July 02, 2020

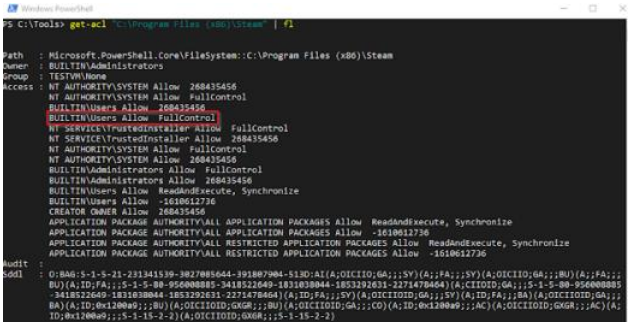
STEAM - ESCALATION OF PRIVILEGES

—

Product: Steam Client Software
Version: 2.10.91.91
Tested on: Windows 10 Pro 2004 x64
Vendor informed: Yes
PoC: This blog post
CVE: CVE-2020-15530

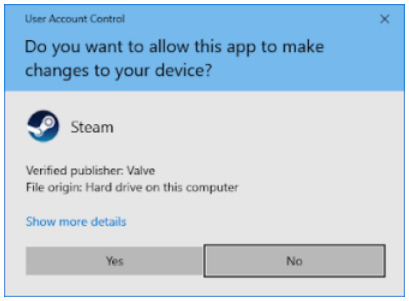
Brief Description: A local attacker can use the steam client updater/installer to execute malicious executables in an elevated context via the installation process. This also results in persistence via a hijacked service executable which will run as NT AUTHORITY\SYSTEM.

Vulnerability Description: If valve´s steam client is installed the installation path can be deleted by normal users.

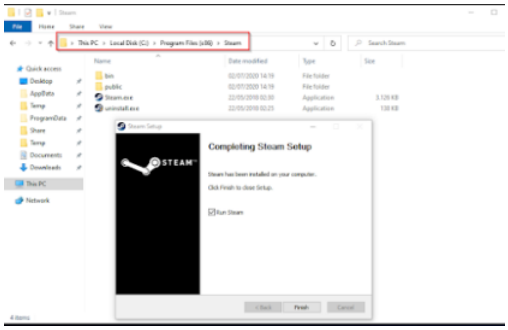


A local attacker can simply delete this folder to force users into a new installation of steam (there is no repair option or similar)

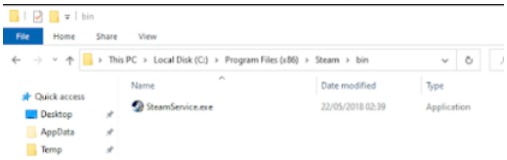
The steam-setup installer needs be executed in an elevated context



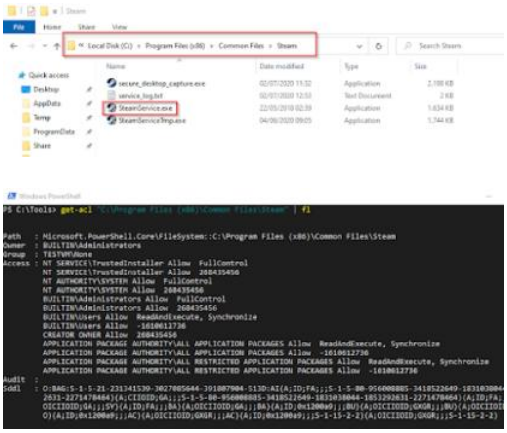
By default the installer will create the folder "C:\Program Files (x86)\Steam" with two sub folders and two files during the installation. The installer also modifies the ACL of this folder so users are able to read, write and delete files as shown above:



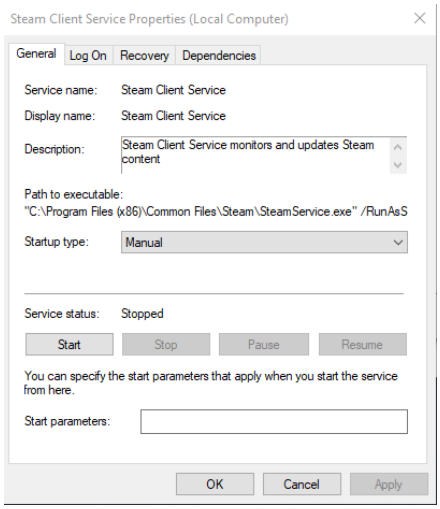
When we look at the "bin" folder we see a "SteamService.exe" file.



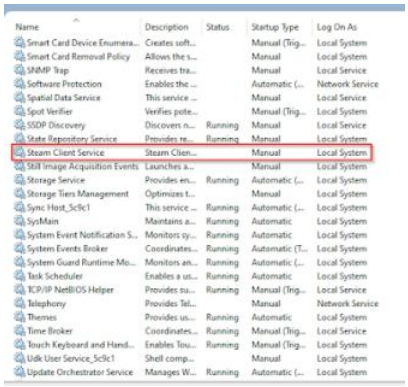
We will also find this file under "C:\Program Files (x86)\Common Files\Steam" which is not under users control



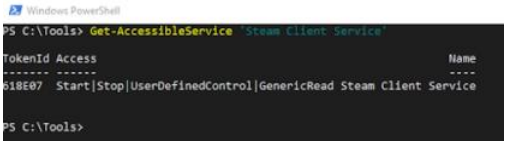
As the name "Steam**Service**" already suggests this file is used by the "Steam Client Service"



This is service only runs when it is needed and is stopped by default. "Steam Client Service" is also executed as "NT AUTHORITY\SYSTEM" ...



... and can be controlled (start|stop) by regular users:



But how can I replace the non writable "SteamService.exe" with a malicious one?

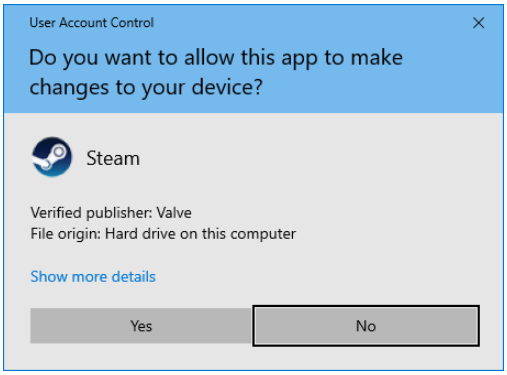
After analyzing the installer I discovered that the installer will install steam in the following way (I left some steps to keep it simple):

1. "C:\Program Files (x86)\Steam" including the two sub folders and files mentioned above will be created
2. A shortcut for the user will be created on the desktop and start menu
3. "SteamService.exe" will be copied from "C:\Program Files (x86)\Steam\bin" to "C:\Program Files (x86)\Common Files\Steam"
4. "C:\Program Files (x86)\Steam\steam.exe" will be launched elevated

To successfully exploit the installer we need to get between step no. 2 and step no. 3 to replace "C:\Program Files (x86)\Common Files\Steam\SteamService.exe" with a malicious one. After the replacement the file should be copied to "C:\Program Files (x86)\Common Files\Steam" and executed as "NT AUTHORITY\SYSTEM" when the service starts.

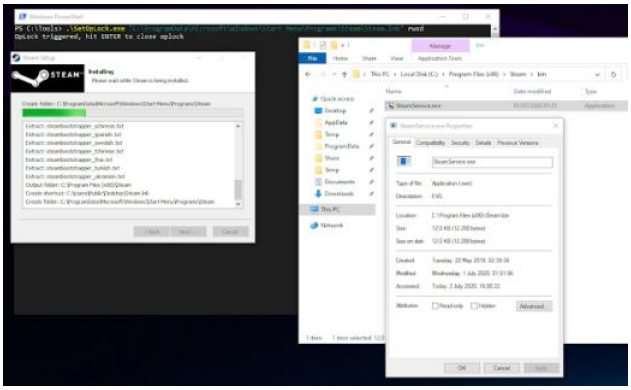
This maybe sounds difficult but is in fact pretty easy ;) In step no. 2 we see that a shortcut for the user will be created **before** "SteamService.exe" will be copied. It would be nice if we can tell the installer to wait until the file is replaced and then continue with the installation. This can be done by using opportunistic locks (oplocks). I use the tool "SetOpLock" by James Forshaw GitHub to set oplocks. When we put all the pieces together we get an attack like this:

The user installs steam (doesn´t matter if the previous installation was deleted/damaged by the attacker before or simply because the user wants to play some games). So the user downloads the installer and runs it after the download. Of course the user has to accept the UAC prompt because otherwise steam will not be installed.

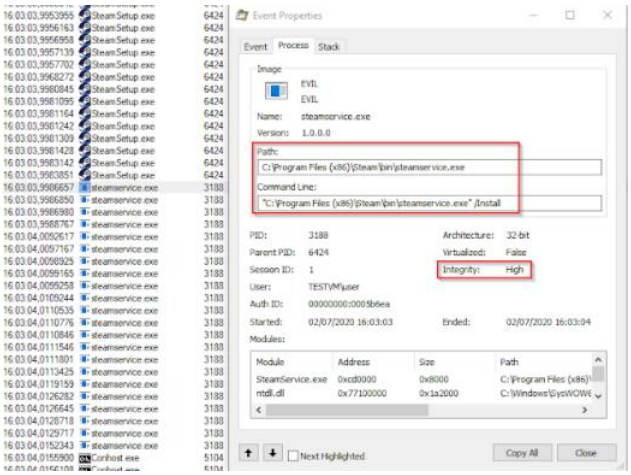


In the background the attacker (usually it is just a script, exe file, office macro etc.) waits for the setup process to be started. Once the setup runs, which can be easily determined by checking running processes, the attacker creates the shortcut "steam.lnk" in the start menu folder "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Steam". If the folder does not exists (because steam is installed for the first time) the attacker simply uses a loop to check if the file was created; once it is created the oplock will be set:

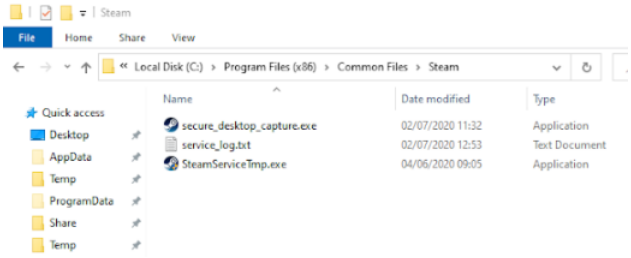
The oplock will be triggered shortly and the uninstaller will stop/pause/wait. The attacker has now time to replace "C:\Program Files (x86)\Steam\bin\SteamService.exe" with an evil version:



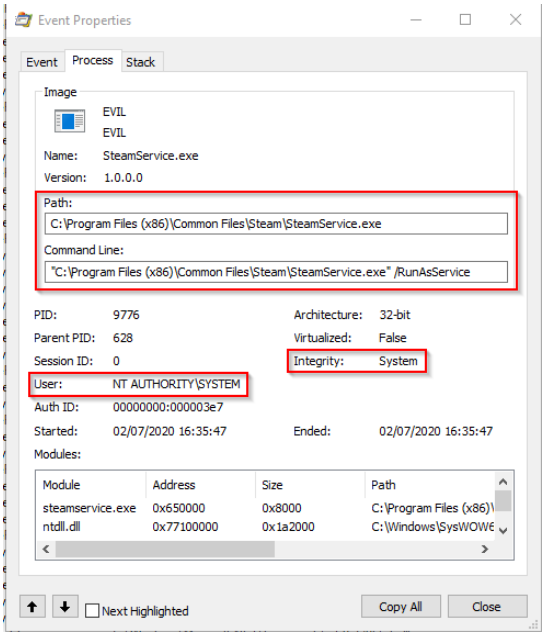
When the oplock is released "SteamService.exe" will be executed elevated:



At this point we have our evil version of "SteamService.exe" launched elevated! But looking at "C:\Program Files (x86)\Common Files\Steam" shows that "SteamService.exe" was not copied automatically:



So the evil version of "SteamService.exe" was executed elevated, but the service will not run our executable when started (because there is no executable with the name "SteamService.exe")... But "SteamService.exe" was executed elevated and therefore an attacker only needs to copy the file to "C:\Program Files (x86)\Common Files\Steam" and start the service. Because the attacker now has elevated access rights the file can be written to "C:\Program Files (x86)\Common Files\Steam" which is not allowed for normal users. When the attacker now starts the service (or the service is started by steam) the malicious version of "SteamService.exe" will be executed as "NT AUTHORITY\SYSTEM"



This service will be started every time the user starts steam. Also steam adds an autostart entry during the installation that will start steam at every user login resulting in executing the malicious "SteamService.exe" as "NT AUTHORITY\SYSTEM" at every user login.

Personal Note: This bug can be mitigated by setting correct ACL on the bin folder. However it is also a good example how users can escalate their privileges starting at the user level over high integrity and finally resulting in code execution as system. I am also pretty sure that somebody discovered this before me and is using it for some evil sh*t.

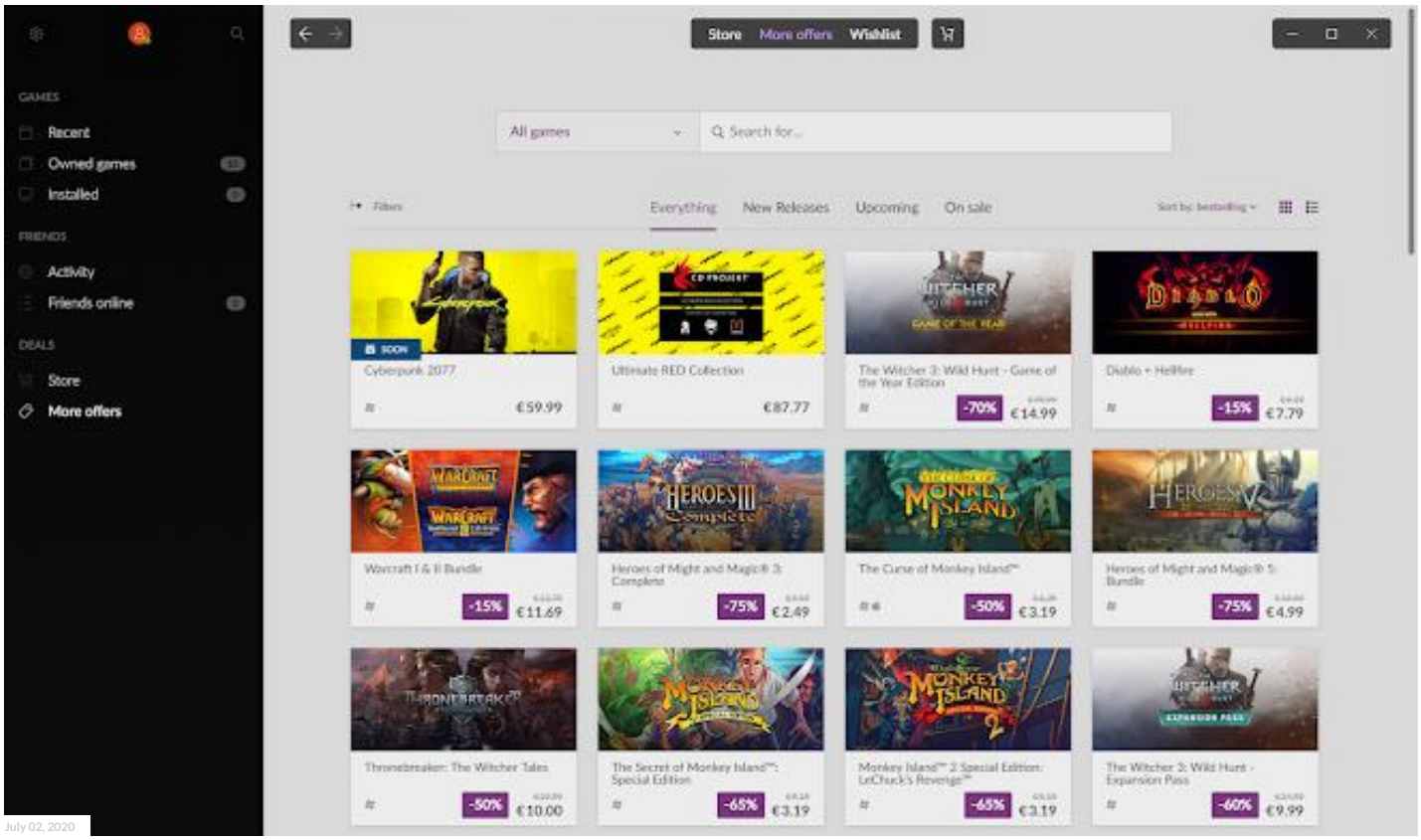
Showing 7 entries (filtered from **1,566** total entries) Search: winsat

Auto-elevated ▲	Executable	DLL	Procedure
✔	winsat.exe	d3d10_1.dll	DllMain
✔		d3d10_1core.dll	DllMain
✔		d3d10.dll	DllMain
✔		d3d10core.dll	DllMain
✔		d3d11.dll	DllMain
✔		dxgi.dll	DllMain
✔		winmm.dll	DllMain

July 30, 2020

UAC BYPASS VIA DLL HIJACKING AND MOCK DIRECTORIES

Share



July 02, 2020

GOG GALAXY - ESCALATION OF PRIVILEGES INCL. CODE EXECUTION

Share

Daniel Gebert

VISIT PROFILE

Archive

Report Abuse