파일 업로드 취약점

asp 웹쉘 사례

bbs.zip 129.0KB

파일 업로드 (첨부파일 기능)

- 웹 서비스 첨부 파일 기능과 환경 설정 미흡을 이용하여 **악의적인 스크립트가 포함된 파일**을 업로드한 후에 웹 서버에 침투하는 공격
- 공격자는 서버 사이드 스크립트를 이용하여 웹쉘(WebShell) 제작
  - 서버 사이트 스크립트 언어 : PHP, JSP, NET, ASP 등
  - 웹쉘(WebShell) : 원격에서 웹 서버를 제어하기 위한 목적으로 제작되었으나, 현재는 악성코드로 분류하여 탐지
- 게시판 첨부 파일, 이력서 첨부 파일, 이미지 첨부 파일, 파일 공유 기능 등에서 발생
- 웹쉘(WebShell)은 직접 제작하거나, 외부 사이트에서 쉽게 구할 수 있음

GitHub GitHub - tennc/webshell: This is a webshell open source proj...

PHP의 웹쉘 주요 함수

함수명	내용
System	외부명령 실행결과를 문자열로 돌려주며 실행결과를 true false 출력함.
Exec	외부명령 실행결과를 문자열로 되돌려주며 실행결과를 출력하지 않음
Passthru	외부명령 실행결과를 문자열로 돌려주며 실행결과를 출력함. (실행결과가 바이너리 파일일 경우 사용)
Popen	외부명령 실행하고 실행결과를 읽음
Shell_exec	외부명령 실행결과를 문자열로 되돌려주며 실행결과를 출력하지 않음 (쉘을 통하여 실행)
Getmygid	프로세스ID를 얻을 때 사용
Array_filter	모든 값에 대해 필터링 하여 원하는 값을 찾을 때 사용
Proc_open	외부명령을 실행하고 실행된 프로세스를 핸들링하기 하기위해 사용

파일 업로드 취약점 대응방안 – ASP 웹쉘 주요 함수

- 외부 프로그램의 실행이나 시스템 명령어 사용을 위해 Shell.Application, Wscript.Shell 객체를 이용
- Shell.Application 객체는 ShellExecute메소드를 사용
- Wscript.Shell 메소드는 Run/Exec을 이용하여 외부 프로그램의 실행이나 시스템 명령어의 사용이 가능

함수명	내용
Run	파일을 동작 시킬 때 사용
Execute	질의어, SQL, 프로시저를 실행하여 결과를 알려줌
Eval	String형태의 소스코드를 동적으로 실행(참/거짓)

파일 업로드 취약점(공격) 조건

1. 파일 첨부 파일 기능에서 확장자 검증 미흡 (서버 사이드 스크립트 업로드 가능해야 함)
    a. 클라이언트 스크립트에서 차단을 하고 있다면 **버프스위트에 우회가 가능**
        i. png 파일 확장자로 준비했다가, POST 요청할 때 확장자를 수정해서 올린다.
    b. **블랙 리스트 방식**으로 서버 사이드 스크립트에 차단!!
        i. php 환경 - php, php3, php5, phtml... 등이 확장자로 사용이 가능
        ii. asp 환경 - asp, aspx, asa, asx, as%70 등이 확장자로 사용이 가능
        iii. 공통 - 대소문자 php, pHp, PhP.. 등 구분 (윈도우, 리눅스 운영체제에 따라)
2. 업로드된 웹쉘 절대경로 위치를 알아야 함

http://192.168.81.138/bWAPP/images/b374k-2.8.php?

1. 업로드된 디렉터리에 스크립트 실행이 가능해야 함

파일 업로드 취약점(공격) 조건

1. 파일 첨부 파일 기능에서 확장자 검증 (시큐어코딩)
    a. 화이트 리스트 방식으로 허용하고자 하는 확장자만 (JPG, PNG, PDF 파일만!!!!) 업로드 되게!!

```
/demoshop/shop_board/shop_download.asp?
strFileName=shell(4).asp&f_path=upload_file
http://211.250.83.36:8183/demoshop/shop_board/upload_file/shell(4).asp
http://211.250.83.36:8183/demoshop/upload_file/shell(4).asp
http://211.250.83.36:8183/upload_file/shell(4).asp
```

1. 업로드된 웹쉘 절대경로 위치를 알지 못하게!!
   a. 업로드되는 절대경로 노출이 되지 않도록 코딩해야 함
      i. shop_download.asp?strFileName=shell.cer&f_path=upload_file HTTP/1.1
      ii. → **(대응) shop_download.asp?strFileName=2&f_path=1 (데이터베이스에서 정보를 가져옴)**

1. 업로드된 디렉터리에 스크립트 실행하지 못하게!!
   a. 디렉터리 권한에 스크립트 실행 권한을 제외를 함!!!
      i. http://192.168.81.138/bWAPP/images 에 images 디렉터리에 스크립트 실행 권한 빼면...
   b. 업로드는 되는 서버를 분리해서 아예 스크립트 실행 원척적으로 차단!!

확장자 화이트리스트 검증 (서버 측)

- 설명: 허용된 파일 확장자만 업로드 가능하도록 서버에서 검증. 블랙리스트 방식은 우회 가능성이 높으므로 화이트리스트 사용 필수.
- 구현:

  php

  ```
  $allowed_extensions = ['jpg', 'jpeg', 'png', 'gif']; $ext = strtolower
  (pathinfo($_FILES['file']['name'], PATHINFO_EXTENSION)); if (!in_array
  ($ext, $allowed_extensions)) { die('Invalid file extension!'); }
  ```

- 주의: .php, .phtml, .php5 등 실행 가능한 확장자는 절대 허용하지 않음.

---

2. MIME 타입 검증

- 설명: 클라이언트가 제공하는 MIME 타입($_FILES['file']['type'])은 조작 가능하므로, 서버에서 파일의 실제 MIME 타입을 확인.
- 구현:

  php

```
$finfo = finfo_open(FILEINFO_MIME_TYPE); $mime = finfo_file($finfo, $_FILES['file']['tmp_name']); $allowed_mimes = ['image/jpeg', 'image/png', 'image/gif']; finfo_close($finfo); if (!in_array($mime, $allowed_mimes)) { die('Invalid MIME type!'); }
```

- 주의: PHP의 finfo 확장 모듈 설치 필요 (apt-get install php-fileinfo).

---

3. 파일 콘텐츠 검증

- 설명: 이미지 파일의 경우, 실제 이미지인지 확인하기 위해 이미지 처리 함수 사용. PHP 코드가 삽입된 파일 차단.
- 구현:

  php

```
$image = getimagesize($_FILES['file']['tmp_name']); if ($image === false) { die('Not a valid image file!'); }
```

- 주의: 이미지 외의 파일(예: PDF)에는 별도 검증 로직 필요.

---

4. 랜덤 파일명 생성

- 설명: 원본 파일명을 사용하면 경로 예측이 쉬워 악용 가능. 고유한 랜덤 파일명으로 저장.
- 구현:

  php

```
$random_filename = uniqid() . '_' . bin2hex(random_bytes(8)) . '.' . $ext; $filepath = 'uploads/' . $random_filename;
```

- 주의: 데이터베이스에 원본 파일명과 매핑 정보 저장 시 SQL Injection 방지.

---

5. 업로드 디렉토리 보안

- 설명: 업로드된 파일이 저장되는 디렉토리는 PHP 실행을 차단하고, 외부 접근을 제한.
- 구현:
  1. 디렉토리 권한 설정:

     bash

     ```
     chmod 755 uploads chown www-data:www-data uploads
     ```

  2. Apache/Nginx 설정으로 PHP 실행 차단:
     - Apache .htaccess:

       apache

       ```
       <FilesMatch "\.(php|php3|php4|php5|phtml|phps)$"> Deny from all
       </FilesMatch>
       ```

     - Nginx:

       nginx

       ```
       location ~* \.(php|php3|php4|php5|phtml|phps)$ { deny all; }
       ```

  3. 디렉토리 인덱싱 비활성화:

     apache

     ```
     Options -Indexes
     ```

- 주의: 업로드 디렉토리를 웹 루트 외부(예: /var/uploads)에 저장하면 더 안전.

---

6. 파일 크기 제한

- 설명: 큰 파일 업로드로 인한 DoS 공격 방지. PHP 설정과 코드에서 크기 제한.

- 구현:
  1. PHP 설정 (php.ini):

     ini

     ```
     upload_max_filesize = 2M post_max_size = 2M
     ```

  2. 코드 레벨:

     php

     ```
     $max_size = 2 * 1024 * 1024;// 2MB if ($_FILES['file']['size'] > $max_size) { die('File size exceeds limit!'); }
     ```

- 주의: php.ini 설정이 우선 적용되므로, 코드와 일치하도록 조정.

파일 업로드 취약점(공격)은 왜 하냐??!!!!

1. 웹쉘을 올려서 웹 서비스(1차 침투)→ DB 연결정보→ DB의 중요한 데이터베이스의 정보를 가져오기 위함!!!
2. 웹서버 침투 (쉘권한 획득) → 근접 내부 시스템을 2차적으로 침투하기 위한 목적!!!
3. (범죄자입장) index.html 페이지 변조 → <iframe>, <script> 삽입해서 악성코드 배포하기 위한 목적