BeeFaauBee Follow

Apr 5, 2021 · 5 min read · Listen

Save

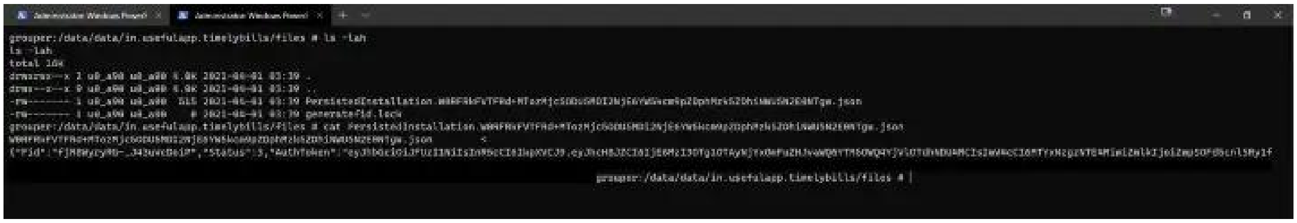
CVE-2021–26833: Cleartext Storage in a File or on Disk in TimelyBills ≤ 1.7.0 for iOS and versions ≤ 1.21.115 for Android

Clear text Storage in a File or on Disk in TimelyBills ≤ 1.7.0 for iOS and versions ≤ 1.21.115 for Android allows attacker who can locally read user's files obtain JWT tokens for user's account due to insufficient cache clearing mechanisms. A threat actor can obtain sensitive user data by decoding the tokens as JWT is signed and encoded, not encrypted

Background

The application requires users to provide username and password or sign-in via social media accounts (Google,Facebook etc.). After validation of these accounts, a JWT token is generated which is further utilized in authentication and authorization while performing various user-level operations.

While conducting security research on the application, It was understood that these JWT tokens are stored in a local file located within the app directory. Furthermore, It was also discovered that the authorization token doesn't expire and remains in the local directory after logging out of the application.



```
grouper:/data/data/in.usefulapp.timelybills/files # ls -lah
ls -lah
total 16K
drwxrwx--x 2 u0_a98 u0_a98 4.0K 2021-04-01 03:39 .
drwx--x--x 9 u0_a98 u0_a98 4.0K 2021-04-01 03:39 ..
-rw------- 1 u0_a98 u0_a98  515 2021-04-01 03:39 PersistedInstallation.[illegible].json
-rw------- 1 u0_a98 u0_a98    0 2021-04-01 03:39 generatefid.lock
grouper:/data/data/in.usefulapp.timelybills/files # cat PersistedInstallation.[illegible].json
[illegible].json        <
{"Fid":"[illegible]","Status":3,"AuthToken":"eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.[illegible]
                                        grouper:/data/data/in.usefulapp.timelybills/files # |
```

Image: Authorization Token stored in Local Directory

Impact

This can really help malicious applications (If installed) to steal such sensitive information from device and utilize the same for application. Furthermore, threat actor can steal confidential information by decoding the JWT token which can reveal information about victim.

Additional Discoveries of Vulnerabilities

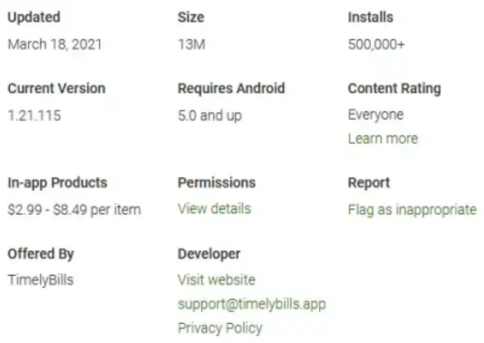
While performing researching over the app it was understood that the application is mainly utilizing functionality as reminder to remind a person about paying their bills on recurring basis or one time. All one has to do is schedule the payments/bills you have to pay and it will remind you of your payments as per your schedule settings. While exploring the application, this thing caught my eye :

> *PRIVACY POLICY*
>
> *TimelyBills knows that you care how information provided by you is used and shared, and we appreciate your trust that we will do so carefully and sensibly. At TimelyBills, we pride ourselves on our commitment to protecting your privacy. This Privacy Statement describes in greater detail the privacy policy at TimelyBills.*

I was pretty much skeptical about their claims of ensuring security of customers data. Few pointers I noted from their privacy statement which were :

- Access of back-end services using secure **authenticated** tokens.
- Data from app, website to our backed servers is transferred over HTTPS which is secured and encrypted using SSL.
- **Security PIN:** You can enable a 4 digit security PIN which protects the app from unauthorized or unwanted access, and the app asks your security PIN to allow access to the app.
- **Password:** Your account and app data is additionally protected with a password encrypted with 256 bit SHA algorithm.

This was enough for me to give it a try and look for the security controls they claim to have implemented in their application. However, to my surprise, with only few hours spent in exploring the app, I had enough evidence to defend my claim about the application have poor security control implementation in their application as well as API. Some stats to share about the application



Updated
March 18, 2021

Size
13M

Installs
500,000+

Current Version
1.21.115

Requires Android
5.0 and up

Content Rating
Everyone
Learn more

In-app Products
$2.99 - $8.49 per item

Permissions
View details

Report
Flag as inappropriate

Offered By
TimelyBills

Developer
Visit website
support@timelybills.app
Privacy Policy

(Image : Stats for app)

I'll begin with all of the findings one by one

1 — Account Takeover

This was very simple and required little information about the victim. All you need to have is account e-mail address or ID that is concatenated during password reset request. Consider the following Scenario :

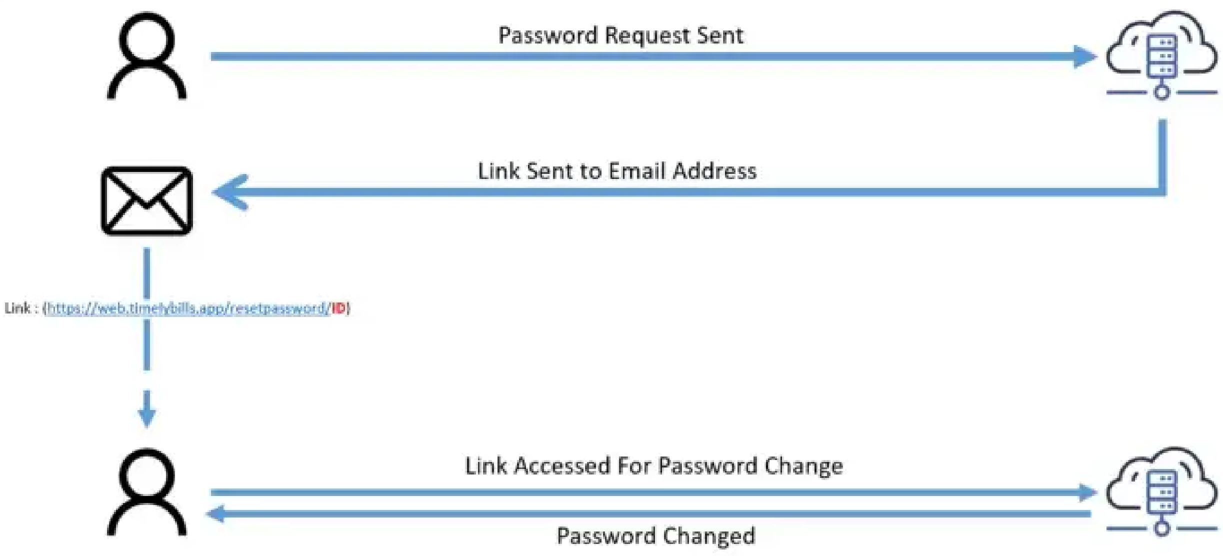
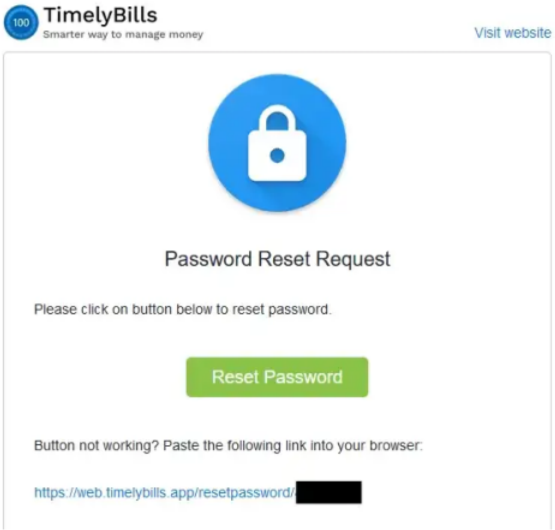

Password Request Sent
Link Sent to Email Address
Link : (https://web.timelybills.app/resetpassword/ID)
Link Accessed For Password Change
Password Changed

Image: Account Takeover Graphical Representation

The part where ID has been highlighted is the interesting one. If an attacker manages to get ID of victim, their account password can be changed by just replacing the ID at the end of request, providing new password and Password is changed successfully.



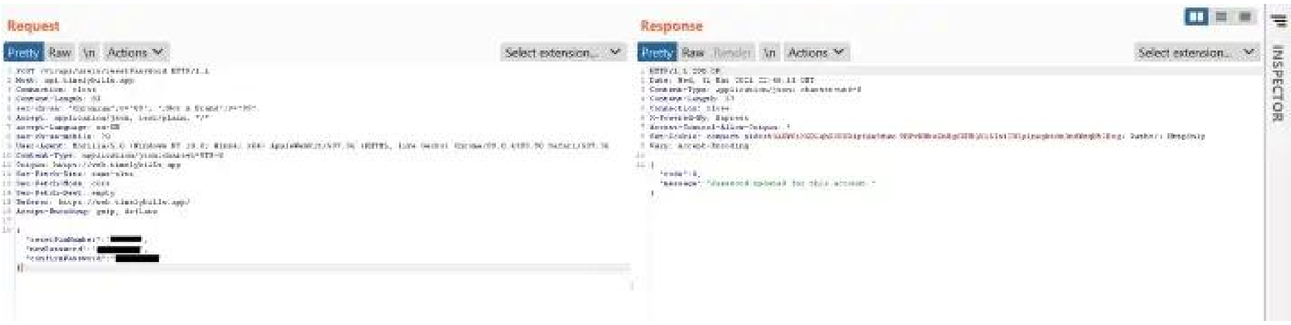
Password Reset Request

Please click on button below to reset password.

Button not working? Paste the following link into your browser:

https://web.timelybills.app/resetpassword/[redacted]

Image: Link For Password Reset



```
POST /v1/api/users/resetPassword HTTP/1.1
Host: api.timelybills.app
Connection: close
Content-Length: 92
sec-ch-ua: "Chromium";v="89", ";Not A Brand";v="99"
Accept: application/json, text/plain, */*
Accept-Language: en-US
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90 Safari/537.36
Content-Type: application/json;charset=UTF-8
Origin: https://web.timelybills.app
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://web.timelybills.app/
Accept-Encoding: gzip, deflate

{
  "resetPinNumber":"[redacted]",
  "newPassword":"[redacted]",
  "confirmPassword":"[redacted]
}
```

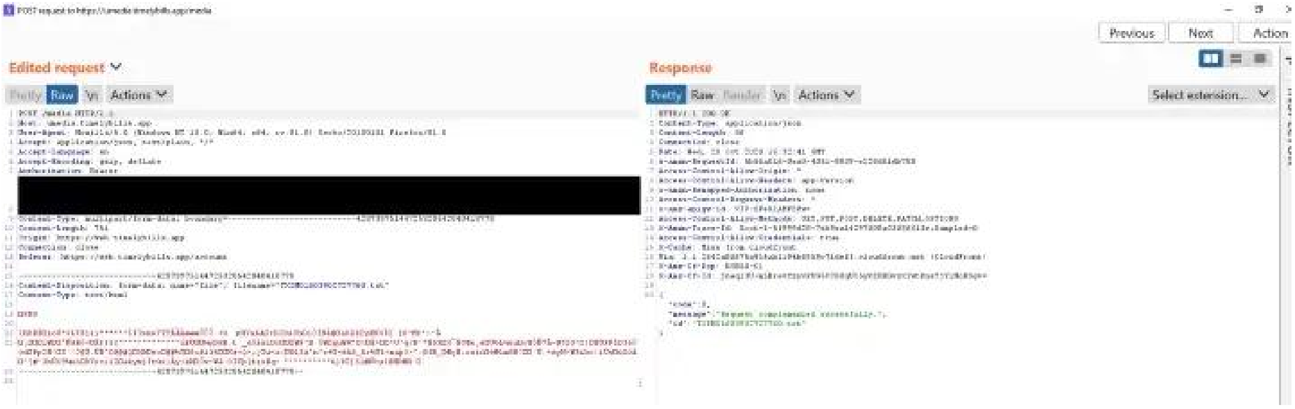
```
HTTP/1.1 200 OK
Date: Wed, 31 Mar 2021 22:48:13 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 57
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
Set-Cookie: [illegible]; Path=/; HttpOnly
Vary: Accept-Encoding

{
  "code":0,
  "message":"Password updated for this account."
}
```

Image: Link for Password Reset Request Manipulation and Response

2 — Uploading Malicious Files on Server

The application provides users to upload or attach their billing details by adding expenses or through different other resources by taking picture of the subjected item and then saving it through application.

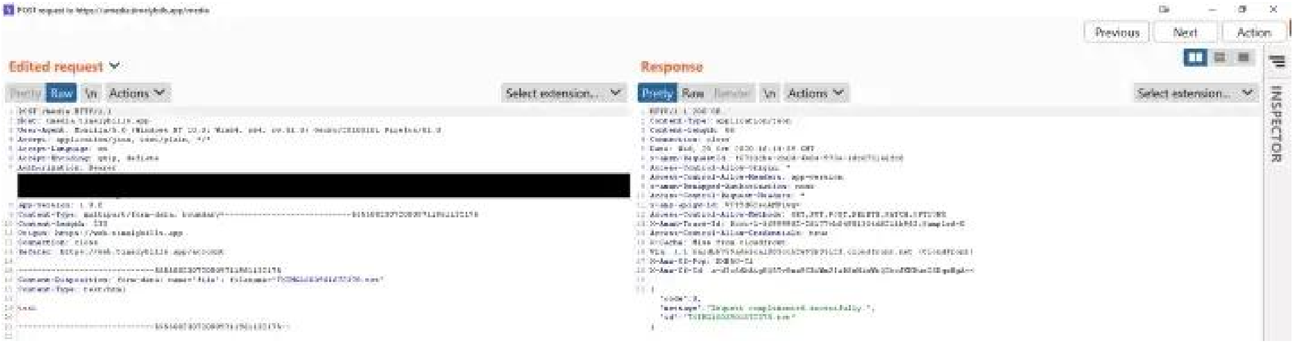
On back-end, the application processes such requests as POST and uploads the same on their cloud infrastructure with unique identifier. While analyzing the application, It was discovered that the app doesn't validate file extensions or content type that has been POST'd to the application API therefore allowing it to process and save the file on server.



```
POST /media HTTP/1.1
Host: media.timelybills.app
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:81.0) Gecko/20100101 Firefox/81.0
Accept: application/json, text/plain, */*
Accept-Language: en
Accept-Encoding: gzip, deflate
Authorization: Bearer
[redacted]
Content-Type: multipart/form-data; boundary=---------------------------[illegible]
Content-Length: 751
Origin: https://web.timelybills.app
Connection: close
Referer: https://web.timelybills.app/account

-----------------------------[illegible]
Content-Disposition: form-data; name="file"; filename="[illegible].txt"
Content-Type: text/html

[illegible]

[illegible]
-----------------------------[illegible]--
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 58
Connection: close
Date: Wed, 28 Oct 2020 20:32:41 GMT
x-amzn-RequestId: [illegible]
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: app-version
x-amzn-Remapped-Authorization: [illegible]
Access-Control-Request-Headers: *
x-amz-apigw-id: [illegible]
Access-Control-Allow-Methods: GET,PUT,POST,DELETE,PATCH,OPTIONS
X-Amzn-Trace-Id: Root=1-[illegible];Sampled=0
Access-Control-Allow-Credentials: true
X-Cache: Miss from cloudfront
Via: 1.1 [illegible].cloudfront.net (CloudFront)
X-Amz-Cf-Pop: [illegible]
X-Amz-Cf-Id: [illegible]

{
  "code":0,
  "message":"Request completed successfully.",
  "id":"[illegible].txt"
}
```

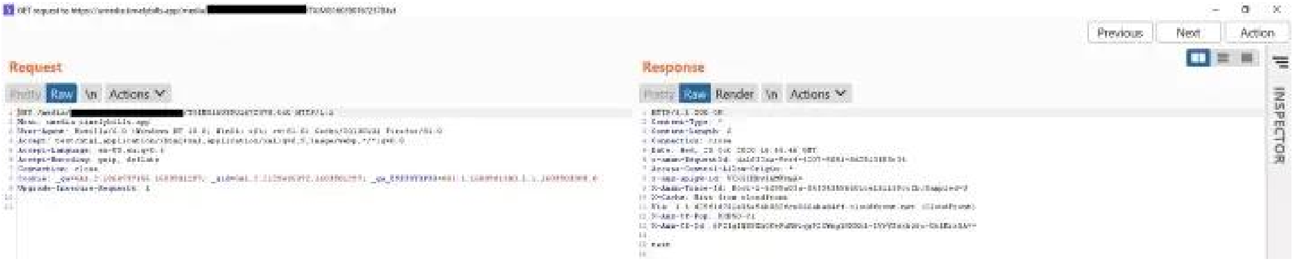
Image: Initial Request of File upload



```
POST /media HTTP/1.1
Host: media.timelybills.app
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:81.0) Gecko/20100101 Firefox/81.0
Accept: application/json, text/plain, */*
Accept-Language: en
Accept-Encoding: gzip, deflate
Authorization: Bearer
[redacted]
App-Version: [illegible]
Content-Type: multipart/form-data; boundary=---------------------------[illegible]
Content-Length: [illegible]
Origin: https://web.timelybills.app
Connection: close
Referer: https://web.timelybills.app/account

-----------------------------[illegible]
Content-Disposition: form-data; name="file"; filename="[illegible]"
Content-Type: text/html

test

-----------------------------[illegible]--
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: [illegible]
Connection: close
Date: [illegible] Oct 2020 [illegible] GMT
x-amzn-RequestId: [illegible]
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: app-version
x-amzn-Remapped-Authorization: none
Access-Control-Expose-Headers: *
x-amz-apigw-id: [illegible]
Access-Control-Allow-Methods: GET,PUT,POST,DELETE,PATCH,OPTIONS
X-Amzn-Trace-Id: Root=1-[illegible];Sampled=0
Access-Control-Allow-Credentials: true
X-Cache: Miss from cloudfront
Via: 1.1 [illegible].cloudfront.net (CloudFront)
X-Amz-Cf-Pop: [illegible]
X-Amz-Cf-Id: [illegible]

{
  "code":0,
  "message":"Request completed successfully.",
  "id":"[illegible]"
}
```

Image: Edited request for File upload



```
GET /media/[redacted][illegible] HTTP/1.1
Host: media.timelybills.app
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:81.0) Gecko/20100101 Firefox/81.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: _ga=[illegible]; _gid=[illegible]; _ga_[illegible]=[illegible]
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Content-Type: *
Content-Length: 4
Connection: close
Date: Wed, 28 Oct 2020 [illegible] GMT
x-amzn-RequestId: [illegible]
Access-Control-Allow-Origin: *
x-amz-apigw-id: [illegible]
X-Amzn-Trace-Id: Root=1-[illegible];Sampled=0
X-Cache: Miss from cloudfront
Via: 1.1 [illegible].cloudfront.net (CloudFront)
X-Amz-Cf-Pop: [illegible]
X-Amz-Cf-Id: [illegible]

test
```

Image: File GET request

Timeline

2020/10/28 — Contact made with Vendor for sharing Information regarding vulnerabilities.

2020/10/28 — Contact Established with Vendor.

2020/10/28 — Provided detailed report and explanation on vulnerabilities that required immediate attention.

2020/10/29 — Vendor Acknowledges details received and forwarded to engineering department.

2020/11/10 — Approached vendor for updates on patches/fixes (No Response)

2020/11/25 — Activity noticed on application new release. Requested if the vulnerabilities shared are highlighted/fixed (No Response)

2020/11/30 — Approached Vendor again and requested for sharing advisory/coordinated vulnerability disclosure for all customers.

2020/12/01 — Vendor acknowledges vulnerabilities and share timeline for fixation by 20th December, 2020

2020/12/21 — Approached Vendor to get update on vulnerabilities shared are fixed (No Response)

2020/12/23— Approached Vendor again and requested for sharing advisory/coordinated vulnerability disclosure for all customers.(No Response)

2020/12/26 — Vendor responded with teams away due to Christmas holidays.

2021/01/15 — Approached Vendor again and requested for sharing advisory/CVD for all customers. (No Response)

2021/02/28 — CVE ID requested from CNA against app

2021/03/31 — CVE issued

2021/04/05 — Public Disclosure

Final Notes

InfoSec community is really one of the greatest one. Earlier, I had uploaded this write-up and few mistakes were identified to which I'm really thankful to John Jackson. Without his support I don't think this whole research would have been compiled in a write-up properly.

Responsible Dis

Android Security

Api Security

Application Security

Information Security