컴퓨터보안 (AWS4016-02)

Week 13 – Security Scenario

컴퓨터공학과
2020112119 강동희
2023. 06. 06

목 차

1. 서론

1.1문제 정의

2. 본론

2.1 환경 세팅

2.2 보물찾기

2.3 보안약점 시나리오 작성

2.4 보안약점 시나리오 구현

3. 결론

3.1 느낀 점

서 론

1.1 문제 정의

[SQL 삽입]

- SQL 삽입 보안약점은 입력된 데이터에 대한 유효성 검사를 하지 않을 경우, 공격자가 입력 폼 및 URL 입력창에 SQL 문장을 삽입하여 DB로부터 정보를 열람하거나 조작할 수 있는 보안약점이다. 내재된 웹 응용프로그램에서는 사용자로부터 입력된 값을 검증 없이 넘겨받아 SQL 문장을 생성하기 때문에 개발자가 의도하지 않은 조작된 SQL 문장이 실행되어 정보유출의 위험성이 있다.

[크로스사이트 스크립트]

- 크로스사이트 스크립트(XSS)는 검증되지 않은 입력값으로 인해 victim의 웹 브라우저에서 의도하지 않은 악성 스크립트가 실행되는 보안약점이다. 공격자는 크로스사이트 스크립트 보안약점을 통해 사용자의 개인정보 및 쿠키정보 탈취, 악성코드 감염, 웹 페이지 변조 등을 진행할 수 있고 희생자에게 비정상적인 기능을 수행하게 할 수 있다. 해당 보안약점은 공격유형에 따라 Reflected XSS, Stored XSS, DOM based XSS 유형으로 분류할 수 있다.

[경로순회]

- 검증되지 않은 외부 입력값을 통해 파일 및 서버 등 시스템 자원에 대한 접근 혹은 식별을 허용하는 경우 입력값 조작으로 시스템이 보호하는 자원에 임의로 접근할 수 있는 보안약점이다. 공격자는 경로저작 및 자원삽입을 통해 허용되지 않은 권한을 획득하고 설정에 관계된 파일을 변경 및 실행시킬 수 있으며 서비스 장애 등을 유발시킬 수 있다.

[역직렬화]

- 역직렬화(Deserialization)은 반대 연산으로 바이너리 파일(Binary File)이나 바이트 스트림(Byte Stream) 으로 객체 구조로 복원할 수 있다. 송신자가 네트워크를 이용하여 직렬화된 정보를 수신자에게 전달하는 과정에서 공격자가 전송 또는 저장된 스트림을 조작할 수 있는 경우에는 역질렬화를 이용하여 악의적으로 데이터와 시스템을 침해할 수 있는 보안약점이다.

본 론

2.1 환경 세팅

[1] Virtual Box 다운로드

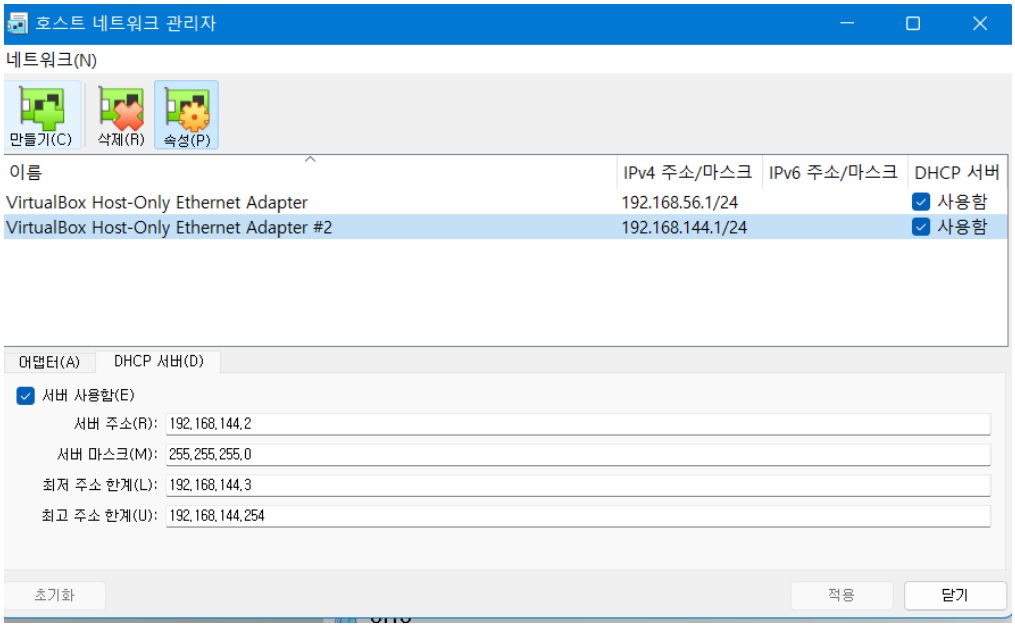
실습 진행을 위해 Virtual Box를 설치한다.

[2] 가상머신 설치

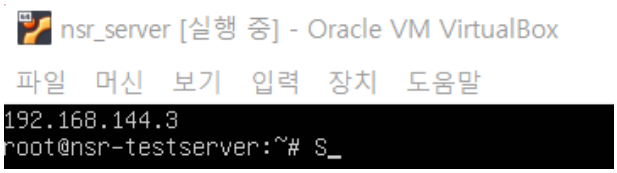
가상머신 이미지 파일을 통해 가상머신과 서버 설치를 진행한다.

[3] 네트워크 세팅

네트워크 관리자를 통해 DHCP 서버와 호스트 네트워크 관리자 설정을 통해 호스트 전용 어댑터 설정한다.



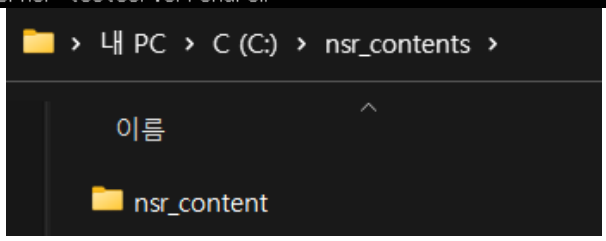
네트워크 설정 후 설정이 완료되었는지 확인하기 위해서 'nsr_server' 가상머신을 실행한다.



[4] 공유 디렉토리 설정

초기 세팅을 통해 공유 디렉토리를 설정한다.

```
root@nsr-testserver:/home/nsr-testserver/share# cp -r /etc/nsr_content/ /home/nsr-testserver/share/
root@nsr-testserver:/home/nsr-testserver/share#
```



2.2 보물찾기

[1] SQL 삽입

회원가입 창	로그인 창	주소 창	검색 창

[2] 크로스사이트 스크립트

Reflected XSS	Stored XSS	DOM based XSS

[3] 경로순회

파일 다운로드	파일 내용 출력	파일 삭제

[4] 역직렬화

데이터 위조	원격 명령어 실행	서비스 거부 공격

[5] Certification & Session 관리

인증된 사용자의 계정이 노출되거나, 다른 사용자의 세션을 탈취하여 권한을 획득한다.

인증 절차 약화	세션 토큰 누출	세션 하이재킹

[6] 파일 업로드 취약점

웹 애플리케이션에 파일을 업로드 하는 기능이 보안 검사 혹은 제한 없이 구현되어 있는 경우

파일 확장자 필터링	악성 파일 업로드	파일 크기 제한 우회	경로 조작 공격

[7] CSRF (Cross-Site Request Forgery)

악의적인 공격자가 인증된 사용자의 권한을 도용하여 웹사이트의 특정 요청을 위조하는 공격

이메일 전송	게시글 작성

[8] Directory List 표시

웹 서버가 디렉토리 내용을 노출하는 기능이 구성되지 않아 공격자가 웹 서버의 파일, 디렉토리 구조에 대한 정보를 쉽게 얻을 수 있는 상태

기본 디렉토리 목록 표시	디렉토리 접근 권한 설정 취약	디렉토리 목록 제한 우회

[9] 보안 설정 부족

시스템에서 보안 구성을 적절히 설정하지 않아 발생하는 취약점

권한 부여 설정 부족	기본 암호 사용	디버그 모드 활성화

[10] 클라이언트 보안 약점

웹 애플리케이션을 사용하는 클라이언트 측에서 발생할 수 있는 보안 취약점

클라이언트 측 인증 우회	악성 다운로드	브라우저 보안 설정 우회

2.3 보안약점 시나리오 작성

1. 보안약점명
CSRF (Cross-Site Request Forgery)
2. 주제 및 시나리오
CSRF 공격을 통한 pale moon 브라우저의 사용자 계정을 악용한 악의적인 동작 수행
3. 보안약점 원인
Pale moon 브라우저에서는 웹사이트 간 요청 위조(CSRF)를 방지하기 위한 내장된 보호 기능이 부족하거나 비활성화되어 있다. 이로 인해 공격자는 사용자의 인증 정보를 이용하여 악의적인 요청을 보낼 수 있다.
4. 공격방법
1. 공격자는 악성 웹사이트를 만들어 사용자들의 접속을 유인한다. 2. 피해자가 악성 웹사이트를 방문하면, 해당 웹사이트에서는 pale moon 브라우저에 대한 악의적인 요청을 생성한다. 3. 피해자의 브라우저는 사용자의 인증 정보(쿠키)를 가지고 있기 때문에, 악성 요청이 pale moon 브라우저에서 유효한 것처럼 보인다. 4. pale moon 브라우저는 악성 요청을 허용하고, 해당 요청을 서버로 전송한다. 5. 서버는 피해자의 인증 정보를 확인하고 악성 요청을 처리한다.
5. 대응방안
CSRF 방어 기능 활성화 : pale moon 브라우저에서 CSRF 방어 기능을 활성화하여, 웹사이트 간 요청 위조 공격을 방지한다. 이를 통해 사용자의 인증 정보를 이용한 악의적인 요청을 차단할 수 있다. SameSite 쿠키 속성 설정 : SameSite 쿠키 속성을 설정하여, 외부 도메인으로의 쿠키 전송을 제한한다. 이를 통해 공격에 사용되는 피싱 사이트나 악의적인 웹사이트로의 쿠키 전송을 방지하여 보안을 강화한다.

1. 보안약점명
디렉토리 목록 표시 취약점
2. 주제 및 시나리오
파일노출 – 디렉토리 목록 표시를 통한 중요 파일 노출
3. 보안약점 원인
Pale moon 웹사이트에서 디렉토리 목록 표시 기능을 제공하며, 이 기능은 적절한 보안 설정이 되어 있지 않다. 따라서 공격자는 디렉토리 목록을 통해 웹사이트 내에 저장된 파일들의 정보를 노출시킬 수 있다.
4. 공격방법
1. 공격자는 Pale moon 웹사이트의 디렉토리 목록 표시 기능을 이용하여 웹사이트 내부의 디렉토리 구조를 탐색한다. 2. 디렉토리 목록에서는 웹사이트에 저장된 파일들의 정보(파일 이름, 크기, 수정일 등)가 포함되어 있으며, 공격자는 이를 이용하여 민감한 정보를 탈취할 수 있다. 3. 노출된 파일 정보를 통해 공격자는 시스템의 구조나 저장된 파일의 위치 등을 파악할 수 있으며, 이를 이용하여 다른 공격을 시도할 수 있게 된다.
5. 대응방안
디렉토리 목록 숨김 : Pale Moon 웹사이트에서 디렉토리 목록 표시 기능을 비활성화하거나 적절한 접근 제어를 설정하여 디렉토리 구조를 외부에 노출시키지 않도록 해야 한다. 접근 권한 설정 : 웹서버에서는 웹사이트 내의 디렉토리에 대한 적절한 접근 권한을 설정해야 한다. 민감한 파일이나 디렉토리에 대한 접근을 제한하여 보안을 강화한다. 웹 방화벽 구성 : 웹 방화벽을 구성하여 디렉토리 목록 표시와 관련된 요청을 필터링하고, 불필요한 정보 노출을 방지한다. 보안 감사 및 취약점 스캐닝 : 정기적인 보안 감사와 취약점 스캐닝을 통해 디렉토리 목록 표시와 관련된 취약점을 식별하고 조치한다.

1. 보안약점명

Remote File Inclusion(원격 파일 포함)

2. 주제 및 시나리오

시스템 제어 – 웹 shell 파일 업로드 및 시스템 명령의 전송을 통한 시스템 제어

3. 보안약점 원인

실습 웹사이트에서 파일 업로드 기능이 존재하며, 파일 업로드 시 확장자 검사나 안전성 검사를 수행하지 않는다.
공격자는 웹으로 직접 실행될 수 있는 스크립트 파일(asp, jsp, php 파일 등)을 업로드 할 수 있다.
업로드된 파일이 실행되면서 시스템 내부 명령어를 실행하거나 외부와의 연결을 통해 시스템을 제어할 수 있다.

4. 공격방법

1. 공격자는 웹사이트의 파일 업로드 기능을 이용하여 웹쉘 파일(webshell.php)을 업로드 한다.

2. 업로드된 웹쉘 파일을 실행하기 위해 공격자는 해당 파일의 URL을 알아낸다.

3. 공격자는 웹쉘 파일을 실행하기 위해 웹쉘 URL을 통해 시스템 명령어를 전송한다.

4. 서버는 웹쉘 파일을 실행하고, 시스템 명령어를 수행하여 시스템 제어가 이루어진다.

5. 대응방안

파일 업로드 시 확장자 검사 및 MIME type 검사를 수행하여 실행 가능한 스크립트 파일의 업로드를 방지한다.

업로드 된 파일의 경로를 외부에서 접근할 수 없도록 보호한다.

업로드 된 파일을 저장할 디렉토리의 실행권한을 제한한다.

웹 서버의 보안 설정을 강화하고, 웹쉘 파일 실행을 방지하는 방화벽을 사용한다.

시큐어 코드 예시(python)

```
// 파일 업로드 시 확장자 검사 및 MIME 타입 검사
if($_FILES['uploaded_file']['error'] === UPLOAD_ERR_OK) {
  $fileInfo = finfo_open(FILEINFO_MIME_TYPE);
  $mimeType = finfo_file($fileInfo, $_FILES['uploaded_file']['tmp_name']);
  $allowedExtensions = array('jpg', 'png', 'gif');
  $allowedMimeTypes = array('image/jpeg', 'image/png', 'image/gif');
  $fileExtension = pathinfo($_FILES['uploaded_file']['name'], PATHINFO_EXTENSION);
  if(!in_array($fileExtension, $allowedExtensions) || !in_array($mimeType,
$allowedMimeTypes)) {
    // 업로드된 파일이 허용된 확장자 또는 MIME 타입이 아닌 경우 에러 처리
    // 예: 파일 업로드 실패 메시지 출력
    die("Invalid file format.");
  }
  // 파일 저장 로직
  // ...
}
```

파일 업로드 시 업로드 된 파일의 확장자와 MIME Type 을 검사하여 허용된 확장자 및 MIME type 인지 확인한다. 만약 허용되지 않은 확장자, 타입의 파일이 업로드 되면, 업로드를 거부하고 예외처리를 수행한다.

2.4 보안 약점 시나리오 구현

1. 보안약점명
CSRF (Cross-Site Request Forgery)
2. 주제 및 시나리오
CSRF 공격을 통한 pale moon 브라우저의 사용자 계정을 악용한 악의적인 동작 수행

```
<!DOCTYPE html>
<html>
<head>
    <title>CSRF Vulnerable Page</title>
</head>
<body>
    <h1>CSRF Vulnerable Page</h1>
    <!-- 사용자의 인증 정보(쿠키)를 이용하여 악성 요청을 보내는 폼 -->
    <form id="csrfForm" action="http://example.com/process" method="POST">
        <!-- 악성 요청을 표시하기 위한 hidden 필드 -->
        <input type="hidden" name="action" value="deleteUser">
    </form>
    <!-- 악성 웹사이트에서 피해자를 유인하기 위한 링크 -->
    <a href="http://malicious-site.com" onclick="submitForm()">Click here!</a>
    <!-- JavaScript 를 사용하여 폼을 자동으로 제출하는 함수 -->
    <script>
        function submitForm() {
            // CSRF 폼 가져오기
            var csrfForm = document.getElementById('csrfForm');
            // CSRF 폼 제출
            csrfForm.submit();
        }
    </script>
</body>
</html>
```

악의적인 웹사이트를 구현했다. 사용자가 "Clink Here'링크를 클릭하면 자바스크립트 함수 'submitForm()' 이 호출되어 CSRF 폼을 자동으로 제출하게 된다. 이 form 은 악성 요청을 표시하는 hidden 필드와 함께 외부 서버('example.com/process)로 POST 요청을 보낸다.

```
<!-- CSRF 토큰 생성 -->
<form id="myForm" action="process.php" method="post">
<input type="hidden" name="csrf_token" value="<?php echo generateCSRFToken(); ?>">
<input type="submit" value="Submit">
</form>
<script>
// CSRF 토큰 검증
document.getElementById("myForm").addEventListener("submit", function(event) {
 var token = document.querySelector('input[name="csrf_token"]').value;
 var xhr = new XMLHttpRequest();
 xhr.open("POST", "check_token.php", false);
 xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
 xhr.send("csrf_token=" + encodeURIComponent(token));
 if (xhr.status !== 200) {
         event.preventDefault(); // 폼 전송 차단
```

```
            alert("CSRF 공격 감지됨!");
    }
  });
</script>
```

CSRF 공격을 방지하기 위한 코드로 JacaScript 를 사용해 CSRF 토큰을 검증한다. 폼 요소가 포함된 HTML 코드는 'myForm' 이라는 ID 를 가진 폼을 생성한다. 해당 form 에는 'csrf_token' 이라는 숨겨진 필드가 있으며, PHP 의 'generateCSRFToken()' 함수를 사용하여 CSRF 토큰을 생성하고 해당 필드에 값을 할당한다. 이는 폼이 제출될 때 CSRF 토큰이 함께 전송된다.

JavaScript 코드는 'myForm' 의 제출 이벤트를 수신하는 리스너를 추가한다. 폼이 제출될 때, JS 는 CSRF 토큰을 가져와 XMLHttpRequest 객체를 사용해 'check_token.php'에 POST 요청을 전송한다. 요청 헤더에는 적절한 콘텐츠 타입을 설정하고, 요청에는 CSRF 토큰을 인코딩하여 포함한다. 서버의 응답 상태 코드가 200 이 아닌 경우, 즉 CSRF 토큰이 유효하지 않은 경우 폼의 제출을 차단하고 경고 메시지를 표시한다.

3 결 론

3.1 느낀 점

다양한 웹사이트의 보안 약점에 대해서 알 수 있었다. 특히 보안 약점 시나리오를 구성하고 구현하기까지 어떤 전제 조건이 필요할지, 어떤 취약점을 어떻게 발견할 수 있을지 모든 상황에 대해서 시뮬레이션을 돌려보고 정리하는 과정이 재미있었다. 특히 실습과제 2~3 에서 보안 약점 시나리오 작성 및 구현을 'CSRF 토큰' 에 대한 보안약점을 위주로 작성했다. 1~2 학년 때 웹서비스 구현 동아리에서 보안을 위한 csrf 토큰에 대해서 구현했었는 데 당시에는 보안을 위해서 사용한다, 정도로만 이해하고 지나쳤던 부분이였다. 하지만 이번 실습을 통해 웹사이트의 보안 취약점에 대해서 정리하면서 왜 csrf 토큰을 통한 인증, 구현이 필요했는지 알 수 있었다. 당시에는 django 를 통한 웹사이트를 구현해서 쉽게 만들 수 있었는데 기초적으로 html 을 사용해서 csrf 토큰의 전송과 인증하는 부분을 구현하려고 하니 다양한 상황을 고려해야 하고, 그중에서도 간단하게 구현하려고하니 조금 힘들었다. 이번 실습에서는 지난 프로젝트 경험에서 궁금했던 점을 이른과 실습을 통해서 확인하여 의미있는 실습이였던 것 같다.