소프트웨어 보안 과제

2016156007

소프트웨어학과 김영민

[1] 93.8 CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

제어되지 않은 사용자의 입력을 통해서 발생하는 오류입니다. 제거되지 않은 입력이 SQL 구문으로 해석될 때 발생합니다. SQL 구문을 제거하지 않으면 해당 입력이 일반 사용자 데이터 대신 SQL 구문으로 해석될 수 있습니다. 이 보안 약점은 쿼리 로직을 조직하여 보안 검사 우회, 백-엔드 데이터베이스 수정을 위한 추가적인 명령문 생성, 시스템 명령어 수행 등의 작업에 이용될 수 있습니다. SQL 삽입은 데이터 베이스 기반의 웹 사이트에서 흔히 발생하는데 이 취약점은 쉽게 발견이 가능하고, 공격이 쉬워서 최소권한 이용자 기반의 웹 사이트나 소프트웨어 패키지인 경우에도 또한 공격에 취약할 수 있습니다.

대부분 웹 양식에서는 이름과 암호 이외의 입력을 차단할 메커니즘이 없기 때문에 예방 조치를 취하지 않는 한 공격자는 입력 박스를 통해서 데이터베이스에 요청을 보내 전체 데이터베이스를 다운로드하거나 다른 불법적인 방식으로 상호작용하는 것이 가능합니다.

[2] 83.3 CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

운영체제 명령에 사용된 특별한 요소를 부적절하게 무효화하는 것으로 공격자가 예상치 못한 실행과, OS상의 위험한 명령을 직접 실행할 수 있도록 허용하게 한다. 이 보안 약점은 공격자가 직접적으로 접근할 수 없는 웹 어플리케이션 환경에서 취약한데 해당 보안 약점이 권한을 가지고 있는 프로그램에서 발생하게 된다면 이것은 공격자로 하여금 일반적으로 접근할 수 없는 명령어를 허용하게 한다. 또한 공격자가 가지지 못한 권한의 대체 명령어를 사용할 수 있게 만든다. 만약에 공격자가 통제하는 명령어가 공격하는 프로그램의 시스템 권한에 작동하여 구동되어 손상이 된다면 프로세스가 주된 기능을 수행하지 못할 수 있어서 문제가 악화된다. 해결을 위해 조금이라도 가능하다면 외부 프로세스 대신 라이브러리 호출을 사용하여야 하고, 실행할 프로그램이 입력 파일 안에서, 또는 기본 입력을 토대로 인수를 지정할 수 있도록 허용할 경우, 명령 줄 대신 해당 모드를 사용하여 인수를 전달하는 방법을 고려해야 합니다. 가능하다면 자동으로 데이터와 코드를 강제로 분리시키는 정형화된 메커니즘을 사용하는 것이 좋습니다.

[3] 79.0 CWE-120 Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')

입력 버퍼의 크기가 출력 버퍼의 크기보다 작은 지 확인하지 않고서 입력 버퍼를 출력 버퍼에 복사하여 버퍼 오버 플로우를 유발하는 것을 말합니다. 프로그램이 보유할 수 있는 것보다 많은 데이터를 버퍼에 넣으려고 할 때 또는 프로그램이 버퍼 경계 외부의 메모리 영역에 데이터를 넣으려고 할 때 오버 플로우가 발생하게 되는데 가장 간단한 유형과 원인으로는 프로그램이 버퍼 길이를 전혀 확인하지 않고 버퍼를 복사하는 사례가 있습니다. 여러가지 다른 변형 사례가 존재하지만 일반적으로 오버 플로우가 존재한다는 것은 프로그래머가 가장 기본적인 보안조치조차 고려하지 않고 있다는 것을 의미합니다. 이 보안 약점은 종종 자동화된 정적 분석 도구를 통해서 발견할 수 있고 최근에 만들어진 최신 도구들을 이용해 탐지하여 탐지 성공률을 높이고 있습니다. 자동 동적분석이나 수동 분석 또한 존재합니다. 대체적으로 버퍼 오버 플로우를 자동으로 완화시키거나 제거할 수 있는 기능을 가지고 있는 언어를 사용하는 것이 바람직합니다. 예를 들어 자바나 펄 같이 자체 메모리 관리를 수행하는 많은 언어에는 버퍼 오버 플로우가 적용되지 않습니다. 하지만 아무리 언어자체가 이론적으로 안전하다고 하더라도 기본 코드에 대한 언어 인터페이스가 여전히 오버 플로우가 될 수 있다는 점은 주의해야 합니다.

[4] 77.7 CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

해당 소프트웨어가 충분하게 사용자 제어 입력을 다른 사용자에게 제공되는 웹 페이지로에 사용되는 출력으로 표시되기 전에 검증, 필터링, 이스케이프 및 인코딩을 하지 않은 경우 발생합니다. 이 보안 약점은 세가지 유형이 있습니다. 첫째로 공격자가 피해자에게 위험한 콘텐츠를 피해자의 취약한 웹 응용 프로그램에 제공한 후에 피해자에게 다시 웹 브라우저에 실행시키는 것이고, 둘째로 데이터베이스, 메시지 포럼, 방문자 로그 또는 기타 신뢰할 수 있는 데이터 저장소에 공격자가 위험한 데이터를 저장해 피해자가 접속하여 악성 콘텐츠를 실행할 경우에 공격자가 명령을 실행해서 피해자의 중요한 데이터를 접속하는 유형이 있고 마지막으로 공격자가 악의적인 스크립트를 주입해서 쿠키와 같은 개인정보를 피해자의 컴퓨터에서 공격자한테 전송할 수 있게 하는 유형이 존재합니다. 대부분의 경우 피해자는 자신도 모르게 공격자에 의해서 자신 또한 공격을 시작할 수 있는데 아무리 신중한 사용자라 하여도 URL 인코딩 또는 유니코드를 이용하는 공격은 알아채기 힘들 수 있습니다. 탐지 방법으로는 자동 정적 분석과 블랙 박스 방법이 있는데 두 방법 모두 완벽한 솔루션은 아닙니다. 이 보안 오류를 완화시키기 위해서는 올바르게 그리고 보다 쉽게 인코딩 된 출력을 생성하는 언어, 라이브러리 또는 프레임 워크를 사용하는 것이 바람직합니다.

[5] 76.9 CWE-306 Missing Authentication for Critical Function

이 보안 약점은 소프트웨어가 권한이 존재하는 응용 프로그램 기능에 대해서 접속을 허용하기 전에 사용자 신원 확인을 수행하지 않은 경우에 발생하게 됩니다. 이 보안 약점은 종종 응용 프로그램 개발 프로세스의 아키텍처 및 디자인 단계에서 발생합니다. 이러한 문제의 예로는 McAfee Advanced Threat Defense의 웹 인터페이스에 있는 약점으로 들 수 있는데 이 보안 약점으로 인해 인증되지 않은 원격 공격자가 특수하게 조작된 HTTP 요청을 보내어 응용 프로그램으로 보내고 프로그램의 구성된 설정을 변경하여 관리 접속 권한을 얻을 수 있게 됩니다. 노출된 기능 및 응용 프로그램 기능에 따라 이 보안 약점의 영향으로는 정보 유출에서부터 응용 프로그램 손상까지 다양합니다. 이 보안 약점을 완화시키기 위해서는 대부분의 인증 관련 문제와 마찬가지로 취약점을 해결하는 방법에 대해서 일반적으로 권장 사항을 제공하는 것은 어렵지만 세가지 기본 규칙을 통해 잠재적으로 부적절한 인증 문제를 제거할 수 있습니다. 응용 프로그램 내에서 모든 권한 있는 자산이 무엇이 있는지 파악하고 식별해야 하고, 응용 프로그램 내에서 사용자의 역할 및 해당 접속 권한을 식별하고, 사용자의 자산에 접속할 수 있는 권한이 있는지 항상 확인해야 합니다.

[6] 76.8 CWE-862 Missing Authorization

이 보안 약점은 아키텍처 및 설계, 구현 및 운영 단계에서 도입하게 되는데 이 약점으로 인해서 소프트웨어는 이용자의 권한 부여, 데이터 접속 또는 일부 조치 수행이 정상적인지 또는 정확한지 여부를 판별할 수 없게 됩니다. 접속 제어 점검이 부족하여 모든 사용자가 원하는 대로 모든 자원에 대해서 접속하게 됩니다. 따라서 정보 노출, 서비스 거부 및 임의 코드 실행을 유발하게 되고 이러한 보안 약점으로 인해서 공격자는 중요한 데이터를 읽고 수정하고 권한이 있는 기능에 접속이 가능해집니다. 역할 기만 접속 제어를 사용해서 적절한 경계에서 시행해야 한다. 이 접근 방식은 수평 인증으로는 보호되지 않을 수 있습니다. 따라서 사용자가 동일한 역할을 가진 사람을 공격하는 것을 방지할 수는 없기 때문에 주의해야합니다. 탐지방법으로는 자동 정적 분석, 자동화된 동적 분석, 수동 분석, 수동 전적 분석 자동화된 결과 해석을 통한 동적 분석 등등 다양한 방법이 존재합니다. 이 보안 약점을 완화시키기 위해서는 소프트웨어를 익명, 일반, 권한 및 관리 영역으로 나누고 데이터와 프로그램 기능을 통해 역할을 신중하게 매핑해서 공격 영역을 가능한 줄이도록 해야 합니다. 도한 이 디자인할 때 이 보안 약점이 발생하지 않도록 하거나 또는 이 취약점을 피하기 쉬운 구조를 제공하는 검증된 라이브러리 또는 프레임 워크를 사용하는 것이 바람직합니다. 예로는 JAAS인증 프레임 워크 및 OWASP ESAPI 접속 제어 기능과 같은 인증 프레임 워크 사용을 고려하는 것이 좋다.

[7] 75.0 CWE-798 Use of Hard-coded Credentials

하드 인코딩 된 비밀번호는 일반적으로 시스템 관리자가 감지하기 어려운 심각한 인증 실패로 이어지게 되는데 일단 탐지되면 수정하기 어려울 수 있으므로 관리자가 제품을 완전히 비활성화해야 할 수 있습니다. 이에는 두가지 중요한 변형이 있습니다. 첫째로 인 바운드로 소프트웨어에 하드 코딩 된 비밀번호를 확인하는 인증 메커니즘이 포함되어 있는 것 말합니다. 둘째로 아웃 바운드로 소프트웨어가 다른 시스템 또는 구성 요소에 연결되며 해당 구성 요소에 연결하기 위한 하드 코드 된 비밀번호가 포함된 상태입니다.

탐지 방법으로는 블랙박스, 자동 정적 분석, 수동 정적 분석, 수동 동적 분석이 존재합니다. 출력에서 호출 트리 또는 유사한 인공물을 사용하여 연관된 동작을 검토하고 입력된 동작을 고정 문자열 또는 값과 비교하는 것으로 보이는지 확인하는 것이 좋습니다. 이 보안 약점을 완화시키기 위해서는 인 바운드 인증에 경우 최초 로그인 시 기본 사용자 이름 및 비밀번호, 키 또는 기타 인증 자격 증명을 하드 코딩하는 대신 사용자가 고유 한 강력한 비밀번호 또는 키를 입력해야 하는 처음 로그인 모드를 사용하는 것이 좋고 아웃 바운드의 경우에는 암호, 키 및 기타 자격 증명을 코드 외부에 강력하게 보호되고 암호화 된 구성 파일 또는 데이터베이스에 저장하고 키를 올바르게 보호하는 것이 좋습니다.

[8] 75.0 CWE-311 Missing Encryption of Sensitive Data

저장 또는 전송 전에 중요하거나 중요한 정보를 암호화하지 않은 것입니다. 적절한 데이터 암호화의 부재는 암호화를 적절하게 구현한 기밀성, 무결성 및 책임성의 보증을 거절하게 됩니다. 이 보안 약점을 완화시키고 점검하기 위해서는 우선 평문으로 전송, 저장 되어있다고 해서 모두 무조건 취약한 것은 아닌 것을 알고, 개인정보(주민등록번호, 여권 번호 등), 금융정보(카드번호, 계좌번호 등), 패스워드 등을 저장할 때에는 암호화하여 저장했는 가에 대해 살펴봐야합니다. 또 통신채널을 통해 민감한 정보를 전송할 때에도 암호화했는지 확인하고 데이터나 자원이 암호화하여 보호해야 할 만큼 가치가 있는 것인지 분명히 하고 관련분야의 전문가들이 강력하다고 간주하는 알고리즘을 사용, 자체적으로 개발한 알고리즘은 사용하지 말아야 하며, 구현방식을 정확히 하고, SSL과 같은 표준 보안기술을 적용하는 것이 좋습니다. 일반적으로 공격자는 암호화 알고리즘 자체를 직접 공격하지 않으며, 정보가 중간에서 도청되어도 해독이 불가능하도록 SSL을 이용한 서비스를 제공하는 것이 바람직합니다. 하지만 중간자 공격과 같은 정보 가래치기 공격은 가능하기 때문에 꾸준히 로컬 네트워크 점검을 통해 꾸준히 수시로 중간자 공격을 예방하는 것이 좋습니다.

[9] 74.0 CWE-434 Unrestricted Upload of File with Dangerous Type

웹사이트에서 파일을 첨부하는 기능을 구현할 때에 각별한 유의가 요구되는데, 공격자가 첨부 기능을 이용한 Web Shell과 같은 악성 파일을 서버에 업로드하고 웹에서 실행하여 서버의 권한을 획득할 수 있고, 폴더나 파일구조, 프로그램 소스내용, DB등 타 서버의 연결 정보 같은 서버의 각종 정보를 이용하여 다양한 악의적 행동을 할 수 있게 됩니다. 결과적으로 공격자가 시스템 관리자 권한을 뺏기는 것과 마찬가지로 가장 위험한 보안 취약점 중 하나에 해당하게 됩니다. 이 보안 약점은 주로 ASP와 PHP로 작성된 응용 프로그램에서 발생하는데 수신자는 수신인으로부터 받은 코드를 사용하여 시스템을 쉽게 제어하고 임의 코드를 실행할 수 있게 됩니다. 탐지 방법으로는 자동화된 결과 해석을 통한 동적 분석, 수동 결과 해석을 통한 동적 분석, 수동 정적 분석, 자동 정적 분석, 건축 또는 디자인 검토가 있습니다. 대책으로 가장 우선시되는 것은 웹서비스가 되지 않는 디렉터리로 파일이 업로드 되도록 하고 파일 확장자를 서버 쪽으로 화이트리스트 기반으로 체크를 하고 업로드가 되도록 하는 것이 바람직합니다. 또한 파일명 특수문자 포함 여부도 검사하여 %, ; 또는 .. 같은 특수 문자를 필터링 하는 규칙을 적용하는 것이 좋습니다.

[10] 73.8 CWE-807 Reliance on Untrusted Inputs in a Security Decision

응용 프로그램은 입력의 존재 또는 값에 의존하는 보호 메커니즘을 사용하지만, 신뢰할 수 없는 행위자에 의해 보호 메커니즘을 우회하는 방식으로 입력을 수정할 수 있는 것을 말합니다. 공격자는 사용자 지정 클라이언트나 다른 공격을 사용하여 입력을 변경할 수 있는데 이 변경 사항이 감지되지 않도록 할 수 있습니다. 이러한 입력 값을 기반으로 하여 인증 및 권한 부여와 같은 보안 결정이 내려지면 공격자는 소프트웨어의 보안을 우회해서 통과하는 것이 가능하게 됩니다. 충분한 암호화, 무결성 검사 또는 기타 메커니즘이 없다면 외부인이 보낸 입력을 신뢰해서는 안됩니다. 공격자는 보안 결정을 우회하여 보호 대상에 액세스 할 수 있습니다. 결과는 관련 기능에 따라 다르지만 신뢰할 수 없는 사용자에게 추가 권한을 부여하는 것부터 중요한 보안 검사를 무시하는 것까지 다양 할 수 있습니다. 궁극적으로 이 취약점으로 인해 민감한 데이터가 노출되거나 수정되거나 시스템 충돌이 발생하거나 임의 코드가 실행될 수 있습니다. 이 보안 약점을 완화하기 위해서는 공격 영역을 식별하고 줄여야 하는데 보안 결정에 사용되는 모든 입력을 식별하고 제출된 입력에 전혀 의존할 필요가 없도록 디자인을 수정할 수 있는지 결정하는 것이 좋습니다. 예를 들어, 사용자 세션에 대한 중요한 정보를 외부 데이터 내에 기록하는 대신 서버 측에서 유지할 수 있습니다.