BLOG // ADVISORIES // JUN 15, 2021

RetroArch for Windows, Versions 1.9.0 - 1.9.4 Advisory

By: Daniel Fulford, Senior Security Associate

Share

ADVISORY SUMMARY

The text-to-speech engine in libretro RetroArch for Windows 1.9.0 - 1.9.4 passes unsanitized input to PowerShell through `platform_win32.c` via the `accessibility_speak_windows` function, which allows attackers who have write access on filesystems that are used by RetroArch to execute code via command injection using specially a crafted file and directory names.

Impact

RetroArch is one of the most trusted emulation front-ends available and has recently become available for download on the Steam platform, making it easily accessible to millions of people worldwide. With the reach of the application across the consumer market, arbitrary code execution presents a very clear and present risk to any host running a vulnerable version.

As the execution of the command can occur simply by the act of reading filenames from a directory for import, malicious commands could be included in file sets meant for import provided in ROM packages or supplemental software or placed in unprivileged folders for later execution. RetroArch does not require a privileged folder for importing content, opening exploitation to anyone with potential access to the host, and will execute as the user running the application.

HIGH RISK LEVEL

Affected Vendor

Libretro	RetroArch for Windows	Versions 1.9.0 - 1.9.4

Product Description

RetroArch is the official front end for libretro, a multi-platform library for emulation. The project's official website is **https://www.libretro.com**. The latest version of the application is 1.9.4, released on May 5, 2021.

Vulnerabilities List:

One vulnerability was identified within the RetroArch for Windows application:

COMMAND INJECTION

The vulnerability is described in the sections below.

Solution

Disable the text-to-speech option in RetroArch until a patch is made available.

VULNERABILITIES

COMMAND INJECTION

The RetroArch for Windows application was affected by a command injection vulnerability that allowed arbitrary commands to be injected into a PowerShell script utilized by the text-to-speech engine. The exploit requires the text-to-speech engine to be enabled. It also requires that a directory or file with a malicious name is read aloud by the text-to-speech engine, commonly done when a user navigates through the menus while the engine is enabled. Successful exploitation can result in full remote code execution under the context of the RetroArch user. If remote file shares are utilized to load content for RetroArch, the vulnerability may be exploited remotely.

CVE ID	Security Risk	Impact	Access Vector
CVE-2021-28927	High	Code execution	Context dependent

The **`accessibility_speak_windows`** function in the **`/frontend/drivers/platform_win32.c`** file handles calls to perform the text-to-speech on any text that needs to be spoken. As shown below, the function invokes an instance of PowerShell and passes in a pre-built script containing the code to perform the text-to-speech as well as the text to be spoken:

```
static bool accessibility_speak_windows(int speed,
      const char* speak_text, int priority)
{
....
   if (USE_POWERSHELL)
   {
      if (strlen(language) > 0)
         snprintf(cmd, sizeof(cmd),
               "powershell.exe -NoProfile -WindowStyle Hidden -Command \"Add-Type -
      else
         snprintf(cmd, sizeof(cmd),
               "powershell.exe -NoProfile -WindowStyle Hidden -Command \"Add-Type -
      if (pi_set)
         terminate_win32_process(g_pi);
      res = create_win32_process(cmd);
      if (!res)
      {
         pi_set = false;
         return true;
      }
...
}
```

The text to be spoken is not sanitized before being injected into the PowerShell command, which leaves it open to injection. To exploit this vulnerability, we first need to be able to control the text that is to be spoken. Since RetroArch perform text-to-speech on menu items, we need to insert a menu item with our exploit code in it. The best way to do that is by adding a malicious file or directory to a location that a user loads ROMs or cores from. When a user scrolls through a list of the files within RetroArch and reaches our malicious filename, it will be passed into the **`accessibility_speak_windows`**function and injected into the PowerShell script. In the below image, Exploit Placeholder represents the directory which will eventually contain our exploit payload:

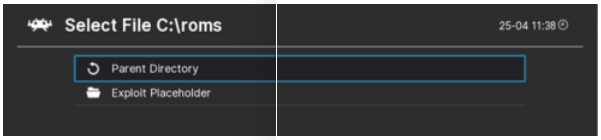


FIGURE 2 - RetroArch file selection screen

Now that we have a way of injecting arbitrary content into the PowerShell script, we must escape from the string. However, if we attempt to create a directory with a quotation character in its name then Windows will reject it. As shown below, Windows restricts the characters that can be included in a file or directory name:

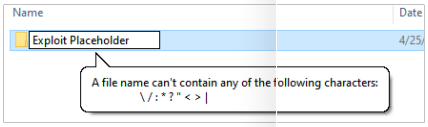


FIGURE 3 – Characters restricted from inclusion in a Windows path

PowerShell supports a subexpression operator in the form of $(expression), which also happens to be evaluated within strings. We can take advantage of this to execute arbitrary expressions within PowerShell strings by simply including it in the string. As an example, if we run the following script then a message box is displayed as soon as the write-out command is executed:

```
write-out "$([System.Windows.MessageBox]::Show('Hello'))"
```

FIGURE 4 *- Sample non-encoded PowerShell 'hello world' payload*

Again however, we are still limited in the characters that we can use in a directory name. Because the colon character is not allowed in file or directory names, it is hard to include full or useful scripts in the filename without encoding them. To get around this limitation, we can simply base64 encode the script that we wish to execute and then pass that as an encoded command into a second instance of PowerShell. It is important to note that PowerShell expects base64 encoded commands to contain UTF-16 strings as opposed to UTF-8. If we want to have PowerShell just print an obligatory Hello World, then our payload would look like the following:

```
$(start-process PowerShell -ArgumentList '-noexit','-enc','dwByAGkAdABlAC0AaABvAHMA
```

FIGURE 5 *- Sample encoded PowerShell payload*

To trigger the exploit, we simply use the Load Content or Load Core options within RetroArch and then scroll the directory containing out payload until we reach it. As soon as the text-to-speech engine attempts to speak the payload, our script will be executed.

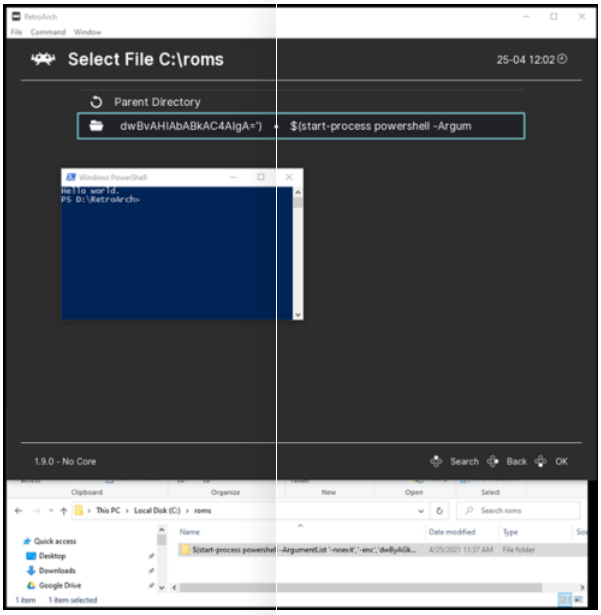


FIGURE 6 - Hello World payload being executed by RetroArch

While the example is created locally for demonstration purposes, RetroArch users commonly download and load content from zip files as well as network file shares. Malicious actors could utilize both vectors to deliver payloads remotely to users.

Credits

Daniel Fulford, Senior Security Consultant I, Bishop Fox (dfulford@bishopfox.com)
Auner Moncada, Security Consultant II, Bishop Fox (amoncada@bishopfox.com)
Robert Punnett, Senior Security Consultant I, Bishop Fox (rpunnett@bishopfox.com)

Timeline

Initial discovery: 03/17/2021

Contact with vendor: 03/17/2021
Vendor acknowledged receipt vulnerability report: 03/18/2021
Advisory published: 06/15/2021

participates in bug bounty programs and has also led security teams for Fortune 500 companies in the transportation industry.

More by Daniel

RECOMMENDED POSTS

You might be interested in these related posts.

Dec 15, 2022

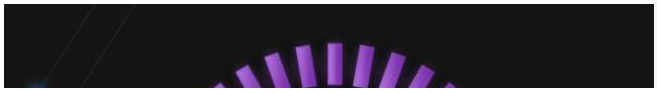
FlowscreenComponents Basepack, Version 3.0.7 Advisory

Nov 21, 2022

Log HTTP Requests, Version 1.3.1, Advisory

Oct 24, 2022

Atlassian Jira Align, Version 10.107.4 Advisory



Jul 13, 2022

Netwrix Auditor Advisory