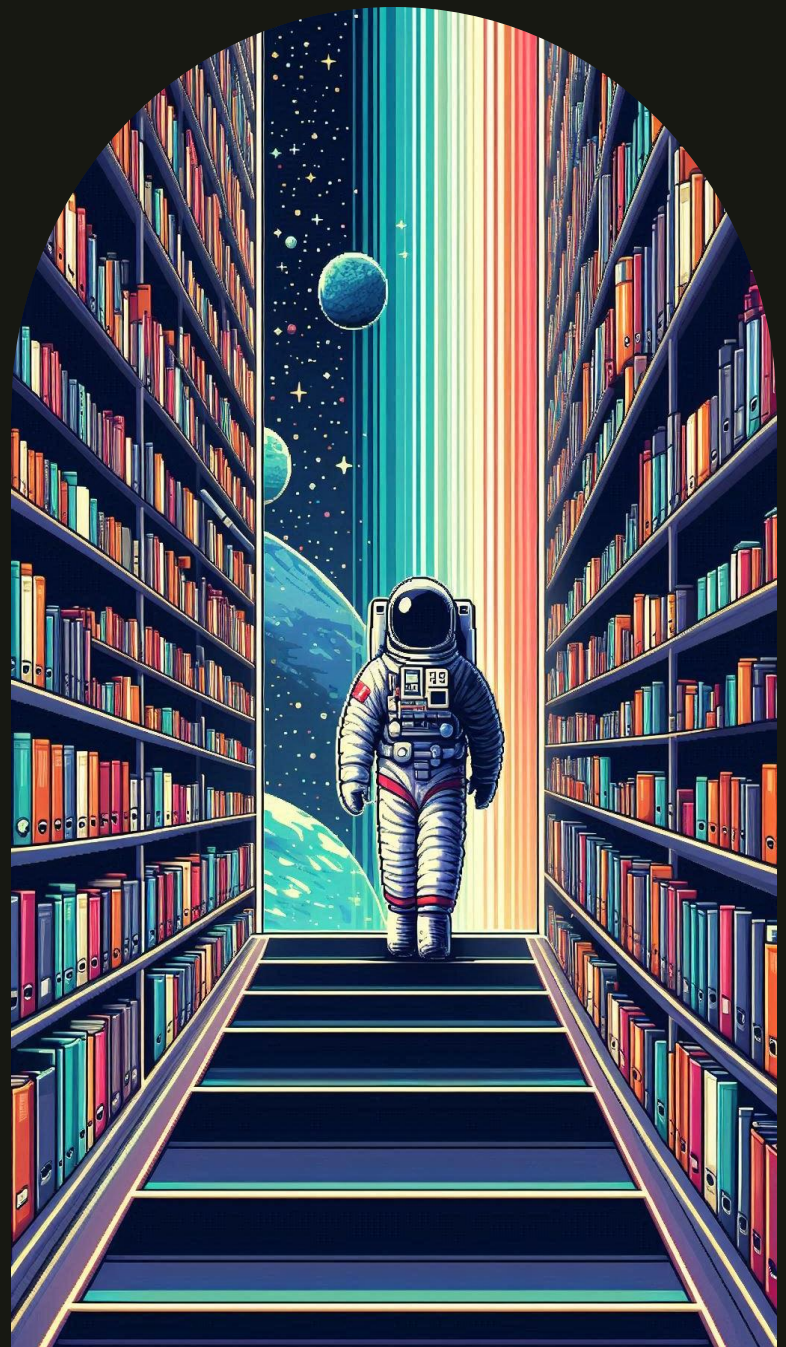

Recurity Labs

Exercise on Mitigating Common Vulnerabilities

Path Traversal

Mitigating Path Traversals

Exercise: Attempt to patch the handler to ensure that path traversals do not succeed

```
@GET
@Path("lfi-string-concat")
public byte[] lfiStringConcat(@QueryParam("x") String filename) throws IOException {
    var path = DIR + filename;
    FileInputStream fis = new FileInputStream(path);
    return fis.readAllBytes();
}
```

Attempt #1

```java
@GET
@Path("lfi-string-concat")
public byte[] lfiStringConcat(@QueryParam("x") String filename) throws IOException {
    var path = DIR + filename;
    if (!path.startsWith(DIR)) {
        throw new RuntimeException(" ... ");
    }
    FileInputStream fis = new FileInputStream(path);
    return fis.readAllBytes();
}
```

Ineffective: The string "*$DIR/../../*" still starts with *$DIR*

Attempt #2

```
@GET
@Path("lfi-string-concat")
public byte[] lfiStringConcat(@QueryParam("x") String filename) throws IOException {
    var path = DIR + filename;
    if (!new File(path).getAbsolutePath().startsWith(DIR)) {
        throw new RuntimeException(" ... ");
    }
    FileInputStream fis = new FileInputStream(path);
    return fis.readAllBytes();
}
```

Ineffective: The function *getAbsolutPath()* returns a string that still contains "*../*" and begins with *$DIR*.

Attempt #3

```java
@GET
@Path("lfi-string-concat")
public byte[] lfiStringConcat(@QueryParam("x") String filename) throws IOException {
    var path = DIR + filename;
    if (!new File(path).toPath().normalize().startsWith(DIR)) {
        throw new RuntimeException(" ... ");
    }
    FileInputStream fis = new FileInputStream(path);
    return fis.readAllBytes();
}
```

Works:

- *.normalize()* expresses the path without ".."-elements.
- *.startsWith()* is now executed on a *Path* object, which also ensures that "*$DIRxyz/file.txt*" does not match.