ReVuln Ltd.
http://revuln.com
@revuln
info@revuln.com

Revision 1

STEAM SERVICE SECURITY

BY LUIGI AURIEMMA

How a malware or an exploit can use the Steam local service to escalate its privileges.

TABLE OF CONTENTS

Contents

Introduction 1
How Steamservice works 3
No signature verification – DLL hijacking 4
Abusing the service for privileged tasks 5
Possible steps for an attacker 8
The proof-of-concept 9
FAQ 10
History 12
Company Information 13

INTRODUCTION

Introduction

STEAM

"Steam[1] is an internet-based digital distribution, digital rights management, multiplayer, and communications platform developed by Valve Corporation. It is used to distribute games and related media from small, independent developers and larger software houses online."[2]

It's not easy to define Steam because it's not just a platform for buying games but also a social network, a market for game items, a framework[3] for integrating various functionalities in games, an anti-cheat, a cloud and more.

But the most important and attractive feature, from a security point of view, is its incredible diffusion[4] [5].

STEAMSERVICE

In 2007 Valve introduced a new local Windows service in Steam for handling the tasks that require Administrator privileges and maintaining the main client process Steam.exe under the normal limited privileges of the current user.

This is a common practice adopted by many software developers moreover after Microsoft introduced the UAC[6] technology from Windows Vista. In fact the secondary job of such service is avoiding to annoy the user with continuous Windows popups requiring the confirmation for using higher privileges.

The service is used also for monitoring the processes of the running games and it's part of Valve anti-cheat (VAC).

In Steam the local service that performs these operations is called "Steam Client Service"[7], a Manual service with SYSTEM privileges. The service can be started by any user but it will terminate immediately if some requirements are not met.

The service is automatically started by Steam when launched and it remains active till Steam is working.

The "Steam Client Service" is a required component[8].

[1] http://steampowered.com
[2] http://en.wikipedia.org/wiki/Steam_(software)
[3] http://www.steampowered.com/steamworks/
[4] http://www.joystiq.com/2014/01/15/steam-has-75-million-active-users-valve-announces-at-dev-days/
[5] http://www.dualshockers.com/2014/06/29/steam-passes-8-million-concurrent-users/
[6] http://en.wikipedia.org/wiki/User_Account_Control
[7] C:\Program Files (x86)\Common Files\Steam\SteamService.exe
[8] https://support.steampowered.com/kb_article.php?ref=9626-UOAC-4950

INTRODUCTION

WHY IT'S INTERESTING

What's interesting about this service is that it can be abused by malicious programs (malware) for performing various tasks with high privileges and that's quite important considering that Steam is one of the most diffused software available.

VULNERABLE VERSIONS

Steam package versions: 1404163764

Steamservice.*: 2.30.30.94

NON-VULNERABLE VERSIONS

None.

As a personal project, this document has been released publicly without contacting Valve.

HOW STEAMSERVICE WORKS

How Steamservice works

The service uses an IPC[9] interface for communicating with the Steam process, the access to the interface is performed using events and shared memory. Named pipes were used in past versions.

Exist many ways[10] to perform IPC and the following are the current steps for starting to communicate with the SteamService:

- create a Global\Valve_SteamIPC_Class event
- create a Steam3Master_SharedMemFile mapped file
- create a Steam3Master_SharedMemLock event
- launch the service, any user without privileges can do it
- open the Global\SteamClientService_SharedMemLock event
- open the Global\SteamClientService_SharedMemFile mapped file
- take the handles from the structure located on the mapped file

Note that such steps are necessary only if we use a stand-alone tool to access SteamService, so if Steam is already running we can inject our code in its process or we can just kill it and use the IPC or replace some Steam libraries and so on.

The service will verify that our process has the steam.exe name and that its own steamservice.dll shared library[11] is correctly signed by Valve.

If steamservice.dll doesn't have a signature or it's signed with a different certificate, the service will terminate immediately.

[9] https://en.wikipedia.org/wiki/Inter-process_communication
[10] http://msdn.microsoft.com/en-us/library/windows/desktop/aa365574(v=vs.85).aspx
[11] C:\Program Files (x86)\Steam\bin\steamservice.dll

NO SIGNATURE VERIFICATION – DLL HIJACKING

No signature verification – DLL hijacking

THE ISSUE

The verification of the signature of steamservice.dll is performed for security reasons because the folder where is located the service executable cannot be modified by the user but the Steam folder used by the dll is fully writable and cannot be trusted.

But this check is completely useless because steamservice.dll depends by other libraries located in the Steam folder that are not verified at all and can be replaced by a malware to execute its code inside the service with SYSTEM privileges.

The following are the libraries that can be used by the malware:

- crashhandler.dll
- dbghelp.dll
- tier0_s.dll
- vstdlib_s.dll
- dnsapi.dll (Windows dll searched in the local folder)
- version.dll (Windows dll searched in the local folder)
- winmm.dll (Windows dll searched in the local folder)

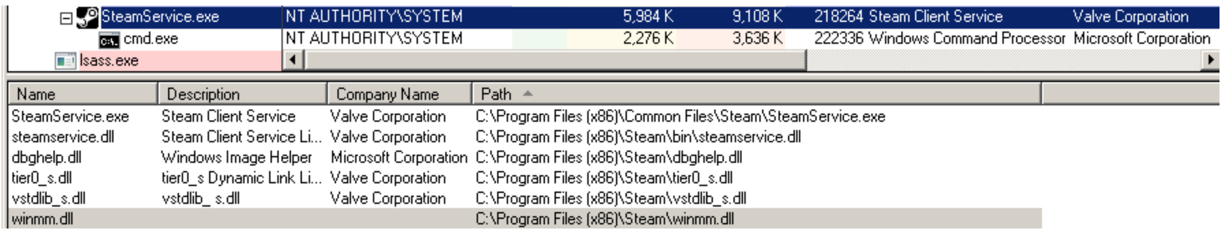
DLL hijacking is an issue which is quite common and it's perfect to escalate privileges to SYSTEM, at this point a malware requires nothing else for its job.

TESTING

Testing this behaviour is very simple:

- terminate Steam
- create your own dll
- copy your dll in the Steam folder with the name winmm.dll
- start the service: sc start "Steam Client Service"

Example of custom winmm.dll executing cmd.exe as SYSTEM:



```
SteamService.exe   NT AUTHORITY\SYSTEM   5,984 K   9,108 K   218264 Steam Client Service          Valve Corporation
  cmd.exe          NT AUTHORITY\SYSTEM   2,276 K   3,636 K   222336 Windows Command Processor     Microsoft Corporation
lsass.exe
```

Name	Description	Company Name	Path
SteamService.exe	Steam Client Service	Valve Corporation	C:\Program Files (x86)\Common Files\Steam\SteamService.exe
steamservice.dll	Steam Client Service Li...	Valve Corporation	C:\Program Files (x86)\Steam\bin\steamservice.dll
dbghelp.dll	Windows Image Helper	Microsoft Corporation	C:\Program Files (x86)\Steam\dbghelp.dll
tier0_s.dll	tier0_s Dynamic Link Li...	Valve Corporation	C:\Program Files (x86)\Steam\tier0_s.dll
vstdlib_s.dll	vstdlib_ s.dll	Valve Corporation	C:\Program Files (x86)\Steam\vstdlib_s.dll
winmm.dll			C:\Program Files (x86)\Steam\winmm.dll

ABUSING THE SERVICE FOR PRIVILEGED TASKS

Abusing the service for privileged tasks

OUR GOAL

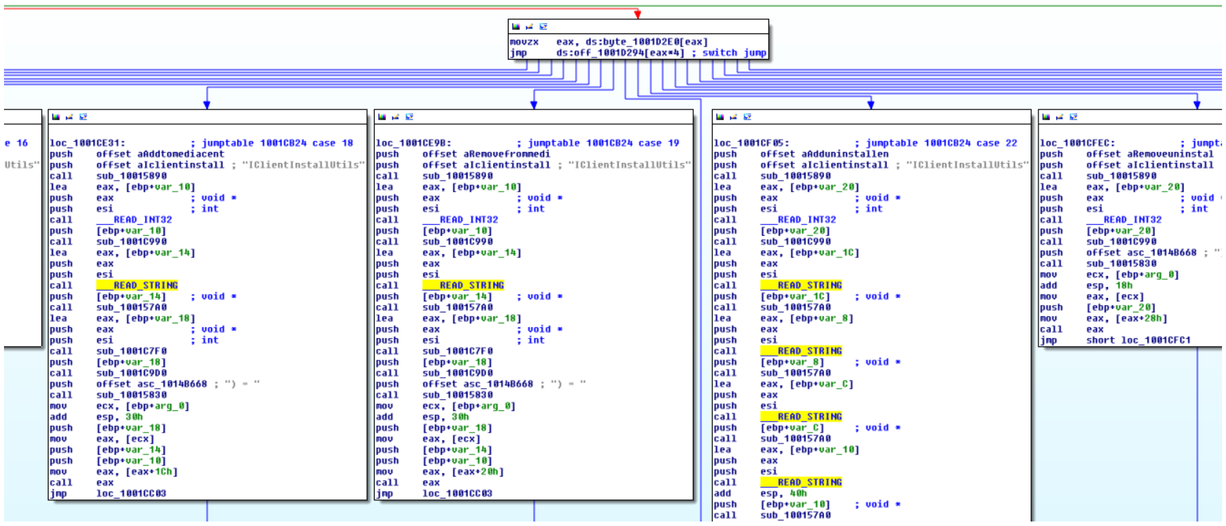
Even if it’s easy to execute the own code as SYSTEM with the previous design issue, this paper would like to focus on how it's possible to abuse a legitimate service without using security vulnerabilities and design issues. So let's take a look at what is possible to do with its features in a scenario in which DLL hijacking is not possible or should be avoided.

THE IPC INTERFACE

The IPC protocol is composed by some commands, the following are the functions of the main one:

- `IClientInstallUtils::SetUniverse`
- `IClientInstallUtils::AddShortcut`
- `IClientInstallUtils::RemoveShortcut`
- `IClientInstallUtils::RemoveFromGameExplorer`
- `IClientInstallUtils::AddRichSavedGames`
- `IClientInstallUtils::RemoveRichSavedGames`
- `IClientInstallUtils::AddToMediaCenter`
- `IClientInstallUtils::RemoveFromMediaCenter`
- `IClientInstallUtils::AddUninstallEntry`
- `IClientInstallUtils::RemoveUninstallEntry`
- `IClientInstallUtils::AddToFirewall`
- `IClientInstallUtils::RemoveFromFirewall`
- `IClientInstallUtils::RegisterSteamProtocolHandler`
- `IClientInstallUtils::FixupSteamClientShortcuts`
- `IClientInstallUtils::RunInstallScript`
- `IClientInstallUtils::AddInstallScriptToWhiteList`
- `IClientInstallUtils::GetInstallScriptExitCode`
- `IClientModuleManager::LoadModule`
- `IClientModuleManager::UnloadModule`
- `IClientModuleManager::CallFunctionAsync`
- `IClientModuleManager::CallFunction`
- `IClientModuleManager::PollResponseAsync`
- `IClientProcessMonitor::RegisterProcess`
- `IClientProcessMonitor::UnregisterProcess`
- `IClientProcessMonitor::TerminateProcess`
- `IRegistryInterface::BGetValueUint`
- `IRegistryInterface::BSetValueBin`
- `IRegistryInterface::BDeleteValue`
- `IRegistryInterface::BDeleteKey`
- `IRegistryInterface::BKeyExists`
- `IRegistryInterface::BSetValueStr`
- `IRegistryInterface::BSetValueUint`
- `IRegistryInterface::BGetSubKeys`
- `IRegistryInterface::BGetValues`
- `IRegistryInterface::BEnumerateKey`
- `IRegistryInterface::BGetValueStr`
- `IRegistryInterface::BGetValueBin`
- `IRegistryInterface::BenumerateValue`

ABUSING THE SERVICE FOR PRIVILEGED TASKS



```
movzx   eax, ds:byte_1001D2E0[eax]
jmp     ds:off_1001D294[eax*4] ; switch jump
```

```
e 16
Utils"
```

```
loc_1001CE31:            ; jumptable 1001CB24 case 18
push    offset aAddtomediacent
push    offset aIclientinstall ; "IClientInstallUtils"
call    sub_10015890
lea     eax, [ebp+var_10]
push    eax              ; void *
push    esi              ; int
call    ___READ_INT32
push    [ebp+var_10]
call    sub_1001C990
lea     eax, [ebp+var_14]
push    eax
push    esi
call    ___READ_STRING
push    [ebp+var_14]     ; void *
call    sub_100157A0
lea     eax, [ebp+var_18]
push    eax              ; void *
push    esi              ; int
call    sub_1001C7F0
push    [ebp+var_18]
call    sub_1001C9D0
push    offset asc_1014B668 ; ") = "
call    sub_10015830
mov     ecx, [ebp+arg_0]
add     esp, 30h
push    [ebp+var_18]
mov     eax, [ecx]
push    [ebp+var_14]
push    [ebp+var_10]
mov     eax, [eax+1Ch]
call    eax
jmp     loc_1001CC03
```

```
loc_1001CE9B:            ; jumptable 1001CB24 case 19
push    offset aRemovefrommedi
push    offset aIclientinstall ; "IClientInstallUtils"
call    sub_10015890
lea     eax, [ebp+var_10]
push    eax              ; void *
push    esi              ; int
call    ___READ_INT32
push    [ebp+var_10]
call    sub_1001C990
lea     eax, [ebp+var_14]
push    eax
push    esi
call    ___READ_STRING
push    [ebp+var_14]     ; void *
call    sub_100157A0
lea     eax, [ebp+var_18]
push    eax              ; void *
push    esi              ; int
call    sub_1001C7F0
push    [ebp+var_18]
call    sub_1001C9D0
push    offset asc_1014B668 ; ") = "
call    sub_10015830
mov     ecx, [ebp+arg_0]
add     esp, 30h
push    [ebp+var_18]
mov     eax, [ecx]
push    [ebp+var_14]
push    [ebp+var_10]
mov     eax, [eax+20h]
call    eax
jmp     loc_1001CC03
```

```
loc_1001CF05:            ; jumptable 1001CB24 case 22
push    offset aAdduninstallen
push    offset aIclientinstall ; "IClientInstallUtils"
call    sub_10015890
lea     eax, [ebp+var_20]
push    eax              ; void *
push    esi              ; int
call    ___READ_INT32
push    [ebp+var_20]
call    sub_1001C990
lea     eax, [ebp+var_1C]
push    eax
push    esi
call    ___READ_STRING
push    [ebp+var_1C]     ; void *
call    sub_100157A0
lea     eax, [ebp+var_8]
push    eax
push    esi
call    ___READ_STRING
push    [ebp+var_8]      ; void *
call    sub_100157A0
lea     eax, [ebp+var_C]
push    eax
push    esi
call    ___READ_STRING
push    [ebp+var_C]      ; void *
call    sub_100157A0
lea     eax, [ebp+var_10]
push    eax
push    esi
call    ___READ_STRING
add     esp, 40h
push    [ebp+var_10]     ; void *
call    sub_100157A0
```

```
loc_1001CFEC:            ; jumpt
push    offset aRemoveuninstal
push    offset aIclientinstall
call    sub_10015890
lea     eax, [ebp+var_20]
push    eax              ; void
push    esi              ; int
call    ___READ_INT32
push    [ebp+var_20]
call    sub_1001C990
push    offset asc_1014B668 ; "
call    sub_10015830
mov     ecx, [ebp+arg_0]
add     esp, 18h
mov     eax, [ecx]
push    [ebp+var_20]
mov     eax, [eax+28h]
call    eax
jmp     short loc_1001CFC1
```

THE EXPLOITABLE COMMANDS

AddShortcut

It's used for placing a “link” file anywhere we desire, it’s commonly used by games for placing their links on the Desktop of all the users.

If there is an Administrator account in the system, it's possible to create a link to our malware in the Startup folder to automatically execute it with such privileges:

- before Vista
  c:\Documents and Settings\Administrator\Start Menu\Programs\Startup\evil.lnk
- Vista/Win7/8
  c:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\evil.lnk

When we call the command RemoveShortcut the extension of the link file must be ".lnk" but with AddShortcut we have no restrictions and so we can also use it to overwrite any file on the system deleting the original content.

AddToFirewall

Function used for adding a firewall rule to allow a specific program to receive incoming TCP connections and UDP packets, a solution for giving network access to a malware.

TerminateProcess

It terminates any desired process by specifying its PID.

It can be used for terminating the privileged processes of various defensive solutions, for example before downloading the malicious code that may be identified by them or to avoid logging and so on.

ABUSING THE SERVICE FOR PRIVILEGED TASKS

If we kill the lsass.exe process we will force the system to reboot, useful if we make certain changes to the registry and the filesystem.

BSetValue*

Functions that write data in the registry but are limited by a set of whitelisted registry locations, so the service can write only under the registry keys listed in the file registrykeys.vdf.

This file is located in the Steam folder and so it's writable but it contains a digital signature "kvsignatures" at its end, this signature is verified by the service.

Even if there is such limitation it's still possible to have room for executing the own code with higher privileges when it's performed the manual uninstalling of software and in some cases during their updating.

One of the whitelisted keys is "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall" so we are able to set WindowsInstaller to zero and the UninstallString value with the path of our malware, and we can perform this operation for all the available registry entries with the effect of executing the malware automatically when one of them is uninstalled.

RunInstallScript

This command is used together with *AddInstallScriptToWhiteList* for automatically executing executables decided by the installscript.vdf script files of the games, for example to install the necessary pre-requisites like DirectX, VC runtime and so on[12].

AddInstallScriptToWhiteList adds the executables listed in the script, usually called installscript.vdf, into the whitelist and *RunInstallScript* executes them only if the same locations are used also in the runasadmin.vdf script.

The installation scripts have a digital signature and currently doesn't seem possible to execute scripts that don't have this field, but an attacker can use an already signed legit script to execute his own code located in any folder he desires.

In fact if we specify an unexistent game ID, Steam will consider the current folder as "installation folder" of the game allowing us to launch batch and executables located in any local and remote webdav/shared folder we desire.

The executables will have the same privileges of the Steam Service, SYSTEM.

[12] The interesting aspect is that these IPC commands are used by Steam only when it runs in "Big Picture" mode, instead normally the Steam application launches the steamservice.exe executable (so doesn't interact with the service) using the /installscript argument and the user must confirm the privileged operation.

POSSIBLE STEPS FOR AN ATTACKER

Possible steps for an attacker

For the design issue related to the unverified libraries in the Steam folder it's enough to put our malware in the Steam folder with a specific filename and we will have SYSTEM privileges immediately after we start the "Steam Client Service".

If this issue will be fixed or limited in future and we must rely on the features of the service, we may think to the following steps (remember that they are performed ever with SYSTEM privileges):

- execute programs
- use TerminateProcess to kill some antivirus processes, please note that this is not enough for defensive solutions that work via drivers[13]
- download the core of the malware
- use AddToFirewall to add a firewall rule for the malware
- check if there is an Administrator account and use AddShortcut to put a link to the malware in its StartUp folder to automatically execute it when he will login
- use the BSetValue* functions to perform the Uninstall trick and being able to execute the malware when a software will be uninstalled by the user

Other features of the service may be abused, those are just the main ones.

[13] http://www.adlice.com/making-an-antivirus-engine-the-guidelines/

THE PROOF-OF-CONCEPT

The proof-of-concept

A proof-of-concept tool has been created to test the IPC interface, the following are the links for the source code and the binary:

- http://revuln.com/files/steam_service_poc_src.zip
- http://aluigi.org/poc/steam_service_poc.zip

It's a simple command-line tool called *Steam.exe* (necessary) that allows to call the service functions listed above and passing them any argument you desire, for example:

- Steam.exe AddToFirewall "c:\windows\notepad.exe" "malicious_firewall_rule"
- Steam.exe TerminateProcess 12345678

The tool works also with a text file containing the list of commands to execute, in the provided package there is an example called *example_commands.ini* which contains some example commands to test but please don't use it if you have no idea of what it does, example:

- Steam.exe example_commands.ini

There is also an archive called *example_execution.zip* which contains all the necessary files to execute a custom executable with SYSTEM privileges using the RunInstallScript function. You need to create a temporary partition or ramdisk on *Z:*\ for the quick test or you have to edit both the text files *example_commands.ini* and *RunAsAdmin.vdf* replacing z:\ with the full path you would like to use.

Then launch "Steam.exe example_commands.ini" and you should see calc.exe spawning through a batch script, so you can easily edit the "Microsoft .NET Framework 4.0.cmd" script to execute what you desire.

Please note that the IPC mechanism used in the tool is not complete and so the Steam Client Service process will remained freezed and you MUST kill it manually. This tool must be considered only a proof-of-concept.

FAQ

Is this issue a security vulnerability?

There is a design issue consisting in the missing verification of the signatures of the shared libraries loaded by the Steam Service that allows dll hijacking.

Additionally the service can be abused to perform various operations and launching executables as SYSTEM without using security vulnerabilities.

Can the service be accessed from remote?

No.

What operating systems are affected?

The issue is not dependent by the system, anyway both Windows 7 and Windows 8 have been tested.

On the operating system before Vista, like Windows XP, the Steam Client Service was not strictly necessary but Steam needs to have Administrator privileges in that case.

What's the main scenario for this issue?

It's a typical post-exploitation scenario, for example a malware/exploit running with the privileges of the current limited user can abuse the Steam Service for gaining SYSTEM privileges or performing some operations with such high privileges with the target of hiding itself and remaining persistent in the system.

What can be done by a malware after it gets SYSTEM privileges?

It's possible to have full access to the Windows registry and the disk allowing the malware to hide itself and disabling any security software just like the Administrator account.

Is this issue critical?

No, but the huge diffusion of Steam and the possibility of becoming SYSTEM without security vulnerabilities or complex exploits and executing certain operations through a trusted service is quite interesting for a malware.

FAQ

Do I need to have Steam or the Steam Client Service running on my computer for being vulnerable?

No, it's enough that Steam has been installed, doesn't matter when and if it's used or not.

What about SteamMachine and the Linux/MacOSX/PlayStation3 versions of Steam?

Sorry, they have not been tested.

How can Steam rebuild the registrykeys.vdf file?

When Steam is launched it automatically verifies all the local files and restores them by taking the original copies from the lzma compressed ZIP files located in the *Package* folder, that's why *registrykeys.vdf* is restored (and not rebuilt) by Steam everytime it gets modified.

Steam doesn't contain any private key or certificate.

Why SteamService.exe is located in both Steam\bin and Common Files\Steam?

The service is located in *"C:\Program Files (x86)\Common Files\Steam"* which is not writable by the normal limited user and it's the exact copy of the one located in the bin folder of Steam. When the service runs it checks if there is a different *steamservice.exe* file in the bin folder (which is writable by the user), then checks the digital signature of that file and then executes it with the */Update* argument. The update is executed with the same privileges of the service (SYSTEM), that's why the new executable will be copied in the *"Common Files\Steam"* folder without user interaction, first as *SteamServiceTmp.exe* and then overwriting the original *SteamService.exe*.

That's how the update process of the service works.

Have you reported these issues to Valve?

No.

HISTORY

History

Mar 2013	initial research on this topic
01 Jun 2014	returned on the research to confirm the issues
10 Jul 2014	public release

COMPANY INFORMATION

Company Information

ReVuln Ltd.
Level 3, Theuma House, 302, St.Paul Street,
Valletta VLT1213
Malta
http://revuln.com
@revuln
info@revuln.com