파일 업로드 취약점(File Upload Vulnerability)

i-Keeper CERT 오나희

목차

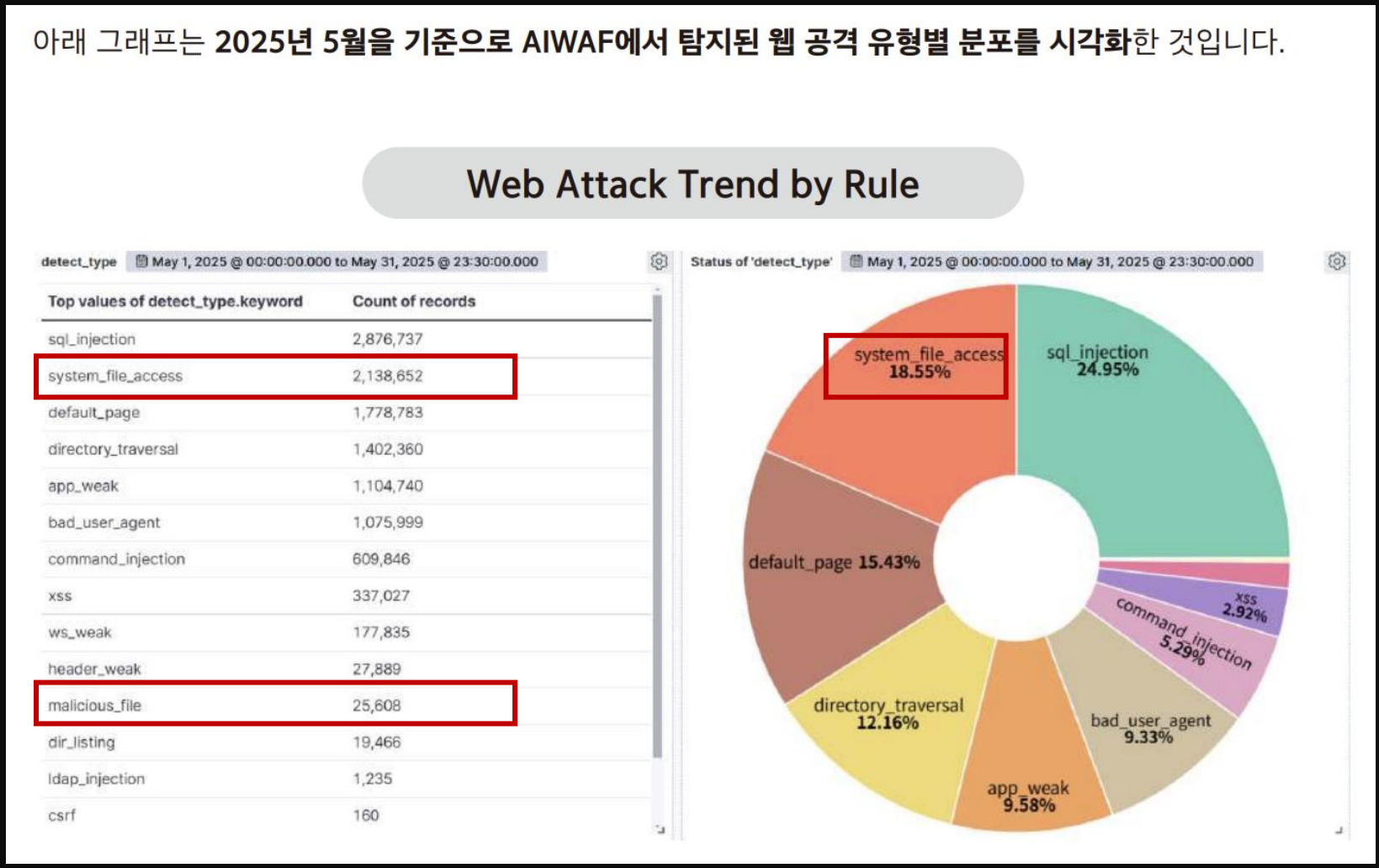
1. 서론
2. 파일 업로드 기능?
3. 파일 업로드 취약점 핵심 원인
4. 공격자가 우회에 사용하는 기법
5. 공격 결과
6. 대응 방법

1. 서론

웹쉘 실행 후 시스템 파일에 접근하려는 시도

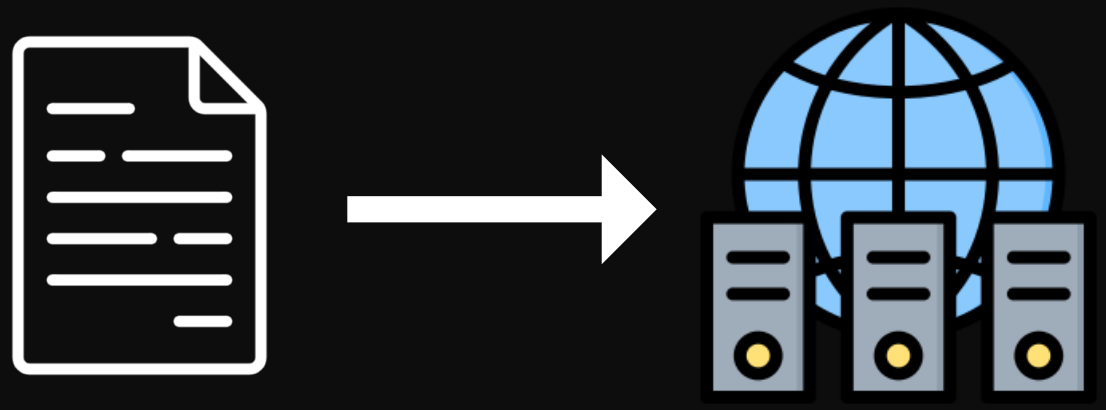
서버에 악성 php, jsp, asp 같은 파일 업로드 시도

아래 그래프는 **2025년 5월을 기준으로 AIWAF에서 탐지된 웹 공격 유형별 분포를 시각화**한 것입니다.



[모니터랩 2025년 5월 웹 공격 동향 보고서]

2. 파일 업로드 기능 ?



사용자가 자신의 디바이스에서 파일을 선택해 웹 서버로 전송하고 저장할 수 있게 해주는 기능

= 웹이나 앱에서 사용자의 파일을 서버로 전송하는 모든 과정

2. 파일 업로드 기능 ?

카카오톡으로 문서, 사진 같은 것들을 주고 받는 것도 파일 업로드 기능이라고 볼 수 있다

사용자 디바이스 → 서버로 파일 전송 → 서버가 저장 → 다른 사용자가 접근 가능하게 처리

사용자가 서버에 직접 파일을 저장할 수 있다는 구조적 특성 때문에 보안적으로는 큰 공격표면을 제공하게 된다

3. 파일 업로드 취약점의 핵심 원인

1. 파일 확장자/MIME/content 검증 미흡
2. 퍼미션 미비
3. 서버측 검증 누락 (클라이언트측 검증만 의존)

3.1. 파일 확장자/MIME/content 검증 미흡

- 서버가 확장자 이름(.jpg, .png)만 확인하고 파일을 허용하는 경우
공격자는 .php.jpg 같은 이중 확장자를 이용해서 우회가 가능하다

- MIME 타입 검증 미흡 (파일의 유형을 나타내는 값)
공격자는 php 파일을 Content-Type을 image/jpeg로 위조할 수 있다 (HTTP 요청)

3.2. 퍼미션 미비

웹 서버(Apache, Nginx 등)는 특정 경로(web root) 하위의 파일에 HTTP 접근을 허용하고
그 중 일부 파일은 코드로 실행(PHP, JSP 등)되도록 설정되어 있다

3.2. 퍼미션 미비

웹서버는 .html, .jpg, .txt 같은 파일은 정적인 파일로 간주해서
내용을 그대로 사용자에게 전달하지만

.php, .jsp, .asp 같은 파일은 동적인 코드로 간주해서
서버가 먼저 실행한 뒤 그 실행 결과만 사용자에게 보여준다 = 웹쉘의 핵심 원리

3.2. 퍼미션 미비

```
/var/www/html/
   ├──── index.html
   ├──── login.php
   └──── uploads/
         └──── shell.php   ← 공격자가 올린 웹쉘
```

파일을 올릴 수는 있다고 해도 실행, 수정 권한까지 있다면…

3.3. 서버 측 검증 누락 (클라이언트측 검증만 의존)

웹사이트에서 파일 업로드 시 보통 브라우저에서 먼저 검사한다
(예 : "이미지만 올릴 수 있어요, "2MB 이하만 가능합니다" 같은 메시지)

하지만 서버측 검증 누락은 html (사용자에게 창 띄우기), js(파일 크기 검사)
즉, 클라이언트 측 검증에만 의존하고 서버에서는 검증을 하지 않는 것

4. 공격자가 우회에 사용하는 기법

1. 이중 확장자 우회
shell.php.jpg 같은 방식으로 서버가 단순히 .jpg, txt 같은 확장자만 검사하면 통과

2. MIME (Multipurpose Internet Mail Extensions) 타입 위조
파일 전송 시 HTTP 헤더의 Content-Type을 조작

3. NULL 바이트 우회 (null byte injection)
shell.php%00.jpg 같은 방식으로 %00은 문자열 종료를 의미 하고, 이를 서버는 .php 로 잘라서 해석

4. 파일 내부 콘텐츠 위장 (이미지 내부에 코드 삽입)
JPEG 파일의 EXIF 메타데이터 영역이나 하단에 PHP 코드 삽입

5. 공격 결과

웹쉘 업로드 후 원격 명령 실행 가능 (ls, whoami..)

*웹쉘이 업로드 된 후 실행되려면 웹서버가 그 파일이 위치한 디렉터리에서 코드 실행을 허용하고 있어야 한다

*웹쉘이 업로드 된 디렉터리도 알고 있어야 한다
(디렉토리 인덱싱 / 업로드 성공 후 응답 메시지 보기 / 추측)

6. 대응 방법

1. 허용된 확장자 화이트 리스트로 제한
2. 파일 MIME 타입과 내용 직접 검사
3. 업로드 파일명 무작위화 / 디렉터리 구조 숨김
4. 파일이 업로드 되는 디렉터리를/var/www/html/ (Apache 기준) 밑으로 두지 않게 빼기
5. 업로드 디렉터리에 실행 권한 제거
6. 파일 크기 제한
7. 악성 파일 탐지 도구 연동

등등…

참고문헌

https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html