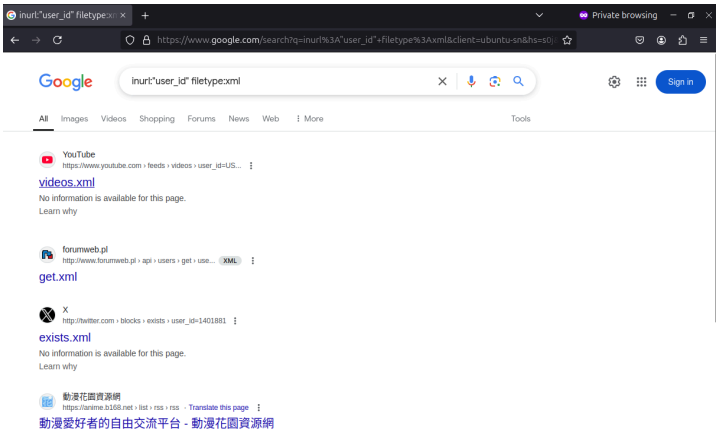
Theory:

A01:2021 - Broken Access Control

- Search ‘inurl:”user_id” filetype:xml’

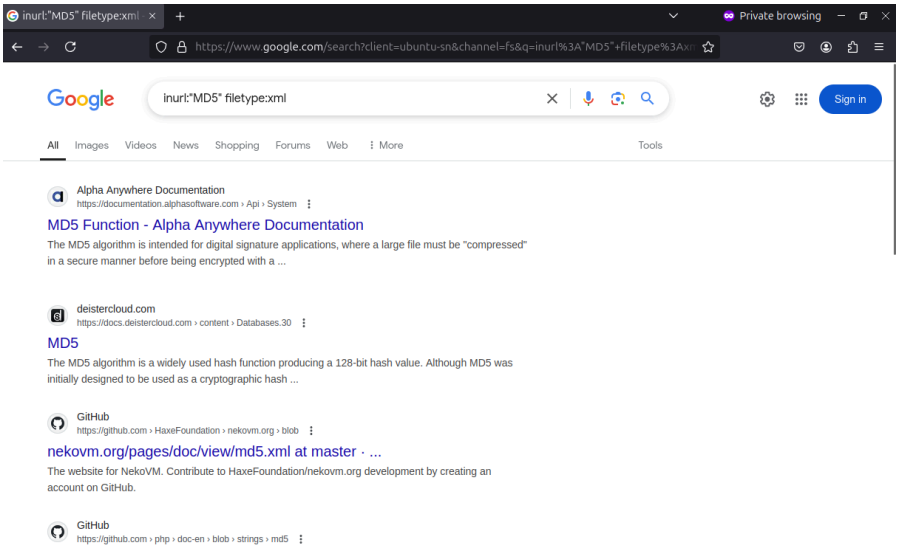


Screenshot of results:

- Found website https://www.forumweb.pl/api/users/get.xml?user_id
- Can alter url query params and access user information, proving that there is broken access control and user information can be accessed by anybody

A02:2021 - Cryptographic Failures

- Search ‘inurl:”MD5” filetype:xml’

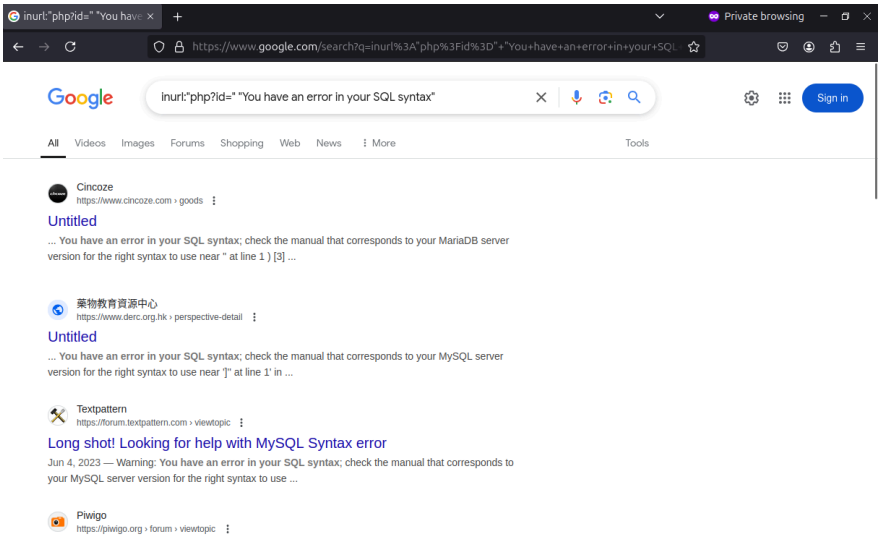


Screenshot of results:

- Found website https://fixtures/pulpproject.org/rpm-with-md5/repodata/repomd.xml
- Can see that they use MD5 encryption and can see encrypted hashes. MD5 is outdated and vulnerable, which is a cryptographic failure

A03:2021 - Injection

- Search ‘inurl:”.php?id=” “You have an error in your SQL syntax”’

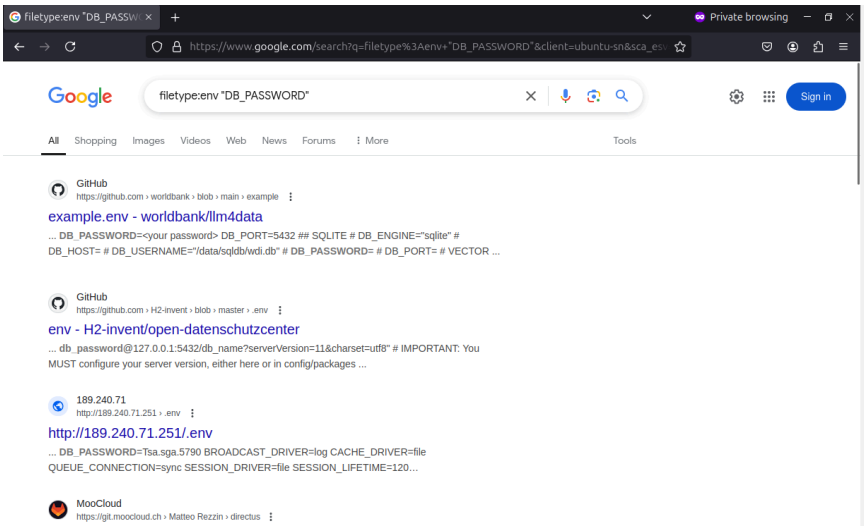


Screenshot of results:

- Found website https://www.derc.org.hk/en/perspective-detail.php?id=114
- Can add AND SLEEP(5) and make server sleep for 5, proving that the service is vulnerable to a SQL injection and the end user can control server actions

A04:2021 - Insecure Design

- Search ‘filetype:env “DB_PASSWORD”’

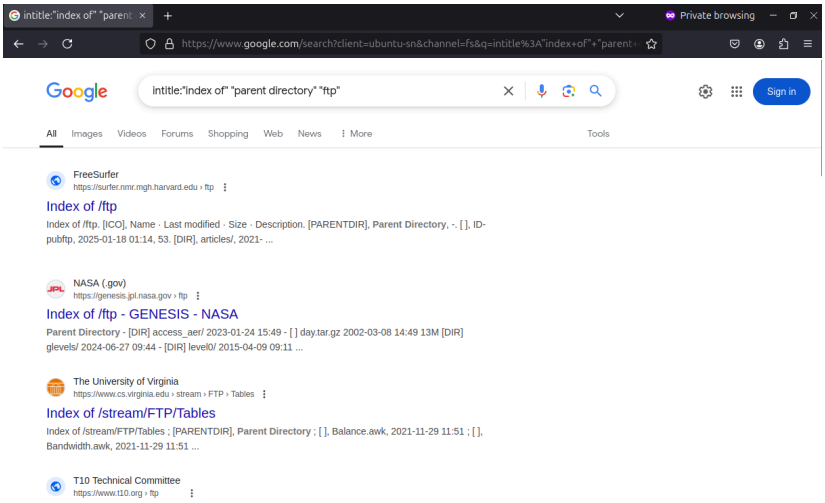


Screenshot of results:

- Found https://yellowslanches.com/.env
- Developers didn’t design with security in mind and accidentally exposed environment variables for api keys, db login, and other sensitive information.

A05:2021 - Security Misconfig

- Search intitle:"index of" "parent directory" "ftp"

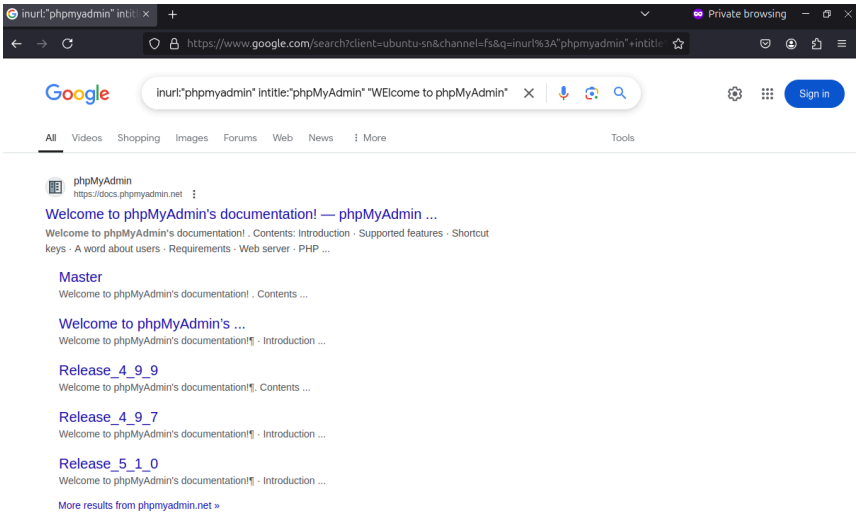


Screenshot of results:

- Found https://surfer.nmr.mgh.hardvard.edu/ftp
- Exposed ftp directory, allows for anybody to see potentially sensitive information.

A06:2021 - Vulnerable and Outdated Components

- Search inurl:”phpmyadmin” intitle:”phpMyAdmin” “Welcome to phpMyAdmin”

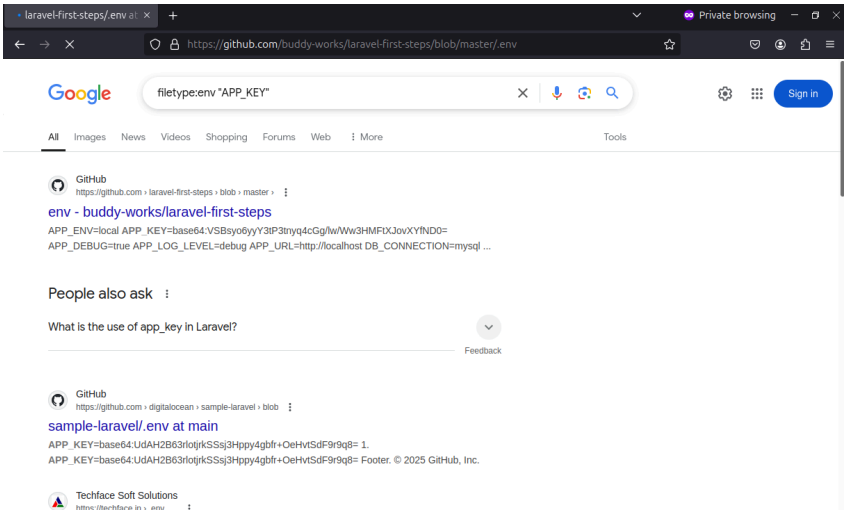


Screenshot of results:

- Found https://web.njit.edu/mysql/phpMyAdmin/doc/html/index.html
- Running outdated 4.2.7.1 with many known vulnerabilities, attackers can take advantage of these vulnerabilities and compromise the service.

A07:2021 - Identification and Authentication failures

- Search ‘filetype:env “APP_KEY”’

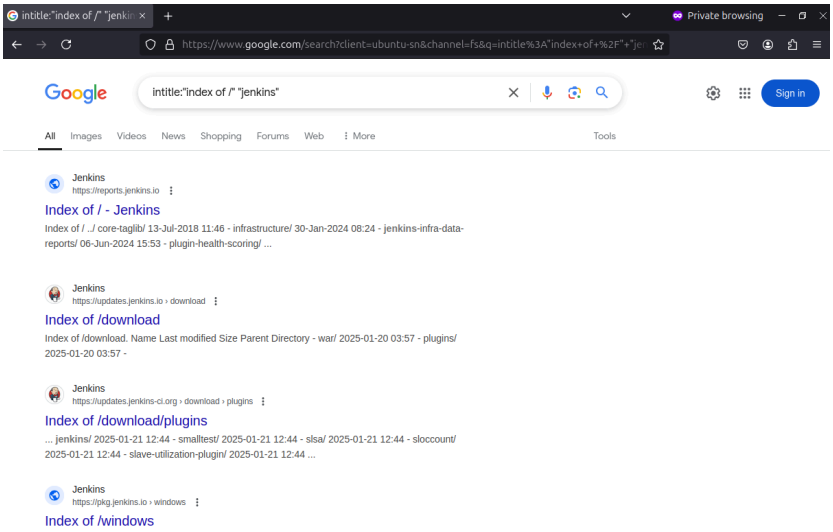


Screenshot of results:

- Found https://techface.in/.env
- Exposed api keys and db login info, users don't have to be authenticated or identified properly to access this sensitive data.

A08:2021 - Software and Data Integrity Failures

- Search 'intitle:"index of /" "jenkins"'

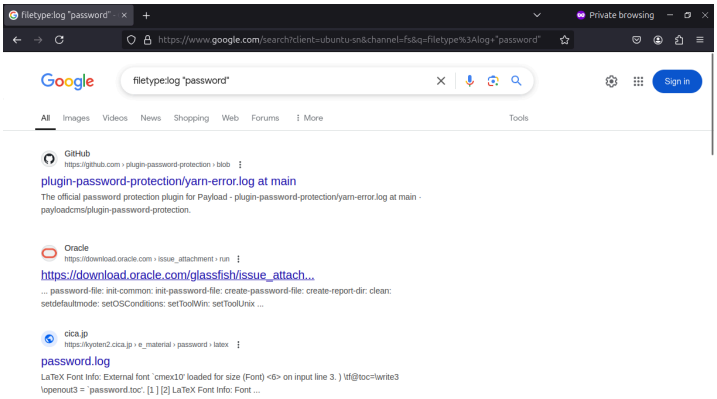


Screenshot of results:

- Found https://eclipse.hello2morrow.com/jenkins/
- Exposed CI/CD jenkins pipelines builds, exposes environment information along with other sensitive data that should not be publicly accessible.

A09:2021 - Security Logging and Monitoring Failures

- Search 'filetype:log "password"

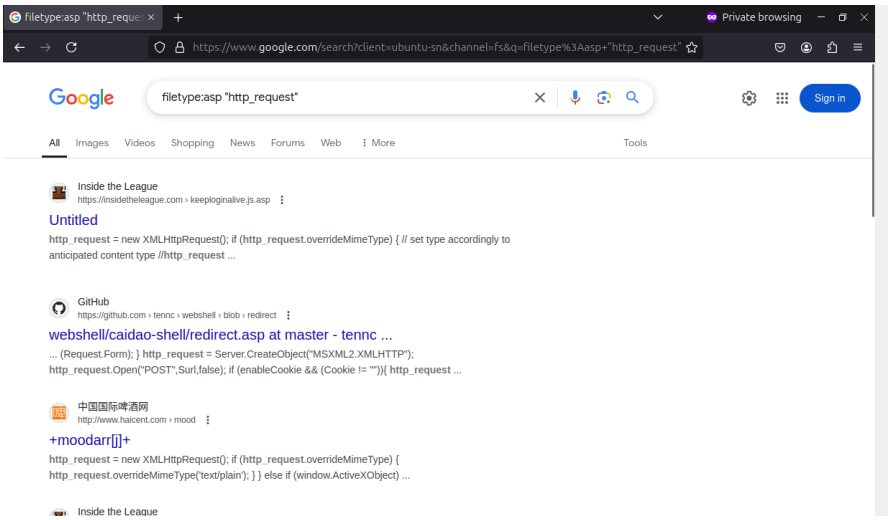


Screenshot of results: People also ask

- Found https://logs.onap.org/onap-integration/daily/onap_daily_nokia_dualstack_master/ci-ds-2021-06-23-00-09-42/infrastructure-healthcheck/k8s/kubernetes-status/pod-dev-aaf-locate-9649df66b-4rftp-aaf-locate-locate-service.log
- Exposed logs that contain sensitive information.

A10:2021 - SSRF

- Search ‘filetype:asp “http_request”’



Screenshot of results:

- Found https://insidetheleague.com/ASPHDWRegistrationAndLogin/keeploginalive.js.asp
- Can use info here to forge server side requests regarding login / keeping a login alive

Practice

Code found here: https://github.com/c2003-tamu/413/tree/main/directory_trav

Demonstration:

Context - in this web server, users should only be able to request files that are found in the project's runtime directory, but without proper sanitization of filepath, users can request files outside the project's scope. To demonstrate this, we can request 2 files: yes.html, which should be accessible as it is in the project's runtime directory and ../no.html, which should not be accessible because it is outside the server's scope.

When we request for yes.html, everything is good:

```
cade@cade-ThinkPad-T480s:~/Desktop/spring2025/csce413/413/directory_trav/runtimedir$ nc localhost 8080
GET yes.html

<p> you should be able to see this </p>

cade@cade-ThinkPad-T480s:~/Desktop/spring2025/csce413/413/directory_trav/runtimedir$
```

But when we request for ../no.html, we can see that we are accessing a file that we should not be able to access:

```
cade@cade-ThinkPad-T480s:~/Desktop/spring2025/csce413/413/directory_trav/runtimedir$ nc localhost 8080
GET ../no.html

<p> you should not be able to see this </p>

cade@cade-ThinkPad-T480s:~/Desktop/spring2025/csce413/413/directory_trav/runtimedir$
```

Server logs:

```
cade@cade-ThinkPad-T480s:~/Desktop/spring2025/csce413/413/directory_trav/runtimedir$ python3 insecure.py
Serving HTTP on localhost port 8080...
127.0.0.1 - - [21/Jan/2025 14:55:22] "GET yes.html" 200 -
127.0.0.1 - - [21/Jan/2025 14:56:46] "GET ../no.html" 200 -
```

This exploitation essentially means that all files on a host machine are accessible and can be accessed with no authentication. To fix this vulnerability, we need to validate the file path as can be seen in secure.py