GitHub Security Lab (GHSL) Vulnerability Report: `GHSL-2022-008`

The GitHub Security Lab team has identified a potential security vulnerability in The OWASP Enterprise Security API.

We are committed to working with you to help resolve this issue. In this report you will find everything you need to effectively coordinate a resolution of this issue with the GHSL team.

If at any point you have concerns or questions about this process, please do not hesitate to reach out to us at `securitylab@github.com` (please include `GHSL-2022-008` as a reference).

If you are *NOT* the correct point of contact for this report, please let us know!

Summary

`getValidDirectoryPath` incorrectly treats sibling of a root directory as a child.

Product

The OWASP Enterprise Security API

Tested Version

v2.2.3.1 (The latest version of "Legacy" 2.x branch as ESAPI 3.x is in early development and has no releases yet.)

Details

Issue: `getValidDirectoryPath` bypass (`GHSL-2022-008`)

`parent` [1] - the third parameter in `getValidDirectoryPath` is used to validate that the `input` [2] path is "inside specified parent" directory [3].

```
public String getValidDirectoryPath(String context, String input /* [2] */, File
parent /* [1] */, boolean allowNull) throws ValidationException,
IntrusionException {
    try {
    if (isEmpty(input)) {
        if (allowNull) return null;
        throw new ValidationException( context + ": Input directory path
required", "Input directory path required: context=" + context + ", input=" +
input, context );
    }

    File dir = new File( input );

    // check dir exists and parent exists and dir is inside parent
```

```
    if ( !dir.exists() ) {
        throw new ValidationException( context + ": Invalid directory name",
"Invalid directory, does not exist: context=" + context + ", input=" + input );
    }
    if ( !dir.isDirectory() ) {
        throw new ValidationException( context + ": Invalid directory name",
"Invalid directory, not a directory: context=" + context + ", input=" + input );
    }
    if ( !parent.exists() ) {
        throw new ValidationException( context + ": Invalid directory name",
"Invalid directory, specified parent does not exist: context=" + context + ",
input=" + input + ", parent=" + parent );
    }
    if ( !parent.isDirectory() ) {
        throw new ValidationException( context + ": Invalid directory name",
"Invalid directory, specified parent is not a directory: context=" + context + ",
input=" + input + ", parent=" + parent );
    }
    if ( !dir.getCanonicalPath().startsWith(parent.getCanonicalPath() ) ) { // <--
-------- [3]
        throw new ValidationException( context + ": Invalid directory name",
"Invalid directory, not inside specified parent: context=" + context + ", input="
+ input + ", parent=" + parent );
    }

    // check canonical form matches input
    String canonicalPath = dir.getCanonicalPath();
    String canonical = fileValidator.getValidInput( context, canonicalPath,
"DirectoryName", 255, false);
    if ( !canonical.equals( input ) ) {
        throw new ValidationException( context + ": Invalid directory name",
"Invalid directory name does not match the canonical path: context=" + context +
", input=" + input + ", canonical=" + canonical, context );
    }
    return canonical;
    } catch (Exception e) {
    throw new ValidationException( context + ": Invalid directory name", "Failure
to validate directory path: context=" + context + ", input=" + input, e, context
);
    }
}
```

If the result of `parent.getCanonicalPath()` is not slash terminated it allows for partial path traversal.

Consider `"/usr/outnot".startsWith("/usr/out")`. The check is bypassed although `outnot` is not under the `out` directory. The terminating slash may be removed in various places. On Linux `println(new File("/var/"))` returns `/var`, but `println(new File("/var", "/"))` - `/var/`, however `println(new File("/var", "/").getCanonicalPath())` - `/var`.

PoC (based on a unittest):

```
Validator instance = ESAPI.validator();
ValidationErrorList errors = new ValidationErrorList();
assertTrue(instance.isValidDirectoryPath("poc", "/tmp/test2", new
File("/tmp/test/"), false, errors));
assertEquals(0, errors.size());
```

Impact

This issue allows to break out of expected directory.

Remediation

Consider using `getCanonicalFile().toPath().startsWith` to compare `Path`:

```
if (
!dir.getCanonicalFile().toPath().startsWith(parent.getCanonicalFile().toPath() ) )
```

GitHub Security Advisories

We recommend you create a private GitHub Security Advisory for this finding. This also allows you to invite the GHSL team to collaborate and further discuss this finding in private before it is published.

Credit

This issue was discovered and reported by GHSL team member @JarLob (Jaroslav Lobačevski).

Contact

You can contact the GHSL team at `securitylab@github.com`, please include a reference to `GHSL-2022-008` in any communication regarding this issue.

Disclosure Policy

This report is subject to our coordinated disclosure policy.