Immunity

Local Privilege Escalation in GOG Galaxy

2020-10-08

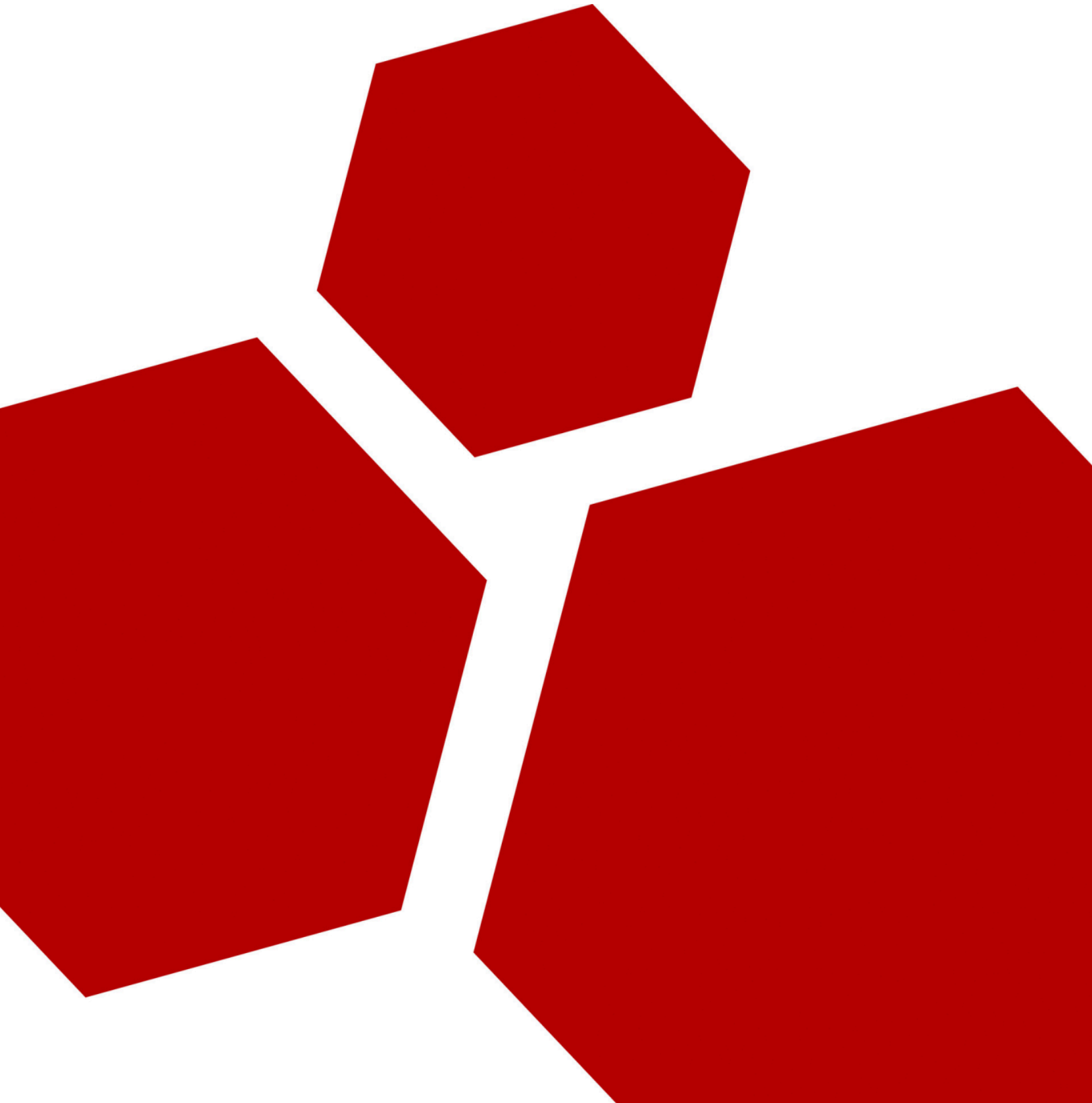

Table of Contents

Advisory Information

Title: Local Privilege Escalation in GOG Galaxy
Vendors contacted: GOG.com
Release mode: Coordinated Release
Credits: This vulnerability was discovered by Juan Pablo De Francesco.

Vulnerability Information

Class: Incorrect Default Permissions [CWE-276]
Affected Version: GOG Galaxy 2.0.16.187 (Windows platform)
Remotely Exploitable: No
Locally Exploitable: Yes
Severity: High - 8.8 (CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
CVE Identifier: CVE-2020-25769

Vulnerability Description

GOG Galaxy 2.0 is a platform developed by CD Projekt designed as a storefront, software delivery, social network and as a unified game launcher; currently offering clients to Windows and macOS operating systems.

An exploitable local privilege escalation vulnerability exists in the latest version of the GOG Galaxy 2.0 Windows client. During the usual workflow the Windows client relies on the *GalaxyClientService* component for handling privileged tasks. On a typical installation, that service is installed and configured to run under a local system account and the presented vulnerability affects this component.

The *GalaxyClientService* uses a TCP channel to communicate with lower-privileged applications. Through this channel the service receives information about the privileged operations that needs to be performed. By default, the service listens for incoming connections on the 9978 TCP port of the loopback interface (127.0.0.1:9978) and the communication is performed using a custom protocol based on Google's Protobuf specification.

To prevent an attacker from sending malicious messages, the *GalaxyClientService* only process incoming messages if they come from trusted applications. In particular, Immunity found that the executable located at the following location is a trusted application:

```
%ALLUSERSPROFILE%\GOG.com\Galaxy\prefetch\desktop-galaxy-updater\GalaxyUpdater.exe
```

But the above application does not exist after a fresh installation of GOG Galaxy 2.0. Moreover, the *prefetch* folder is not even present. If we check the permissions of the parent *Galaxy* folder it can be seen that any user (BUILTIN\Users) can create subfolders there:

```
C:\ProgramData\GOG.com>accesschk -nobanner -dl %ALLUSERSPROFILE%\GOG.com\Galaxy
C:\ProgramData\GOG.com\Galaxy
  DESCRIPTOR FLAGS:
      [SE_DACL_PRESENT]
      [SE_DACL_PROTECTED]
  OWNER: DESKTOP-1EBQOLD\juan
  [0] ACCESS_ALLOWED_ACE_TYPE: NT AUTHORITY\SYSTEM
          [OBJECT_INHERIT_ACE]
          [CONTAINER_INHERIT_ACE]
          [INHERITED_ACE]
        FILE_ALL_ACCESS
  [1] ACCESS_ALLOWED_ACE_TYPE: BUILTIN\Administrators
          [OBJECT_INHERIT_ACE]
          [CONTAINER_INHERIT_ACE]
          [INHERITED_ACE]
        FILE_ALL_ACCESS
  [2] ACCESS_ALLOWED_ACE_TYPE: DESKTOP-1EBQOLD\juan
          [INHERITED_ACE]
        FILE_ALL_ACCESS
  [3] ACCESS_ALLOWED_ACE_TYPE: CREATOR OWNER
          [OBJECT_INHERIT_ACE]
          [CONTAINER_INHERIT_ACE]
          [INHERIT_ONLY_ACE]
          [INHERITED_ACE]
        GENERIC_ALL
  [4] ACCESS_ALLOWED_ACE_TYPE: BUILTIN\Users
          [OBJECT_INHERIT_ACE]
          [CONTAINER_INHERIT_ACE]
          [INHERITED_ACE]
        FILE_LIST_DIRECTORY
        FILE_READ_ATTRIBUTES
        FILE_READ_EA
        FILE_TRAVERSE
        SYNCHRONIZE
        READ_CONTROL
  [5] ACCESS_ALLOWED_ACE_TYPE: BUILTIN\Users
          [CONTAINER_INHERIT_ACE]
          [INHERITED_ACE]
        FILE_ADD_FILE
        FILE_ADD_SUBDIRECTORY
        FILE_WRITE_ATTRIBUTES
        FILE_WRITE_EA
```

Figure 1 - %ALLUSERSPROFILE%\GOG.com\Galaxy folder permissions.

A resulting consequence of the relaxed folder permissions is that any user will be able to deploy a fake *GalaxyUpdater.exe* executable at:

```
%ALLUSERSPROFILE%\GOG.com\Galaxy\prefetch\desktop-galaxy-updater\GalaxyUpdater.exe
```

On the other hand, if the latest version was obtained from the update of an older version, the *GalaxyUpdater.exe* file will be present having the following permissions:

```
C:\ProgramData\GOG.com>accesschk -nobanner -l %ALLUSERSPROFILE%\GOG.com\Galaxy\prefetch\
desktop-galaxy-updater\GalaxyUpdater.exe
C:\ProgramData\GOG.com\Galaxy\prefetch\desktop-galaxy-updater\GalaxyUpdater.exe
  DESCRIPTOR FLAGS:
      [SE_DACL_PRESENT]
      [SE_DACL_PROTECTED]
  OWNER: BUILTIN\Administrators
  [0] ACCESS_ALLOWED_ACE_TYPE: NT AUTHORITY\SYSTEM
          [INHERITED_ACE]
        FILE_ALL_ACCESS
  [1] ACCESS_ALLOWED_ACE_TYPE: BUILTIN\Administrators
          [INHERITED_ACE]
        FILE_ALL_ACCESS
  [2] ACCESS_ALLOWED_ACE_TYPE: DESKTOP-1EBQOLD\juan
          [INHERITED_ACE]
        FILE_ALL_ACCESS
```

Figure 2 – GalaxyUpdater.exe file permissions after an update (Case 1).

As shown above, the regular user will have all kinds of access on that file, including the ability to overwrite it and replace it with an attacker-controlled *GalaxyUpdater.exe* executable.
Is worth mentioning that in some cases Immunity found that after an update the file and the relevant folders all remain with more restrictive permissions:

```
C:\ProgramData\GOG.com>accesschk -nobanner -l %ALLUSERSPROFILE%\GOG.com\Galaxy\prefetch\
desktop-galaxy-updater\GalaxyUpdater.exe
C:\ProgramData\GOG.com\Galaxy\prefetch\desktop-galaxy-updater\GalaxyUpdater.exe
  DESCRIPTOR FLAGS:
      [SE_DACL_PRESENT]
      [SE_DACL_PROTECTED]
  OWNER: BUILTIN\Administrators
  [0] ACCESS_ALLOWED_ACE_TYPE: NT AUTHORITY\SYSTEM
          [INHERITED_ACE]
        FILE_ALL_ACCESS
  [1] ACCESS_ALLOWED_ACE_TYPE: BUILTIN\Administrators
          [INHERITED_ACE]
        FILE_ALL_ACCESS
```

Figure 3 - GalaxyUpdater.exe file permissions after an update (Case 2).

Even though it is unclear why there are different behaviors observed, it can be stated that in the first two cases (fresh install and update-Case1), an attacker will be able to fully control the *GalaxyUpdater.exe* executable, and as a consequence, will be able to successfully send arbitrary messages to the *GalaxyClientService*.

The messages, based on Google's Protobuf format, also contain an HMAC (SHA-512) that the service uses to verify its authenticity before processing it. Immunity was able to extract the original key used to generate the HMAC codes. At this point, it is possible to successfully send arbitrary messages to the service and be sure that those will be processed by *GalaxyClientService*.

One of the many messages that can be sent to the service is the *"LaunchElevatedRequest"*, which may be used to launch an arbitrary process. By properly setting special options in the message,

the process will run using a token duplicated from the current service's token. As by default this service runs using a local system account (NT Authority\System), it will be possible to run arbitrary processes as NT Authority\System.

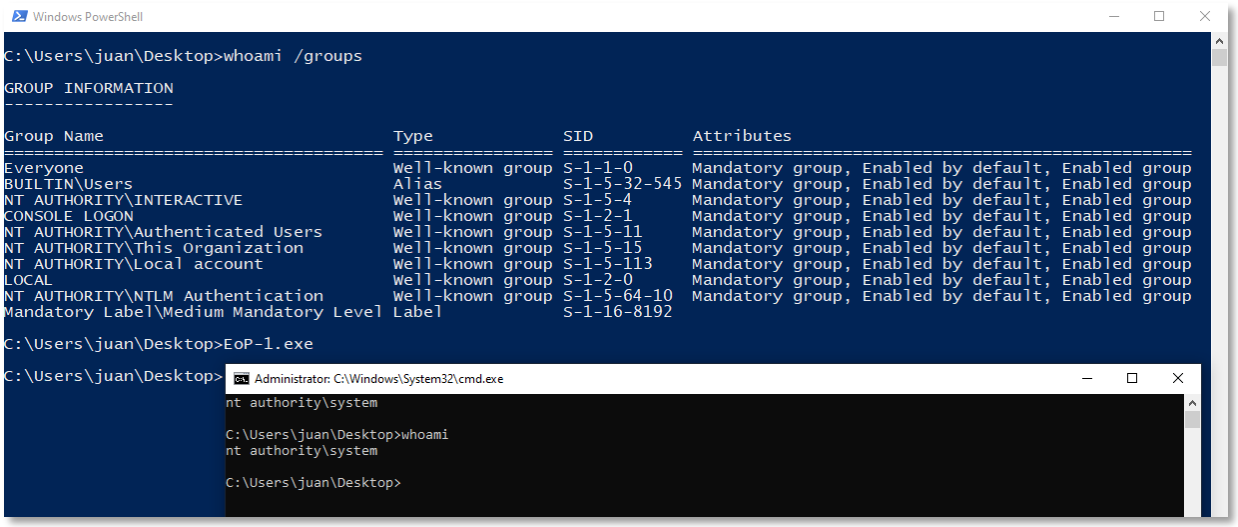

Figure 4 – Running our own developed PoC that exploits this vulnerability.

Report Timeline

2020-06-29: Initial contact with the vendor via security@gog.com.
2020-07-03: Second attempt to contact the vendor via security@gog.com.
2020-07-06: Third attempt to contact the vendor via security@gog.com and using the support form from the vendor web site.
2020-07-06: GOG.com Support replies sharing an email of a direct contact.
2020-07-06: Fourth attempt to contact the security team via bstyczynski@gog.com.
2020-07-14: Fifth attempt to contact the security team via bstyczynski@gog.com.
2020-07-15: GOG.com Security Team requested more information about the vulnerability.
2020-07-15: Immunity requested public key to share the vulnerability information.
2020-07-16: GOG.com Security Team sends a public key.
2020-07-16: A draft report with technical details and a proof of concept was sent to the vendor.
2020-07-20: Immunity asks for confirmation of receiving the report.
2020-07-23: GOG.com confirmed the reception of the report.
2020-07-31: Immunity requests a status update.
2020-08-07: Immunity requests a status update.
2020-08-19: Immunity requests a status update.
2020-08-24: Immunity requests a status update. Due to the lack of response from the vendor Immunity schedules advisory release to 2020-09-28.
2020-08-26: GOG.com communicates that the development team is working on the issue.
2020-08-27: GOG.com communicates that the fix for the vulnerability will we available shortly.
2020-08-27: Immunity requests to be notified 5 business days before the fix is released to coordinate advisory release date.

2020-08-28: GOG.com communicates they do not disclose exact publication dates for their patches and fixes.
2020-08-31: Immunity communicates that GOG.com should share the fix release date in order to coordinate the advisory release date.
2020-09-08: Immunity requests a status update.
2020-09-16: GOG.com schedules the fix release date to 2020-09-23.
2020-09-16: Immunity acknowledges the receipt of the fix release date.
2020-09-18: Immunity sent a request to Mitre for the CVE ID.
2020-09-18: Mitre assigns CVE-2020-25769.
2020-09-18: GOG.com notifies that they need to work on a new date for the fix release date but will happen not later than 2020-10-01.
2020-09-18: Immunity requests to be notified 5 business days before the fix is released to coordinate advisory release date.
2020-10-01: GOG.com release the fix on version 2.0.21 Beta.
2020-10-08: Immunity published the advisory.

Disclaimer