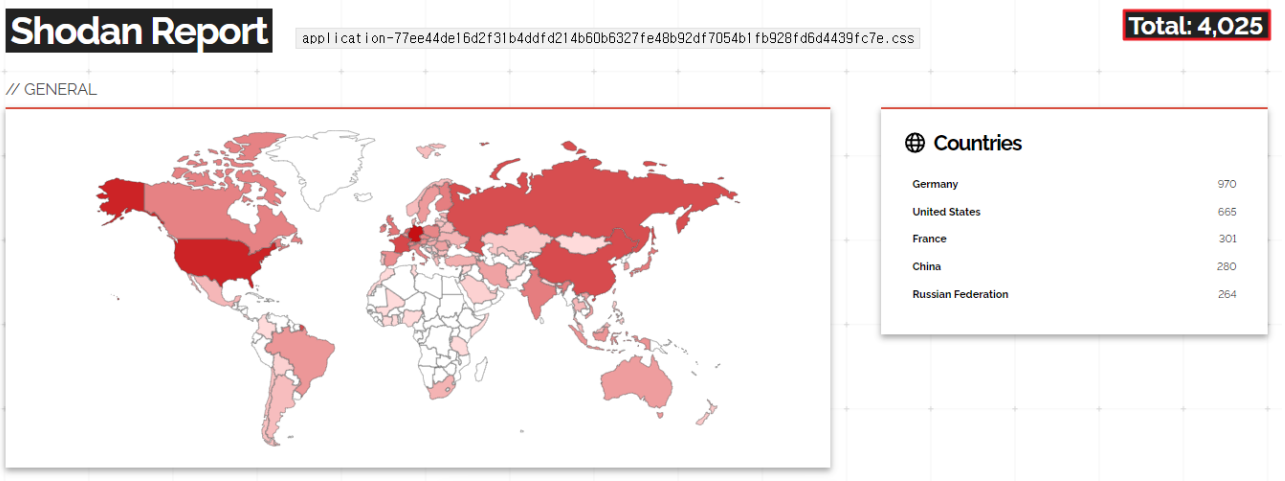
Research & Technique

GitLab 임의 파일 읽기 취약점 (CVE-2023-2825)

■ 취약점 개요

2023 년 5 월, 개인 또는 조직이 소프트웨어 개발 및 협업을 위해 사용하는 Git 리포지토리 관리 솔루션 GitLab 에서 임의 파일 읽기 취약점이 발견됐다. 해당 취약점은 경로 탐색 취약점을 활용해 서버의 임의의 파일을 읽거나 다운받을 수 있기 때문에 GitLab 에서는 CVSS[1] 기준 10.0 점으로 평가했다. 특히, 인증되지 않은 공격자가 공개 프로젝트의 첨부파일 다운로드 경로를 조작해 잠재적으로 서버 주요 데이터 파일인 구성 세부 정보, 기업의 소스 코드, 민감한 사용자 데이터 등에 접근할 수 있어 주의가 필요하다.

인터넷상에서 공개된 취약한 GitLab 은 Shodan 등의 OSINT 검색 엔진을 통해 확인할 수 있다. 지난 6 월 28 일 Shodan 을 이용해 취약한 서버를 검색한 결과, 약 4,000 개의 취약한 GitLab 이 존재하는 것으로 나타났다. 따라서 취약한 버전을 사용한다면 각별한 주의가 필요하다.



*출처: Shodan Report

그림 1. 취약한 서버 검색 결과

[1] CVSS(Common Vulnerability Scoring System)란 컴퓨터 시스템 보안 취약점의 심각도를 평가하기 위한 무료 공개 산업 표준이다.

■ 영향받는 소프트웨어 버전

CVE-2023-2825 에 취약한 GitLab 의 버전은 다음과 같다.

S/W 구분	취약 버전
GitLab CE(Community Edition)/EE(Enterprise Edition)	16.0.0

※ 취약점이 동작하기 위해서는 최소 5 개의 그룹이 존재해야 한다는 조건이 존재한다. 조건은 아래의 취약점 상세 분석을 통해 확인할 수 있다.

■ 공격 시나리오

CVE-2023-2825 취약점을 이용한 공격 시나리오는 다음과 같다.

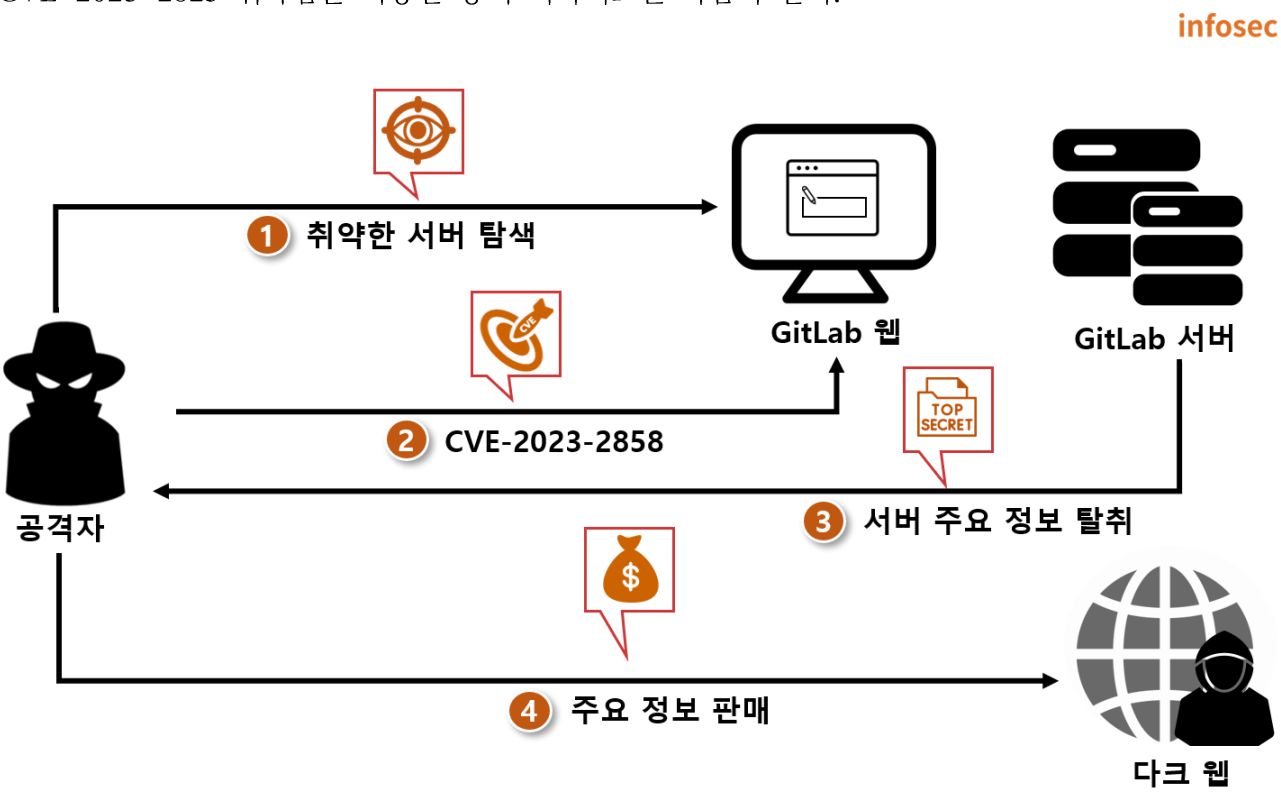


그림 2. 공격 시나리오

① 공격자는 OSINT 검색 엔진을 통해 취약한 GitLab 웹 서버를 탐색
② 공격자는 CVE-2023-2825 취약점을 이용해 피해자 서버에 접근
③ 공격자의 요청을 받은 서버는 주요 정보(개발 소스 코드, 서버 환경 구성 정보 등)를 공격자에게 반환
④ 공격자는 획득한 주요 정보를 다크 웹이나 다른 경쟁사에게 판매

■ 테스트 환경 구성 정보

테스트 환경을 구축하여 CVE-2023-2825 의 동작 과정을 살펴본다.

이름	정보
피해자	Ubuntu 20.04.5 LTS (192.168.100.162) GitLab 16.0.0
공격자	Kali Linux 6.1.0-kali5-amd64 (192.168.100.152)

■ 취약점 테스트

Step 1. 환경 구성

1) 피해자 PC 에 docker hub 에서 지원하는 GitLab CE 이미지 중, 취약점이 존재하는 GitLab 16.0.0 버전의 서버를 구축한다.

명령어	$ docker run -d -p 80:80 gitlab/gitlab-ce:16.0.0-ce.0 -d 옵션: detach 모드로 백그라운드로 docker 를 실행시키는 옵션 -p 옵션: local 포트와 docker 에서 실행할 포트를 지정하는 옵션

```
root@ubuntu:/home/eqst# docker run -d -p 80:80 gitlab/gitlab-ce:16.0.0-ce.0
Unable to find image 'gitlab/gitlab-ce:16.0.0-ce.0' locally
16.0.0-ce.0: Pulling from gitlab/gitlab-ce
1bc677758ad7: Pull complete
633fcf47bc79: Pull complete
472c1ac0c258: Pull complete
5b665b492973: Pull complete
0bd8b5a23fe7: Pull complete
b385dd2cb2ca: Pull complete
38ac4d68d24c: Pull complete
e4588a97b783: Pull complete
Digest: sha256:ab90cdb096c4f81247088357b0e051f5b8a999284b2186cbd1b1ec1a41cca7e8
Status: Downloaded newer image for gitlab/gitlab-ce:16.0.0-ce.0
3e524103ef6858b7825c530db4ce0d2dd3c1eb5f1e36776ef413574655d61784
```

그림 3. docker 를 통한 환경 구축

2) GitLab 루트 계정의 패스워드를 재설정하기 위해 컨테이너의 터미널을 열고 아래의 명령어를 실행한다.

명령어	컨테이너 접근 명령어 : $ docker exec -it [container 명 또는 container ID] /bin/bash 패스워드 변경 명령어 : # gitlab-rake "gitlab:password:reset[root]"

```
root@ubuntu:/home/eqst# docker ps
CONTAINER ID   IMAGE                          COMMAND             CREATED
  STATUS                    PORTS
NAMES
3e524103ef68   gitlab/gitlab-ce:16.0.0-ce.0   "/assets/wrapper"   4 minutes ago
  Up 4 minutes (healthy)   22/tcp, 443/tcp, 0.0.0.0:80->80/tcp, :::80->80/tcp
distracted_heyrovsky
root@ubuntu:/home/eqst# docker exec -it 3e524103ef68 /bin/bash
root@3e524103ef68:/# gitlab-rake "gitlab:password:reset[root]"
Enter password:
Confirm password:
Password successfully updated for user with username root.
```

컨테이너 접근 명령어

패스워드 변경 명령어

그림 4. GitLab 루트 계정 패스워드 재설정

3) 취약점 테스트를 위해 PoC 가 저장된 git 파일을 공격자의 PC 로 복사한다.

PoC 가 저장된 GitHub URL 은 다음과 같다.

- URL: https://github.com/Occamsec/CVE-2023-2825.git

명령어	$ git clone https://github.com/Occamsec/CVE-2023-2825.git

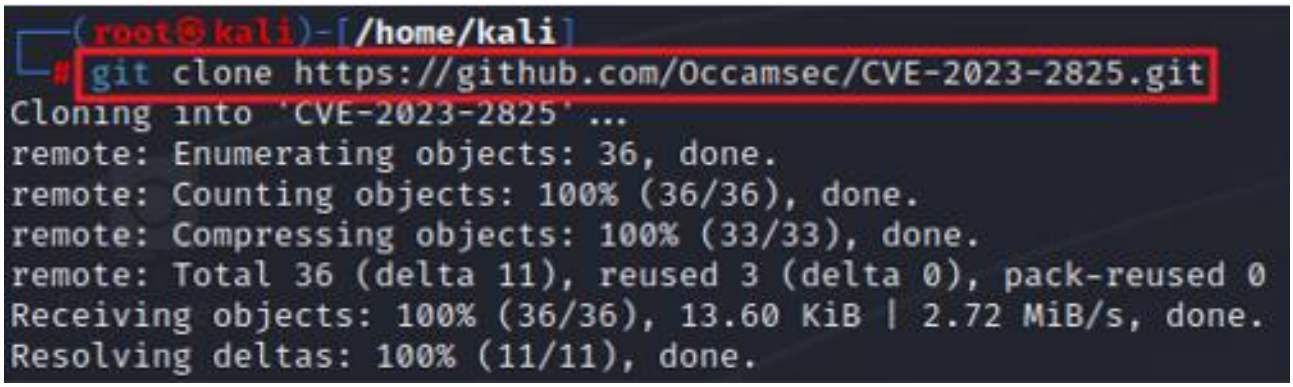


그림 5. PoC 복사 및 경로 확인

4) 편집기를 활용해 PoC 파일에 피해자 서버의 정보를 입력한다.

※ root 계정을 넣은 이유는 PoC 테스트를 위한 프로젝트 생성을 위함으로, 실제 취약점에서는 인증 정보가 없는 사용자가 공개된 프로젝트를 대상으로 공격할 수 있다.

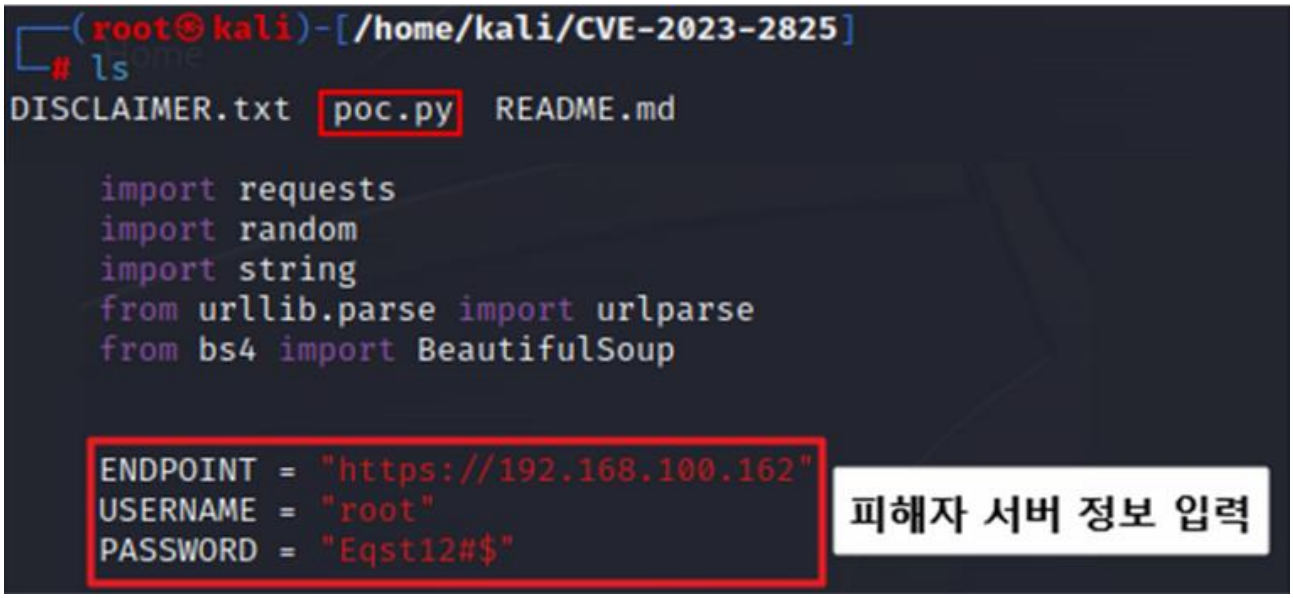


그림 6. 피해자 서버 정보 입력

5) PoC가 동작해 피해자 서버의 /etc/passwd 파일을 확인할 수 있다.

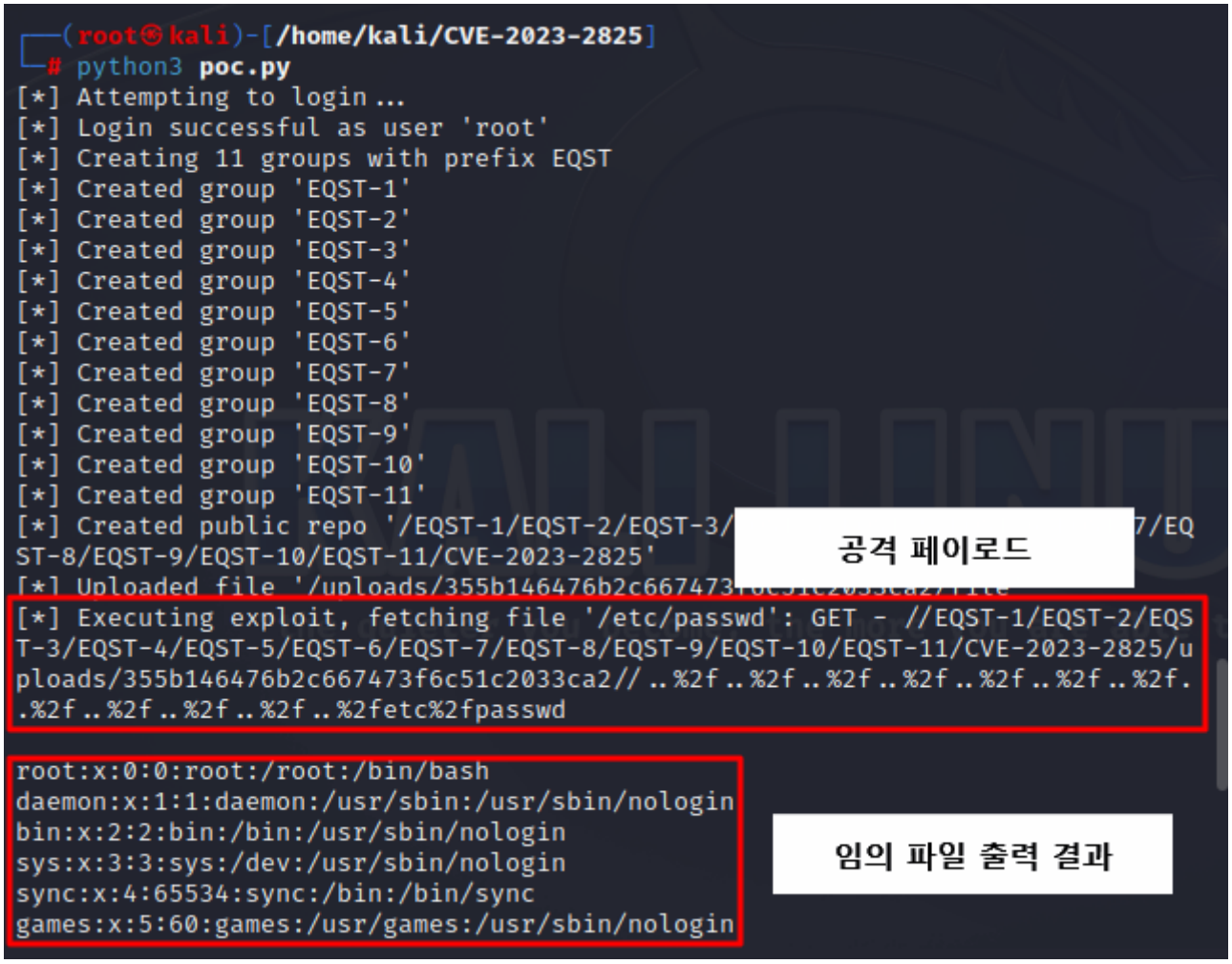


그림 7. 취약점으로 인한 임의 파일 노출

■ 취약점 상세 분석

Step 1) 취약점 개요

CVE-2023-2825 취약점은 인증되지 않은 공격자가 공개된 취약한 버전의 GitLab 프로젝트 또는 Snippet[2] 등 첨부파일에 접근할 경우 취약점이 동작한다. GitLab 서버는 요청받은 URL 경로에 파일이 존재하지 않을 경우, puma[3]로 전달 이후 디코딩하여 처리한다. 이후, GitLab 서버는 전달받은 URL 에서 파일명을 받아온다. 하지만, 파일명을 검사하는 로직이 누락되어 있어 취약점이 발생한다.

```
scope path: :uploads do
  # Note attachments and User/Group/Project/Topic avatars
  get "-/system/:model/:mounted_as/:id/:filename",
    to: "uploads#show",
    constraints: { model: %r{note|user|group|project|projects\/topic|achievements\/achievement},
    mounted_as: /avatar|attachment/, filename: %r{[^/]+} }
```

그림 8. puma 를 통한 디코딩하는 소스

공격자가 패킷을 조작해 첨부파일 명에 Path Traversal 구문을 인코딩하여 전달하면, 서버에서는 디코딩하여 처리된 문자열에서 파일명을 해석하기 때문에 취약점이 발생한다. 따라서 인코딩 문자열 "..%2f", "%2e%2e%2f"를 이용해 상위 디렉터리에 존재하는 파일에 접근할 수 있다.

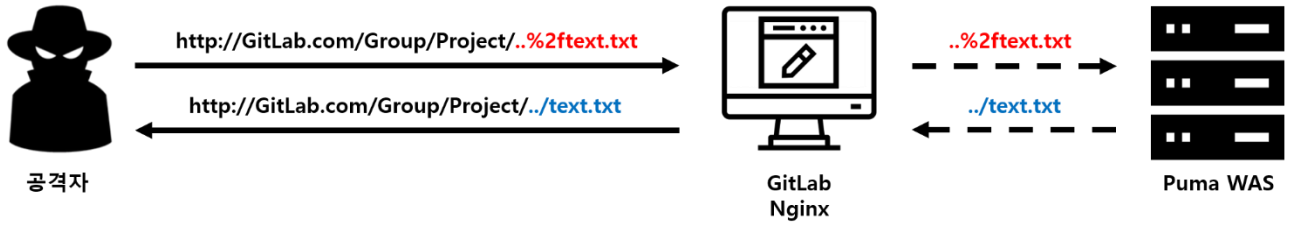


그림 9. 디코딩 해석 그림

[2] Snippet 이란 자주 사용되는 코드나 다른 사용자와 공유하기 위한 코드, 텍스트 등을 저장하기 위한 페이지다.

[3] puma 란 WAS(Web Application Server)의 한 종류로 Ruby 응용 프로그램을 위한 서버다. GitLab 의 Rails(Ruby 의 Web framework 의 일종) 응용 프로그램을 실행하기 위해 사용된다.

취약점을 이해하기 위해서는 GitLab 의 첨부파일 관련 저장 구조를 알아야 한다. 그룹의 프로젝트를 생성하거나, Snippet 을 생성할 때는 업로드 경로와 특정 디렉터리 경로가 결합되어 저장된다. 아래의 예시는 정상적으로 파일을 업로드 했을 때, 서버에 저장되는 경로이다.

그림 10. 저장되는 경로

하지만, 다운로드 요청 URL 경로에 존재하지 않은 첨부파일 다운로드 요청 시, GitLab 은 puma 서버를 통해 아래의 심볼릭 링크[4]에서 파일을 전송한다.

그림 11. 심볼릭 링크 디렉터리 경로

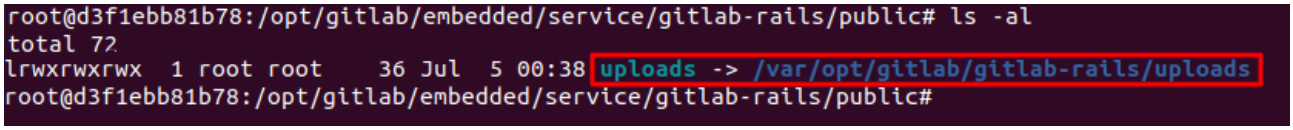


그림 12. 서버의 심볼릭 링크 경로

GitLab 의 첨부파일 업로드 시 생성되는 다운로드 요청 URL 규칙은 다음과 같다. 프로젝트에서 중첩된 하위 그룹의 생성 수만큼 디렉터리가 추가적으로 생성된다.

GitLab 주소/[Group]/[하위 그룹]/Project/uploads/[hash]/file

GitLab 주소/[Group]/[하위 그룹1][하위 그룹 2]...[하위 그룹 N]/Project/uploads/[hash]/file

그림 13. 다운로드 경로 요청 URL 구성

[4] 심볼릭 링크란 특정 파일이나 디렉터리에 대한 참조를 포함하는 파일로, Windows 운영체제의 바로 가기와 같은 기능을 한다.

하위 그룹이 하나만 존재하는 다운로드 요청 URL 일 경우는 아래의 그림과 같이 WebRoot 디렉터리인 5 개의 경로만 이동이 가능하며, 심볼릭 링크의 uploads 디렉터리까지만 이동이 가능하다.

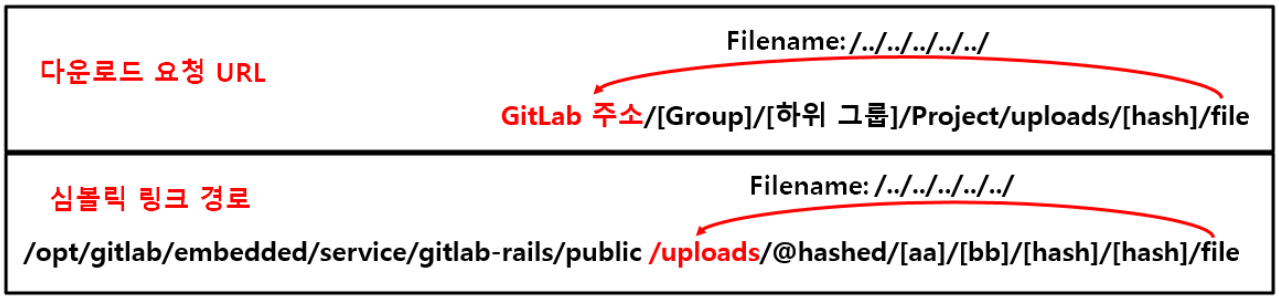


그림 14. 웹 루트 디렉터리로 이동

하지만, 다운로드 요청 URL 은 중첩된 하위 그룹의 생성 수만큼 디렉터리의 개수가 생성되고, 심볼릭 링크의 디렉터리 개수는 고정이므로, WebRoot 디렉터리 보다 상위 디렉터리까지 접근이 가능하다.

그림 15. 상위 디렉터리로 이동

서버가 첨부파일 다운로드 요청을 전달받을 때, 참조하는 심볼릭 링크의 예시를 도식화한 그림은 아래와 같다.

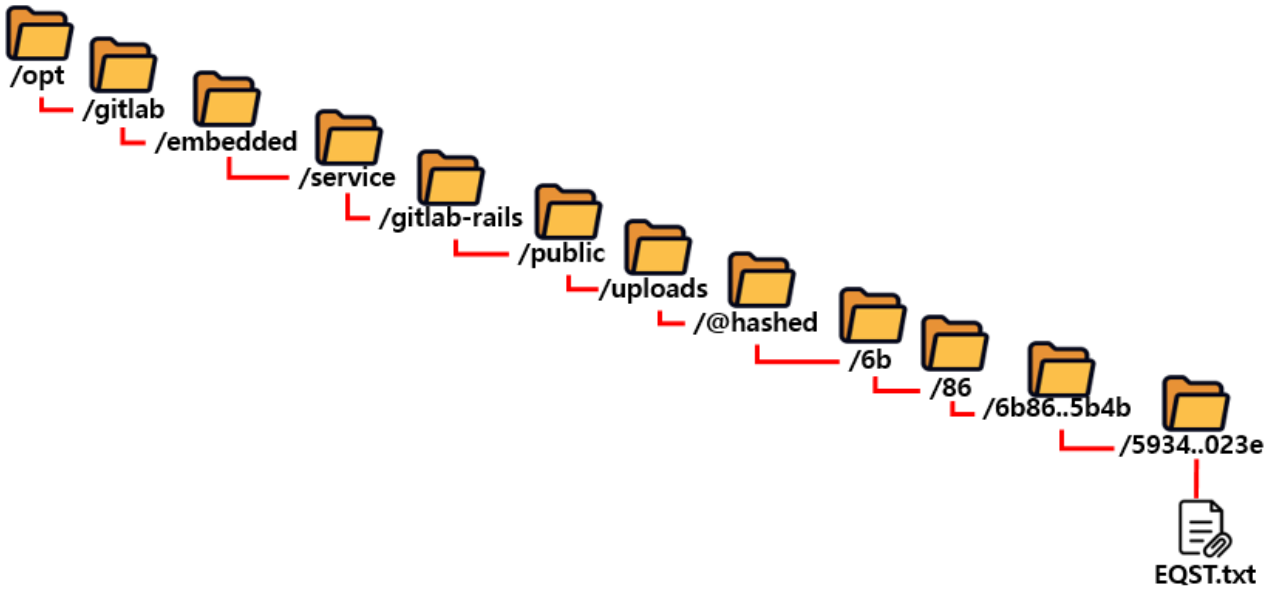


그림 16. 심볼릭 링크의 예시 도식화

Step 2) 동작 상세 분석

취약점 상세 분석을 위해 취약한 버전의 GitLab 에 그룹(EQSTLab)을 생성하고 공개 프로젝트(Insight)를 생성한다.

※ 취약점을 분석하기 위해서 현재 경로를 출력하는 print.txt 파일을 각각의 디렉터리에 생성했다.

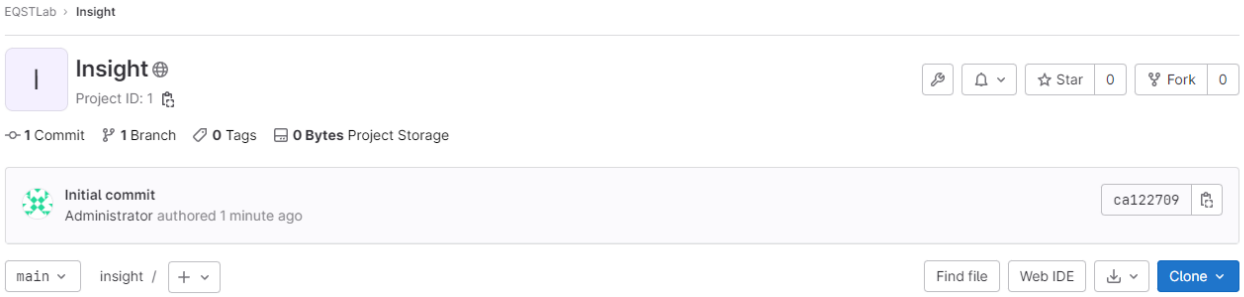


그림 17. 생성 화면

프로젝트 생성 이후, 첨부파일을 악용하기 위해서 해당 프로젝트에 관련 내용을 적을 수 있는 공간인 issue 를 생성하여 첨부파일(EQST.txt)을 업로드한다.

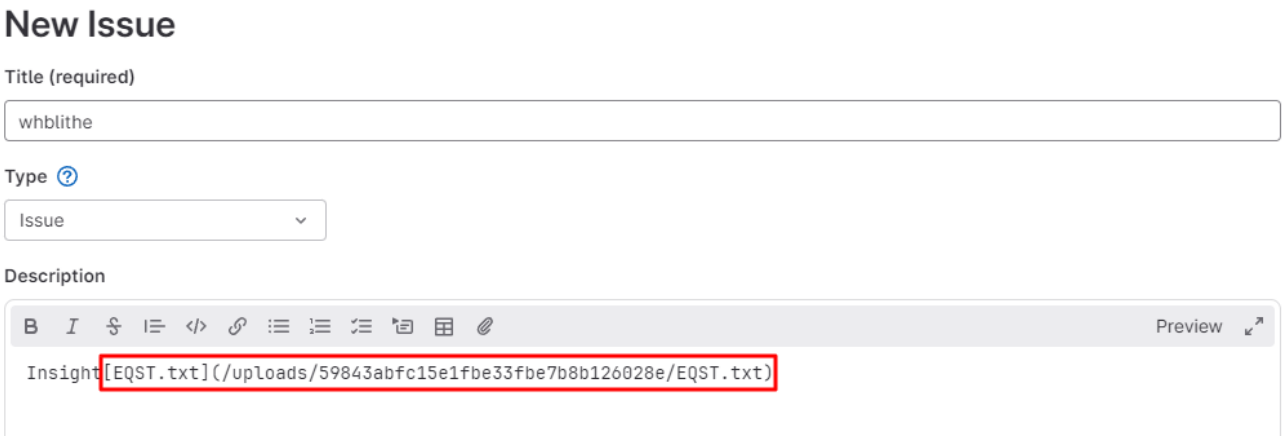


그림 18. 첨부파일 등록

아래의 경로를 통해 첨부파일(EQST.txt)를 다운로드 받는다.

- http://192.168.100.162/eqstlab/insight/uploads/59843abfc15e1fbe33fbe7b8b126028e/EQST.txt

그림 19. 파일 다운로드

CVE-2023-2825 취약점을 악용하기 위해서는 프록시 툴을 이용해, 파일명을 변조하여"../"문자열을 URL 인코딩한 후 "..%2fprint.txt" 또는 "%2e%2e%2fprint.txt" 페이로드를 피해자 서버에 전달하면 상위 경로에 도달할 수 있다.

프록시 툴을 이용해 상위 경로의 현재 경로를 출력하는 print.txt 의 응답 값은 다음과 같다.

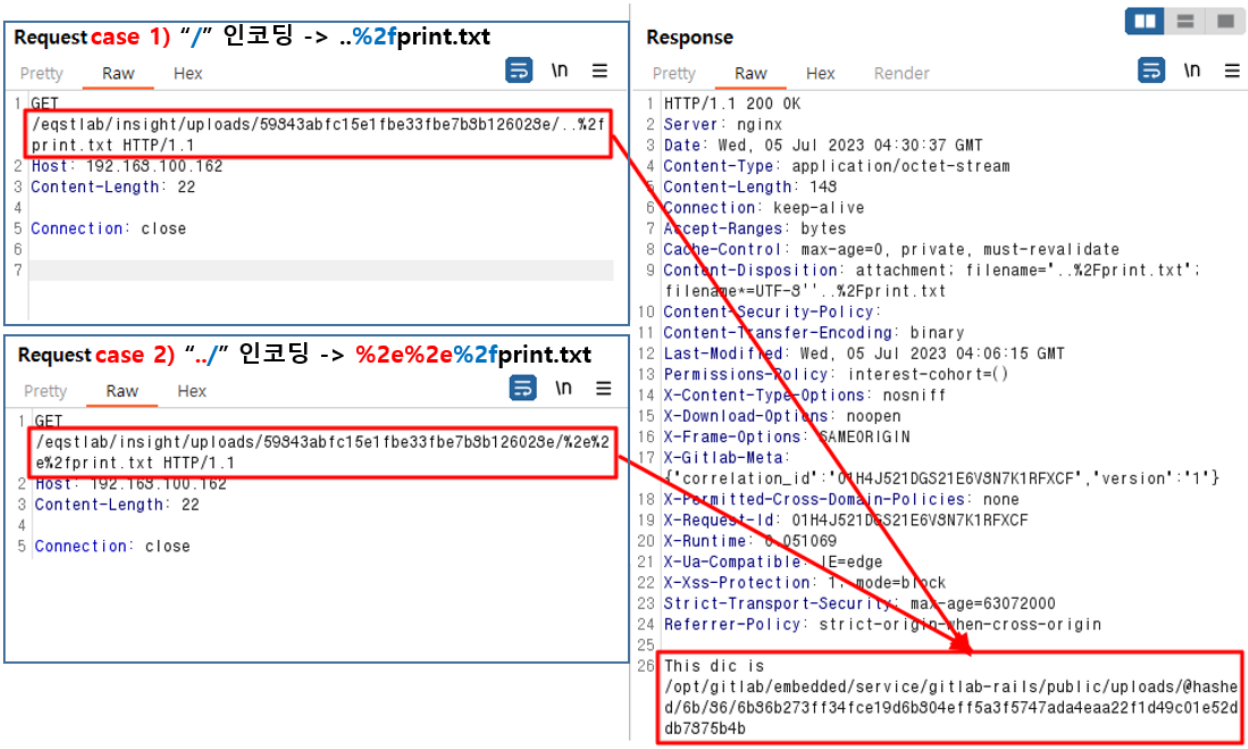


그림 20. 상위 경로로 이동 후 파일 정보 출력

한 단계씩 상위 경로로 이동을 반복하다, 상위 5 개의 디렉토리 이동할 때 400 Bad Request 에러를 반환한다. 이는 하위 그룹을 만들지 않은 프로젝트 구성이기 때문에, 다운로드 요청 URL 에는 /eqstlab/insight/uploads/59843abfc15e1fbe33fbe7b8b126028e/ 4 개의 디렉터리만 포함되어 있다. 따라서, WebRoot 디렉터리 보다 상위 디렉터리인 5 개 디렉터리 이동은 불가능하다.

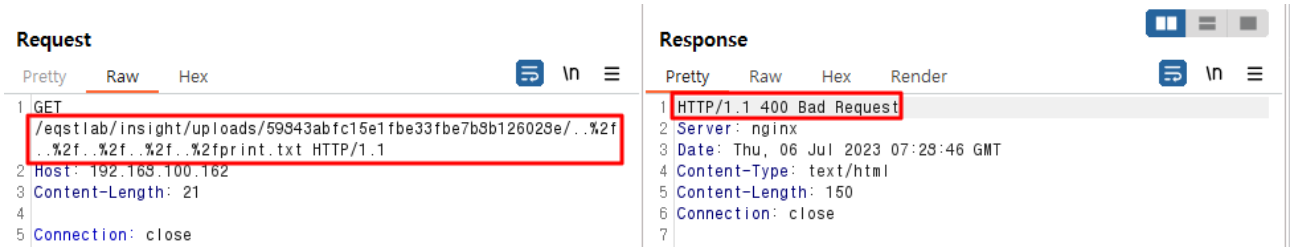


그림 21. 에러 반환

아래 그림은 WebRoot 디렉터리 보다 상위 디렉터리인 5 개의 경로를 이동 시, 400 Bad Request 에러를 반환하는 과정을 도식화한 그림이다.

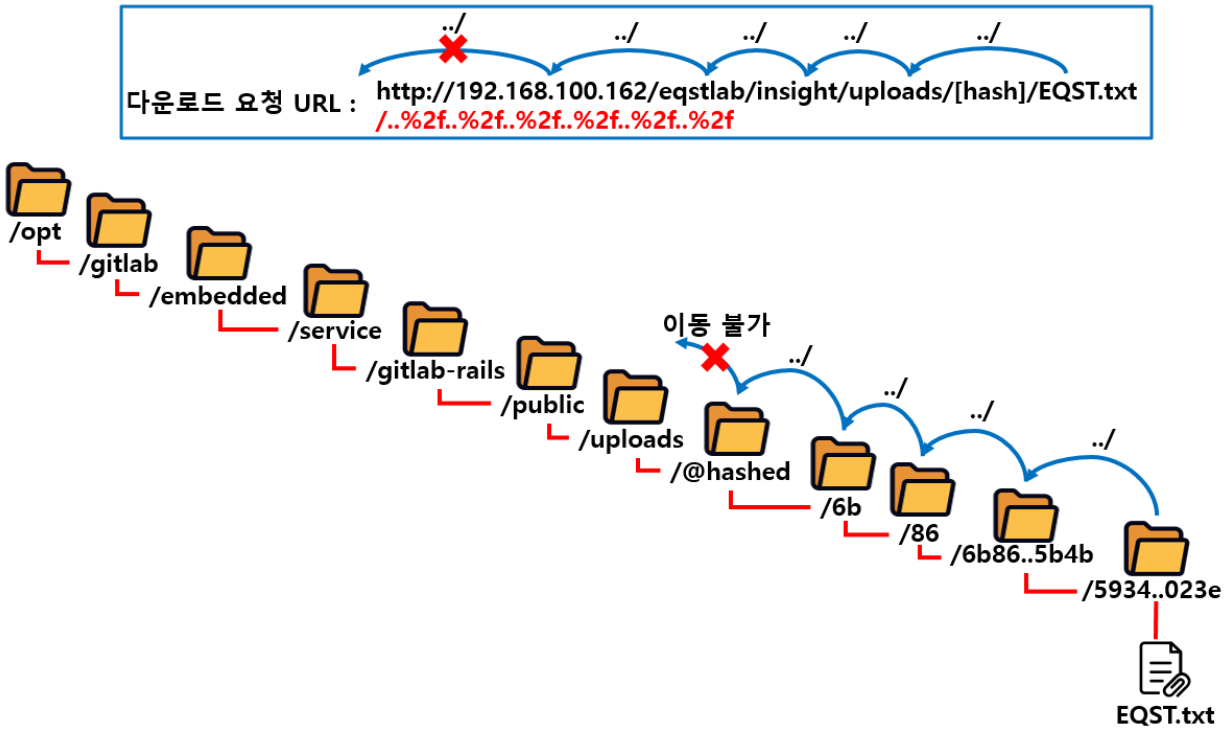


그림 22. 이동 불가 도식화

CVE-2023-2825 취약점을 활용해 WebRoot 디렉터리 보다 상위 디렉터리에 접근하기 위해서는 중첩된 하위 그룹을 추가하여, 다운로드 요청 URL 내의 포함된 디렉터리 수를 늘려야 한다.

따라서, WebRoot 보다 상위 디렉터리에 위치한 주요 정보 중 하나인 /opt/gitlab/embedded/service/gitlab-rails/config/secrets.yml 에 접근하기 위해서는 기존의 4 개의 디렉터리에 3 개를 추가하여 총 7 개의 상위 디렉터리로 이동해야 한다. 따라서 중첩된 하위 그룹 3 개를 추가해 CVE-2023-2825 취약점을 악용한다.

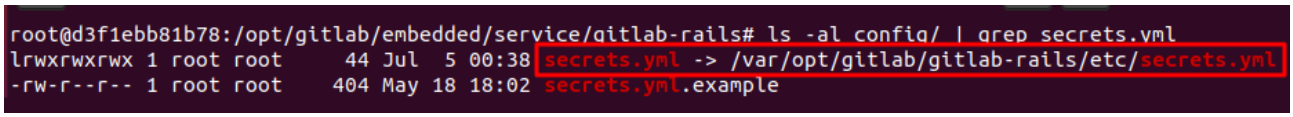


그림 23. 서버 내의 /config/secrets.yml 경로

7 개의 상위 디렉터리로 접근을 위해 3 개의 중첩된 하위 그룹을 생성한 그림은 아래와 같다.

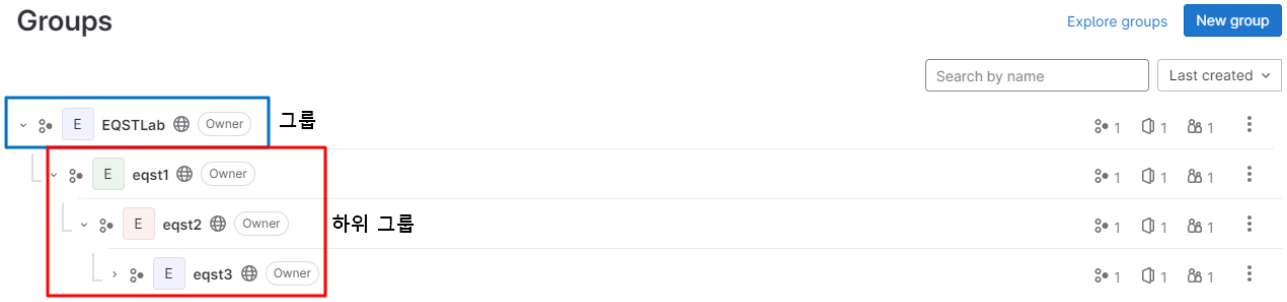


그림 24. 하위 그룹 생성

아래 그림은 중첩된 하위 그룹 3 개를 추가하여, WebRoot 디렉터리 보다 상위 디렉터리인 gitlab-rails 에 존재하는 /config/secrets.yml 파일에 접근하는 과정을 도식화한 그림이다.

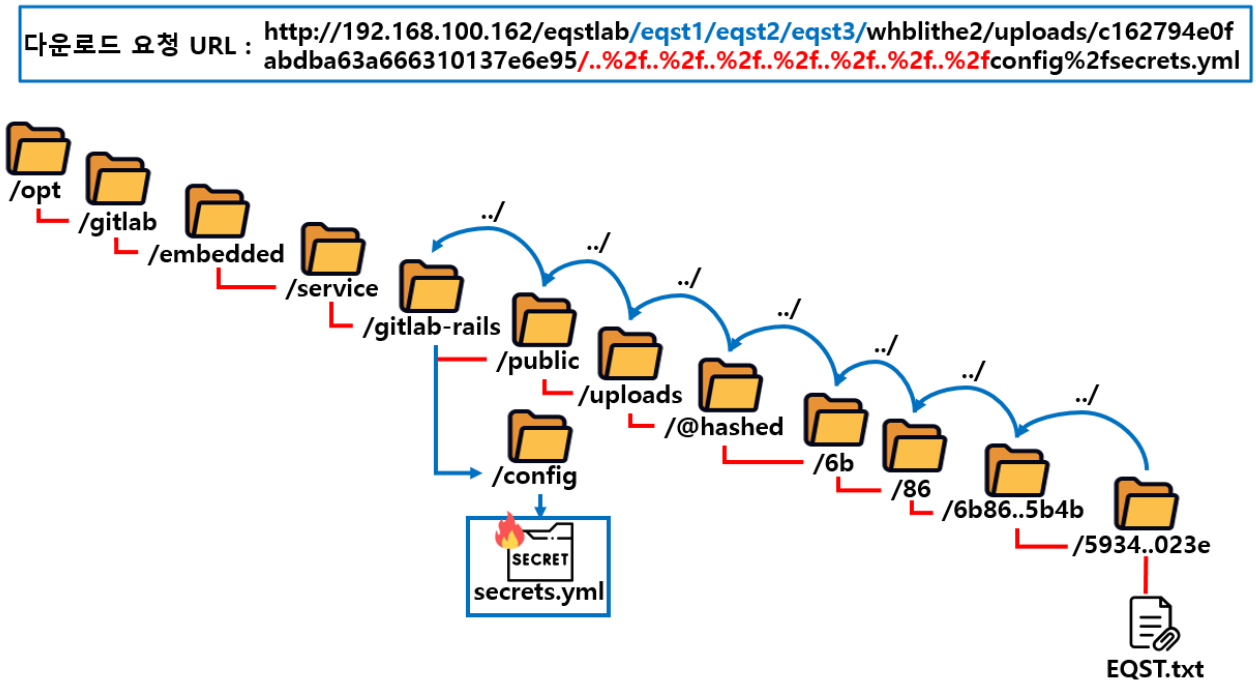


그림 25. /config/secrets.yml 접근 도식화

앞선 과정을 통해 중첩된 하위 그룹을 생성한 뒤 /config/secrets.yml 정보를 출력하는 페이로드는 아래와 같다.

그림 26. 상위 디렉터리의 /config/secrets.yml 파일 접근

따라서, 최상위 루트 디렉터리로 접근하기 위해서는 총 12 개의 상위 디렉터리로 이동해야 하므로, 1 개 그룹과 8 개의 하위 그룹 프로젝트의 첨부파일 경로를 조작해 /etc/passwd 까지 접근이 가능하다.

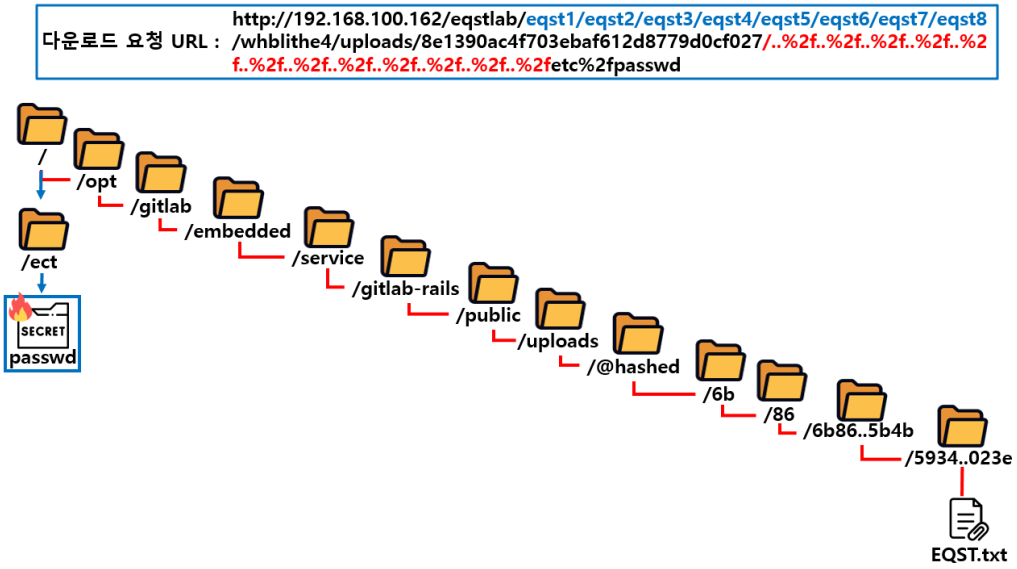


그림 27. 루트 디렉터리 접근

앞선 과정을 통해 12 개의 상위 디렉터리로 이동해 최상의 루트 디렉터리의 이동해 /etc/passwd 출력하거나 다운로드 받을 수 있다.

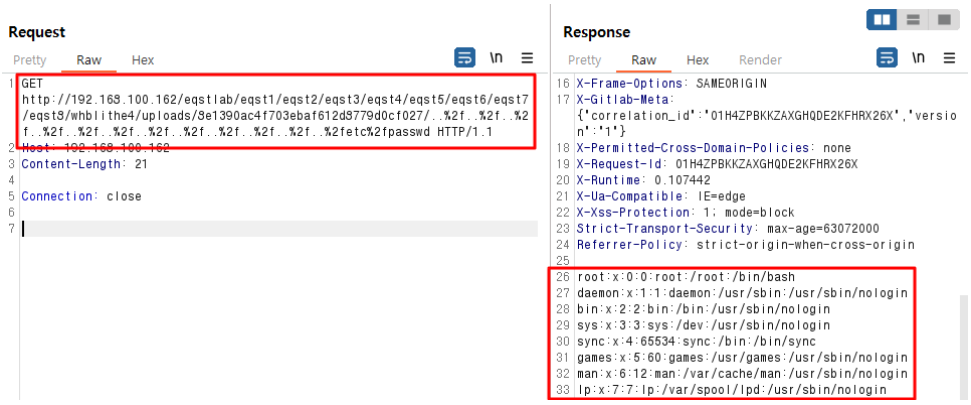


그림 28. root 폴더의 /etc/passwd 파일 정보 획득

URL 주소에 입력하면 /etc/passwd 파일을 다운로드할 수 있다.

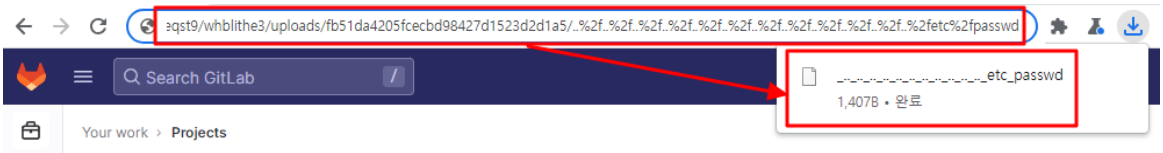


그림 29. /etc/passwd 파일 다운로드

■ 대응 방안

취약한 버전의 GitLab 서버를 운영 중이라면, 공격자가 프로젝트에 issue 를 생성하거나 공개된 Snippet 에 첨부파일을 등록하여 취약점을 악용할 수 있다. 이에 대응하기 위해서는 정규식 표현을 기반으로 디코딩한 문자열이 'Path Traversal 패턴'인지 검사하는 로직이 추가된 GitLab 16.0.1 이상 버전으로 업데이트하는 것이 안전하다.

```
def check_path_traversal!(path)
  return unless path

  path = path.to_s if path.is_a?(Gitlab::HashedPath)
  raise PathTraversalAttackError, 'Invalid path' unless path.is_a?(String)

  path = decode_path(path)
  path_regex = %r{(\A(\.{1,2})\z|\A\.\.[/\\]|[/\\]\.\.\z|[/\\]\.\.[/\\]|\n)}

  if path.match?(path_regex)
    logger.warn(message: "Potential path traversal attempt detected", path: "#{path}")
    raise PathTraversalAttackError, 'Invalid path'
  end

  path
end
```

그림 30. 정규식 표현을 통한 Path Traversal 탐지

16.0.1 이상 버전은 업로드를 관여하는 모듈에서 check_path_traversal 로직이 추가되어 Path Traversal 이 탐지될 경우 bad_reqeust 를 반환하고 있음을 확인할 수 있다.

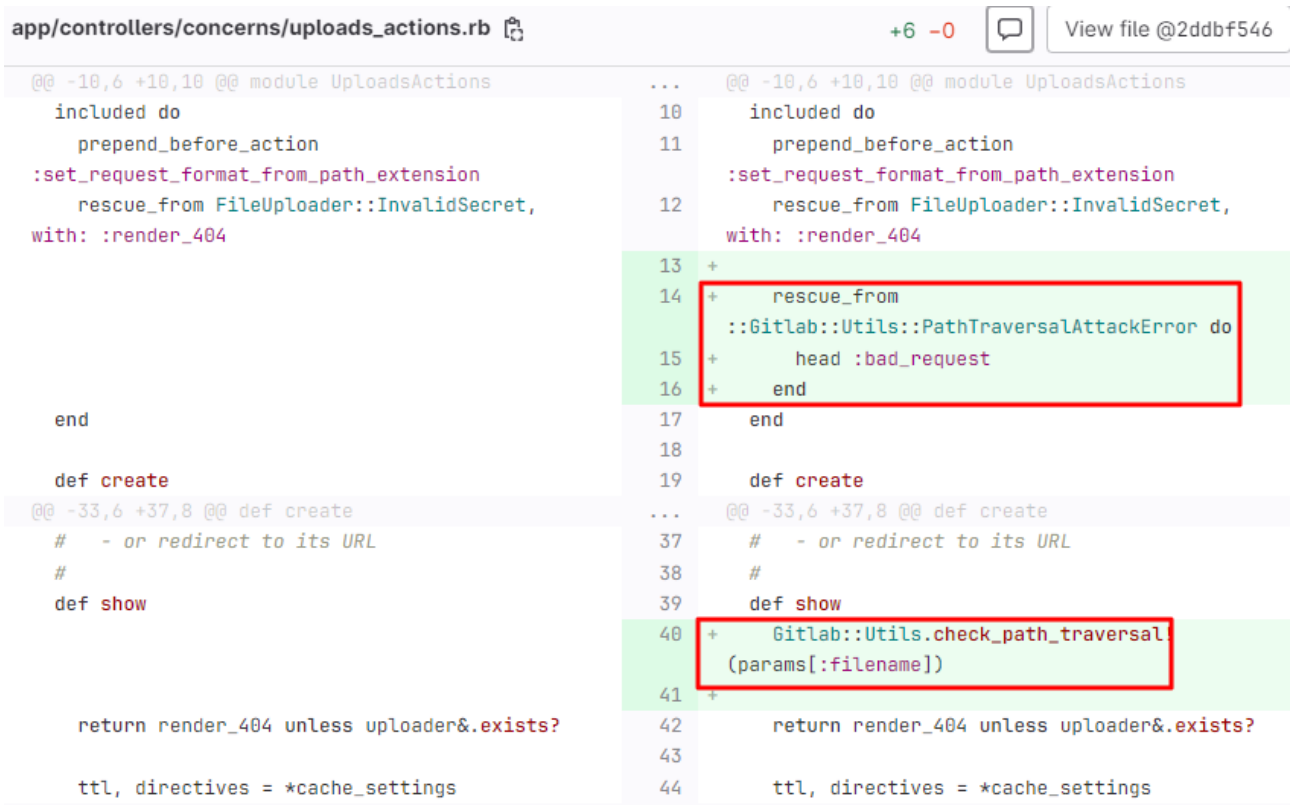


그림 31. 검사 로직 추가

불가피하게 버전 업데이트가 어려울 경우 공개된 프로젝트를 비활성화하거나, 웹 방화벽을 이용해 패킷을 검사하는 것이 필요하다. 하지만 이는 취약점에 대한 완벽한 대응 방안이 아니므로 검사 로직이 추가된 16.0.01 이상 버전으로 업데이트 하는 것을 권장한다.

■ 참고 사이트

- URL: https://labs.watchtowr.com/gitlab-arbitrary-file-read-gitlab-cve-2023-2825-analysis/
- URL: https://github.com/Occamsec/CVE-2023-2825.git
- URL: https://about.gitlab.com/releases/2023/05/23/critical-security-release-gitlab-16-0-1-released/