주요정보통신기반시설

기술적 취약점 분석·평가 방법 상세가이드

08

Web($)



|  |  |  |
| --- | --- | --- |
| 1. 버퍼 오버플로우 ···································· | 645 | 15. 크로스사이트 리퀘스트 변조(CSRF) ··· 684 |
| 2. 포맷스트링 ············································ | 647 | 16. 세션 예측 686 |
| 3. LDAP 인젝션 ······································· | 649 | 17. 불충분한 인가 688 |
| 4. 운영체제 명령 실행 ······························ | 651 | 18. 불충분한 세션 만료 690 |
| 5. SQL 인젝션 ········································· | 653 | 19. 세션 고정 693 |
| 6. SSI 인젝션 ··········································· | 659 | 20. 자동화 공격 694 |
| 7. XPath 인젝션 ······································· | 661 | 21. 프로세스 검증 누락 696 |
| 8. 디렉터리 인덱싱 ···································· | 663 | 22. 파일 업로드 699 |
| 9. 정보 누출 ············································· | 668 | 23. 파일 다운로드 707 |
| 10. 악성 콘텐츠 ········································ | 672 | 24. 관리자 페이지 노출 711 |
| 11. 크로스사이트 스크립팅 ························ | 673 | 25. 경로 추적 714 |
| 12. 약한 문자열 강도 ································ | 678 | 26. 위치 공개 716 |
| 13. 불충분한 인증 ····································· | 680 | 27. 데이터 평문 전송 719 |
| 14. 취약한 패스워드 복구 ························· | 682 | 28. 쿠키 변조 721 |

08. Web(웹) 보안



Web 취약점 분석·평가 항목

|  |  |  |
| --- | --- | --- |
| 점검항목 | 항목 중요도 | 항목코드 |
| 버퍼 오버플로우 | 상 | BO |
| 포맷스트링 | 상 | FS |
| LDAP 인젝션 | 상 | LI |
| 운영체제 명령 실행 | 상 | OC |
| SQL 인젝션 | 상 | SI |
| SSI 인젝션 | 상 | SS |
| XPath 인젝션 | 상 | XI |
| 디렉터리 인덱싱 | 상 | DI |
| 정보 누출 | 상 | IL |
| 악성 콘텐츠 | 상 | CS |
| 크로스사이트 스크립팅 | 상 | XS |
| 약한 문자열 강도 | 상 | BF |
| 불충분한 인증 | 상 | IA |
| 취약한 패스워드 복구 | 상 | PR |
| 크로스사이트 리퀘스트 변조(CSRF) | 상 | CF |
| 세션 예측 | 상 | SE |
| 불충분한 인가 | 상 | IN |
| 불충분한 세션 만료 | 상 | SC |
| 세션 고정 | 상 | SF |
| 자동화 공격 | 상 | AU |
| 프로세스 검증 누락 | 상 | PV |
| 파일 업로드 | 상 | FU |
| 파일 다운로드 | 상 | FD |
| 관리자 페이지 노출 | 상 | AE |
| 경로 추적 | 상 | PT |
| 위치 공개 | 상 | PL |
| 데이터 평문 전송 | 상 | SN |
| 쿠키 변조 | 상 | CC |

643



웹(Web)

|  |  |
| --- | --- |
| BO (상) | 1. 버퍼 오버플로우 |
| 취약점 개요 | |
| 점검내용 | * 사용자가 입력한 파라미터 값의 문자열 길이 제한 확인 |
| 점검목적 | * 웹 사이트에서 사용자가 입력한 파라미터 값의 문자열 길이 제한 여부를 점검하여 비정상적 오류 발생을 차단하기 위함 |
| 보안위협 | * 웹 사이트에서 사용자가 입력한 파라미터 값의 문자열 길이를 제한하지 않는 경우 개발 시에 할당된 저장 공간보다 더 큰 값의 입력이 가능하고 이로 인한 오류 발생 시 의도되지 않은 정보 노출, 프로그램에 대한 비인가 접근 및 사용 등이 발생할 수 있음 |
| 참고 | ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드 |
| 판단기준 | 양호 : 파라미터 값에 다량의 다양한 포맷 문자열 입력 시 에러 페이지나 오류가 발생하지 않는 경우 |
| 취약 : 파라미터 값에 대한 검증 미흡으로 에러 페이지나 오류가 발생하는 경우 |
| 조치방법 | 파라미터 값을 외부에서 입력받아 사용하는 경우 입력 값 범위를 제한하며, 허용 범위를 벗어나는 경우 에러 페이지가 반환되지 않도록 조치 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 로그인 페이지에서 계정 정보 입력 시 대량의 문자열을 입력하여 에러 페이지나 오류가 발생하는지 점검  Step 2) 로그인 후 정보변경 페이지에서 가입 정보 수정 시 대량의 문자열을 입력하여 에러 페이지나 오류가 발생하는지 점검 | |

웹(Web)

|  |  |
| --- | --- |
| BO (상) | 1. 버퍼 오버플로우 |
| Step 3) 검색어 입력 시 대량의 문자열을 입력하여 에러 페이지나 오류가 발생하는지 점검  Step 4) 게시글 작성 시 대량의 문자열을 입력하여 에러 페이지나 오류가 발생하는지 점검  Step 5) URL 파라미터 값에 대량의 문자열 입력 시 에러 페이지나 오류가 발생하는지 점검  ※ 텍스트 필드에 입력 값 검증(특수문자 제한, maxlength 등)이 설정된 경우 우회 시도 필요 (예: 로컬 프록시를 사용하여 요청 시 파라미터 값에 대량의 문자열 직접 입력 등)   * 보안설정방법 * 웹 서버, 웹 애플리케이션 서버 버전을 안정성이 검증된 최신 버전으로 패치 * 웹 애플리케이션에 전달되는 파라미터 값을 필요한 크기만큼만 받을 수 있도록 변경하고 입력 값 범위를 초과한 경우에도 에러 페이지를 반환하지 않도록 설정 * 동적 메모리 할당을 위해 크기를 사용하는 경우 그 값이 음수가 아닌지 검사하여 버퍼 오버플로우를 예방하는 형태로 소스 코드 변경 * 버퍼 오버플로우를 점검하는 웹 스캐닝 툴을 이용하여 주기적으로 점검 | |
| 조치 시  영향 | 일반적으로 영향 없음 |



웹(Web)

|  |  |
| --- | --- |
| FS (상) | 2. 포맷 스트링 |
| 취약점 개요 | |
| 점검내용 | * 웹 애플리케이션에 포맷 스트링 취약점 존재 여부 점검 |
| 점검목적 | * 공격자의 포맷 스트링 취약점을 통한 악의적인 행위를 차단하기 위함 |
| 보안위협 | * C언어로 만드는 프로그램 중 변수의 값을 출력하거나 입력받을 때 입력받은 값을 조작하여 프로그램의 메모리 위치를 반환받아 메모리 주소를 변조하여 시스템의 관리자 권한을 획득할 수 있음 |
| 참고 | ※ 포맷 스트링 버그(format string bug): printf 등의 함수에서 문자열 입력 포맷을 잘못된 형태로 입력하는 경우 나타나는 취약점으로 루트 권한을 획득하는 것도 가능함. 포맷 스트링의 종류에는 여러 가지가 있으며 그 중 C언어에서 일반적으로 Data(변수)를 입·출력문에서 일정한 형태로 받아들이거나 출력하기 위하여 사용하는 기호로는 다음과 같은 것들이 있음  (예) %d, %f, %c, %s, %x, %p ...  %d : 정수형 10진수 상수  %f : 실수형 상수  %lf : 실수형 상수  %c : 문자값  %s : 문자 스트링  %u : 양의 정수(10진수)  %o : 양의 정수(8진수)  %x : 양의 정수(16진수)  %n : 쓰인 총 바이트 수  ※ %n 은 이전까지 입력되었던 문자열의 길이(Byte)수 만큼 해당 변수에 저장시키기 때문에 메모리의 내용도 변조 가능하므로 Format String 취약점에서 핵심이기도 함. 문자열의 길이를 변조시키고 싶은 값의 길이만큼 만든 후 %n을 써주게 되면 메모리상에 공격자가 원하는 값을 넣을 수 있게 됨  ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 기반 C/S 프로그램 |
| 판단기준 | 양호 : 포맷 스트링 버그를 발생시키는 문자열 입력 시 검증 로직이 존재하여 오류가 발생하지 않는 경우 |
| 취약 : 포맷 스트링 버그를 발생시키는 문자열 입력 시 검증 로직이 미흡하여 오류가 발생하는 경우 |
| 조치방법 | 웹 서버 프로그램을 최신 버전으로 업데이트하고 포맷 스트링 버그를 발생시키는 문자열에 대한 검증 로직 구현 |
| 점검 및 조치 사례 | |
| * 점검방법 | |

웹(Web)

|  |  |
| --- | --- |
| FS (상) | 2. 포맷 스트링 |
| Step 1) 웹 사이트에서 사용자가 입력한 파라미터 값에 아래와 같은 패턴 입력 후 전송 시 다른 값을 입력했을 때는 발생하지 않는 에러 반응이나 멈추는 등 이상 반응을 보이는지 확인  패턴1 - %n%n%n%n%n%n%n%n%n%n, 패턴2 - %s%s%s%s%s%s%s%s%s%s,  패턴3 - %1!n!%2!n!%3!n!%4!n!%5!n!%6!n!%7!n!%8!n!%9!n!%10!n!  패턴4 - %1!s!%2!s!%3!s!%4!s!%5!s!%6!s!%7!s!%8!s!%9!s!%10!s!   * 보안설정방법 * 컴파일러에서 문자열 입력 포맷에 대한 자체적인 검사를 내장하고 있으므로 문자열 입력 포맷 검증 후 소스 코드에 적용   (예) GCC 컴파일러는 문자열 입력 포맷과 실제 입력이 맞지 않는 경우 경고 옵션이 존재하나 해당 방식은 컴파일 과정에서만 검증 가능하며, 런타임 상황에서는 Fuzz testing을 이용하여 포맷 스트링 버그가 존재하는지 검증이 필요함   * 웹 서버 프로그램 최신 보안패치 적용 * 웹 사이트에서 사용자가 입력한 파라미터 값 처리 중에 발생한 경우 사용자 입력 값의   유효성에 대한 검증 로직을 구현 | |
| 조치 시  영향 | 일반적으로 영향 없음 |



웹(Web)

|  |  |
| --- | --- |
| LI (상) | 3. LDAP 인젝션 |
| 취약점 개요 | |
| 점검내용 | * 웹페이지 내 LDAP 인젝션 취약점 점검 |
| 점검목적 | * 취약한 시스템에 신뢰할 수 없는 LDAP 코드 삽입 공격을 통한 비인가자의 악의적인 행위를 차단하기 위함 |
| 보안위협 | * 응용 프로그램이 사용자 입력 값에 대한 적절한 필터링 및 유효성 검증을 하지 않아 공격자는 로컬 프록시를 사용함으로 LDAP 문의 변조가 가능함 * 공격 성공 시 승인되지 않은 쿼리에 권한을 부여하고, LDAP 트리 내의 내용 수정이나 임의의 명령 실행을 가능하게 하므로 적절한 필터링 로직을 구현하여야 함 |
| 참고 | ※ LDAP 인젝션: 사용자 입력을 기반으로 LDAP(Lightweight Directory Access Protocol)구문을 구축하여 웹 기반 응용 프로그램을 악용하는 데 사용되는 공격  ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 방화벽 |
| 판단기준 | 양호 : 임의의 LDAP 쿼리 입력에 대한 검증이 이루어져 변조된 쿼리가 실행되지 않는 경우 |
| 취약 : 임의의 LDAP 쿼리 입력에 대한 검증이 이루어지지 않아 변조된 쿼리가 실행되는 경우 |
| 조치방법 | 지정된 문자열만 입력 허용하고, 임의의 LDAP 쿼리 입력에 대한 검증 로직 구현 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 사용자 입력 값에 변조된 LDAP 쿼리 삽입 후 실행되는지 확인 | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ‘ | “ | -- | # | ( | ) |
| = | \*/ | /\* | + | < | > |
| user\_tables | user\_table\_columns | | table\_name | column\_name | Syscolumns |
| union | select | insert | drop | update | and |
| or | If | join | substring | from | where |
| declare | substr | openrowset | xp\_ | sysobject | % |
| \* | ; | & | | |  |  |

웹(Web)

|  |  |
| --- | --- |
| LI (상) | 3. LDAP 인젝션 |
| * 보안설정방법 * 사용자 입력 값을 화이트 리스트로 지정하여 영문(a-z, A-Z)과 숫자(0-9)만을 허용 * DN과 필터에 사용되는 사용자 입력 값에는 특수문자가 포함되지 않도록 특수문자 제거 * 특수문자를 사용해야 하는 경우 특수문자(DN에 사용되는 특수문자는 '\', 필터에 사용되는 특수문자는 =, +, <, >, #, ;, \ 등)에 대해서는 실행 명령이 아닌 일반문자로 인식되도록 처리   ※ 필터링 대상   * 웹 방화벽에 LDAP 관련 특수문자를 필터링하도록 룰셋 적용 | |
| 조치 시  영향 | 일반적으로 영향 없음 |



웹(Web)

|  |  |
| --- | --- |
| OC (상) | 4. 운영체제 명령 실행 |
| 취약점 개요 | |
| 점검내용 | * 웹 사이트 내 운영체제 명령 실행 취약점 존재 여부 점검 |
| 점검목적 | * 적절한 검증절차를 거치지 않은 사용자 입력 값에 의해 의도하지 않은 시스템 명령어가 실행되는 것을 방지하기 위함 |
| 보안위협 | * 해당 취약점이 존재하는 경우 부적절하게 권한이 변경되거나 시스템 동작 및 운영에 악영향을 줄 가능성이 있으므로 "|", "&", ";", "`" 문자에 대한 필터링 구현이 필요함 |
| 참고 | ※ CVE/NVD - 공개적으로 알려진 취약점 검색 가능 <http://cve.mitre.org/cve/search_cve_list.html> https://nvd.nist.gov/vuln/search  ※ 사용 중인 웹 서버 버전 확인, 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 방화벽 |
| 판단기준 | 양호 : 임의의 명령어 입력에 대한 검증이 이루어지는 경우 |
| 취약 : 임의의 명령어 입력에 대해 명령이 실행되는 경우 |
| 조치방법 | 취약한 버전의 웹 서버 및 웹 애플리케이션 서버는 최신 버전으로 업데이트를 적용해야 하며, 애플리케이션은 운영체제로부터 명령어를 직접적으로 호출하지 않도록 구현하는 게 좋지만, 부득이하게 사용해야 할 경우 소스 코드나 웹 방화벽에서 특수문자, 특수 구문에 대한 검증을 할 수 있도록 조치해야 함 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 에러 페이지 또는 HTTP 응답 헤더에 노출되는 웹 서버 버전 정보를 수집하여 운영체제 명령 실행 관련 알려진 취약점 검색  ※ Apache Struts 2 RCE 취약점 참고사이트 https://cwiki.apache.org/confluence/display/WW/Security+Bulletins  Step 2) 웹 애플리케이션에 전달되는 파라미터 값에 공개적으로 알려진 운영체제 명령 실행 코드 삽입 후 명령어가 실행되는지 확인 | |

웹(Web)

|  |  |
| --- | --- |
| OC (상) | 4. 운영체제 명령 실행 |
| * 보안설정방법 * 웹 방화벽에 모든 사용자 입력 값을 대상으로 악용될 수 있는 특수문자, 특수 구문 등을 필터링 할 수 있도록 규칙 적용 * 애플리케이션은 운영체제로부터 명령어를 직접적으로 호출하지 않도록 구현 * 명령어를 직접 호출하는 것이 필요한 경우에는, 데이터가 OS의 명령어 해석기에 전달되기 전에 입력 값을 검증/확인하도록 구현 * 입력 값에 대한 파라미터 데이터의 "&", "|", ";", "`" 문자에 대한 필터링 처리   ※ 참고: “&”, “|”, “`” 문자 설명   * & : 윈도우 명령어 해석기에서 첫 번째 명령이 성공했을 경우만 두 번째 명령어를 실행 * | : 첫 번째 명령어가 성공하는지에 상관없이 두 번째 명령어를 실행 * ` : 쉘 해석기가 명령어를 해석하다 역 작은따옴표(`) 내에 포함된 명령어를 만나면 기존 명령어를 계속 실행하기 전에 역 작은따옴표로 둘러싸인 명령어를 먼저 실행   (예) `ls –al`   * 웹 서버 및 웹 애플리케이션 서버는 공개적으로 알려진 취약점이 제거된 상위 버전으로 업데이트해야 함   ※ KISA 인터넷 보호나&KrCERT 보안공지 참고 https://[www.boho.or.kr/data/secNoticeList.do](http://www.boho.or.kr/data/secNoticeList.do)   * 클라이언트에서 전송되는 요청(Request) 값에 대한 엄격한 필터링 적용 및 OGNL (Object Graph Navigation Language) 표현식 사용을 금지하여 원격에서 임의의 명령어가 실행되지 않도록 구현해야 함 | |
| 조치 시  영향 | 일반적으로 영향 없음 |



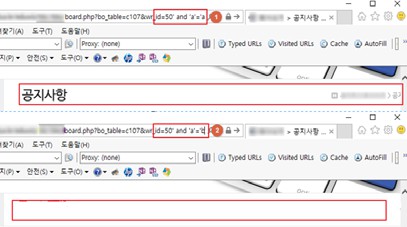
웹(Web)

|  |  |
| --- | --- |
| SI (상) | 5. SQL 인젝션 |
| 취약점 개요 | |
| 점검내용 | * 웹페이지 내 SQL 인젝션 취약점 존재 여부 점검 |
| 점검목적 | * 대화형 웹 사이트에 비정상적인 사용자 입력 값 허용을 차단하여 악의적인 데이터베이스 접근 및 조작을 방지하기 위함 |
| 보안위협 | * 해당 취약점이 존재하는 경우 비정상적인 SQL 쿼리로 DBMS 및 데이터(Data)를 열람하거나 조작 가능하므로 사용자의 입력 값에 대한 필터링을 구현하여야 함 |
| 참고 | ※ SQL인젝션: 사용자의 입력 값으로 웹 사이트 SQL 쿼리가 완성되는 약점을 이용하며, 입력 값을 변조하여 비정상적인 SQL 쿼리를 조합하거나 실행하는 공격. 개발자가 생각지 못한 SQL문을 실행되게 함으로써 데이터베이스를 비정상적으로 조작 가능함  ※ SQL인젝션 공격 관련 코드 검토 필요  ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 방화벽 |
| 판단기준 | 양호 : 임의로 작성된 SQL 쿼리 입력에 대한 검증이 이루어지는 경우 |
| 취약 : 임의로 작성된 SQL 쿼리 입력에 대한 검증이 이루어지지 않는 경우 |
| 조치방법 | 소스코드에 SQL 쿼리를 입력 값으로 받는 함수나 코드를 사용할 경우, 임의의 SQL 쿼리 입력에 대한 검증 로직을 구현하여 서버에 검증되지 않는 SQL 쿼리 요청 시 에러 페이지가 아닌 정상 페이지가 반환되도록 필터링 처리하고 웹 방화벽에 SQL 인젝션 관련 룰셋을 적용하여 SQL 인젝션 공격을 차단함 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 사용자 입력 값에 특수문자나 임의의 SQL 쿼리를 삽입하여 DB 에러 페이지가 반환되는지 확인 | |

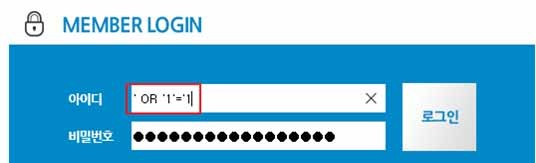
웹(Web)

SI (상) 5. SQL 인젝션

Step 2) 사용자 입력 값에 임의의 SQL 참, 거짓 쿼리를 삽입하여 참, 거짓 쿼리에 따라 반환되는 페이지가 다른지 확인



Step 3) 로그인 페이지에 참이 되는 SQL 쿼리를 전달하여 로그인되는지 확인



* 보안설정방법
* SQL 쿼리에 사용되는 문자열의 유효성을 검증하는 로직 구현
* 아래와 같은 특수문자를 사용자 입력 값으로 지정 금지 (아래 문자들은 해당 데이터베이스에 따라 달라질 수 있음)

|  |  |
| --- | --- |
| 문자 | 설명 |
| ‘ | 문자 데이터 구분기호 |
| ; | 쿼리 구분 기호 |
| --, # | 해당라인 주석 구분 기호 |
| /\* \*/ | \* 와 \*/ 사이 구문 주석 |

* Dynamic SQL 구문 사용을 지양하며 파라미터에 문자열 검사 필수적용



웹(Web)

|  |  |
| --- | --- |
| SI (상) | 5. SQL 인젝션 |
| * 시스템에서 제공하는 에러 메시지 및 DBMS에서 제공하는 에러 코드가 노출되지 않도록 예외처리 * 웹 방화벽에 인젝션 공격 관련   □ ASP.net   * 문자열 유효성 검증 로직 구현   (예) 특정 문자열 필터링 적용 (※ 예로 제시한 것으로, 구현 시 다를 수 있음)  request로 입력 값을 가져오는 경우 입력 값에서 특수문자를 제거하여 바인딩하는 소스 삽입 replaceAll() 메소드를 사용하여 구현  private string SafeSqlLiteral(string inputSQL)  {  Str = inputSQL.Replace("'","''");  Str = str. Replace(";","");  Str = str. Replace("--","");  Str = str. Replace("|","");  Str = str. Replace(":","");  Str = str. Replace("+","");  Str = str. Replace("\","");  Str = str. Replace("/","");  .....  return str;  }   * Dynamic SQL   Private void cmdLogin\_Click(object sender, System.EventArgs e) { string strCnx = ConfigurationSettings.AppSettings[“cnxNWindBad”]; Using (SqlConnection cnx = new SqlConnection(strCnx))  {  SqlParameter prm; Cnx.Open();  string strQry =  “SELECT Count(\*) FROM Users WHERE UserName = @username “ + “AND Password = @password”;  Int intRecs;  SqlCommand cmd = new SqlCommand(strQry, cnx); cmd.CommandType = CommandType.Text;  prm = new SqlParameter(“@username”,SqlDbType.VarChar,50); prm.Direction = ParameterDirection.Input;  prm.Value = txtUser.Text; cmd.Parameters.Add(prm);  prm = new SqlParameter(“@password”,SqlDbType.VarChar,50); prm.Direction = ParameterDirection.Input;  prm.Value = txtPassword.Text; cmd.Parameters.Add(prm);  intRecs = (int) cmd.ExecuteScalar(); if(intRecs > 0) {  FormsAuthentication.RedirectFromLoginPage(txtUser.Text, false);  }  else {  lblMsg.Text = “Login attempt failed.”;  }  }  } | |

웹(Web)

|  |  |
| --- | --- |
| SI (상) | 5. SQL 인젝션 |
| □ JSP   * 문자열 유효성 검증 로직 구현   (예) 특정 문자열 필터링 적용 (※ 예로 제시한 것으로, 구현 시 다를 수 있음)  request로 입력 값을 가져오는 경우 입력 값에서 특수문자를 제거하여 바인딩하는 소스 삽입 replaceAll() 메소드를 사용하여 구현  public static String makeQuery(String str) {  String result = ""; if(str != null) {  result = chkNull(replace(str, "'", ""));  result = chkNull(replace(str, ";", ""));  result = chkNull(replace(str, "--", ""));  result = chkNull(replace(str, "|", ""));  result = chkNull(replace(str, ":", ""));  result = chkNull(replace(str, "+", ""));  result = chkNull(replace(str, "\", ""));  result = chkNull(replace(str, "/", ""));  result = chkNull(replace(str.toLowerCase(), "select", "")); result = chkNull(replace(str.toLowerCase(), "update", "")); result = chkNull(replace(str.toLowerCase(), "delete", "")); result = chkNull(replace(str.toLowerCase(), "insert", "")); result = chkNull(replace(str.toLowerCase(), "where", "")); result = chkNull(replace(str.toLowerCase(), "from", "")); result = "'"+result+"'";  }  return result;  }  public static String chkNull(String str) { if (str == null)  return "";  else  return str;  }   * Dynamic SQL 구문 사용 금지   (예1) PreparedStatement 객체 사용 (※ 예로 제시한 것으로, 구현 시 다를 수 있음)  try{  String tableName = props.getProperty("jdbc.tableName"); String name = props.getProperty("jdbc.name")  String qury = "SELECT \* FROM ? WHERE Name = ?"; stmt = con.perpareStatement(query);  stmt.setString(1, tableName); stmt.setString(2, name);  rs = stmt.executeQuery();  .....  }  catch (SQLException sqle){ } finally { }  (예2) JDO API 사용 시 외부 입력 값이 위치하는 부분을 “?”로 설정하여 실행 시 해당 파라미터가 실행되도록 수정 (※ 예로 제시한 것으로, 구현 시 다를 수 있음)  try{ | |



웹(Web)

|  |  |
| --- | --- |
| SI (상) | 5. SQL 인젝션 |
| Properties props = new Properties(); String filename = “contacts.txt”;  FileInputStream in = new FileInputStream(filename); Props.load(in);  name = props.getProperty(“name”);  if (name = null || “”.equals(name)) return null; query += “ where name = ?”;  }  catch (IOException e)  {  Javax.jdo.Query q = pm.newQuery(query); return (List<Contact>) q.execute(name);  }  (예3) J2EE Persistence API 사용 시 파라미터를 받는 쿼리를 생성하고 파라미터를 설정하여 실행 (※ 예로 제시한 것으로, 구현 시 다를 수 있음)  try{  Properties props = new Properties(); String filename = “contacts.txt”;  FileInputStream in = new FileInputStream(filename); Props.load(in);  String id = props.getProperty(“id”);  If (id == null || “”.equals(id)) id = “itemid”;  Query query = em.createNativeQuery(“Select OBJECT(i) from Item I where i.itemID > :id”);  Query.setParameter(“id”, id);  . }  (예4) mybatis Data Map 사용 시 쿼리에 삽입되는 Name 파라미터를 #name# 형태로 받아  실행 (※ 예로 제시한 것으로, 구현 시 다를 수 있음)  <?xml version=”1.0” encoding=”UTF-8”?>  .....  <!-- static SQL 사용 -->  <delete id=”delStudent” parameterClass=”Student”> DELETE STUDENTS  WHERE NUM = #num# AND Name = ‘#name#’  </delete>  □ PHP  \* 문자열 유효성 검증 로직 구현  (예1) addslashes 함수를 이용한 특정 문자열 필터링 적용 (※ 예로 제시한 것으로, 구현 시 다를 수 있음)  $query = sprintf("SELECT id,password,username FROM user\_table WHERE\_ id='%s';",addslashes($id));  // id 변수를 문자형으로 받고, id 변수의 특수문자를 일반문자로 변환  // @로 php 에러 메시지를 막음  $result = @OCIParse($conn, $query);  if (!@OCIExecute($result)) error("SQL 구문 에러"); exit;  @OCIFetchInto($result,&$rows);  … 중략 … | |

웹(Web)

|  |  |
| --- | --- |
| SI (상) | 5. SQL 인젝션 |
| (예2) eregi\_replace 함수를 이용한 특정 문자열 필터링 적용 (※ 예로 제시한 것으로, 구현 시 다를 수 있음)  function SQL\_Injection($get\_Str) {  return eregi\_replace("  ( select| union| insert| update| delete| drop|\"|\'|#|\/\\*|\\*\/|\\\|\;)","",  $get\_Str);  }  (예3) php.ini 설정 중 magic\_quotes\_gps 옵션을 이용하여 특정 문자열 필터링 적용  # GPC(Get, Post, Cookie)를 통해 넘어오는 문자열 중 ', ", \, NULL 값의 앞에 자동으로  백슬래쉬 문자를 붙여주는 기능을 함 (PHP 6.0 이후 버전 사용 불능)  magic\_quotes\_gpc = on  \* Dynamic SQL 구문 사용 금지  (예1) Static SQL 구문 사용 (※ 예로 제시한 것으로, 구현 시 다를 수 있음)  $sql = 'SELECT ID, PASSWORD, USER\_NAME FORM DB WHERE VALUES = ? ';  $stmt = $mysqli->prepare($sql);  $stmt->bind\_param('s', '1');  $stmt->execute();  $stmt->bind\_result($ID, $PASSWORD, $USER\_NAME); // 칼럼수만큼 변수로 지정  while($stmt->fetch()) {  printf("%s %s\n", $ID, $PASSWORD, $USER\_NAME);  }  $stmt->close();  $mysqli->close();  (예2) mybatis Data Map 사용 시 쿼리에 삽입되는 Name 파라미터를 #name# 형태로 받아 실행 (※ 예로 제시한 것으로, 구현 시 다를 수 있음)  <?xml version=”1.0” encoding=”UTF-8”?>  .....  <!-- static SQL 사용 -->  <delete id=”delStudent” parameterClass=”Student”> DELETE STUDENTS  WHERE NUM = #num# AND Name = ‘#name#’  </delete> | |
| 조치 시 영향 | 문자열 유효성 검증 로직 구현 시 웹 서비스에서 사용하고 있는 명령어 및 특수문자가 필터링 되어 장애가 발생할 수 있으므로 사전 영향도 분석 필요 |



웹(Web)

|  |  |
| --- | --- |
| SS (상) | 6. SSI 인젝션 |
| 취약점 개요 | |
| 점검내용 | * 웹페이지 내 SSI 인젝션 공격 가능성 점검 |
| 점검목적 | * 적절한 입력 값 검증 절차를 마련하여 악의적인 파일을 include 시키지   못하도록 하여 불법적인 데이터 접근을 차단하기 위함 |
| 보안위협 | * 해당 취약점이 존재할 경우 웹 서버 상에 있는 파일을 include 시켜 명령문이 실행되게 함으로 불법적으로 데이터에 접근할 수 있음 * 공통 SSI 구현은 외부의 파일을 Include 할 수 있는 명령어를 제공하며, 웹 서버의 CGI 환경 변수를 설정하고 출력할 수 있고, 외부의 CGI 스크립트나 시스템 명령어들을 실행할 수 있으므로 사용자 입력 값에 대한 검증 로직을 추가로 구현하여야 함 |
| 참고 | ※ SSI(Server-Side Includes): CGI 프로그램을 작성하거나 혹은 서버사이드 스크립트를 사용하는 언어로, 웹 서버가 사용자에게 페이지를 제공하기 전에 구문을 해석하도록 지시하는 역할을 함  ※ SSI(Server-Side Includes) 인젝션: HTML 문서 내 입력받은 변수 값을 서버 측에서 처리할 때 부적절한 명령문이 포함 및 실행되어 서버의 데이터가 유출되는 취약점  ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 서버, 웹 방화벽 |
| 판단기준 | 양호 : 사용자 입력 값에 대한 검증이 이루어지는 경우 |
| 취약 : 사용자 입력 값에 대한 검증이 이루어지지 않는 경우 |
| 조치방법 | 사용자 입력 값에 대한 검증 로직 추가 구현 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 사용자가 입력 가능한 파라미터 값에 <!--#echo var=”DOCUMENT\_ROOT” -->를 삽입하여 전송 후 반환되는 페이지에 사이트의 홈 디렉터리가 표시되는지 확인 | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 변경 전 | < | > | “ | ( | ) | # | & |
| 변경 후 | &lt; | &gt; | &quot; | &#40 | &#41 | &#35 | &amp |

웹(Web)

|  |  |
| --- | --- |
| SS (상) | 6. SSI 인젝션 |
| Step 2) 사용자가 입력 가능한 파라미터 값에 <!-- #exec cmd=”ls -al” -->를 삽입하여 전송 후 반환되는 페이지에 디렉터리의 파일 리스트가 표시되는지 확인  Step 3) HTTP 요청(Request) 헤더에 명령어를 삽입하여 실행되는지 확인 (※ 예로 제시한 것으로, 웹 사이트 환경에 맞춰 점검하여야 함)  GET / HTTP/1.0  Referer: <!--#exec cmd="/bin/ps ax"-->  User-Agent: <!--#include virtual="/proc/version"-->   * 보안설정방법 * 사용자 입력으로 사용 가능한 문자들을 정해놓음 * 정해진 문자들을 제외한 나머지 모든 문자들을 필터링 함 * 필터링 해야 하는 대상은 GET 질의 문자열, POST 데이터, 쿠키, URL, 그리고 일반적으로 브라우저와 웹 서버가 주고받는 모든 데이터를 포함하며, 아래는 특수문자에 대한 Entity 형태를 표시한 것임 * 웹 서버의 SSI 기능을 사용하지 않거나, 웹 방화벽에 특수문자를 필터링하도록 룰셋 적용 | |
| 조치 시  영향 | 일반적으로 영향 없음 |



웹(Web)

|  |  |
| --- | --- |
| XI (상) | 7. XPath 인젝션 |
| 취약점 개요 | |
| 점검내용 | * 웹페이지 내 조작된 XPath 쿼리 공격 가능성 점검 |
| 점검목적 | * XPath 쿼리에 대한 적절한 필터링을 적용하여 웹 사이트의 로직 손상 및 특정 데이터 추출을 차단하기 위함 |
| 보안위협 | * 해당 취약점이 존재할 경우 프로그래머가 의도하지 않았던 문자열을 전달하여 쿼리문의 의미를 왜곡시키거나 그 구조를 변경하고 임의의 쿼리를 실행하여 인가되지 않은 데이터를 열람할 수 있으므로 적절한 필터링 로직 구현이 필요함 |
| 참고 | ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 방화벽 |
| 판단기준 | 양호 : 쿼리 입력 값에 대해 검증이 이루어지는 경우 |
| 취약 : 쿼리 입력 값에 대해 검증이 이루어지지 않는 경우 |
| 조치방법 | 쿼리 입력 값에 대해 검증 로직 추가 구현 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) ['and'a'='a, 'and'a'='b], [ and 1=1, and 1=2]의 셋트의 값을 각각 삽입하여 쿼리의 참, 거짓에 따라 반환되는 페이지가 다른지 확인  Step 2) 다음 값을 입력해서 에러가 발생하지 않는지 확인  ‘ or count(parent::\*[position()=1])=0 or ‘a’='b ‘ or count(parent::\*[position()=1])>0 or ‘a’='b 1 or count(parent::\*[position()=1])=0  1 or count(parent::\*[position()=1])>0 | |

웹(Web)

|  |  |
| --- | --- |
| XI (상) | 7. XPath 인젝션 |
| * 보안설정방법   \* XPath 쿼리에 사용자가 값을 입력할 수 있는 경우, 엄격한 입력 값 검증을 통해 필요 문자만을 받아들이게 함 ( ) = ‘ [ ] : , \* / 등 XPath 쿼리를 파괴하는 특수문자는 입력하지 못하게 하여야 하며, 특정 특수문자만을 필터링하는 것이 아닌 허용된 문자 이외의 모든 입력을 허용하지 않아야 함 | |
| 조치 시  영향 | 일반적으로 영향 없음 |



웹(Web)

|  |  |
| --- | --- |
| DI (상) | 8. 디렉터리 인덱싱 |
| 취약점 개요 | |
| 점검내용 | * 웹 서버 내 디렉터리 인덱싱 취약점 존재 여부 점검 |
| 점검목적 | * 디렉터리 인덱싱 취약점을 제거하여 특정 디렉터리 내 불필요한 파일 정보의 노출을 차단 |
| 보안위협 | * 해당 취약점이 존재할 경우 브라우저를 통해 특정 디렉터리 내 파일 리스트를 노출하여 응용시스템의 구조를 외부에 허용할 수 있고, 민감한 정보가 포함된 설정 파일 등이 노출될 경우 보안상 심각한 위험을 초래할 수 있음 |
| 참고 | ※ 디렉터리 인덱싱 취약점: 특정 디렉터리에 초기 페이지 (index.html, home.html, default.asp 등)의 파일이 존재하지 않을 때 자동으로 디렉터리 리스트를 출력하는 취약점 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 서버 |
| 판단기준 | 양호 : 디렉터리 파일 리스트가 노출되지 않는 경우 |
| 취약 : 디렉터리 파일 리스트가 노출되는 경우 |
| 조치방법 | 웹 서버 설정을 변경하여 디렉터리 파일 리스트가 노출되지 않도록 설정 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) URL 경로 중 확인하고자 하는 디렉터리까지만 주소창에 입력하여 인덱싱 여부 확인  Step 2) 디렉터리 끝에 %3f.jsp 문자열을 붙여 디렉터리 인덱싱이 되는지 확인 | |

웹(Web)

|  |  |
| --- | --- |
| DI (상) | 8. 디렉터리 인덱싱 |
| * 보안설정방법   \* 웹 서버 환경설정에서 디렉터리 인덱싱 기능 제거  ※ 웹 서버 별 상세 설정   * Apache   httpd.conf 파일 내 DocumentRoot 항목의 Options에서 Indexes 제거 Indexes가 해당 디렉터리의 파일 목록을 보여주는 지시자임  설정 전  <Directory “/var/www/html”>  Options Indexes  </Directory>  설정 후  <Directory “/var/www/html”>  Options  </Directory>   * IIS 7.0   설정 > 제어판 > 관리도구 > “인터넷 서비스 관리자” 선택 후 해당 웹 사이트에서 우클릭 후 등록 정보 > [홈 디렉터리] 탭 > [디렉터리 검색] 체크 해제   * IIS 7.5/8.0/8.5/10.0   IIS(인터넷 정보 서비스) 관리자 > [해당 웹 사이트] > [IIS] > [디렉터리 검색] 선택  우측의 [사용 안 함] 버튼을 눌러 비활성화 | |



웹(Web)

|  |  |
| --- | --- |
| DI (상) | 8. 디렉터리 인덱싱 |
| * WebtoB 설정   Step 1) ${WEBTOBDIR}/config/http.m 파일 Options 항목에서 index 옵션 삭제 또는, –index 옵션으로 설정 ( default: -index )  Step 2) ${WEBTOBDIR}/config/http.m에서 확인  # ${WEBTOBDIR}/config/http.m  \*NODE  GuideSample WEBTOBDIR=”/home/user/webtob”,  SHMKEY = 54000,  DOCROOT=”/home/user/webtob/docs”, PORT = “8080”,  HTH = 1, LOGGING = "log1",  ERRORLOG = "log2",  Options = “-index”  Step 3) 확인 후 설정파일 컴파일 및 재구동  # wscfl -i http.m (http.m 파일 컴파일) # wsdown  # wsboot (재구동)  ∎iPlanet  Step 1) 관리자 콘솔에서 설정 (※ 1번 또는, 2번 방법 중 선택 적용)  관리자 콘솔 > Server Name > Virtual Server Class > Class Manage > Content  Mgmt > Document Preferences > Directory Indexing 항목 "None" 설정 | |

웹(Web)

|  |  |
| --- | --- |
| DI (상) | 8. 디렉터리 인덱싱 |
| Step 2) 설정 파일에서 설정  /[iPlanet Dir]/https-[Server\_name]/config/obj.c | |



웹(Web)

|  |  |
| --- | --- |
| DI (상) | 8. 디렉터리 인덱싱 |
| ∎%3f.jsp 취약점 제거  웹 서버를 Apache로 사용한다면 아래와 같이 설정하여 %3f.jsp 문자를 필터링해야 하며, Resin이나 Tomcat을 사용한다면 최신 버전으로 업그레이드함  <LocationMatch “/(%3f|\?)\.jsp”>  AllowOverride None Deny from all  </LocationMatch>  Resin 2.1.x 버전은 최신 버전으로 업그레이드하거나 아래와 같이 설정할 수 있음  Step 1) Resin 환경설정 파일 (resin.conf)에서 가상 디렉터리 설정 부분인 “web-app id”를 찾음  Step 2) 아래 내용 추가  <directory-servlet>none</directory-servlet>  ※ 주의할 점: 모든 가상 디렉터리에 적용 필요 | |
| 조치 시  영향 | 일반적으로 영향 없음 |

웹(Web)

|  |  |
| --- | --- |
| IL (상) | 9. 정보 누출 |
| 취약점 개요 | |
| 점검내용 | * 웹 서비스 시 불필요한 정보가 노출되는지 여부 점검 |
| 점검목적 | * 웹 서비스 시 불필요한 정보가 노출되는 것을 방지함으로써 2차 공격에   활용될 수 있는 정보 노출을 차단하기 위함 |
| 보안위협 | * 웹 사이트에 중요정보(개인정보, 계정정보, 금융정보 등)가 노출되거나 에러 발생 시 과도한 정보(애플리케이션 정보, DB 정보, 웹 서버 구성 정보, 개발 과정의 코멘트 등)가 노출될 경우 공격자들의 2차 공격을 위한 정보로 활용될 수 있음 |
| 참고 | ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 서버 |
| 판단기준 | 양호 : 웹 사이트에 중요정보가 노출되지 않고, 에러 발생 시 과도한 정보가 노출되지 않는 경우 |
| 취약 : 웹 사이트에 중요정보가 노출되거나, 에러 발생 시 과도한 정보가 노출되는 경우 |
| 조치방법 | 웹 사이트에 노출되는 중요정보는 마스킹을 적용하여야 하며, 발생 가능한 에러에 대해 최소한의 정보 또는 사전에 준비된 메시지만 출력함 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 웹 사이트에 중요정보가 평문으로 노출되고 있는지 확인  Step 2) 웹페이지에 마스킹 된 중요정보가 웹페이지 소스에 평문으로 노출되고 있는지 확인 | |



웹(Web)

|  |  |
| --- | --- |
| IL (상) | 9. 정보 누출 |
| Step 3) 에러 메시지 또는 에러 페이지에서 과도한 정보가 노출되는지 확인  Step 4) 인코딩된 중요정보는 디코딩 가능한지 확인  Step 5) 임의의 계정으로 로그인을 시도하여 반환되는 에러 메시지를 통해 특정 ID의 가입 여부를 식별할 수 있는지 확인   * 보안설정방법 * 사용자가 주민등록번호 뒷자리, 비밀번호 입력 시 별표 표시하는 등 마스킹 처리를 하여 주변 사람들에게 노출되지 않도록 함 * 개인정보의 조회, 출력 시 아래와 같은 원칙으로 일부 정보에 마스킹을 적용하여 표시   1. 성명 중 이름의 가운데 글자 (ex : 홍\*동)   2. 생년월일 (ex : \*\*\*\*년 \*\*월 \*\*일   3. 전화번호 또는 휴대전화번호 (ex : 02-\*\*\*\*-5678, 010-\*\*\*\*-5678)   4. 주소의 읍/면/동 (ex : 서울시 송파구 \*\*\*동)   5. IP v4 주소의 경우 17~24bit, IP v6 주소의 경우 113~128bit * 웹페이지를 운영 서버에 이관 시 주석은 모두 제거하여 이관 * 중요정보(개인정보, 계정정보, 금융정보 등)를 HTML 소스에 포함하지 않도록 함 * 로그인 실패 시 반환되는 에러 메시지는 특정 ID의 가입 여부를 식별할 수 없게 구현 (예: ‘가입하지 않은 아이디이거나, 잘못된 비밀번호입니다.’) * 일반적으로 웹에서 발생하는 에러 메시지는 400, 500번대의 에러 코드를 반환하는데 이러한 | |

웹(Web)

|  |  |
| --- | --- |
| IL (상) | 9. 정보 누출 |
| 에러 코드에 대해 별도의 에러 페이지로 Redirect 하거나 적절한 에러처리 루틴을 설정하여 처리되도록 함(전체적인 통합 에러 페이지를 작성한 후 모든 에러 코드에 대해 통합 에러 페이지로 Redirect 되도록 설정)  ※ 웹 서버 별 상세 설정   * Apache   ErrorDocument 500 “Error Message” ErrorDocument 404 “/your web root/error.html” ErrorDocument 404 “/your web root/error.html”  ErrorDocument 402 http:/xxx.com/error.html  위와 같이 특정 에러 코드에 대해 에러 메시지를 출력할 수도 있고 특정 웹 페이지로 Redirect 시킬 수 있으며, 이 설정은 httpd.conf 의 전역 설정에 추가하거나 원하는 가상 호스트의 <VirtualHost> </VirtualHost> 사이에 추가하면 됨   * IIS 5.0, 6.0   인터넷 정보 서비스(IIS) 관리 > 속성 > [사용자 지정 오류] 탭에서 400, 401, 403, 404, 500 등 웹 서비스 에러에 대해 별도 페이지 지정   * IIS 7.0, 7.5, 8.0, 8.5, 10.0 설정   Step 1) 에러 메시지 설정  인터넷 정보 서비스(IIS) 관리자 > 해당 웹 사이트 > [오류 페이지]에서 400, 401, 403, 404,  500 등 웹 서비스 에러에 대해 별도 페이지 지정 | |



웹(Web)

|  |  |
| --- | --- |
| IL (상) | 9. 정보 누출 |
| Step 2) 오류 페이지 설정 편집  인터넷 정보 서비스(IIS) 관리자 > 해당 웹 사이트 > 오류 페이지 > [기능 설정 편집]에서 “서버 오류 발생 시 다음 반환“ 항목을 ”사용자 지정 오류 페이지“로 설정 | |
| 조치 시  영향 | 일반적으로 영향 없음 |

웹(Web)

|  |  |
| --- | --- |
| CS (상) | 10. 악성 콘텐츠 |
| 취약점 개요 | |
| 점검내용 | * 게시판 등에 악성 콘텐츠 삽입 및 실행 여부 점검 |
| 점검목적 | * 사이트 내 악의적인 콘텐츠 삽입 및 실행을 방지하기 위함 |
| 보안위협 | * 웹 사이트 게시판, 댓글, 자료실 등에 정상적인 콘텐츠 대신 악성 콘텐츠를 주입하여 실행될 경우 사용자가 해당 콘텐츠 열람 시 악성코드 감염 및 웹 페이지 변조 등 보안상 심각한 위험에 노출될 수 있음 |
| 참고 | ※ 기반시설 특성상 원칙적으로 업로드 기능을 제한해야 하나 꼭 사용해야 하는 경우 특정 사용자만 허용된 확장자의 콘텐츠 파일을 업로드 할 수 있게 구현 필요  ※ 관련 점검 항목 : XS(상), FU(상)  ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 방화벽 |
| 판단기준 | 양호 : 악의적 콘텐츠가 실행되지 않는 경우 |
| 취약 : 악의적 콘텐츠가 입력되며, 실행되는 경우 |
| 조치방법 | 사용자 입력 값에 대한 검증 로직 추가 및 실행 제한 설정 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 콘텐츠 삽입 및 파일 업로드 제한 필터링 적용 여부 점검  Step 2) 게시판 등의 페이지에서 강제적으로 이뤄지는 악의적인 프로그램 다운로드 및 콘텐츠 자동 실행이나 악의적인 사이트로의 이동이 발생하는지 확인   * 보안설정방법 * 악성 콘텐츠가 삽입되어있는 페이지에 대하여 증거자료(화면, 소스 등)를 남기고, 삽입된 악성 콘텐츠를 삭제하거나 페이지의 삭제 등을 실시함   취득한 증거자료를 가지고 악성 콘텐츠의 삽입 원인에 대하여 분석하여 원인을 제거할 것을 권고함   * 게시판의 글 등록 및 파일 업로드 기능에 Flash 파일이나 avi 동영상 파일, exe 실행 파일 등 악성코드가 포함될 수 있는 콘텐츠를 삽입 또는 업로드 하지 못하게 필터링 적용 * 주기적으로 업로드된 파일을 대상으로 바이러스 검사 실시 | |
| 조치 시  영향 | 일반적으로 영향 없음 |



|  |  |
| --- | --- |
| XSS에 취약한 페이지 유형 | 1. HTML을 지원하는 게시판 2. Search Page 3. Join Form Page 4. Referrer를 이용하는 Page 5. 그 외 사용자로부터 입력받아 화면에 출력하는 모든 페이지에서   발생 가능 |
| XSS를 유발할 수 있는 스크립트 | <script> … </script>  <img src=“javascript:…..”>  <div style=“background-image:url(javascript…)”></div>  <embed>…</embed> |

웹(Web)

|  |  |
| --- | --- |
| XS (상) | 11. 크로스사이트 스크립팅 |
| 취약점 개요 | |
| 점검내용 | * 웹 사이트 내 크로스사이트 스크립팅 취약점 존재 여부 점검 |
| 점검목적 | * 웹 사이트 내 크로스사이트 스크립팅 취약점을 제거하여 악성 스크립트의 실행을 차단 |
| 보안위협 | * 웹 애플리케이션에서 사용자 입력 값에 대한 필터링이 제대로 이루어지지 않을 경우, 공격자는 사용자 입력 값을 받는 게시판, URL 등에 악의적인 스크립트(Javascript, VBScript, ActiveX, Flash 등)를 삽입하여 게시글이나 이메일을 읽는 사용자의 쿠키(세션)를 탈취하여 도용하거나 악성코드 유포 사이트로 Redirect 할 수 있음 |
| 참고 | ※ 크로스사이트 스크립팅: 악의적인 사용자가 공격하려는 사이트에 스크립트를 넣는 기법으로 공격 방식은 크게 stored 공격 방식과 reflected 공격 방식으로 나누어 짐  ※ OWASP - XSS 필터링 관련 참고사항 [https://www.owasp.org/index.php/XSS\_Filter\_Evasion\_Cheat\_Sheet](http://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)  ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 방화벽 |
| 판단기준 | 양호 : 사용자 입력 값에 대한 검증 및 필터링이 이루어지는 경우 |
| 취약 : 사용자 입력 값에 대한 검증 및 필터링이 이루어지지 않으며, HTML 코드가 입력·실행되는 경우 |
| 조치방법 | 웹 사이트의 게시판, 1:1 문의, URL 등에서 사용자 입력 값에 대해 검증 로직을 추가하거나 입력되더라도 실행되지 않게 하고, 부득이하게 웹페이지에서 HTML을 사용하는 경우 HTML 코드 중 필요한 코드에 대해서만 입력되게 설정 |
| 점검 및 조치 사례 | |
| * 점검방법   ※ XSS 취약 유형 | |

웹(Web)

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| XS (상) | 11. 크로스사이트 스크립팅 |
| <iframe></iframe>  ※ Filtering을 우회하기 위해 다양한 표현 가능   * %3Cscript%3E........%3Cscript%3E ◆ Jav&#97;script; * Java&#13;script ◆ Java&#0013;script   Step 1) 사용자 입력 값을 전달받는 애플리케이션(회원정보 변경, 게시판, 댓글, 자료실 등)에 스크립트 입력 후 실행되는지 확인  [게시글에 스크립트 삽입(stored)]  [스크립트 코드 동작] | |



웹(Web)

|  |  |
| --- | --- |
| XS (상) | 11. 크로스사이트 스크립팅 |
| Step 2) 사용자 입력 값을 전달받는 애플리케이션(검색, URL)에 스크립트 입력 후 실행되는지 확인   * 보안설정방법 * 웹 사이트에 사용자 입력 값이 저장되는 페이지는 공격자가 웹 브라우저를 통해 실행되는 스크립트 언어(HTML, Javascript, VBScript 등)를 사용하여 공격하므로 해당되는 태그 사용을 사전에 제한하고, 사용자 입력 값에 대한 필터링 작업이 필요함 * 게시물의 본문뿐만 아니라 제목, 댓글, 검색어 입력 창, 그 외 사용자 측에서 넘어오는 값을 신뢰하는 모든 form과 파라미터 값에 대해서 필터링을 수행함 * 입력 값에 대한 필터링 로직 구현 시 공백 문자를 제거하는 trim, replace 함수를 사용하여 반드시 서버 측에서 구현되어야 함 * URLDecoder 클래스에 존재하는 decode 메소드를 통해 URL 인코딩이 적용된 사용자 입력 값을 디코딩함으로써 우회 공격 차단 * 웹 방화벽에 모든 사용자 입력 폼(회원정보 변경, 게시판, 댓글, 자료실, 검색, URL 등)을 대상으로 특수문자, 특수 구문 필터링하도록 룰셋 적용   ※ 필터링 조치 대상 입력 값   * 스크립트 정의어 : <SCRIPT>, <OBJECT>, <APPLET>, <EMBED>, <FORM>, <IFRAME> 등 * 특수문자 : <, >, ", ', &, %, %00(null) 등   ※ 웹 애플리케이션 별 상세 설정   * ASP   <%  … 중략 …  If use\_HTML Then  content = Server.HTMLEncode(content)  … 중략 …  Sub ReplaceStr(content, byref str) content = replace(content, "'", " \'")  content = replace(content, "&", "&amp;") content = replace(content, " \'", "&quot") content = replace(content, "<", "&lt") content = replace(content, ">", "&gt")  str = content | |

웹(Web)

XS (상) 11. 크로스사이트 스크립팅

End Sub

… 중략 …

%>

* PHP

… 중략 …

if($use\_html == 1) // HTML tag를 사용해야 하는 경우 부분 허용

$memo = str\_replace("<", "&lt", $memo);// HTML TAG 모두 제거

$tag = explode(",", $use\_tag);

for($i=0; $i<count($tag); $i++) { // 허용할 TAG만 사용할 수 있도록 변경

$memo = eregi\_replace("&lt".$tag[$i]." ", "<".$tag[$i]." ", $memo);

$memo = eregi\_replace("&lt".$tag[$i].">", "<".$tag[$i].">", $memo);

$memo = eregi\_replace("&lt/".$tag[$i], "</".$tag[$i], $memo); }

else // HTML tag를 사용하지 못하게 할 경우

$memo = str\_replace("<", "&lt", $memo);

$memo = str\_replace(">", "&gt", $memo);

… 중략 …

* JSP

<%

… 중략 …

string subject = request.getParameter("subject\_BOX"); subject = subject.replaceAll(“<”, “&lt”);

subject = subject.replaceAll(“>”, “&gt”);

… 중략 …

%>

※ 참고: 필터링 대상

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| < | > | &lt; | &gt; | innerHTML |
| javascript | eval | onmousewheel | onactive | onfocusout |
| expression | charset | ondataavailable | oncut | onkeyup |
| applet | document | onafteripudate | onclick | onkeypress |
| meta | string | onmousedown | onchange | onload |
| xml | create | onbeforeactivate | onbeforecut | onbounce |
| blink | append | onbeforecopy | ondbclick | onmouseenter |
| link | binding | onbeforedeactivate | ondeactivate | onmouseout |
| style | alert | ondatasetchaged | ondrag | onmouseover |
| script | msgbox | cnbeforeprint | ondragend | onsubmit |
| embed | refresh | cnbeforepaste | ondragenter | onmouseend |
| object | void | onbeforeeditfocus | ondragleave | onresizestart |
| iframe | cookie | onbeforeuload | ondragover | onuload |
| frame | href | onbeforeupdate | ondragstart | onselectstart |
| frameset | onpaste | onpropertychange | ondrop | onreset |
| ilayer | onresize | ondatasetcomplete | onerror | onmove |



웹(Web)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| layer | onselect | oncellchange | onfinish | onstop |
| bgsound | base | onlayoutcomplete | onfocus | onrowexit |
| title | onblur | onselectionchange | vbscript | onerrorupdate |
| onbefore | onstart | onrowsinserted | onkeydown | onfilterchage |
| onmouseup | onfocusin | oncontrolselected | onrowsdelete | onlosecapture |
| onrowenter | onhelp | onreadystatechange | onmouseleave | onmousemove |
| oncontextmenu |  |  |  |  |

|  |  |
| --- | --- |
| XS (상) | 11. 크로스사이트 스크립팅 |
|  | |
| 조치 시  영향 | 일반적으로 영향 없음 |

웹(Web)

|  |  |
| --- | --- |
| BF (상) | 12. 약한 문자열 강도 |
| 취약점 개요 | |
| 점검내용 | * 웹페이지 내 로그인 폼 등에 약한 강도의 문자열 사용 여부 점검 |
| 점검목적 | * 유추 가능한 취약한 문자열 사용을 제한하여 계정 및 패스워드 추측 공격을 방지하기 위함 |
| 보안위협 | * 해당 취약점 존재 시 유추가 용이한 계정 및 패스워드의 사용으로 인한 사용자 권한 탈취 위험이 존재하며, 해당 위험을 방지하기 위해 값의 적절성 및 복잡성을 검증하는 로직을 구현하여야 함 |
| 참고 | ※ 약한 문자열 강도 취약점: 웹 사이트에서 취약한 패스워드로 회원가입이 가능할 경우 공격자는 추측 및 주변 정보를 수집하여 작성한 사전 파일로 대입을 시도하여 사용자 계정을 탈취할 수 있는 취약점  ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드 |
| 판단기준 | 양호 : 관리자 계정 및 패스워드가 유추하기 어려운 값으로 설정되어 있으며, 일정 횟수 이상 인증 실패 시 로그인을 제한하고 있는 경우 |
| 취약 : 관리자 계정 및 패스워드가 유추하기 쉬운 값으로 설정되어 있으며, 일정 횟수 이상 인증 실패 시 로그인을 제한하고 있지 않은 경우 |
| 조치방법 | 계정 및 비밀번호의 체크 로직 추가 구현 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 웹 사이트 로그인 페이지의 로그인 창에 추측 가능한 계정이나 패스워드를 입력하여 정상적으로 로그인되는지 확인   * 취약한 계정: admin, administrator, manager, guest, test, scott, tomcat, root, user, operator, anonymous 등 * 취약한 패스워드: Abcd, aaaa, 1234, 1111, test, password, public, blank 패스워드, ID와 동일한 패스워드 등   Step 2) 일정 횟수(3~5회) 이상 인증 실패 시 로그인을 제한하는지 확인 | |



웹(Web)

|  |  |
| --- | --- |
| BF (상) | 12. 약한 문자열 강도 |
| * 보안설정방법 * 취약한 계정 및 패스워드를 삭제하고, 사용자가 취약한 계정이나 패스워드를 등록하지 못하도록 패스워드 규정이 반영된 체크 로직을 회원가입, 정보변경, 패스워드 변경 등 적용 필요한 페이지에 모두 구현하여야 함   ※ 규정 예시  Step 1) 다음 각 목의 문자 종류 중 2종류 이상을 조합하여 최소 10자리 이상 또는 3종류 이상을 조합하여 최소 8자리 이상의 길이로 구성   * 1. 영문 대문자(26개)   2. 영문 소문자(26개)   3. 숫자(10개)   4. 특수문자(32개)   Step 2) 연속적인 숫자나 생일, 전화번호 등 추측하기 쉬운 개인정보 및 아이디와 비슷한 비밀번호는 사용하지 않는 것을 권고  Step 3) 비밀번호에 유효기간을 설정하여 반기별 1회 이상 변경 Step 4) 최근 사용되었던 패스워드 재사용 금지   * 로그인 시 패스워드 입력 실패가 일정 횟수(3~5회) 이상 초과할 경우 관리자에게 통보 및 계정 잠금   ※ 인증 실패 횟수를 Client Side Script(Javascript, VBScript 등)를 사용하면 사용자가  임의로 수정할 수 있으므로 Server Side Script(PHP, ASP, JSP 등)를 통하여 구현 | |
| 조치 시  영향 | 일반적으로 영향 없음 |

웹(Web)

|  |  |
| --- | --- |
| IA (상) | 13. 불충분한 인증 |
| 취약점 개요 | |
| 점검내용 | * 중요 페이지 접근 시 추가 인증 요구 여부 점검 |
| 점검목적 | * 중요 페이지에 추가 인증으로 접근을 강화하여 불필요한 정보의 노출 및 변조를 차단하기 위함 |
| 보안위협 | * 중요정보(개인정보 변경 등) 페이지에 대한 인증 절차가 불충분할 경우 권한이 없는 사용자가 중요정보 페이지에 접근하여 정보를 유출하거나 변조할 수 있으므로 중요정보 페이지에는 추가적인 인증 절차를 구현하여야 함 |
| 참고 | ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드 |
| 판단기준 | 양호 : 중요정보 페이지 접근 시 추가 인증을 하는 경우 |
| 취약 : 중요정보 페이지 접근에 대한 추가 인증을 하지 않는 경우 |
| 조치방법 | 중요정보 페이지에 대한 추가 인증 로직 추가 구현 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 중요정보(개인정보 변경 등) 페이지 접근 시 재인증 여부 확인  Step 2) 인증 후 페이지에 아이디만을 인증 값으로 하여 변수로 관리되고 있는지 확인 | |



웹(Web)

|  |  |
| --- | --- |
| IA (상) | 13. 불충분한 인증 |
| * 보안설정방법 * 중요정보(개인정보 변경 등)를 표시하는 페이지에서는 본인 인증을 재확인하는 로직을 구현하고, 사용자가 인증 후 이용 가능한 페이지에 접근할 때마다 승인을 얻은 사용자인지 페이지마다 검증하여야 함 * 접근 통제 정책을 구현하고 있는 코드는 구조화, 모듈화가 되어 있어야 함 * 접근제어가 필요한 모든 페이지에 통제수단(로그인 체크 및 권한 체크)을 구현해야 하며 특히, 하나의 프로세스가 여러 개의 페이지 또는 모듈로 이루어져 있을 때 권한 체크가 누락되는 경우를 방지하기 위해서 공통 모듈을 사용하는 것을 권장함 * 인증 과정을 처리하는 부분에 Client Side Script(Javascript, VBScript 등)를 사용하면 사용자가 임의로 수정할 수 있으므로 Server Side Script(PHP, ASP, JSP 등)를 통하여 인증 및 필터링 과정을 수행함 | |
| 조치 시  영향 | 일반적으로 영향 없음 |

웹(Web)

|  |  |
| --- | --- |
| PR (상) | 14. 취약한 패스워드 복구 |
| 취약점 개요 | |
| 점검내용 | * 웹 사이트 내 패스워드 복구 절차의 적절성 점검 |
| 점검목적 | * 패스워드 복구 로직을 유추하기 어렵게 구현하고, 인증된 사용자 메일이나 SMS에서만 복구 패스워드를 확인할 수 있도록 하여 비인가자를 통한 사용자 패스워드 획득 및 변경을 방지하기 위함 |
| 보안위협 | * 취약한 패스워드 복구 로직(패스워드 찾기 등)으로 인하여 공격자가 불법적으로 다른 사용자의 패스워드를 획득, 변경할 수 있음 |
| 참고 | ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드 |
| 판단기준 | 양호 : 패스워드 재설정 시 난수를 이용하여 재설정되고 인증된 사용자 메일이나 SMS로 재설정된 패스워드 혹은 패스워드 재설정을 위한 링크 전송 시 |
| 취약 : 패스워드 재설정 시 일정 패턴으로 재설정되고 웹 사이트 화면에 바로 출력 시 |
| 조치방법 | 패스워드 복구 로직을 변경하고 인증된 사용자 메일이나 SMS에서만 재설정된 패스워드를 확인할 수 있도록 조치 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 재설정(또는 패스워드 찾기)되는 패스워드 몇 개를 획득하여 사용자의 연락처, 주소, 메일 주소, 일정 패턴을 패스워드로 이용하고 있는지 확인하고 재설정된 패스워드를 인증된 사용자 메일이나 SMS로 전송하는지 확인 | |



웹(Web)

|  |  |
| --- | --- |
| PR (상) | 14. 취약한 패스워드 복구 |
| * 보안설정방법 * 사용자의 개인정보(연락처, 주소, 메일 주소 등)로 패스워드를 생성하지 말아야 하며, 난수를 이용한 불규칙적이고 최소 길이(6자 이상 권고) 이상의 패턴이 없는 패스워드를 발급하여야 함 * 사용자 패스워드를 발급해주거나 확인해줄 때 웹 사이트 화면에 바로 출력해주는 것이 아니라 인증된 사용자 메일이나 SMS로 전송해주어야 함 * 패스워드 재발급 검증 실패에 대한 임계값을 설정하여 일정 횟수 이상 실패한 경우 다른 방식으로 패스워드 찾기 기능을 제공하여야 한다. 검증 후 기존의 패스워드가 아닌 임시패스워드를 발급하도록 설계해야 하며, 사용자가 임시패스워드를 발급받은 즉시 새로운 패스워드로 재설정하도록 구현하여야 함 | |
| 조치 시  영향 | 일반적으로 영향 없음 |

웹(Web)

|  |  |
| --- | --- |
| CF (상) | 15. 크로스사이트 리퀘스트 변조(CSRF) |
| 취약점 개요 | |
| 점검내용 | * 사용자의 신뢰(인증) 정보의 변조 여부 점검 |
| 점검목적 | * 사용자 입력 값에 대한 적절한 필터링 및 인증에 대한 유효성을 검증하여 신뢰(인증) 정보 내의 요청(Request)에 대한 변조 방지 |
| 보안위협 | * 사용자의 신뢰(인증) 정보 내에서 사용자의 요청(Request)을 변조함으로써 해당 사용자의 권한으로 악의적인 공격을 수행할 수 있음 |
| 참고 | ※ CSRF(Cross Site Request Forgery): 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 특정 웹 사이트에 요청하게 하는 공격 유형  ※ OWASP - CSRF 관련 참고사항 https://owasp.org/www-community/attacks/csrf  ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 방화벽 |
| 판단기준 | 양호 : 사용자 입력 값에 대한 검증 및 필터링이 이루어지는 경우 |
| 취약 : 사용자 입력 값에 대한 필터링이 이루어지지 않으며, HTML 코드(또는 스크립트)를 입력하여 실행되는 경우 |
| 조치방법 | 사용자 입력 값에 대해 검증 로직 및 필터링 추가 적용 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) XSS 취약점이 존재하는지 확인 | |



웹(Web)

|  |  |
| --- | --- |
| CF (상) | 15. 크로스사이트 리퀘스트 변조(CSRF) |
| Step 2) 등록 및 변경 등의 데이터 수정 기능의 페이지가 있는지 조사함  Step 3) 데이터 수정 페이지에서 전송되는 요청(Request) 정보를 분석하여 임의의 명령을 수행하는 스크립트 삽입 후 해당 게시글을 타 사용자가 열람하였을 경우 스크립트가 실행되는지 확인   * 보안설정방법 * 웹 사이트에 사용자 입력 값이 저장되는 페이지는 요청이 일회성이 될 수 있도록 설계 * 사용 중인 프레임워크에 기본적으로 제공되는 CSRF 보호 기능 사용 * 사용자가 정상적인 프로세스를 통해 요청하였는지 HTTP 헤더의 Referer 검증 로직 구현 * 정상적인 요청(Request)과 비정상적인 요청(Request)를 구분할 수 있도록 Hidden Form을 사용하여 임의의 암호화된 토큰(세션 ID, Timestamp, nonce 등)을 추가하고 이 토큰을 검증하도록 설계 * HTML이나 자바스크립트에 해당되는 태그 사용을 사전에 제한하고, 서버 단에서 사용자 입력 값에 대한 필터링 구현 * HTML Editor 사용으로 인한 상기사항 조치 불가 시, 서버 사이드/서블릿/DAO(Data Access Object) 영역에서 조치하도록 설계 * XSS 조치 방안 참조 | |
| 조치 시  영향 | 일반적으로 영향 없음 |

웹(Web)

|  |  |
| --- | --- |
| SE (상) | 16. 세션 예측 |
| 취약점 개요 | |
| 점검내용 | * 단순한 방법(연속된 숫자 할당 등)으로 생성되는 세션 ID를 예측하여 세션   탈취 여부 점검 |
| 점검목적 | * 사용자의 세션ID를 추측 불가능하도록 난수로 생성하여 공격자의 불법적인 접근을 차단하기 위함 |
| 보안위협 | * 사용자에게 전달하는 세션 ID가 일정한 패턴을 가지고 있는 경우 공격자가 세션 ID를 추측하여 불법적인 접근을 시도할 수 있음 |
| 참고 | ※ 세션(Session): 일정 시간 동안 같은 사용자(브라우저)로 부터 들어오는 일련의 요구를 하나의 상태로 보고 그 상태를 일정하게 유지시키는 기술  ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드 |
| 판단기준 | 양호 : 추측 불가능한 세션 ID가 발급되는 경우 |
| 취약 : 세션 ID가 일정한 패턴으로 발급되는 경우 |
| 조치방법 | 추측 불가능한 세션 ID가 발급되도록 로직 구현 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 각기 다른 IP 주소와 다른 사용자명, 시간적 차이로 세션 ID를 발급받음 Step 2) 발급받은 세션 ID에 일정한 패턴이 있는지 조사 | |



웹(Web)

|  |  |
| --- | --- |
| SE (상) | 16. 세션 예측 |
| Step 3) 일정한 패턴이 확인되고, 패턴에 의해 사용 가능한 세션 ID의 예측이 가능한지 확인   * 보안 설정 방법   \* 아무리 길이가 길고 복잡한 항목으로 세션 ID가 만들어져도 공격자가 충분한 시간과 자원이 있다면 뚫는 것은 불가능하지 않으므로 강력한 세션 ID를 생성하여야 함  주된 목적은 수많은 대역폭과 처리 자원을 가지고 있는 공격자가 하나의 유효한 세션 ID를 추측하는데 최대한 오랜 시간이 걸리게 하여 쉽게 추측하지 못하게 하는 것에 있음  단순 조합보다는 상용 웹 서버나 웹 애플리케이션 플랫폼에서 제공하는 세션 ID를 사용하고, 가능하다면 맞춤형 세션 관리 체계를 권고함  세션 ID는 로그인 시마다 추측할 수 없는 새로운 세션 ID로 발급하여야 함 | |
| 조치 시  영향 | 일반적으로 영향 없음 |

웹(Web)

|  |  |
| --- | --- |
| IN (상) | 17. 불충분한 인가 |
| 취약점 개요 | |
| 점검내용 | * 민감한 데이터 또는 기능에 접근 및 수정 시 통제 여부 점검 |
| 점검목적 | * 접근 권한에 대한 검증 로직을 구현하여 비인가자의 악의적인 접근을 차단하기 위함 |
| 보안위협 | * 접근제어가 필요한 중요 페이지의 통제수단이 미흡한 경우, 비인가자가 URL 파라미터 값 변경 등의 방법으로 중요 페이지에 접근하여 민감한 정보 열람 및 변조 가능함 |
| 참고 | ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드 |
| 판단기준 | 양호 : 접근제어가 필요한 중요 페이지의 통제수단이 적절하여 비인가자의 접근이 불가능한 경우 |
| 취약 : 접근제어가 필요한 중요 페이지의 통제수단이 미흡하여 비인가자의 접근이 가능한 경우 |
| 조치방법 | 접근제어가 필요한 모든 페이지에 권한검증 로직 구현 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 비밀 게시글(또는 개인정보 변경, 패스워드 변경 등) 페이지에서 다른 사용자와의 구분을 ID, 일련번호 등의 단순한 값을 사용하는지 조사  Step 2) 게시글을 구분하는 파라미터 값을 변경하는 것만으로 다른 사용자의 비밀 게시글 (또는 개인정보 변경, 패스워드 변경 등)에 접근 가능한지 확인 | |



웹(Web)

|  |  |
| --- | --- |
| IN (상) | 17. 불충분한 인가 |
| * 보안설정방법 * 접근제어가 필요한 중요 페이지는 세션을 통한 인증 등 통제수단을 구현하여 인가된 사용자 여부를 검증 후 해당 페이지에 접근할 수 있도록 함 * 페이지별 권한 매트릭스를 작성하여 접근제어가 필요한 모든 페이지에서 권한 체크가 이뤄지도록 구현하여야 함 | |
| 조치 시  영향 | 일반적으로 영향 없음 |

웹(Web)

|  |  |
| --- | --- |
| SC (상) | 18. 불충분한 세션 만료 |
| 취약점 개요 | |
| 점검내용 | * 세션의 만료 기간 설정 여부 점검 |
| 점검목적 | * 세션 타임아웃 기능을 구현하여 공격자가 만료되지 않은 세션 활용을 방지하기 위함 |
| 보안위협 | * 세션의 만료 기간을 정하지 않거나, 만료기한을 너무 길게 설정된 경우 악의적인 사용자가 만료되지 않은 세션을 활용하여 불법적인 접근이 가능할 수 있음 |
| 참고 | ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 서버 |
| 판단기준 | 양호 : 세션 종료 시간이 설정되어 있는 경우 |
| 취약 : 세션 종료 시간이 설정되어 있지 않아 세션 재사용이 가능한 경우 |
| 조치방법 | 세션 종료 시간 설정 또는 자동 로그아웃 기능 구현(세션 종료 시간은 사이트의 특성에 따라 달라질 수 있으므로 사이트의 특성에 맞게 적정 시간 설정) |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 인증 후 정상적으로 세션이 발행된 페이지의 리퀘스트를 취득하여 일정 시간 (사이트에 따라 다름)이 지난 후에 재전송 시 정상 처리가 되는지 확인  [로그인 후 세션 발급]  [일정 시간 경과 후 세션 유지 여부 확인] | |



웹(Web)

SC (상) 18. 불충분한 세션 만료

* 보안설정방법

\* 세션 타임아웃 구현 시 타임아웃 시간은 10분으로 설정할 것을 권고함

※ 웹 애플리케이션 별 상세 설정

* ASP

접속자별로 세션을 생성하여 사용자 정보를 각각 저장할 수 있는 세션 오브젝트로 타임아웃

기능을 구현함

※ 세션 오브젝트: 페이지 접근을 허가하거나 금지할 때 또는, 사용자별로 정보를 저장할 때 많이 사용하며 접속자의 브라우저에서 쿠키 기능을 지원해야 세션 오브젝트 사용이 가능함

다음과 같은 설정이 적용될 경우 사용자가 로그아웃 시 세션은 바로 삭제되며, 로그아웃 하지 않고 10분 동안 웹 서버로의 요청이 없으면 세션은 없어지게 됨

… 중략 …

// Session의 유지 시간 Setting Session.timeout = 10

… 중략 …

|  |  |  |
| --- | --- | --- |
| 구분 | | 설 명 |
| Property | SessionID | 사용자마다 갖게 되는 고유한 세션 값 |
| Timeout | 세션이 유지되는 기간 |
| Method | Abandon | 강제로 세션을 소멸시키는 함수 |
| Event | Onstart | 각각의 사용자가 처음 방문할 때 발생 |
| Onend | 사용자의 세션이 끝나는 시점에 발생 |

* JSP

세션 타임아웃 기능을 구현하는 방법은 session.getLastAccessedTime()를 이용하여 세션의 마지막 접근 시간으로부터 일정 시간동안 세션 접근을 하지 않은 경우 자동으로 세션을 종료하도록 함

세션의 타임아웃은 두 가지 방법으로 설정할 수 있음

Step 1) web.xml 파일에서 <session-config> 태그를 사용하여 타임아웃을 지정하는 방법. web.xml, Weblogic.xml 중 한 곳에만 설정 (만약, 두 곳 모두 설정 시 우선순위에 의해 web.xml 설정이 적용됨)

Web.xml : “분” 단위

<session-config>

<session-timeout>10</session-timeout>

</session-config>

웹(Web)

|  |  |
| --- | --- |
| SC (상) | 18. 불충분한 세션 만료 |
| Weblogic.xml: “초” 단위  <session-descriptor>  <timeout-secs>600</timeout-secs>  </session-descriptor> 또는,  <session-param>  <param-name>TimeoutSecs</param-name>  <param-value>600</param-value>  </session-param>  Step 2) 세션 기본 객체가 제공하는 setMaxInactiveInterval() 메소드 사용  ※ 주의할 점: web.xml 에서는 타임아웃 시간 단위가 분이지만 메소드에서는 초 단위임  … 중략 …  // Session의 유지 시간을 Setting  String strTime = Param.getPropertyFromXML("SessionPersistenceTime"); if (strTime == null) {  session.setMaxInactiveInterval(600);  } else {  session.setMaxInactiveInterval((new Integer(strTime)).intValue());  }  … 중략 … | |
| 조치 시  영향 | 일반적으로 영향 없음 |



웹(Web)

|  |  |
| --- | --- |
| SF (상) | 19. 세션 고정 |
| 취약점 개요 | |
| 점검내용 | * 사용자 로그인 시 항상 일정하게 고정된 세션 ID 값을 발행하는지 여부 확인 |
| 점검목적 | * 로그인할 때마다 예측 불가능한 새로운 세션 ID를 발행하여 세션 ID의 고정 사용을 방지하기 위함 |
| 보안위협 | * 사용자 로그인 시 항상 일정하게 고정된 세션 ID가 발행되는 경우 세션 ID를 도용한 비인가자의 접근 및 권한 우회가 가능 |
| 참고 | ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드 |
| 판단기준 | 양호 : 로그인할 때마다 예측 불가능한 새로운 세션 ID가 발행되고, 기존 세션 ID는 파기될 경우 |
| 취약 : 로그인 세션 ID가 고정 사용되거나 새로운 세션 ID가 발행되지만 예측 가능한 패턴으로 발행될 경우 |
| 조치방법 | 사용자가 로그인할 때마다 예측 불가능한 새로운 세션 ID 생성 로직 구현하고 기존 세션 ID는 파기함 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 로그인 시(1) 세션 ID가 발행되는지 확인하고 로그아웃 후 다시 로그인(2)할 때 예측 불가능한 새로운 세션 ID가 발급되는지 확인   * 보안설정방법   \* 로그인할 때마다 예측 불가능한 새로운 세션 ID를 발급받도록 해야 하고 기존 세션 ID는 파기해야 함 | |
| 조치 시  영향 | 일반적으로 영향 없음 |

웹(Web)

|  |  |
| --- | --- |
| AU (상) | 20. 자동화 공격 |
| 취약점 개요 | |
| 점검내용 | * 웹 애플리케이션의 특정 프로세스(로그인 시도, 게시글 등록, SMS 발송 등)에 대한 반복적인 요청 시 통제 여부 확인 |
| 점검목적 | * 무차별 대입 공격 및 자동화 공격으로 웹 애플리케이션에 자원이 고갈되는   것을 방지하기 위함 |
| 보안위협 | * 웹 애플리케이션의 특정 프로세스에 대한 반복적인 요청을 통제하지 않을 경우 무차별 대입 공격으로 인해 사용자 계정을 탈취할 수 있고, 자동화 공격으로 게시글 등록 또는 SMS 발송 요청을 반복하여 웹 애플리케이션 자원을 고갈시킬 수 있음 |
| 참고 | ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 방화벽 |
| 판단기준 | 양호 : 웹 애플리케이션의 특정 프로세스에 대한 반복적인 요청 시 통제가 적절한 경우 |
| 취약 : 웹 애플리케이션의 특정 프로세스에 대한 반복적인 요청 시 통제가 미흡한 경우 |
| 조치방법 | 웹 애플리케이션의 특정 프로세스에 대한 대량 사용 통제 로직 구현 및 웹 방화벽 룰셋 설정을 통해 대량의 불특정 프로세스 요청 차단 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 로그인 시도, 게시글 등록, SMS 발송 등에 대한 정상적인 요청 정보를 식별하여 반복적으로 요청 시 통제가 이루어지는지 확인   * 보안설정방법   \* 로그인 시도, 게시글 등록, SMS 발송 등에 대한 사용자 요청이 일회성이 될 수 있도록, 캡차(이미지를 이용하여 확인 값을 표시하고 사용자가 값을 등록하여 인증함) 등 일회성 확인 로직을 구현하여야 함  ※ 캡차(CAPTCHA): 자동화된 컴퓨터와 사람을 판별하기 위한 기술의 일종 | |



웹(Web)

|  |  |
| --- | --- |
| AU (상) | 20. 자동화 공격 |
| \* 자동화 공격을 시도하면 짧은 시간에 다량의 패킷(양)이 전송되므로 이를 공격으로 감지하고 방어할 수 있는 IDS/IPS 시스템을 구축하여야 함. 서버에 요청되는 패킷(양)의 모니터링이 불가능한 경우 적시에 적절한 대응이 어려움 | |
| 조치 시  영향 | 일반적으로 영향 없음 |

웹(Web)

|  |  |
| --- | --- |
| PV (상) | 21. 프로세스 검증 누락 |
| 취약점 개요 | |
| 점검내용 | * 인증이 필요한 웹 사이트의 중요(관리자 페이지, 회원변경 페이지 등) 페이지에 대한 접근제어 설정 여부 확인 |
| 점검목적 | * 인증이 필요한 모든 페이지에 대해 유효 세션임을 확인하는 프로세스 및 주요 정보 페이지에 접근 요청자의 권한 검증 로직을 적용하여, 비인가자가 하위 URL 직접 접근, 스크립트 조작 등의 방법으로 중요한 페이지에 접근을 시도하는 것을 차단하기 위함 |
| 보안위협 | * 인증이 필요한 웹 사이트의 중요(관리자 페이지, 회원변경 페이지 등) 페이지에 대한 접근 제어가 미흡할 경우 하위 URL 직접 접근, 스크립트 조작 등의 방법으로 중요한 페이지에 대한 접근이 가능함 |
| 참고 | ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드 |
| 판단기준 | 양호 : 인증 후에 접근해야 하는 웹 사이트의 하위 URL을 로그인하지 않고 직접 접근할 때 접근이 불가능한 경우 |
| 취약 : 웹 사이트의 하위 URL을 로그인하지 않고 직접 접근할 때 접근이 가능한 경우 |
| 조치방법 | 인증이 필요한 페이지의 경우 페이지별 권한 체크 로직 구현 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 업무프로세스 파악  Step 2) 권한의 종류 및 범위 파악  Step 3) 페이지의 모든 기능을 수집하여 프로세스 상에 통제된 페이지 접근이 가능한지 확인 | |



웹(Web)

|  |  |
| --- | --- |
| PV (상) | 21. 프로세스 검증 누락 |
| * 보안설정방법 * 우회될 수 있는 플로우를 차단하여야 하며, 페이지별 권한 매트릭스를 작성하여 페이지에 부여된 권한의 타당성을 체크한 후 권한 매트릭스를 기준으로 전 페이지에서 권한 체크가 이뤄지도록 구현하여야 함 * 인증이 필요한 모든 페이지에 대해 유효 세션임을 확인하는 프로세스 및 주요 정보 페이지에 접근 요청자의 권한 검증 로직을 적용함 * 유효 세션의 검증 및 페이지에 대한 접근 권한을 Client Side Script에 의존할 경우 사용자가   임의로 수정할 수 있으므로 Server Side Script로 구현된 프로세스를 사용  ※ 웹 애플리케이션 별 상세 설정   * ASP   (예) 인증이 필요한 페이지 소스 코드  <% - 인증 성공 시 세션값 세팅 Session(“sessionChk”) = True Session(“UserID”) = userID Session(“UserGrp”) = userGrp  Session(“UserIP”) = Request.Servervariables(“REMOTE\_ADDR”)  … 중략 …   * 사용자 그룹 리턴 함수   … 중략 …  Function GetUserGroup(strUserID) End function … 중략 …   * 페이지에 접근 가능한 UserGroup 설정값이 ‘100’ 가정 시 ChkUserGrp = GetUserGroup(userID)   //세션 userID값을 통해 DB에 저장된 사용자 그룹 리턴 … 중략 …  If Session\_Check and Session(“UserGrp”) = ChkUserGrp Then If Session(“UserGrp”) <> 100 Then  Response.Write(“권한이 없습니다.”) Response.End  End Else  Response.Redirect “Login.asp”  Response.End End if  … 중략 … %>   * JSP   (예) 인증이 필요한 페이지 소스 코드  <%  … 중략 …  PortalSessionManager sessionMgr = (PortalSessionManager)  session.getAttribute("sessionMgr");  if (sessionMgr == null || sessionMgr.getUserId() == null) {  (new FailToAuthenticateCmd()).execute(request,response);  }  … 중략 …  String usrGrp = session.getAttribute("Usrgrp") == null ? | |

웹(Web)

|  |  |
| --- | --- |
| PV (상) | 21. 프로세스 검증 누락 |
| "" : (String)session.getAttribute("Usrgrp");  if (!usrGrp.equals("") || !userGrp.equals(Code.getMarket())) {  // 접근 권한을 인가할 수 없음.  (new FailToPermissionCmd()).execute(request,response); }  중략 …  %> | |
| 조치 시  영향 | 일반적으로 영향 없음 |



웹(Web)

|  |  |
| --- | --- |
| FU (상) | 22. 파일 업로드 |
| 취약점 개요 | |
| 점검내용 | * 웹 사이트의 게시판, 자료실 등에 조작된 Server Side Script 파일 업로드 및 실행 가능 여부 점검 |
| 점검목적 | * 업로드되는 파일의 확장자에 대한 적절성 여부를 검증하는 로직을 통해 공격자가 조작된 Server Side Script 파일 업로드 방지 및 서버상에 저장된 경로를 유추하여 해당 Server Side Script 파일 실행을 불가능하게 하기 위함 |
| 보안위협 | * 해당 취약점이 존재할 경우 공격자는 조작된 Server Side Script 파일을 서버에 업로드 및 실행하여 시스템 관리자 권한 획득 또는 인접 서버에 대한 침입을 시도할 수 있음 |
| 참고 | ※ Server Side Script: 웹에서 사용되는 스크립트 언어 중 서버 측에서 실행되는 스크립트  ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 서버, 웹 방화벽 |
| 판단기준 | 양호 : 업로드되는 파일에 대한 확장자 검증이 이루어지는 경우 |
| 취약 : 업로드되는 파일에 대한 확장자 검증이 이루어지지 않는 경우 |
| 조치방법 | 업로드되는 파일에 대한 확장자 검증 및 실행 권한 제거 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 웹 사이트에 파일 업로드 기능이 존재하는 경우, 확장자가 jsp, php, asp, cgi 등 Server Side Script 파일들이 업로드 가능한지 확인  ※ 클라이언트에서 JavaScript, VBScript 등의 스크립트로 파일 첨부를 차단하는 경우 차단 기능을 수정하여 파일 첨부함 | |

웹(Web)

|  |  |
| --- | --- |
| FU (상) | 22. 파일 업로드 |
| Step 2) 웹 사이트에 있는 디렉터리 정보를 이용하여 첨부한 Server Side Script 파일의 위치를 조사한 후 브라우저 주소창에 해당 경로를 입력하여 실행 가능한지 확인   * 보안설정방법   ※ 사용자가 파일을 업로드할 수 있는 모든 모듈에 적용 필요   * 화이트 리스트 방식으로 허용된 확장자만 업로드 가능토록 서버 측 통제 적용 * 업로드되는 파일을 디렉터리에 저장할 때 파일명과 확장자를 외부 사용자가 추측할 수 없는 문자열로 변경하여 저장(파일 이름은 DB에 저장) | |



웹(Web)

|  |  |
| --- | --- |
| FU (상) | 22. 파일 업로드 |
| * 업로드 파일을 위한 전용 디렉터리를 별도로 생성하여 웹 서버 데몬 설정 파일(httpd.conf 등)에서 실행 설정을 제거함으로써, Server Side Script가 업로드되더라도 웹 엔진이 실행하지 않는 환경을 설정함 * 파일 업로드 필드를 대상으로 특수문자 필터링하도록 웹 방화벽 룰셋 적용   ※ 유형 별 상세 설정   * 웹 애플리케이션 * ASP   Step 1) 수용 가능한 파일의 확장자만 업로드 허용(Positive 방식)   * 이미지 파일의 경우 (JPG, GIF, BMP 등) * 문서 파일의 경우 (XLS, PDF, PPT, DOC 등)   (예) doc, hwp, pdf, jpg 파일만 업로드 허용 (※ 예로 제시하는 것으로, 구현 시 다를 수 있음)  확장자 검증 시 대소문자 구분 없이 문자열 비교  ....  FunctionIsAllowExtension(originFilename,mAllowExtension) Dim ReturnValue, eregObj, matches, PatternStr, FileNameExt  FileNameExt = Mid(originFilename, InStrRev(originFilename, “.”) +1)  ReturnValue = False  if IsNull(mAllowExtention) Then IsAllowExtension = True  Exit Function End if  PatternStr = “^(“ & Replace(mAllowExtention, “,”,”|”) & “)$”  Set eregObj = New RegExp  With eregObj  .IgnoreCase = True  .Global = True  .Pattern = PatternStr ReturnValue = .test(FileNameExt) End with  Set eregObj = Nothing IsAllowExtention = ReturnValue End Function  .....  If Not IsAllowExtention(“파일명.txt”, “doc,hwp,pdf,jpg”) Then  response.write “허용되지 않은 확장자 입니다.” End if  Step 2) MIME TYPE 확인을 통한 실행 파일 업로드 차단 (예) MIME TYPE을 통한 악성 파일 업로드 차단  if UploadForm("UPFILE").MineType<> "image" then Response.write "Permit only Image files"  Response.end end if   * ASP.net   Step 1) 수용 가능한 파일의 확장자만 업로드 허용(Positive 방식)   * 이미지 파일의 경우 (JPG, GIF, BMP 등) | |

웹(Web)

|  |  |
| --- | --- |
| FU (상) | 22. 파일 업로드 |
| * 문서 파일의 경우 (XLS, PDF, PPT, DOC 등)   (예) doc, hwp, pdf, jpg 파일만 업로드 허용 (※ 예로 제시하는 것으로, 구현 시 다를 수 있음)  확장자 검증 시 대소문자 구분 없이 문자열 비교  string upload\_Image(FileUpload fileupload, string ImageSavedPath)  {  FileUpload fu = fileupload; string imagepath = "";  if (fileupload.HasFile)  {  string filepath = Server.MapPath(ImageSavedPath);  String fileExtension = System.IO.Path.GetExtension(fu.FileName).ToLower(); String[] allowedExtensions = { ".doc", ".hwp", ".pdf", ".jpg" };  for (int i = 0; i < allowedExtensions.Length; i++)  {  if (fileExtension == allowedExtensions[i])  {  try  {  string s\_newfilename = DateTime.Now.Year.ToString()+ DateTime.Now.Month.ToString() + DateTime.Now.Day.ToString()+ DateTime.Now.Hour.ToString() + DateTime.Now.Minute.ToString()+ DateTime.Now.Second.ToString() + fileExtension; fu.PostedFile.SaveAs(filepath + s\_newfilename);  imagepath = ImageSavedPath + s\_newfilename;  }  catch (Exception ex)  {  Response.Write("File could not be uploaded.");  }  }  }  }  return imagepath;  }  Step 2) MIME TYPE 확인을 통한 실행 파일 업로드 차단 (예) MIME TYPE을 통한 악성 파일 업로드 차단  public void validateFileToUpload(FileUpload objFile)  { intMAX\_FILE\_SIZE=(4\*1024\*1024);  int fileSize = objFile.PostedFile.ContentLength; if(fileSize>MAX\_FILE\_SIZE)  {  returnMessage="FileUploadFailed"; returnreturnMessage;  }  string chosenFileExtension = System.IO.Path.GetExtension(objFile.FileName); string[]allowedExtensions= {".doc",".xls",".ppt",".pptx",".txt" }; if(!allowedExtensions.Contains(chosenFileExtension))  {  returnMessage="FileUploadFailed"; returnreturnMessage;  }  string[] allowedMimeTypes = { "text/plain", "text/xml" }; stringchosenFileMiMeType=objFile.PostedFile.ContentType; if(!allowedMimeTypes.Contains(chosenFileMiMeType)) | |



웹(Web)

|  |  |
| --- | --- |
| FU (상) | 22. 파일 업로드 |
| {  returnMessage="FileUploadFailed"; returnreturnMessage;  }  }   * JSP   Step 1) 수용 가능한 파일의 확장자만 업로드 허용(Positive 방식)   * 이미지 파일의 경우 (JPG, GIF, BMP 등) * 문서 파일의 경우 (XLS, PDF, PPT, DOC 등)   (예) doc, hwp, pdf, jpg 파일만 업로드 허용 (※ 예로 제시하는 것으로, 구현 시 다를 수 있음).  확장자 검증 시 대소문자 구분 없이 문자열 비교  .....  publievoidupload(HttpServletRequestrequest)throwsServletException  {  MultipartHttpRequest multi = (MultipartHttpRequest) request; String next = (String) multi.getFileNames().next(); MultipartFile file = multi.getFile(next);  If (file == null ) return;  // 화이트 리스트 방식으로 업로드 파일 확장자 체크  if (fileName != null)  {  If (fileName,endsWith(“.doc”) || fileName,endsWith(“.hwp”) || fileName,endsWith(“.pdf”) || fileName,endsWith(“.jpg”))  {  //file 업로드 루틴: 저장 시 파일명을 외부 사용자가 추측할 수 없는 형태로 변경  .....  Step 2) MIME TYPE 확인을 통한 실행 파일 업로드 차단 (예) MIME TYPE을 통한 악성 파일 업로드 차단  <%  String[] validExt = {"jpg","gif","png" }; // 파일 허용 확장자 String[] validType = {"application/octet-stream", "application/x-msdownload",  "application/x-sh" }; // 파일 MIME 타입 제한 MultipartRequest mRequest = new MultipartRequest(request, SITE\_UPLOAD\_DIR+strUploadDir, intUploadMaxSize,  "UTF-8", new DefaultFileRenamePolicy());  uploadFileSystemName1 = mRequest.getFilesystemName("attach1");  //저장파일명  File strGetfile1= mRequest.getFile("attach1"); uploadFileExt1 =  uploadFileSystemName1.substring(uploadFileSystemName1.lastIndexOf('.')  + 1); // 파일 확장자  uploadFileType1 = mRequest.getContentType("attach1"); //파일 MIME 타입 for(int i=0; i< validType.length; i++) { if(uploadFileType1.equalsIgnoreCase(validType[i])) { out.print("<script>alert('업로드 금지 파일')</script>");  commUtil.deleteFile(SITE\_UPLOAD\_DIR+strUploadDir+"/", uploadFileSystemName1); return;  }  }  %> | |

웹(Web)

|  |  |
| --- | --- |
| FU (상) | 22. 파일 업로드 |
| * PHP   Step 1) 수용 가능한 파일의 확장자만 업로드 허용(Positive 방식)   * 이미지 파일의 경우 (JPG, GIF, BMP 등) * 문서 파일의 경우 (XLS, PDF, PPT, DOC 등)   (예) doc, hwp, pdf, jpg 파일만 업로드 허용 (※ 예로 제시하는 것으로, 구현 시 다를 수 있음)  확장자 검증 시 대소문자 구분 없이 문자열 비교  .....  // 파일 이름에 특수문자가 있을 경우 업로드를 금지시킴 if (eregi("[^a-z0-9 \.\_ \-]",$\_FILES['userfile']['name'])) print "파일 이름의 특수문자 체크";  exit;  // 파일 확장자 중 업로드를 허용할 확장자를 정의함  $full\_filename = explode(".", $\_FILES['userfile']['name']);  $extension = $full\_filename[sizeof($full\_filename)-1];  $extension= strtolower($extension);  if (!( ereg($extension","hwp") || ereg($extension","pdf") || ereg($extension","jpg")) ) print "업로드 금지 파일 입니다";  exit;  .....  Step 2) MIME TYPE 확인을 통한 실행 파일 업로드 차단 (예) MIME TYPE을 통한 악성 파일 업로드 차단  <?  // 허용된 확장자를 가진 파일에 대해서 파일 업로드 성공  If (($\_FILES["file"]["type"] == "image/gif") || ($\_FILES["file"]["type"] == "image/jpeg") ||  ($\_FILES["file"]["type"] == "image/JPG") || ($\_FILES["file"]["type"] == "text/plain"))  {  echo "파일 업로드 성공"  }  else  {  echo "파일 업로드 실패. 허용된 파일의 형식이 아닙니다."  }  ?>   * 웹 서버 * IIS   설정 > 제어판 > 관리도구 > 인터넷 서비스(IIS) 관리자 선택  해당 업로드 폴더에서 우클릭 > 등록 정보 > 디렉터리> 실행 권한 “없음” 설정 | |



웹(Web)

|  |  |
| --- | --- |
| FU (상) | 22. 파일 업로드 |
| * Apache   Apache 설정 파일인 httpd.conf에 해당 디렉터리에 대한 문서 타입을 컨트롤 하기 위해 Directory 섹션의 AllowOverride 지시자에서 FileInfo 또는, “All” 추가  <Directory “/usr/local/apache”>AllowOverride FileInfo (또는, All)….  …  <Directory>  파일 업로드 디렉터리에 .htaccess 파일을 만들고 다음과 같이 AddType 지시자를 이용, 현재 서버에서 운영되는 Server Side Script 확장자를 text/html로 MIME Type을 재조정하여 업로드된 Server Side Script가 실행되지 않도록 설정  또는, FileMatch 지시자를 이용하여 \*.ph, \*.inc, \*lib 등의 Server Side Script 파일에 대해서 직접 URL 호출을 금지시킴  <.htaccess><FilesMatch “ \.(ph|inc|lib)”>  Order allow, deny  Deny from all  </FilesMatch>  AddType text/html .html .htm .php .php3 .php4 .phtml .phps .in .cgi .pl .shtml .jsp  ※ 주의할 점   * Apache 서버의 경우 AllowOverride 지시자 변경 시 Apache Restart 필요 | |

웹(Web)

|  |  |
| --- | --- |
| FU (상) | 22. 파일 업로드 |
| * 파일 업로드 되는 디렉터리에 운영에 필요한 Server Side Script가 존재하는지 확인 * 파일 다운로드 프로그램이 아닌 직접 URL 호출을 통해 파일을 다운받는 경우 FileMatch 지시자를 사용하면 차단 설정한 확장자의 파일 다운로드는 거부됨 * 첨부 파일 확장자 필터링 처리로 사용자가 첨부 파일의 업로드 시도 시, 업로드 파일의 확장자를 검토하여 적절한 파일인지 검사하는 루틴을 삽입하여, 적합한 파일의 확장자 이외의 파일에 대해서 업로드가 불가능하도록 하며, 이런 필터링 규칙은 서버에서 구현하여야 함 * 시스템 보안 설정 시 웹 서버 구동은 반드시 관리자 권한이 아닌 일반 사용자 권한으로 구동함 * 외부 사용자가 첨부 파일을 이용하여 권한을 획득할지라도 최소한의 권한만을 사용할 수 있도록 함 * 업로드된 디렉터리에서 실행 권한을 제거하는 방법은 임시적이기는 하지만 소스 코드의 수정 없이 간단히 수행될 수 있음 * 웹 방화벽   \* 웹쉘에 주로 사용되는 문자열, 오브젝트, 메소드 등을 시그니처로 지정하여 파일 업로드 시  탐지 및 차단함 | |
| 조치 시  영향 | 일반적으로 영향 없음 |



웹(Web)

|  |  |
| --- | --- |
| FD (상) | 23. 파일 다운로드 |
| 취약점 개요 | |
| 점검내용 | * 웹 사이트에서 파일 다운로드 시 허용된 경로 외 다른 경로의 파일 접근이 가능한지 여부 점검 |
| 점검목적 | * 파일 다운로드 시 허용된 경로 외 다른 경로의 파일 접근을 방지하여 공격자가 임의의 위치에 있는 파일을 열람하거나 다운받는 것을 불가능하게 하기 위함 |
| 보안위협 | * 해당 취약점이 존재할 경우 공격자는 파일 다운로드 시 애플리케이션의 파라미터 값을 조작하여 웹 사이트의 중요한 파일(DB 커넥션 파일, 애플리케이션 파일 등) 또는 웹 서버 루트에 있는 중요한 설정 파일(passwd, shadow 등)을 다운받을 수 있음 * cgi, jsp, php 등 파일 다운로드 기능을 제공해주는 애플리케이션에서 입력되는 경로를 검증하지 않는 경우 임의의 문자(../.. 등)나 주요 파일명의 입력을 통해 웹 서버의 홈 디렉터리를 벗어나서 임의의 위치에 있는 파일을 열람하거나 다운받는 것이 가능함 |
| 참고 | ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 서버, 웹 방화벽 |
| 판단기준 | 양호 : 다운로드 파일이 저장된 디렉터리 이외에 접근이 불가능한 경우 |
| 취약 : 다운로드 파일이 저장된 디렉터리 이외에 접근이 가능한 경우 |
| 조치방법 | 다운로드 시 허용된 경로 이외의 디렉터리와 파일에 접근할 수 없도록 구현 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 웹 사이트에 cgi, jsp, php 등의 애플리케이션을 이용하여 파일을 다운받는 페이지가 있는지 조사 | |

웹(Web)

|  |  |
| --- | --- |
| FD (상) | 23. 파일 다운로드 |
| Step 2) 웹 사이트에서 파일 다운로드 시 요청(Request) 정보에 파일 경로를 웹 서버(웹 사이트 포함) 중요 파일(winnt\win.ini, /etc/passwd 등)의 상대 경로(../)로 치환한 후 전송했을 때 해당 경로 파일들을 다운로드 가능한지 확인  ../../../../../../../../../../../../etc/passwd  ../../../../../../../../../../../../winnt/win.ini  ../../../../../../../../../../../../boot.ini  Step 3) "Step 2"에서 파일 다운로드가 불가능한 경우 변조한 파일 경로를 아래의 인코딩(또는 치환, 종단문자추가)을 적용하여 재전송 후 다운로드 가능한지 확인  ※ URL 인코딩: .(%2e), /(%2f), \(%5c)  ※ 16bit 유니코드 인코딩: .(%u002e), /(%u2215), \(%u2216)  ※ 더블URL 인코딩: .(%252e), /(%252f), \(%255c)  ※ 경로 치환: …//, ….\\, ….\/, …./\  ※ 종단 문자 추가: [파일명]%00.jpg, [파일명]%0a.jpg | |



웹(Web)

FD (상) 23. 파일 다운로드

* 보안설정방법
* 파일 다운로드의 취약성은 주로 파일의 이름을 조작하는 데서 비롯되므로 다운로드 파일 이름을 데이터베이스에 저장하고 다운로드 수행 시 요청 파일 이름과 비교하여 적절한지 확인하여 사용자가 조작할 수 있는 변수를 제거함
* 다운로드 애플리케이션 소스 파일을 수정하여 파일을 다운받을 수 있는 디렉터리를 특정 디렉터리로 한정하고 이 외의 다른 디렉터리에서는 파일을 다운받을 수 없도록 설정해야 함
* PHP를 사용하는 경우 php.ini 에서 magic\_quotes\_gpc를 On으로 설정하여 .\./ 와 같은 역 슬러시 문자 입력 시 치환되도록 설정
* 파일 다운로드의 절대 경로 설정 및 DocBase의 상위경로 또는 타 드라이브로 설정을 변경함
* 다운로드 경로 정보를 자바스크립트나 js 소스에서 확인할 수 없게 제한하며, 웹 서버 서블릿 내부 또는 별도의 설정 파일에서 관리
* 다운로드를 제공하는 페이지의 유효 세션 체크 로직 필수 적용
* 다운로드 시 사용되는 파라미터 값 대상으로 아래의 특수 문자를 필터링하도록 웹 방화벽 룰셋 적용

|  |  |
| --- | --- |
| 문자 | 설명 |
| . | Path Traversal 가능성의 확인 |
| / | 특정 Path의 접근 가능성을 확인 |
| \ | 운영환경에 따른 Path 접근 확인 |
| % | UTF 인코딩 파라미터 |

[참고 : 필터링문자]

※ 웹 애플리케이션 별 상세 설정

* ASP

(예) 필터링 처리 (※ 예로 제시하는 것으로, 구현 시 다를 수 있음)

.....

file = Request.form(“file”) Response.ContentType = “application/unknown”

Response.AddHeader “Content-Disposition”, “attachment; filename=” & file Set objStream = Server.CreateObject(“ADODB.Stream”)

strFile = Server.MapPath(“./uploadfiles”) & “ \” & file strFname = Mid(Fname, InstrRev(file, “ \”) +1)

if strFile = strFPath Then

.....

* ASP.net

(예) ASP.net 예외처리 (※ 예로 제시하는 것으로, 구현 시 다를 수 있음)

.NET 환경은 자체적으로 Path Traversal을 막고 있으므로, 소스 자체적인 별다른 조치는 필요가 없으나 일부 .NET 버전에 해당 보안 매커니즘을 우회할 수 있는 취약점이 발견된 사례가 있으므로, 최신 패치를 설치할 것을 권고함. 해당 패치가 설치되어 있지 않은 경우 Global.asax에 다음과 같은 내용을 추가하여야 함

웹(Web)

|  |  |
| --- | --- |
| FD (상) | 23. 파일 다운로드 |
| <script language="C#" runat="server">  void Application\_BeginRequest(object source, EventArgs e) { if (Request.Path.IndexOf(' \ \') >= 0 ||  System.IO.Path.GetFullPath(Request.PhysicalPath) != Request.PhysicalPath) {  throw new HttpException(404, "not found");  }  }</script>   * PHP   (예) 필터링 처리 (※ 예로 제시하는 것으로, 구현 시 다를 수 있음)  if (preg\_match(“/[^a-z0-9\_-]/I”, $up\_dir)) print “디렉터리의 특수 문자 체크”;  exit;  if(preg\_match(“/[^\xA1-\xFEa-z0-9.\_-]/I”, urldecode($dn\_file\_name))) print "파일 이름의 특수문자 체크";  exit;   * JSP   (예) 필터링 처리 (※ 예로 제시하는 것으로, 구현 시 다를 수 있음)  String UPLOAD\_PATH= "/var/www/upload/";  String filename= response.getParameter("filename");  String filepathname = UPLOAD\_PATH + filename;  if(filename.equalsIgnoreCase(".") || filename.equalsIgnoreCase("/")|| filename.equalsIgnoreCase(" \"))  // 파일명 체크  return 0;  // 파일 전송 루틴 response.setContentType("application/unknown; charset=euc-kr");  response.setHeader("Content-Disposition","attachment;filename=" + filename + ";"); response.setHeader("Content-Transfer-Encoding:" , "base64");  try {  BufferedInputStream in = new BufferedInputStream(new FileInputStream(filepathname));  .........  } catch(Exception e) {  // 에러 체크 [파일 존재 유무 등]  } | |
| 조치 시  영향 | 일반적으로 영향 없음 |



웹(Web)

|  |  |
| --- | --- |
| AE (상) | 24. 관리자 페이지 노출 |
| 취약점 개요 | |
| 점검내용 | * 유추하기 쉬운 URL로 관리자 페이지 및 메뉴 접근의 가능 여부 점검 |
| 점검목적 | * 관리자 페이지 URL이 유추하기 쉬운 이름(admin, manager 등) 및 웹 사이트 설계 오류를 수정하여 비인가자의 관리자 메뉴 접근을 방지하고자 함 |
| 보안위협 | * 웹 관리자의 권한이 노출될 경우 웹 사이트의 변조뿐만 아니라 취약성 정도에 따라서 웹 서버의 권한까지도 노출될 수 있음 |
| 참고 | ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 서버, 웹 방화벽 |
| 판단기준 | 양호 : 유추하기 쉬운 URL로 관리자 페이지 접근이 불가능한 경우 |
| 취약 : 유추하기 쉬운 URL로 관리자 페이지 접근이 가능한 경우 |
| 조치방법 | 유추하기 어려운 이름(포트 번호 변경 포함)으로 관리자 페이지를 변경하여 비인가자가 관리자 페이지에 접근할 수 없도록 하고 근본적인 해결을 위해 지정된 IP만 관리자 페이지에 접근할 수 있도록 제한하여야 함  단, 부득이하게 관리자 페이지를 외부에 노출해야 하는 경우 관리자 페이지 로그인 시 2차 인증(otp, vpn, 인증서 등) 적용 필요함 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 추측하기 쉬운 관리자 페이지 경로(/admin, /manager, /master, /system 등) 접근을 시도하여 관리자 페이지가 노출되는지 확인 | |

웹(Web)

|  |  |
| --- | --- |
| AE (상) | 24. 관리자 페이지 노출 |
| Step 2) 추측하기 쉬운 포트(7001, 8080, 8443, 8888 등) 접속을 시도하여 관리자 페이지가 노출되는지 확인  Step 3) 관리자 페이지의 로그인 창에 기본 관리자 계정(admin, administrator, manager 등) 및 패스워드를 입력하여 로그인 가능한지 확인 | |



웹(Web)

|  |  |
| --- | --- |
| AE (상) | 24. 관리자 페이지 노출 |
| Step 4) 관리자 페이지 로그인 후 식별된 하위 페이지(/admin/main.asp, /admin/menu.html 등) URL을 새 세션에서 직접 입력하여 인증 과정 없이 접근 가능한지 확인   * 보안설정방법 * 일반 사용자의 접근이 불필요한 관리자 로그인 페이지 주소를 유추하기 어려운 이름으로 변경하고 관리자 페이지 접근 포트도 변경함 * 관리자 페이지의 하위 페이지 URL을 직접 입력하여 접근하지 못하도록 페이지마다 세션 검증이 필요함 * 관리자 페이지 이외에도 특정 사용자만 접근 가능한 페이지들은 정상적인 프로세스에 따라 접근할 수 있도록 페이지마다 세션 검증이 필요함 * 웹 방화벽을 이용하여 특정 IP만 접근 가능할 수 있도록 룰셋 적용 | |
| 조치 시  영향 | 일반적으로 영향 없음 |

웹(Web)

|  |  |
| --- | --- |
| PT (상) | 25. 경로 추적 |
| 취약점 개요 | |
| 점검내용 | * 웹 서버와 웹 애플리케이션의 파일 또는 디렉터리의 접근 통제 여부 점검 |
| 점검목적 | * 웹 서버 또는 웹 애플리케이션의 중요한 파일과 데이터의 접근 및 실행을 방지하고자 함 |
| 보안위협 | * 웹 서버와 웹 애플리케이션의 파일 또는 디렉터리 접근이 통제되지 않아 웹 서버 또는 웹 애플리케이션의 중요한 파일과 데이터에 접근을 허용하는 취약점으로 웹 루트 디렉터리에서 외부의 파일까지 접근하여 이를 실행할 수 있음 |
| 참고 | ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 서버, 웹 방화벽 |
| 판단기준 | 양호 : 웹 루트 디렉터리보다 상위 디렉터리(예. /root)에 접근이 불가능한 경우 |
| 취약 : 웹 루트 디렉터리보다 상위 디렉터리에 접근이 가능한 경우 |
| 조치방법 | 사용자가 임의로 접근할 수 있는 최상위 디렉터리를 웹 루트 디렉터리로 설정하여 웹 서버의 시스템 루트 디렉터리로 접근하지 못하게 제한 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 웹 브라우저에 표시할 페이지를 지정하는 파라미터 값을 임의의 경로가 포함된 값으로 변조 후 전송하여 해당경로의 파일 내용이 웹 브라우저에 표시되는지 확인  ../../../../../../../../../../../../etc/passwd  ../../../../../../../../../../../../winnt/win.ini  ../../../../../../../../../../../../boot.ini  Step 2) "Step 1)"에서 변조하여 전송한 파라미터 값을 아래의 인코딩(또는 치환, 종단문자 추가)을 적용하여 재전송 후 해당경로의 파일 내용이 웹 브라우저에 표시되는지 확인  ※ URL 인코딩: .(%2e), /(%2f), \(%5c)  ※ 16bit 유니코드 인코딩: .(%u002e), /(%u2215), \(%u2216)  ※ 더블URL 인코딩: .(%252e), /(%252f), \(%255c) | |



웹(Web)

|  |  |
| --- | --- |
| PT (상) | 25. 경로 추적 |
| ※ 경로 치환: …//, ….\\, ….\/, …./\  ※ 종단 문자 추가: [파일명]%00.jpg, [파일명]%0a.jpg   * 보안설정방법   \* 웹 사이트에서 접근하려는 파일이 있는 디렉터리에 chroot 환경1) 적용 시 경로 추적 공격을 최소화할 수 있음  1) chroot 디렉터리는 해당 디렉터리가 루트처럼 다뤄짐. chroot 파일 시스템은 대부분의 유닉스를 기반으로 한 플랫폼에서 지원 가능하며, 윈도우 플랫폼에서는 적절한 시작 디렉터리를 새로운 논리 드라이브로 만들어 웹 사이트에서 해당 드라이브를 통하여 접근하게 함 (예: 웹 사이트의 최상위 디렉터리를 웹 루트 디렉터리로 제한)  □ IIS  인터넷 정보서비스(IIS) 관리> [해당 웹 사이트]> [속성]> [홈 디렉터리]탭> [구성] 버튼 선택> [옵션] 탭에서 [부모 경로 사용] 체크 해제   * 애플리케이션 소스 파일을 수정하여 파일 내용을 웹 브라우저에 표시할 수 있는 디렉터리를 특정 디렉터리로 한정하고 이 외의 다른 디렉터리에서는 파일 내용을 표시할 수 없도록 설정해야 함 * PHP를 사용하는 경우 php.ini 에서 magic\_quotes\_gpc를 On으로 설정하여 .\./ 와 같은 역 슬러시 문자 입력 시 치환되도록 설정 * 웹 사이트에서 사용되는 파라미터 값 대상으로 특수 문자를 필터링하도록 웹 방화벽 룰셋   적용 | |
| 조치 시  영향 | 일반적으로 영향 없음 |

웹(Web)

|  |  |
| --- | --- |
| PL (상) | 26. 위치 공개 |
| 취약점 개요 | |
| 점검내용 | * 예측 가능한 폴더의 위치 사용 여부 및 불필요한 파일의 존재 여부 점검 |
| 점검목적 | * 공격자가 폴더의 위치를 예측하여 파일 및 정보 획득을 방지하고자 함 |
| 보안위협 | * 폴더나 파일명의 위치가 예측 가능하여 쉽게 노출될 경우 공격자는 이를 악용하여 대상에 대한 정보를 획득하고 민감한 데이터에 접근 가능 |
| 참고 | - |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 서버 |
| 판단기준 | 양호 : 불필요한 파일이 존재하지 않고, 샘플 페이지가 존재하지 않을 경우 |
| 취약 : 불필요한 파일이 존재하거나, 샘플 페이지가 존재하는 경우 |
| 조치방법 | 웹 루트 디렉터리 이하 모든 불필요한 파일 및 샘플 페이지 삭제 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 웹 루트 디렉터리 내 웹 서비스에 불필요한 확장자(.bak, .backup, .org, .old, .zip, .log,  .sql, .new, .txt, .tmp, .temp) 파일이 존재하는지 확인 | |



웹(Web)

|  |  |
| --- | --- |
| PL (상) | 26. 위치 공개 |
| Step 2) 각종 샘플페이지(cgi-bin, manual, usage, iissamples, scripts, iisHelp, IISAdmin,  \_vit\_bin, Printers, phpinfo.php, examples, jsp, servlets)의 디렉터리 및 파일 존재 여부 확인  Step 3) 네트워크 다이어그램 및 구성, 사용자 이름/암호, 오류 메시지 내용, 웹 사이트 개발, 테스트 및 UAT, 준비 버전, 민감한 정보를 포함한 디렉터리 검색 등 정보를 확인하고자 하는 모든 내용을 아래의 검색엔진을 사용하여 결과 확인  ※ Baidu, Binsearch, Bing, DuckDuckGo, ixquick/Startpage, Google, Shodan, PunkSPIDER 등 | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| \*.bak | \*.backup | \*.org | \*.old | \*.new | \*.txt |
| \*.zip | \*.log | \*.! | \*.sql | \*.tmp | \*.temp |

웹(Web)

|  |  |
| --- | --- |
| PL (상) | 26. 위치 공개 |
| * 보안설정방법 * robots.txt 파일 작성을 통해 검색 차단할 디렉터리, 확장자, 페이지 등을 지정할 수 있으며   HTML의 HEAD 태그 내에 META 태그를 추가하여 검색엔진의 인덱싱을 차단   * 웹 디렉터리를 조사하여 아래의 삭제해야 할 파일 확장자에 포함된 백업 파일을 모두 삭제하고, \*.txt 확장자와 같이 작업 중 생성된 일반 텍스트 파일이나 이미지 파일 등도 제거함   ※ 삭제해야 할 파일 확장자 예시   * 백업 파일은 백업 계획을 수립하여 안전한 곳에 정기적으로 백업해야 하며 웹 서버에서는 운영에 필요한 최소한의 파일만을 생성하여야 함 * 웹 서버 설정 후 디폴트 페이지와 디폴트 디렉터리 및 Banner를 삭제하여 Banner Grab에 의한 시스템 정보 유출을 차단함 * Apache, IIS, Tomcat 등 각 웹 서버 설정 시 함께 제공되는 샘플 디렉터리 및 매뉴얼   디렉터리, 샘플 애플리케이션을 삭제하여 보안 위험을 최소화함 | |
| 조치 시  영향 | 일반적으로 영향 없음 |



웹(Web)

|  |  |
| --- | --- |
| SN (상) | 27. 데이터 평문 전송 |
| 취약점 개요 | |
| 점검내용 | * 서버와 클라이언트 간 통신 시 데이터의 암호화 여부 점검 |
| 점검목적 | * 서버와 클라이언트 간 통신 시 데이터의 암호화 전송 미흡으로 정보 유출의 위험을 방지하고자 함 |
| 보안위협 | * 웹상의 데이터 통신은 대부분 텍스트 기반으로 이루어지기 때문에 서버와 클라이언트 간에 암호화 프로세스를 구현하지 않으면 간단한 도청(Sniffing) 을 통해 정보를 탈취 및 도용할 수 있음 |
| 참고 | ※ Sniffing: 스니퍼(sniff: 냄새를 맡다, 코를 킁킁거리다)를 이용하여 네트워크상의 데이터를 도청하는 행위  ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드, 웹 서버 |
| 판단기준 | 양호 : 중요정보 전송구간에 암호화 통신이 적용된 경우 |
| 취약 : 중요정보 전송구간에 암호화 통신이 이루어지지 않는 경우 |
| 조치방법 | 사이트의 중요정보 전송구간(로그인, 회원가입, 회원정보관리, 게시판 등)  암호화 통신(https, 애플리케이션방식) 적용 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 중요정보(인증정보, 개인정보 등)를 송수신하는 페이지 존재 여부 확인 | |

웹(Web)

|  |  |
| --- | --- |
| SN (상) | 27. 데이터 평문 전송 |
| Step 2) 중요정보 송수신 페이지가 암호화 통신(https, 데이터 암호화 등)을 하는지 확인  Step 3) 취약한 버전의 암호 프로토콜 사용 시 암호화된 통신 내용이 유출될 수 있어 취약한 버전의 SSL(SSL 2.0, 3.0) 사용 여부를 점검   * 보안설정방법 * 웹상에서의 전송 정보를 제한하여 불필요한 비밀번호, 주민등록번호, 계좌정보와 같은 중요정보의 전송을 최소화하여야 하며, 중요정보에 대해서는 반드시 SSL 등의 암호화 통신을 사용하여 도청으로부터의 위험을 제거함 * 쿠키와 같이 클라이언트 측에서 노출되는 곳에 비밀번호, 인증인식 값, 개인정보 등의 정보를 기록하지 않음 * 암호화 전송 시 프로토콜 설계의 결함이 있는 SSLv2, SSLv3, TLSv1.0, TLSv1.1은 비활성화 필수, TLSv1.2 이상 사용을 권장함   ※ 웹 서버 별 상세 설정   * Apache   httpd-ssl.conf 또는 ssl.conf의 SSL 관련 VirtualHost 설정에 아래를 추가 SSLProtocol all -SSLv2 -SSLv3 -TLSv1 -TLSv1.1   * IIS   [SSL v2 사용 안 함] [HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHAN NEL\Protocols\SSL 2.0\Server]  하위에 '새로만들기' > 'DWord(32비트)' 값 선택 > 이름 부분에 'Enabled' 입력 > 데이터 부분에 '0' 입력 > 시스템 재부팅  [SSL v3 사용 안함] [HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHAN NEL\Protocols\SSL 3.0\Server]  하위에 '새로만들기' > 'DWord(32비트)' 값 선택 > 이름 부분에 'Enabled' 입력 > 데이터 부분에 '0' 입력 > 시스템 재부팅 | |
| 조치 시  영향 | IIS 웹 서버의 경우 취약한 프로토콜 비활성화 후 시스템 재부팅 필요 |



웹(Web)

|  |  |
| --- | --- |
| CC (상) | 28. 쿠키 변조 |
| 취약점 개요 | |
| 점검내용 | * 쿠키 사용 여부 및 사용하는 경우 안전한 알고리즘으로 암호화 여부 점검 |
| 점검목적 | * 쿠키를 사용하는 경우 안전한 알고리즘으로 암호화하여 공격자가 쿠키 인젝션 등과 같은 쿠키 값 변조를 통한 다른 사용자로의 위장 및 권한 변경을 방지하고자 함 |
| 보안위협 | * 클라이언트에 전달되는 쿠키에 사용자 식별 값이 평문으로 노출될 경우 쿠키 변조를 통해 다른 사용자의 유효한 세션을 취득할 수 있으며, 기타 중요정보의 유출 및 변조 가능함 |
| 참고 | ※ 쿠키(Cookie): 인터넷 사용자가 어떠한 웹 사이트를 방문할 경우 그 사이트가 사용하고 있는 서버에서 인터넷 사용자의 컴퓨터에 설치하는 작은 기록 정보 파일  ※ 소스코드 및 취약점 점검 필요 |
| 점검대상 및 판단기준 | |
| 대상 | * 웹 애플리케이션 소스코드 |
| 판단기준 | 양호 : 쿠키를 사용하지 않고 Server Side Session을 사용하고 있거나, 쿠키(또는 Session)를 사용하는 경우 안전한 알고리즘(SEED, 3DES, AES)이 적용되어있는 경우 |
| 취약 : 안전한 알고리즘이 적용되어있지 않은 쿠키(또는 Session)를 사용하거나, Client Side Session을 사용하는 경우 |
| 조치방법 | 쿠키 대신 Server Side Session 방식을 사용하거나, 쿠키를 통해 인증 등 중요한 기능을 구현해야 할 경우엔 안전한 알고리즘(SEED, 3DES, AES 등) 적용 |
| 점검 및 조치 사례 | |
| * 점검방법   Step 1) 쿠키 내용 및 발행되는 쿠키에 중요정보(인증을 위한 ID, 권한을 위한 구분자 등)의 노출 여부 조사 | |

웹(Web)

|  |  |
| --- | --- |
| CC (상) | 28. 쿠키 변조 |
| Step 2) 쿠키의 중요정보를 변경하여 다른 사용자 및 권한으로 정상 이용이 가능한지 확인   * 보안설정방법 * 쿠키 대신 보안성이 강한 Server Side Session 방식 사용. Client Side Session 방식인 쿠키는 그 구조상 다양한 취약점에 노출될 수 있음 * 쿠키(또는 Session)를 사용해서 중요정보나 인증을 구현해야 할 경우엔 안전한 알고리즘   (SEED, 3DES, AES 등) 적용   * HTTP 헤더에 아래와 같이 설정하여 세션 ID 값은 HTTPS를 통해서만 전송되도록 설정하고, 자바스크립트를 통해 세션 ID 값 등 쿠키 정보가 유출되지 않도록 보호   Set-Cookie : secure, HttpOnly  Set-Cookie : domain=app.mysite.com  ※ HttpOnly 설정 관련 참고사항  https://msdn.microsoft.com/en-us/library/system.web.httpcookie.httponly.aspx https://[www.owasp.org/index.php/HttpOnly](http://www.owasp.org/index.php/HttpOnly) | |
| 조치 시  영향 | 일반적으로 영향 없음 |