New issue

Jump to bottom

Null injection hardening #143

Merged

bturner merged 3 commits into `brettwooldridge:master` from `benhumphreys:harden-null-injection` on Sep 20

Conversation 14 Commits 3 Checks 0 Files changed 2

benhumphreys commented on Sep 19 Contributor

In `NuProcessBuilder` throw an `IllegalArgumentException` if commands that include a null character are passed. This hardens against command line injection type attacks in applications that use NuProcess.

The issue this aims to fix is described below. On Linux if you start a process with this list of arguments:

```
NuProcessBuilder pb = new NuProcessBuilder(Arrays.asList("/tmp/main", "--foo", "--bar=\0--bad", "--extra"));
```

One might expect you'd get arguments like this at the forked process (and indeed on MacOS you do, more on that later):

```
argv[0]: /tmp/main
argv[1]: --foo
argv[2]: --bar=
argv[3]: --extra
```

Or perhaps you'd expect an exception is thrown. But instead you get this:

```
argv[0]: /tmp/main
argv[1]: --foo
argv[2]: --bar=
argv[3]: --bad
```

The problem exists in NuProcess since 1.2.0 when it switched to using `Java_java_lang_UNIXProcess_forkAndExec()` and happens because the command array gets turned into a null separated string here: https://github.com/brettwooldridge/NuProcess/blob/master/src/main/java/com/zaxxer/nuprocess/linux/LinuxProcess.java#L141-L147

That is necessary because `Java_java_lang_UNIXProcess_forkAndExec()` takes a string. And then in `Java_java_lang_UNIXProcess_forkAndExec()` it gets converted back to an array/vector, with the extra/injected null obviously being interpreted as a delimiter too:
https://github.com/adoptium/jdk8u/blob/master/jdk/src/solaris/native/java/lang/UNIXProcess_md.c#L608
https://github.com/adoptium/jdk8u/blob/master/jdk/src/solaris/native/java/lang/childproc.c#L166-L176

On MacOS this problem doesn't exist the forked process gets:

```
argv[0]: /tmp/main
argv[1]: --foo
argv[2]: --bar=
argv[3]: --extra
```

This is because MacOS uses `posix_spawnp()` which takes an array, so doesn't have this conversion to a null delimited string. I'm not sure how Windows behaves, I didn't test it.

The solution I used in our NuProcess fork was just to apply the same approach as Java's ProcessBuilder uses: https://github.com/adoptium/jdk8u/blob/master/jdk/src/share/classes/java/lang/ProcessBuilder.java#L1022-L1026

There is some scope to do some deduplication of code in the constructors of `NuProcessBuilder`. But I've just kept it simple for now. Happy to do that deduplication, or any other changes should that be desirable.

`Null injection hardening` … ✓ `d4005b6`

bturner commented on Sep 19 Collaborator

Checking in the constructors for `NuProcessBuilder` won't catch the case where a caller is mutating the list returned by `command()`, which is something the API allows. I would probably forego those checks and, instead, I'd hook up the check in `start` and `run`, where `ensureListener` is called. That will catch `NUL` characters regardless of how they were provided, whether up front via a constructor or after the fact by modifying `command()`. We could double up on the checks, by having the constructors still validate, but in practice I'm not sure that adds anything; it just shifts how early in the builder chain the exception is thrown when the `NUL` characters are present up front.

❤️ 1

`Null injection hardening (Rework)` … ✓ `fabf505`

bturner commented on Sep 19 Collaborator

Looks good. Can you add a couple unit tests somewhere to verify the exception? I'd add two, one where you pass the `NUL` in the constructor call, and another where you pass it by mutating the `command()` list. Perhaps in one of the two you can call `run()` to end the chain and in the other you can call `start()`. (Or you could add 4 if you prefer; I don't think you need to, though.)

`Null injection hardening (Adds tests)` ✓ `467b28a`

benhumphreys commented on Sep 19 Contributor Author

I've added tests. There were probably a few ways I could have written them, including using assertThrows() to verify it is actually the run() or start() call that throws the exception. However in Java 7 without lambdas this obviously requires a bit more boilerplate, so I went with the simple approach. Happy to take feedback and refactor them if there is a preference.

bturner commented on Sep 19 Collaborator

Thanks Ben. I'm planning to merge this and release 2.0.5 with this fix tonight. We're also working on a Github Security Advisory for the issue, which should help notify other projects on Github that use NuProcess (and more broadly) so they can upgrade.

brettwooldridge commented on Sep 20 • edited ▾ Owner

Are we sure this is the only attack vector? For example, semi-colon targeted attacks against shells (to execute additional commands). Maybe if those exist we can handle it as a separate issue?

Or, maybe that is a supported use-case, and there’s not much to be done about it. At some point, there is a responsibility on the user of the library to sanitize inputs.

brettwooldridge commented on Sep 20 Owner

BTW, these changes look fine. Feel free to merge.

bturner commented on Sep 20 Collaborator

> Are we sure this is the only attack vector? For example, semi-colon targeted attacks against shells (to execute additional commands).

In my opinion there's a bit of a difference to those sorts of attacks and the one being fixed here, in that those aren't attacking NuProcess--they're attacking the software that's using NuProcess. NuProcess doesn't run a shell on its own; software using NuProcess can choose to build a shell command line or not. If it does, then I think it's reasonable to expect that software to be sanitizing its inputs to prevent command line injection attacks. At a library level, I don't think it's possible for NuProcess to reliably know whether a semi-colon in the command is valid or not.

bturner merged commit **`29bc09d`** into `brettwooldridge:master` on Sep 20 View details

1 check passed

benhumphreys commented on Sep 20 Contributor Author

> Are we sure this is the only attack vector?

It is a good question, and I have spent some time looking for similar vectors in NuProcess and in our product and couldn't find any. I've also done testing with some lists of sample command injection payloads you can find around the internet and none resulted in NuProcess chopping up the argument list like this, or doing anything else untoward.

As for the question of additional sanitization, and I share the same thoughts as Bryan as to the separation of responsibilities.

brettwooldridge commented on Sep 20 Owner

I agree with of the above in general. And while I don't disagree with the fix in this case (NUL), I also consider that an issue of sanitization rather than a security flaw in the library. This is no different than a SQL injection attack IMHO, and applications should never 'pass thru' user input (API or interactive) unchecked. Particularly in this case, where code execution is involved. The NUL-sanitization could just as easily be performed at the application layer, in front of NuProcess, though as I said above I do not object to this fix as a convenience.

bturner commented on Sep 21 • edited ▾ Collaborator

> And while I don't disagree with the fix in this case (NUL), I also consider that an issue of sanitization rather than a security flaw in the library.

Just for the sake of clarity, I want to elaborate a little on why I view this differently.

Consider `WindowsProcess` and `WindowsCreateProcessEscape.quote` . NuProcess is *not* escaping quotes to prevent them from being propagated to the started process because they might be dangerous; it's escaping them for exactly the *opposite* reason: To ensure they are presented *verbatim* to that process, exactly as they were provided to NuProcess.

To me, that is one of NuProcess's responsibilities as a library: To ensure, when a process is started, that process receives *exactly* the command line the code that called NuProcess provided.

In this case, when the command provided contains `NUL` characters, NuProcess is *not* successfully providing that command line, verbatim, to the started process. Because of the *internal* API for `Java_java_lang_UNIXProcess_forkAndExec`, and the way that handles `NUL` characters, NuProcess ends up starting the process with a different `argv` than what the caller provided to `NuProcessBuilder`.

To further highlight this, I think it's worth noting that NuProcess 1.1.3 and earlier *did not have this problem*. Because those versions used `posix_spawnp` directly, which accepts `argv` as an array rather than a single `String`, they would faithfully recreate the command line as provided no matter the contents. It was only in 1.2.0, with the switch to `Java_java_lang_UNIXProcess_forkAndExec`, that this vulnerability was introduced. Further to that, NuProcess is only vulnerable to this issue, where you can inject new argument using `NUL` bytes, on Linux. On macOS, as Ben showed, that specific argument is truncated at the `NUL` and on Windows the *entire command line* is truncated at the `NUL`:

```
argv[0]: C:\Temp\main.exe
argv[1]: --foo
argv[2]: --bar=
```

If we don't agree that this is a meaningful distinction, that's fair. My goal here isn't to try and browbeat consensus. Rather, I just want to ensure I've clearly conveyed why I think this fix matters, and why I view it differently from other types of sanitization issues. In other cases, you're sanitizing your input to prevent the process, upon receiving the command line you provided, from doing something you don't want it to do, but which is *correct* for the process to do, given those arguments. In this case, you're trying to prevent the process from receiving a different command line from the one you provided. (And, since there's no way to escape `NUL` bytes such that they can be passed through correctly, at least that I'm aware of, we do the next best thing and report up front that we can't successfully reproduce the provided command line and fail cleanly.)

bturner commented on Sep 21 Collaborator

I have built, tagged and released NuProcess 2.0.5. It will take some time for the artifacts to appear in Central.

🎉 1

brettwooldridge commented on Sep 21 Owner

@bturner You convinced me.

brettwooldridge commented on Sep 24 Owner

@bturner Can you update the maven version in the README.md and update the CHANGES.md? Thanks.

bturner commented on Sep 26 Collaborator

Sure. I'll get it done later today

Reviewers

No reviews

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

3 participants