2022년도

지금도 놓치고 있는 웹 취약점#2

본 문서는 ㈜파이오링크 침해대응센터에서 수행한 웹 침투 테스트의 실제 사례를 웹 공격 시나리오로 구성하여 악의적인 공격에 쉽게 활용되고 있는 위험성을 공유하고 각 고객사의 환경에 맞는 대응 방안을 마련하기 위해 작성되었습니다.

PIOLINK

SQL 인젝션 → 소스코드 탈취 및 분석 → 취약점 확인 → 파일 업로드 및 시스템 장악

여전히 많은 웹 사이트에서 SQL Injection, XSS 등의 공격과 관련된 특수문자 등에 대한 필터링이 미흡하다. SQL Injection을 이용하여 DB조작 뿐만 아니라 특정 함수 등을 사용할 경우 File Upload, File Download, Command Injection 등이 가능하며 시스템 장악에 이를 수 있다.

① SQL Injection 공격에 취약함을 확인하고 load_file()함수를 통해 서버 자원을 탈취할 수 있다.

② 서버의 자원 경로는 robot.txt, view-source, Directory Scan 등을 통해 서버의 자원 정보(1차 정보)를 획득할 수 있었으며, 이렇게 확보한 정보를 토대로 내부 자원(2차 정보)를 획득 할 수 있다.

③ 다운받은 내부 자원에 대한 코드 검사를 진행하여 include 정보 등 추가적인 내용(3차 정보)를 획득 할 수 있으며, 이를 통해 내부 자원(웹 개발 소스코드)를 전부 추출할 수 있다.

④ 이렇게 추출한 내부 자원에 대한 소스코드 진단 등을 통해 File Upload, SSRF, Session Injection 등의 취약점을 확인하여 시스템 장악에 활용할 수 있다.

⑤ File Upload가 가능한 취약 지점에 접근 시 관리자 권한이 필요하여 Session Injection 을 통해 관리자 권한을 획득하고 해당 취약지점에서 WebShell 업로드, 시스템 제어(시스템 명령 실행) 단계에 이를 수 있다.

① SQL Injection

[공격]

외부에서 SQL Injection 취약점이 존재하는 게시판에 대하여 공격을 진행, WAF 등에 의해 차단되었으나 다음과 같은 보안솔루션을 우회하는 공격패턴을 이용하여 공격할 수 있다.

- 게시판?취약파라미터=(0)/*!or(1=0)union*/select+1,2,3,4,(select+group_concat(i_id)+from+intranet_member+where+1),6,7,8 … , 58,59%23&f_name=&x=17&y=18
- 게시판?취약파라미터 =/*PIO_*/(0)/*!or(1=0)union*/select+1,2,3,4,(select load_file({file_name})),6,7,8,9,10,… , 59#

[탐지]

alert tcp any any -> any any (msg:"SQL Injection bypass Prevention 1"; content:"GET |POST "; content:"₩/₩*!"; content:"₩*₩/"; content: "select|union|insert"; flow:to_server)

alert tcp any any -> any any (msg:"SQL Injection to filedownload"; content:"GET |POST "; content:"%27|₩'|%22|₩""; content“select"; content:“load_file("; flow:to_server)

[위협 요소 및 대응]

SQL Injection에 대하여 방어를 하고 있음에도 우회 공격에 대한 대응이 미비하여 공격이 성공한 케이스로 해당 공격이 성공함에 따라 서버 자원 및 계정정보를 비롯한 차후 공격에도 활용되어질 정보들이 유출되었다. 이러한 공격에 대한 대응책이 필요하고 우회 공격 및 최신 취약점 등에 대한 지속적인 관리가 필요하다. 또한 File Download, RCE등과의 연계가능성이 높기 때문에 더욱 주의해야한다. 또한 File Upload 후에 업로드 된 파일에 대한 경로를 찾을 수 없을 때 SQL Injection을 활용하였던 바, 반드시 취약요소를 없애기 위해 노력해야 한다.

SQL 인젝션 → 소스코드 탈취 및 분석 → 취약점 확인 → 파일 업로드 및 시스템 장악

② 소스코드 탈취 및 분석

[공격]

다운받은 소스코드 분석을 통해 File Download, SQL Injection 등의 취약점을 확인할 수 있으며, 이러한 취약점을 악용하여 추가적인 서버의 자원 획득 및 시스템 장악이 가능하다. 이러한 방법은 외부에서 침투 지점을 찾아내는 것보다 훨씬 정교하고 치명적인 공격이 가능하도록 할 수 있다.

[위협 요소 및 대응]

시큐어코딩 및 소스코드 진단 등을 통해서 취약점이 발생하지 않도록 미리 관리해야 하며, 불필요한 파일이나 경로에 대한 노출을 최소화해야 한다. 특히 오픈소스 사용이 늘어가고 있는 상황에서 해당 오픈소스의 버전 정보, 디렉토리 구조, 페이지 이름 등을 토대로 어떤 오픈소스가 사용되었는지를 공격자는 알 수 있기 때문에 오픈소스 사용 시에 이러한 점을 주의하여 커스터 마이징하거나 취약점에 관련된 정보를 주기적으로 확인하여 조치해야 한다.

③ 세션 주입

[공격]

앞서 소스코드 탈취 및 분석을 통해 취약한 함수를 사용하고 있음을 확인하였고 해당 취약점을 관리자의 세션을 주입하여 을 획득 할 수 있다. 관리페이지는 관리 목적의 많은 기능들을 포함하고 있어 추가 공격에 활용 될 수 있다.

[위협 요소 및 대응]

취약한 함수의 사용을 지양하고 부득이하게 취약한 함수를 사용하게 될 경우 원하는 기능만 수행될 수 있도록 필터링 해야 한다.

SQL 인젝션 → 소스코드 탈취 및 분석 → 취약점 확인 → 파일 업로드 및 시스템 장악

④ 파일 업로드

[공격]

관리자 페이지에 파일을 업로드 하는 부분이 존재했고 해당 부분에 대한 소스코드를 확보, 분석하여 필터링을 쉽게 우회하여 공격할 수 있다.

[위협 요소 및 대응]

업로드 가능한 확장자와 관련하여 **Black list**기반으로 필터링하고 있는 경우, 필터링되지 않은 확장자로 업로드 할 수 있다. **Blacklist**보다는 **Whitelist** 기반으로 필터링해야 한다. 또한 공격을 탐지하기 위해 룰을 생성함에 있어 최초 공격을 탐지하기 위해 노력해야하고 뿐만 아니라 **WebShell**을 통한 시스템 명령 실행, 그에 대한 **Response**를 탐지하여 누락이 없도록 해야한다.

[탐지]

alert tcp any any -> any any (msg:"WebShell Detect 1"; content:"eval(gzinflate(base64_";)

alert tcp any any -> any any (msg:“WebShell Detect 2"; content:"GET |POST "; content:“eval(|passthru(|exec(|system(|shellexec("; nocase;)

alert tcp any any -> any any (msg:“WebShell Detect 3"; content:"GET |POST "; content:“subprocess.run|os.run|start-process"; nocase;)

alert tcp any any -> any any (msg:“Command Execute Detect 1”; content:“/bin/bash”; nocase; pcre:“/([0-9]{1,3}₩.){3}[0-9]{1,3}/”; pcre:“/[0-65535]/”)

alert tcp $INTERNAL_NET any -> $EXTERNAL_NET any (msg:“List Command Execute Detect 2”; pcre:“/([ldrwxt-]{10})₩s+(₩d)+ ([A-z-_0-9]+)₩s([A-z-_0-9]+)₩s+(₩d)+₩s+(₩d{4}-₩d{2}-₩d{2})₩s(₩d{2}:₩d{2})₩s(₩w+)/”; nocase; flow:to_client, established;)

㈜ 파이오링크 I 02-2025-9665 I cert@piolink.com