코딩 규칙까지 충분히 감안해야 하기 때문에 필터를 직접 만드는 것은 매우 어려운 작업이다. 직접 XSS Filter를 만드는 것보다 이미 검증된 필터를 활용하는 것을 권장하고 있다.

대표적인 XSS Filter는 Lucy XSS Filter이다. 해당 필터는 화이트리스트 정책을 사용하여 스크립트 공격에 위험이 있는 입력값에 대해 HTML 인코딩을 적용하여 허용하지 않은 목록을 정리하여 적용하는 블랙리스트 방식보다 안전하다.