STEAM BROWSER PROTOCOL INSECURITY

(WHEN LOCAL BUGS GO REMOTE)

Luigi Auriemma[1] and Donato Ferrante[2]
ReVuln
http://revuln.com
info@revuln.com
http://twitter.com/revuln
15 October 2012

Abstract *In this paper we will uncover and demonstrate a novel and interesting way to convert local bugs and features in remotely exploitable security vulnerabilities by using the well known Steam[3] platform as attack vector against remote systems.*

1 STEAM

From Wikipedia: "Steam is a digital distribution, digital rights management, multiplayer and communications platform developed by Valve Corporation".
With Steam users can buy games, download demos and free-to-play games, find multiplayer matches, communicate with other users, share stats and so on. Steam is the digital delivery platform having the biggest user base (approximately 50 millions users) and supporting several platforms: Windows, MacOS, PS3, mobile devices and Linux.

2 STEAM BROWSER PROTOCOL

Steam, like other software, uses its own URL handler to enhance experience by integrating web-based functionality directly in its own platform.

Steam uses the *steam://* URL protocol in order to:

- Install and uninstall games
- Backup, validate and defrag game files
- Connect to game servers
- Run games
- Reach various pages and sections where it's possible to buy or activate games, download tools, read news, check user profiles and so on

The *Steam Browser Protocol* has several commands, most of them are listed on Valve[4] website along with a non-updated list[5] of games using Steam as platform. The list of commands is not a complete reference of all the commands available

[1] http://twitter.com/luigi_auriemma
[2] http://twitter.com/dntbug
[3] http://steampowered.com
[4] https://developer.valvesoftware.com/wiki/Steam_browser_protocol
[5] https://developer.valvesoftware.com/wiki/Steam_Application_ID

with the *Steam Browser Protocol*, as several commands are partially documented or not documented at all on Valve website.

In the next sections we are going to cover how Steam URLs are handled by web browsers and other software, in order to get a good understanding of the possible ways in which it's possible to trigger remote attacks via such URLs. Figure 1 gives an overview of one of the possible attack scenario.

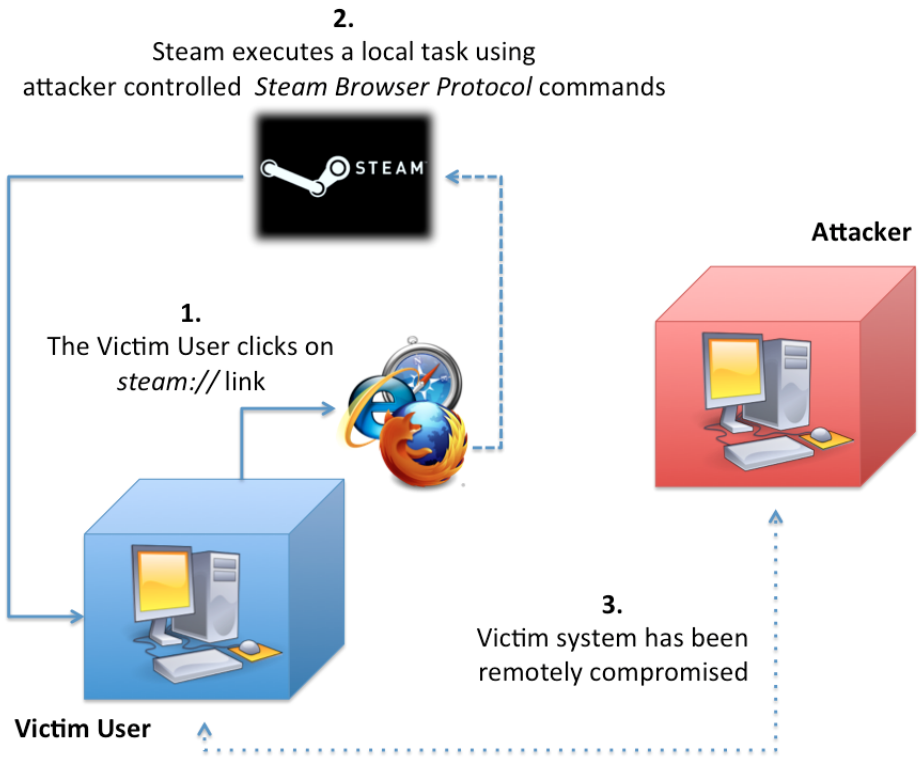

Figure 1: Remote Steam Protocol Commands exploitation: overview

2.1 Strategy 1: Web Browsers

First we checked all of the most known web browsers in order to verify how they react while handling external (not handled by the browser itself) URL protocols (i.e. *rtsp://*, *mms://*, *steam://* and so on).

According to the results reported in Table 1 all the browsers that execute external URL handlers directly without warnings and those based on the Mozilla engine (like Firefox and SeaMonkey) are a perfect vector to perform silent *Steam Browser Protocol* calls. Additionally for browsers like Internet Explorer and Opera it's still possible to hide the dodgy part of the URL from being showed in the warning message by adding several spaces into the *steam://* URL itself.

Internet Explorer	Warning including the URL, and in case of IE9 a possible additional warning ("protected mode") without any detail
Firefox	No URL visualized, only request for confirmation (no warnings)
Chrome	Warning with a detailed description or the URL and the program to call
Opera	Warning with only 40 chars of the URL visualized
Safari	Direct execution without warnings
Webkit	Direct execution without warnings
MaxThon	Direct execution without warnings
Avant	Direct execution without warnings
Lunascape	Direct execution without warnings
SeaMonkey	See Firefox
PaleMoon	See Firefox
SRWare Iron	See Chrome

Table 1: Web browser and Steam protocol survey

2.2 Strategy 2: Alternatives

Apart from web browsers, there is additional software that may be used to perform calls to external protocol handlers. Most of them rely on the default browser but some of them don't.

The following are some of the software (tested during our research) that doesn't show any warnings while performing external URL protocol calls:

- Steam browser (Steam's custom web browser)
- RealPlayer embedded browser
- Other software able to process html pages

In our opinion the Steam browser is a very interesting alternative to the common browsers, except for the following limitations:

- The websites one can visit from within the Steam browser are generally limited to locations owned by Valve like *steampowered.com* and *steamcommunity.com* domains
- Valve prevents *steam://* protocol injections by performing several checks on users provided links
- References to external websites get redirected via *steam://openurl/website* calls, which rely on the default browser instead of the Steam one

For the sake of completeness, it's worth mentioning that the web browser used in the in-game Overlay Interface of Steam acts differently and it allows all the websites, except *steam://* links get ignored completely, so this browser flavor can't be used as vector. As you may have argued, we want to be able to open links that we are not supposed to open by using the Steam browser.

If you are familiar with Steam, you know that every user gets a personal profile page and on this page it's possible to include information like pictures and videos. While pictures provided by the users get uploaded on Steam, videos are just links to YouTube videos. If a user tries to view a video attached to a profile, the user will get a page in which there is only the video, so no comments or description coming from YouTube. But if the user clicks on the title of the video (i.e. to leave comments on the YouTube video) then a new window is opened with all the details about the video including comments and description. So a malicious user can include links to external hosts, which can remotely invoke Steam commands by using the usual *steam://* URLs. With this strategy the Steam browser will execute the protocol handler calls without any warnings. Please see Figure 2 for a graphical recap of this approach.

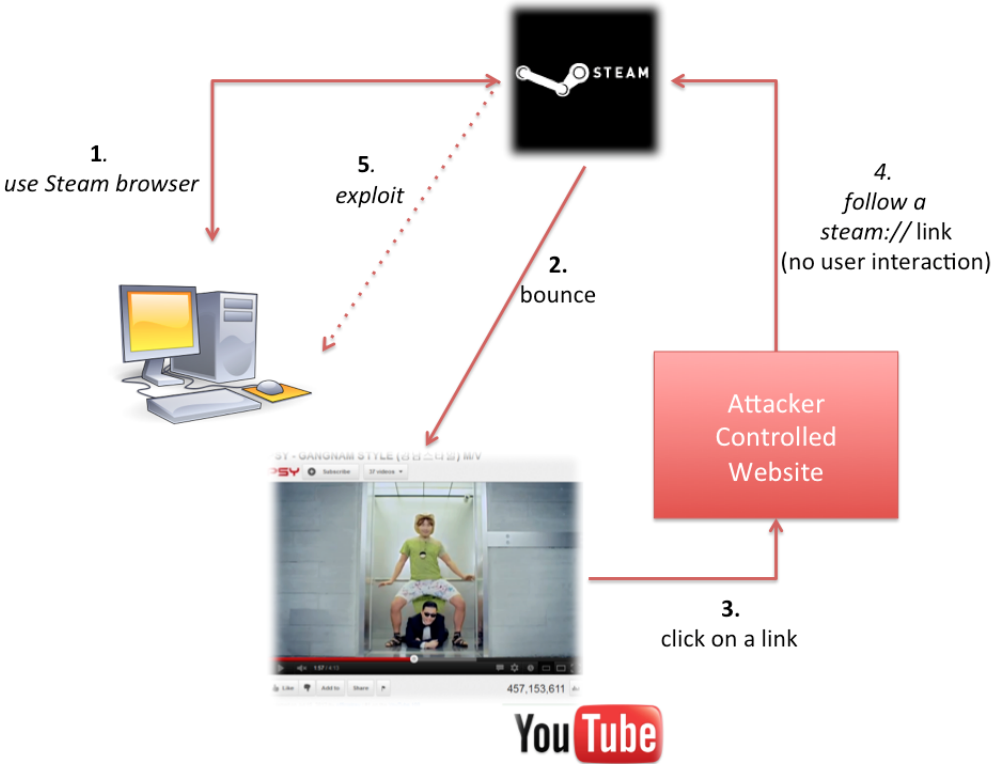

Figure 2: Remote exploitation via Steam browser and YouTube bouncing

3 Insecurity Time

At this point we know the pros and cons of various ways to launch *steam://* URLs, so we can start exploring some security vulnerabilities and what we can achieve with them. In the following sections we will report some new vulnerabilties we found during this research, please note that all of them are exploitable remotely by simply using the *Steam Browser Protocol* as trigger.

3.1 Steam Browser Protocol Commands

The *retailinstall* command is an undocumented feature (not a bug) of the *Steam Browser Protocol* that allows installing and restoring backups from a local directory. One of its parameter is *path* that is used to specify this local directory but obviously this directory can be a Windows network folder available on a remote host. When Steam executes the *retailinstall* command, Steam checks and loads two files: *splash.tga* (an image) and *sku.sis* (an install file). The splash image gets displayed immediately (Figure 3) to the user as soon as the command gets executed.

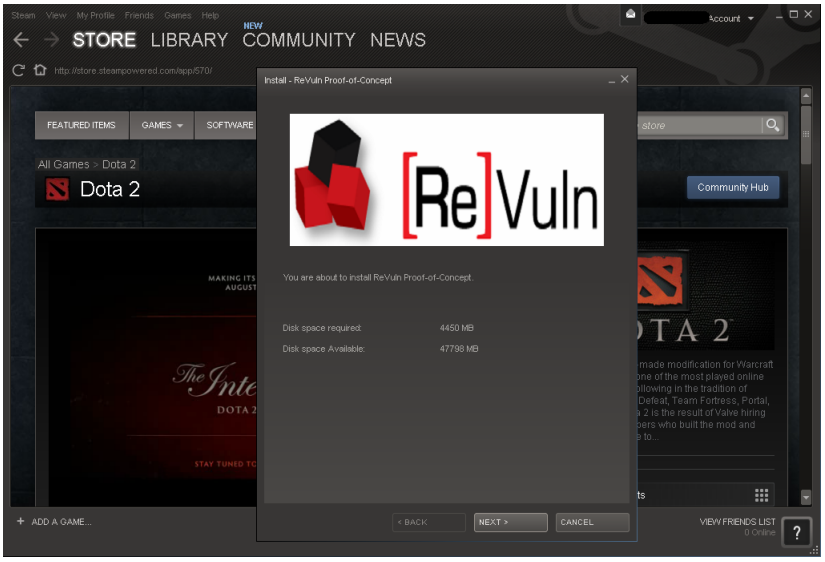

Figure 3: Steam loading *splash.tga* while executing the *retailinstall* command

The Steam function in charge of processing the splash images is vulnerable to an integer overflow vulnerability while processing malformed TGA files. The problem is located in *LoadTGA* function of *vgui2_s.dll* that loads TGA files in memory (Figure 4). The result is a heap-based buffer-overflow that may allow executing malicious code on the Steam process.

```
73EB0E65  .  85C9          TEST ECX,ECX
73EB0E67  .v 74 0B         JZ SHORT 73EB0E74
73EB0E69  .  8BD7          MOV EDX,EDI
73EB0E6B  .  0FAFD3        IMUL EDX,EBX
73EB0E6E  .  03D2          ADD EDX,EDX
73EB0E70  .  03D2          ADD EDX,EDX
73EB0E72  .  8911          MOV DWORD PTR DS:[ECX],EDX
73EB0E74  >  03C0          ADD EAX,EAX
73EB0E76  .  03C0          ADD EAX,EAX
73EB0E78  .  50            PUSH EAX                          [Arg1, width * height * 4
73EB0E79  .  E8 0204FEFF   CALL 73E91280                     vgui2_s.73E91280, malloc
```

Figure 4: ASM code related to the integer overflow condition in *LoadTGA*

3.2 Steam Browser Protocol Game Command-line Parameters

In Steam it's possible to launch installed games by using one of the following *steam://* commands:

- *run*

- *rungameid*
- *runsafe*
- *rungame*

As with most of the software, the games available on Steam accept command line arguments. Steam allows you to pass such arguments to games but there is no official documentation about any strategy to do that, except for the *-applaunch* command that can't be used in a universal and silent way, because of different URL encoding strategies used by web browsers.

Most of the four commands that can be used to run games via Steam URLs are undocumented, anyway the following are their formats:

- *steam://run/id/language/url_encoded_parameters*
- *steam://rungameid/id/language_bug/url_encoded_parameters*
- *steam://runsafe/id*
- *steam://rungame/id/lobby_id/parameters*

The only commands suitable for remote environments are *run* and *rungameid* where *url_encoded_parameters* is an URL encoded string passed to the *Q_URLDecode* function that stores the decoded result in a buffer of *128* bytes. The *Q_URLDecode* function allows you to use any character and also demonstrates that there are some commands designed to be used remotely via browser. The limitation of *128* chars for the parameters doesn't affect exploitation of any of the following bugs, because if we need more room we can just use some JavaScript to join chunks of commands.

3.2.1 Game Exploitation 1: Source Engine

As first example of game exploitation via Steam we have chosen the game engine with the biggest user base: Source[6] .

The following are the most known games based on such engine: Half-Life 2, Counter-Strike: Source, Half-Life: Source, Day of Defeat: Source, Team Fortress 2, Portal 2, Left 4 Dead 2, Dota 2, Alien Swarm, SiN Episodes, Dark Messiah of Might and Magic, The Ship, Zombie Panic! Source, Age of Chivalry, Synergy, D.I.P.R.I.P., Eternal Silence, Pirates Vikings & Knights II, Dystopia, Insurgency, Nuclear Dawn and Smashball.

Most of them include the basis commands[7] available in the Source engine, which we are going to use for writing files with custom content in arbitrary locations. For exploiting this engine we have opted for the following command-line options:

- *+con_logfile*, allows you to specify a file that will receive the content of the console (it can't be a Windows remote share)

[6] `http://source.valvesoftware.com`
[7] `https://developer.valvesoftware.com/wiki/Command_Line_Options`

- *+echo*, used to put custom data in the log file
- *+quit*, (optional) closes the game
- *-hijack*, (optional) useful in case the user already has an instance of the game running and we want to send additional commands that are limited by the *Q_URLDecode 128* chars

Our choice for exploiting this bug is to create a *.bat file* in the *Startup folder* of the user account which will execute our commands injected through *+echo* at the next login of the user on the system. There is also an interesting scenario against dedicated servers by specifying the *motd.txt* of the game as logfile and launching the *cvarlist* command that will dump all the game variables in such file that is visible to any player who joins the server. Team Fortress 2[8] is one of the most played games based on this engine and it's free-to-play.

3.2.2 Game Exploitation 2: Unreal Engine

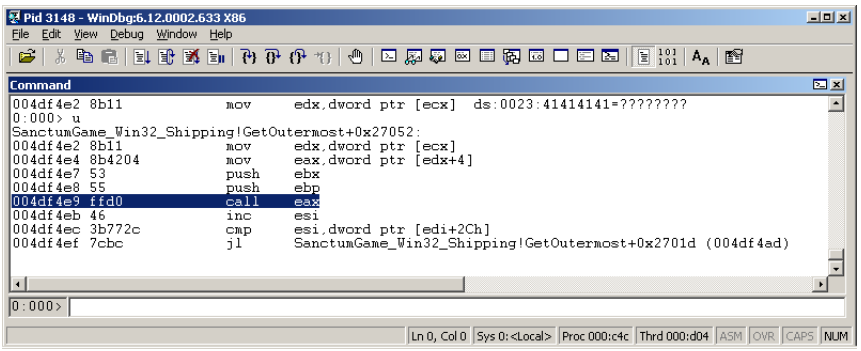

Figure 5: Remote exploitation via Steam of Unreal Engine

For games based on the Unreal Engine[9] we opted for exploiting a real security vulnerability that occurs while loading content that resides on remote computers (Windows remote WebDAV or SMB share) which we can load via command-line parameters:

steam://run/ID/server \\HOST\evil.upk -silent

Indeed this engine is affected by many integer overflow vulnerabilities (maybe we will document them one of these days) that allow execution of malicious code.

A full list of command-line parameters available for the Unreal Engine 3 is available online[10].

3.2.3 Game Exploitation 3: APB Reloaded

All Points Bulletin[11] is a well known Massive Multiplayer Online (MMO) game that includes a customizable auto-update feature. In this case we decide an arbitrary

[8] http://www.teamfortress.com
[9] http://www.unrealengine.com
[10] http://udn.epicgames.com/Three/CommandLineArguments.html
[11] http://www.gamersfirst.com/apb/

update server via command-line and exploit a directory traversal for overwriting or creating any file we desire with our custom content.

On Steam there are tons of MMO games free-to-play like APB so the user base is very big and most of them can be exploited with such techniques. Additionally most of these games use anti-cheating solutions and require to be launched with Administrator permissions (we are in the gaming world where people don't have security knowledge, having such privileges is quite common) so the whole system can be compromised.

3.2.4 Game Exploitation 4: MicroVolts

MicroVolts[12] is another example of known MMO game exploitable via auto-update, just another directory traversal vulnerability.

3.3 Proof-Of-Concept

Please refer to vimeo.com/revuln[13] channel, for a proof-of concept video[14], illustrating all the issues reported in this paper.

4 Possible Fix And Workaround

In this section we propose some solutions to avoid or reduce the impact of the the issues we found during our research.

4.1 User-side

The issue can be limited by disabling the *steam://* URL handler or using a browser that doesn't allow the direct execution of the *Steam Browser Protocol*.

4.2 Steam-side

A solution would be avoiding to pass command-line arguments to third party software and undocumented commands accessible from external and untrusted sources like the Internet.

4.3 Game-side

The main problem of the *Steam Browser Protocol* is that it allows abusing local features of games (like using log files) so the developers can't do much in this situation, except trying to reduce possible related issues, by:

- Adopting secure programming techniques also in non-network related code
- Using certificates validation while performing auto-patching

5 Conclusion

In this paper we proved that the current implementation of the *Steam Browser Protocol* handling mechanism is an excellent attack vector, which enables attackers to exploit local issues in a remote fashion. We also detailed as proof of the effectiveness of this new attack vector, five new remotely exploitable issues, including one

[12] `http://www.microvolts.com`
[13] `http://vimeo.com/revuln`
[14] `http://vimeo.com/51438866`

in Steam, and two in widely used game engines (Source and Unreal). Because of the big audience (more than 50 million people), the support for several different platforms including Windows, MacOs and Linux and the amount of effort required to exploit bugs via *Steam Browser Protocol* commands, Steam can be considered a high-impact attack vector.

6 FAQ

- Is this a Windows-only issue?
  - No. If you can install and run Steam on your OS then you are vulnerable.
- Is this a browser-only issue?
  - No. Anything that is able to process common URL links can be used as a trigger.
- Is this a Safari-only issue?
  - No. Safari is just one of the possible triggers.
- Will browsers always show a warning/popup to the user?
  - No. Users can suppress the warning for *steam://* links by using the browser settings. This is quite common for gamers that use the Steam protocol to join online game servers.
- Are the issues related only to the four games you listed?
  - No. Games usually share the same engine (i.e. Source Engine, Unreal Engine, and so on), so an engine related bug affects several games.
- Did you test all the games available on Steam?
  - No. Our only purpose was to detail the *Steam Browser Protocol* issues. Moreover with our examples we covered two different engines (Source and Unreal) and two well known MMOs.
- Is the *retailinstall* issue remotely exploitable?
  - Yes.
- Does the victim user have to click on a malicious *steam://* link?
  - No. In fact all the links used in our PoC video point to normal HTML pages.
- Does the victim user always see the real link shown in the browser status bar?
  - No.
- Are users using only the Steam browser safe?
  - No. As demonstrated in the YouTube bouncing scenario.

- I can't replicate some of the bugs shown in the video. Why?
  - Because after our public paper, Valve has just limited the *con_logfile* command[15] and APB has just removed a legacy command[16].

7 Revision History

- 18 October 2012: Version 1.1 released: FAQ section added.
- 15 October 2012: Version 1.0 released.

[15] http://store.steampowered.com/news/9120/

[16] http://www.reddit.com/r/Games/comments/11kib1/steam_security_issue_crosspost_rgaming/c6nrr0i