ADMINISTRATIVE REVIEW TRIBUNAL
MELBOURNE REGISTRY
INTELLIGENCE AND SECURITY JURISDICTIONAL AREA

ART 2025/4916

BETWEEN

FRASER TWEEDALE
Applicant

CEO, SERVICES AUSTRALIA
Respondent

AFFIDAVIT

I, **Fraser Tweedale** of 29 Cape Street, Holland Park 4121, affirm:

1. I hold a Bachelor of Engineering (Computer Systems Engineering) from the University of Queensland, and have worked as a software engineer since 2007. I am currently employed as a Principal Software Engineer at Red Hat, LLC, working on authentication and identity management solutions, a role I have held since 2014. I am involved in open source software development, both in my work and in a personal capacity.

2. Unless otherwise stated, the facts set out in this affidavit are within my own knowledge.

Background

3. I have applied to the Administrative Review Tribunal (**Tribunal**) under s 57A of the *Freedom of Information Act 1982* (Cth) (**FOI Act**) for review of a decision of Services Australia to refuse my request under the FOI Act for access to the source code of the Android version of the *myGov Code Generator* application (**App**).

4. The App generates a six-digit, time-based one-time password (**TOTP code)** which users enter alongside their username and password when logging into their myGov account. A TOTP code is valid for 30 seconds and serves as a single-use authentication code. TOTP codes are derived from a secret, randomly generated seed value (the '**TOTP shared secret**'), which is possessed by the App (after the App has been enrolled) and by the myGov Service (where the unique shared secrets are associated with corresponding myGov user accounts).

5. The respondent in the proceeding, the CEO of Services Australia, relies upon the 'open' and 'confidential' affidavits of Mr Garrett McDonald, the Chief Information Security Officer of Services Australia.

6. The respondent has applied to the Tribunal under s 70 of the *Administrative Review Tribunal Act 2024* (Cth) (**ART Act**) for confidentiality orders prohibiting disclosure of Mr McDonald's confidential affidavit (**Confidential Affidavit**) in its entirety. The terms of the proposed confidentiality orders would prevent disclosure to the applicant and his legal representatives.

7. One matter critical to the making of the proposed confidentiality orders is whether information in the Confidential Affidavit is, in fact, confidential. To the extent that information in the Confidential Affidavit is in the public domain or can be readily ascertained through technical examination of the App, it cannot be regarded as confidential and cannot be the subject of a confidentiality order.

Lodged on behalf of the Applicant
Applicant contact: Jason Bosland

Prepared by: Jason Bosland

Wise Law
141A Como Parade East
Parkdale Vic 3195

Telephone: 0421 193 711

Email: JB@wiselaw.com.au

8. I make this affidavit to support my opposition to the respondent's application for confidentiality orders over the Confidential Affidavit. The affidavit sets out and explains evidence concerning information about the App that is already in the public domain.

GitHub

9. There are two separate entries on the public platform, GitHub, which describe and/or replicate some details of the App.

10. **GitHub** is a public source code hosting and collaboration service, currently owned and operated by Microsoft. Millions of developers and companies around the world use GitHub to host and collaborate on source code. Many projects and code 'repositories' are public (open source).

11. GitHub is based on the 'Git' version control system and tracks changes to code repositories over time. GitHub includes an 'issues' feature, which allows users to report problems or request features in a repository, and a 'pull requests' feature that enables GitHub users to propose changes to the code in a repository (which can subsequently be merged or rejected by a repository maintainer). Both 'issues' and 'pull requests' features support discussions, where users can comment on the content of the issue or pull request.

12. **GitHub Gist** is a service operated by GitHub for publishing 'gists', which are typically source code snippets, experiments, how-to guides, and proof-of-concept code. GitHub Gist web pages allow other Github users to leave comments on Gists.

13. I have been an active GitHub user since 2012.

mygov-totp-enroll repository

14. On 11 April 2019, GitHub user, 'abrasive' (James Wah), published on GitHub the 'mygov-totp-enroll' repository (**mygov-totp-enroll repository**).

    Annexed hereto and marked '**FT-1**' are screenshots of the landing page of the Github repository posted by user, 'abrasive', as at 1 July 2026: https://github.com/abrasive/mygov-totp-enroll.

15. The mygov-totp-enroll repository contained an alternative implementation of the myGov Code Generator enrollment workflow, serving as a functional substitute of the official App. In this workflow, a user performs an initial authentication to the official myGov Service using their account username and password, and the application requests a TOTP shared secret, and activates the TOTP two-factor authentication method for the user's myGov account. The behaviour of the mygov-totp-enroll application which is contained in the GitHub repository diverges from the behaviour of the official App in what it does with the TOTP shared secret after retrieval. The official App stores it securely on the user's mobile device on which the App is running. In contrast, the mygov-totp-enroll application presents the TOTP shared secret in a form that enables the user to enroll the TOTP shared secret in their preferred TOTP authenticator app (for example, Google Authenticator).

16. The source code contained in the 'mygov-totp-enroll' repository includes:

    16.1. Approximately 90 lines of HyperText Markup Language (HTML) code across two files, defining the user interface;

    Annexed hereto and marked '**FT-2**' are screenshots of the two files containing the HTML code, as at 1 July 2026: https://github.com/abrasive/mygov-totp-enroll/blob/master/instructions.html and https://github.com/abrasive/mygov-totp-enroll/blob/master/ui.html.

16.2. Approximately 200 lines of application code written in the JavaScript programming language, in 1 file. Of this, approximately 90 lines implement the interactions with the myGov services, approximately 15 lines implement the presentation of the TOTP secret as a QR code, and the remainder deals with initialisation of the application and user interface and other "boilerplate";

Annexed hereto and marked '**FT-3'** are screenshots of the code written in JavaScript programming language, as at 1 July 2026: https://github.com/abrasive/mygov-totp-enroll/blob/master/main.js.

16.3. Invocations of behaviour provided by third party software packages related to network requests, data encoding and decoding, and QR code generation.

17. Following the publication of the mygov-totp-enroll repository, a series of improvements and bug fixes were made to the repository by various contributors including the original author.

18. GitHub users also filed problem reports or discussed proposed changes to this repository using the 'issues' and 'pull request' functions. It is evident from the issues and pull requests on the mygov-totp-enroll repository, and the associated discussions, that numerous people had success using the mygov-totp-enroll program to enable TOTP authentication on their myGov account. It follows that the implementation of the mygov-totp-enroll program, in respect of the TOTP enrollment protocol, its use of myGov Service Application Programming Interfaces (APIs), and its implementation and parameter selection for the TOTP algorithm, was correct such that it served as a functional substitute for the official *myGov Code Generator* mobile application.

Annexed hereto and marked '**FT-4**' are screenshots of the 'issues' and 'pull request' entries in which users discuss how to use the mygov-totp-enroll repository to enable TOTP authentication for their myGov account, as at 1 July 2026.

19. Based on the information reproduced in FT-1, FT-2, FT-3 and FT-4, it can be concluded that the source code in the mygov-totp-enroll repository published on GitHub contains identical or functionally equivalent information about the myGov service domain names, authentication protocol, API "endpoints", and payload formats that would be present in the source code of the official App. Therefore, I consider that this information about the App and its interaction with the myGov Service is in the public domain.

mygov_totp_storage Gist

20. On 27 April 2021, GitHub user, 'hacker1024' (Joshua Leivenzon), published on GitHub Gist a 'gist' entitled 'Information about TOTP token storage in the App, as well as token extraction instructions' (**hacker1024 gist**).

Annexed hereto and marked '**FT-5**' are screenshots of the GitHub gist posted by hacker1024, including comments from other users as at 1 July 2026: https://gist.github.com/hacker1024/5d0845863e2dced27fd5eebc4ac95a39

Annexed hereto and marked '**FT-6**' are screenshots of subsequent revisions to the gist as at 1 July 2026: https://gist.github.com/hacker1024/5d0845863e2dced27fd5eebc4ac95a39/revisions

21. For the purposes of this affidavit, the term 'hacker' should be understood as someone with deep expertise in computer systems and who applies their skills enthusiastically to solve problems. The GitHub account name 'hacker1024' should therefore not be associated with malicious intent or nefarious activities.

22. The hacker1024 gist contained instructions for extracting the TOTP shared secret from the App's internal storage directory. These instructions include the following information about the App:

    22.1. Where and how the encrypted TOTP shared secret and associated encryption key are stored within the App's internal storage directory;

    22.2. The encryption algorithms and parameters that the App uses to encrypt the TOTP shared secret;

    22.3. A 'hard-coded' (static) password which is used to decrypt the secondary encryption key, which in turn decrypts the TOTP shared secret;

    22.4. The TOTP algorithm parameters, particularly digest algorithm, number of digits, and time duration;

    22.5. The steps a user can follow to decrypt and decode the TOTP shared secret, and prepare it for use with other TOTP authenticator applications.

23. As evident from the screenshot reproduced in FT-5 (pp 26 and 27), several users wrote comments indicating that, by following the instructions, they successfully extracted the TOTP seed secret from the App's internal data storage. The publisher account, hacker1024, also posted a follow-up comment providing additional information about device requirements to successfully execute the instructions (FT-5, p 27).

24. As a result of the hacker1024 gist and associated comments from users reproduced in FT-5, I consider that accurate information about the App and its implementation, in particular how the App stores the TOTP shared secret, is in the public domain.

Reddit discussion

25. A Reddit discussion initiated by Reddit user 'j_l-w' in the r/australia forum on April 12, 2019, entitled 'for all the myGov fake 2FA haters', discusses myGov's two-factor authentication system (**j_l-w discussion**). The initial post in the discussion includes a link to the 'mygov-totp-enroll' GitHub repository owned by the GitHub user 'abrasive' (James Wah). It is apparent from the text of the initial post, and the correspondence of the name 'j_l-w' to 'James Wah', that Reddit user 'j_l-w' and GitHub user 'abrasive' are one and the same person.

    Annexed hereto and marked '**FT-7**' is the Reddit post by user 'j_l-w' in the r/Australia forum as at 1 July 2026:
    https://www.reddit.com/r/australia/comments/bc6gmv/for_all_the_mygov_fake_2fa_haters/

26. A Reddit user, 'Thingface', published two comments in the j_l-w discussion explaining how to intercept the network communications of the official App to read the TOTP shared secret, and an analysis of the protocol and intercepted message payloads (**Thingface comments**).

    Annexed hereto and marked '**FT-8**' are screenshots of the two comments made by user, 'Thingface', on the Reddit post as at 1 July 2026:
    https://www.reddit.com/r/australia/comments/bc6gmv/for_all_the_mygov_fake_2fa_haters/

27. Reddit user 'j_l-w' posted a follow-up message in response to the Thingface comments that includes a statement that he took a similar approach to analysing the behaviour of the official App.

    Annexed hereto and marked '**FT-9**' is a screenshot of the follow-up message by Reddit user 'j_l-w' as at 1 July 2026:
    https://www.reddit.com/r/australia/comments/bc6gmv/for_all_the_mygov_fake_2fa_haters/

28. I consider that the information about the protocol and messages published on Reddit by user "Thingface" as captured in annexure FT-8 is in the public domain. I also consider that any information about the protocol, message or payload content or formats, or service domain names and endpoint network addresses or locations, that could be revealed by applying the network traffic interception techniques discussed in annexures FT-8 and FT-9 to reveal information about the App or its interactions with myGov services, is in the public domain.

Sworn by the deponent	)	
Fraser Tweedale at Brisbane	)	
on 2 July 2026	)	……..
before me:	)	Signature of deponent

…….
Signature of witness

Jason Bosland
11 Oak Grove, Ripponlea, Victoria, 3000
An Australian Legal Practitioner within
the meaning of the Legal Profession
Uniform Law (Victoria)

This affidavit was witnessed electronically by audiovisual link in accordance with the *Oaths and Affirmations Act 2018* (Vic).

ADMINISTRATIVE REVIEW TRIBUNAL
MELBOURNE REGISTRY
INTELLIGENCE AND SECURITY JURISDICTIONAL AREA

2025/4916

FRASER TWEEDALE
Applicant

CEO, SERVICES AUSTRALIA
Respondent

ANNEXURE FT-1

The following is the annexure marked **FT-1** referred to in the affidavit of Fraser Tweedale, affirmed at Brisbane on 2 July 2026. This annexure contains a screenshot of the landing page of the GitHub repository posted by user, 'abrasive', as at July 1 2026: https://github.com/abrasive/mygov-totp-enroll.

.....
Jason Bosland
Australian Legal Practitioner
Melbourne

Lodged on behalf of the Applicant

Applicant contact: Jason Bosland

Prepared by: Jason Bosland

Wise Law
141A Como Parade East
Parkdale Vic 3195

Telephone: 0421 193 711

Email: JB@wiselaw.com.au

abrasive / **mygov-totp-enroll** Public

Code Issues 8 Pull requests 2 Actions Security and quality Insight

About

Enroll a real TOTP client to access myGov

Readme
Activity
108 stars
9 watching
8 forks
Report repository

Releases

No releases published

Packages

No packages published

Contributors 4


abrasive James Wah


lachlanhunt Lachlan Hunt

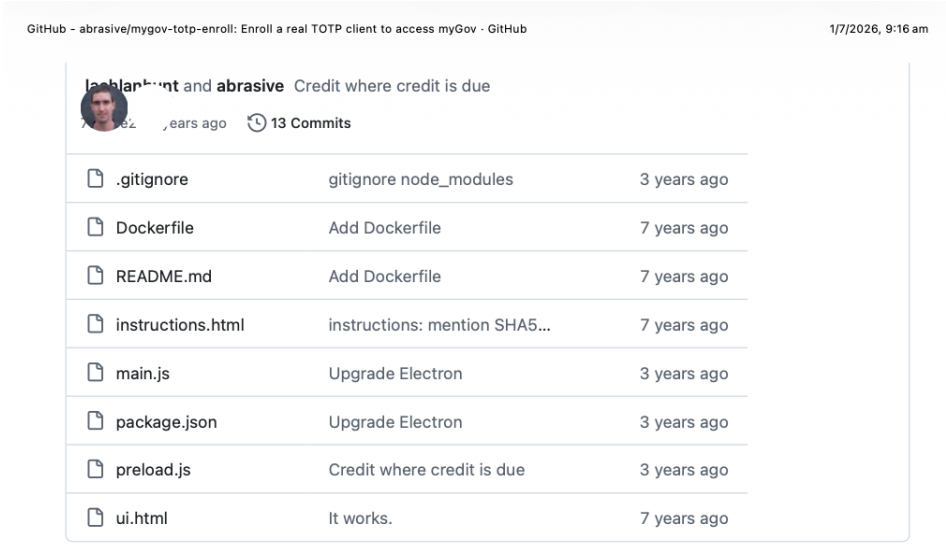

alexlance Alex Lance


danielgruber8 Daniel

Languages

JavaScript 65.7% **HTML** 32.0% **Dockerfile** 2.3%

1 Branch 0 Tags Go to file Go to file Code



[illegible] and **abrasive** Credit where credit is due

[illegible] ears ago **13 Commits**

.gitignore	gitignore node_modules	3 years ago
Dockerfile	Add Dockerfile	7 years ago
README.md	Add Dockerfile	7 years ago
instructions.html	instructions: mention SHA5...	7 years ago
main.js	Upgrade Electron	3 years ago
package.json	Upgrade Electron	3 years ago
preload.js	Credit where credit is due	3 years ago
ui.html	It works.	7 years ago

README

mygov-totp-enroll

This tool lets you enroll a TOTP authenticator (eg. andOTP) to access myGov.

You can't have more than one authenticator set up, so if you want multiple copies or backups, better do them all at once.

It shouldn't be possible to fuck up your myGov account using this tool, because you have to enter a correct code from your newly configured authenticator for the TOTP login requirement to be activated. However, if you do screw up somehow, there's no way to recover a myGov account, and you have to make a new one. Consequently I make no guarantees about this code, and if it flattens your cat or makes your lollies taste funny, well, you were warned.

This is an Electron app because you absolutely have to install a handler for full custom URL schemes, as the myGov OAuth endpoint insists on returning you to au.gov.my://app, which is pretty reasonable for them from a security point of view and atrocious for us from a reasonable software point of view. So sorry for the bloat, but there you go.

Instructions for use

1. `npm install`
2. `npm start`
3. pray
4. follow the instructions

Or if you have docker installed

1. `docker build -t mygov .`
2. `docker run -e DISPLAY --net=host -it mygov npm start`

ADMINISTRATIVE REVIEW TRIBUNAL
MELBOURNE REGISTRY
INTELLIGENCE AND SECURITY JURISDICTIONAL AREA

2025/4916

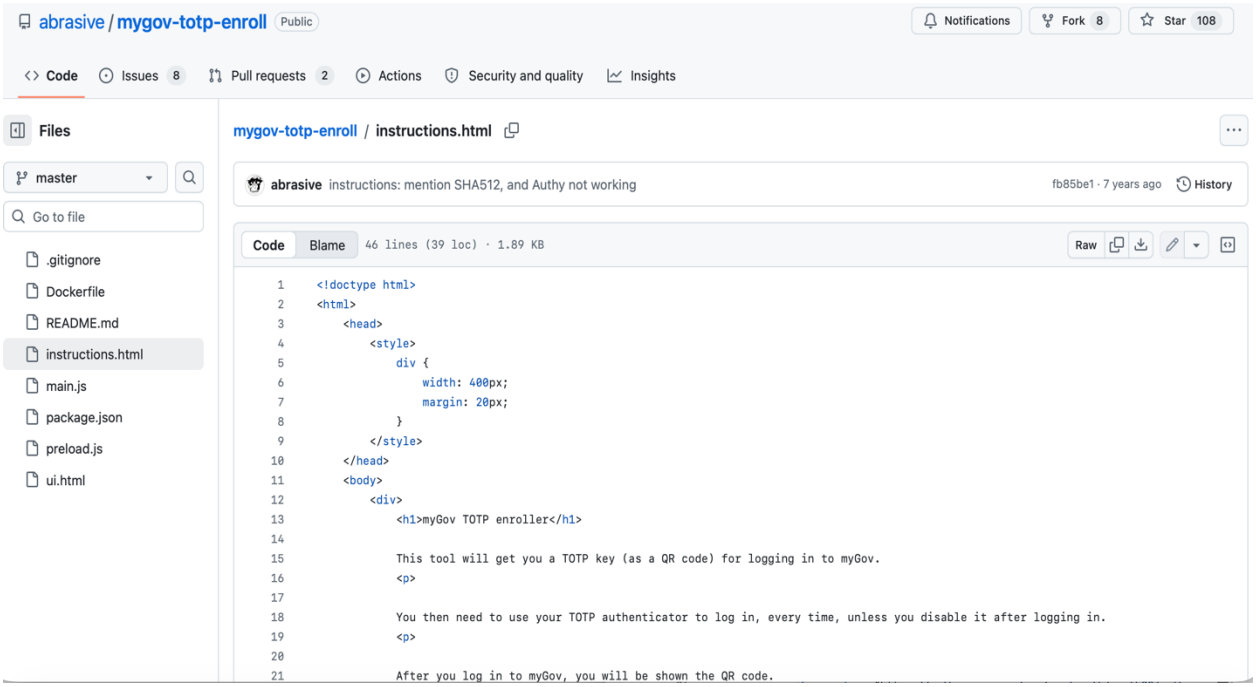
FRASER TWEEDALE
Applicant

CEO, SERVICES AUSTRALIA
Respondent

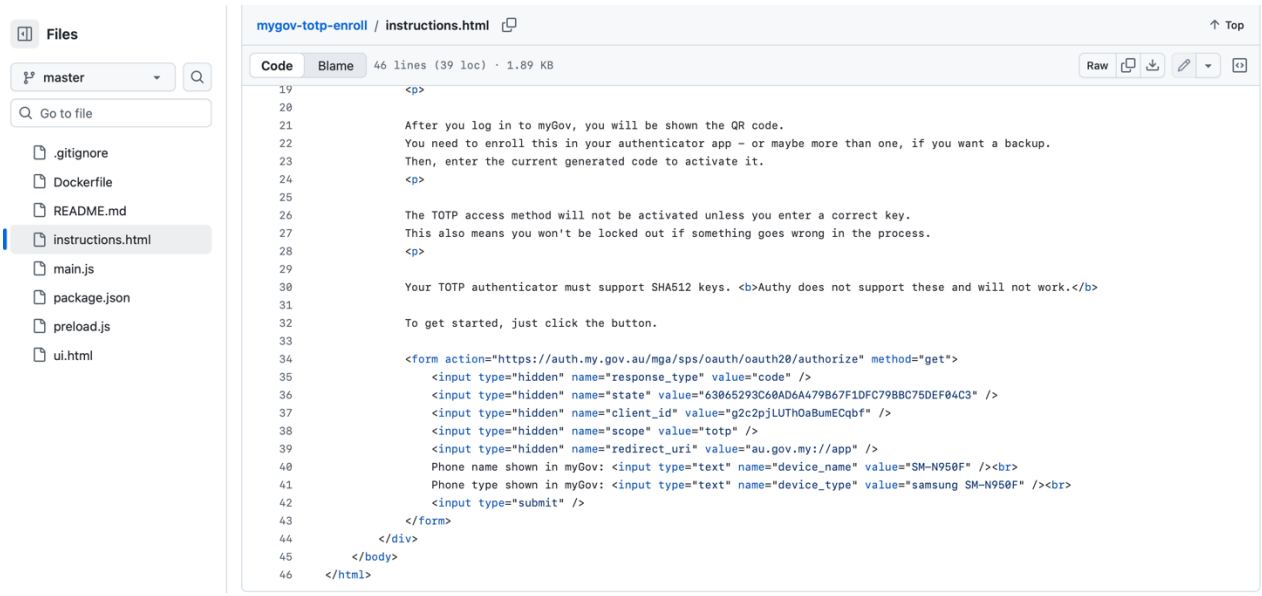
ANNEXURE FT-2

The following is the annexure marked **FT-2** referred to in the affidavit of Fraser Tweedale, affirmed at Brisbane on 2 July 2026. This annexure includes screenshots of the two files containing the HTML source code in the 'mygov-totp-enroll' GitHub repository as at July 1 2026 and available at: https://github.com/abrasive/mygov-totp-enroll/blob/master/instructions.html and https://github.com/abrasive/mygov-totp-enroll/blob/master/ui.html.

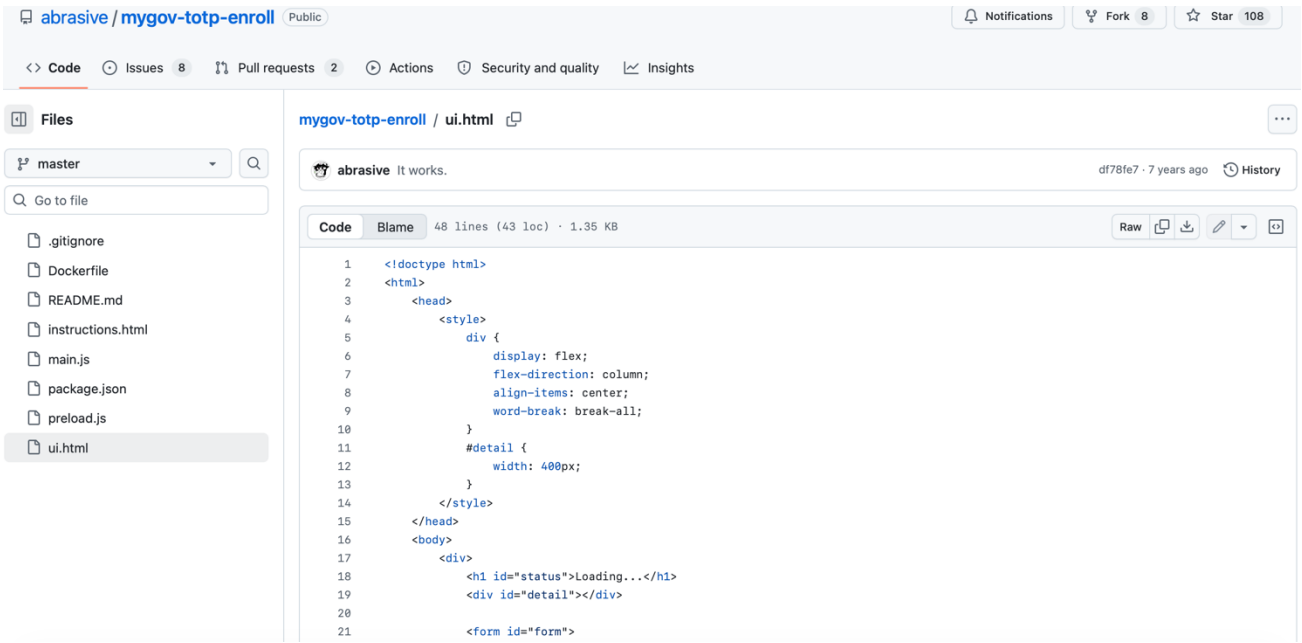
.....
Jason Bosland
Australian Legal Practitioner
Melbourne


abrasive / mygov-totp-enroll Public
Notifications
Fork 8
Star 108
Code
Issues 8
Pull requests 2
Actions
Security and quality
Insights
Files
master
Go to file
.gitignore
Dockerfile
README.md
instructions.html
main.js
package.json
preload.js
ui.html
mygov-totp-enroll / instructions.html
abrasive instructions: mention SHA512, and Authy not working
fb85be1 · 7 years ago
History
Code
Blame
46 lines (39 loc) · 1.89 KB
Raw
<!doctype html>
<html>
<head>
<style>
div {
width: 400px;
margin: 20px;
}
</style>
</head>
<body>

<h1>myGov TOTP enroller</h1>
This tool will get you a TOTP key (as a QR code) for logging in to myGov.
<p>
You then need to use your TOTP authenticator to log in, every time, unless you disable it after logging in.
<p>
After you log in to myGov, you will be shown the QR code.

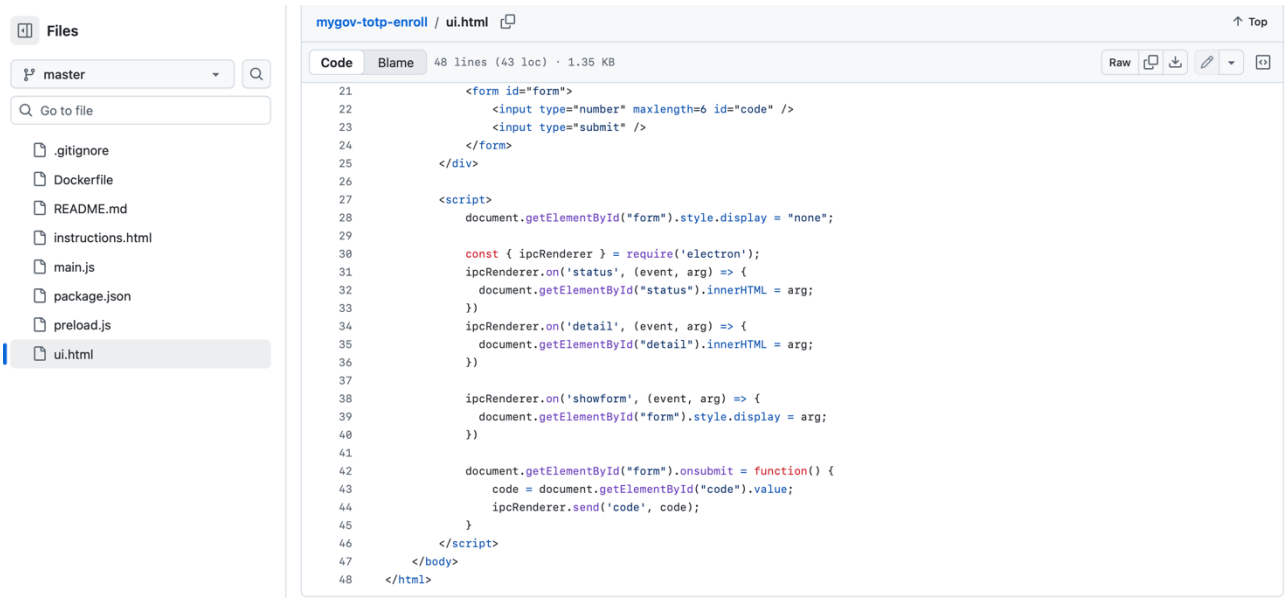

Files
master
Go to file
.gitignore
Dockerfile
README.md
instructions.html
main.js
package.json
preload.js
ui.html
mygov-totp-enroll / instructions.html
Top
Code
Blame
46 lines (39 loc) · 1.89 KB
Raw
<p>
After you log in to myGov, you will be shown the QR code.
You need to enroll this in your authenticator app - or maybe more than one, if you want a backup.
Then, enter the current generated code to activate it.
<p>
The TOTP access method will not be activated unless you enter a correct key.
This also means you won't be locked out if something goes wrong in the process.
<p>
Your TOTP authenticator must support SHA512 keys. Authy does not support these and will not work.
To get started, just click the button.
<form action="https://auth.my.gov.au/mga/sps/oauth/oauth20/authorize" method="get">
<input type="hidden" name="response_type" value="code" />
<input type="hidden" name="state" value="63065293C60AD6A479B67F1DFC79BBC75DEF04C3" />
<input type="hidden" name="client_id" value="g2c2pjLUThOaBumECqbf" />
<input type="hidden" name="scope" value="totp" />
<input type="hidden" name="redirect_uri" value="au.gov.my://app" />
Phone name shown in myGov: <input type="text" name="device_name" value="SM-N950F" />
Phone type shown in myGov: <input type="text" name="device_type" value="samsung SM-N950F" />
<input type="submit" />
</form>

</body>
</html>


abrasive / mygov-totp-enroll Public
Notifications
Fork 8
Star 108
Code
Issues 8
Pull requests 2
Actions
Security and quality
Insights
Files
master
Go to file
.gitignore
Dockerfile
README.md
instructions.html
main.js
package.json
preload.js
ui.html
mygov-totp-enroll / ui.html
abrasive It works.
df78fe7 · 7 years ago
History
Code
Blame
48 lines (43 loc) · 1.35 KB
Raw
<!doctype html>
<html>
<head>
<style>
div {
display: flex;
flex-direction: column;
align-items: center;
word-break: break-all;
}
#detail {
width: 400px;
}
</style>
</head>
<body>

<h1 id="status">Loading...</h1>

<form id="form">


Files
master
Go to file
.gitignore
Dockerfile
README.md
instructions.html
main.js
package.json
preload.js
ui.html
mygov-totp-enroll / ui.html
Top
Code
Blame
48 lines (43 loc) · 1.35 KB
Raw
21 <form id="form">
22 <input type="number" maxlength=6 id="code" />
23 <input type="submit" />
24 </form>
25
26
27 <script>
28 document.getElementById("form").style.display = "none";
29
30 const { ipcRenderer } = require('electron');
31 ipcRenderer.on('status', (event, arg) => {
32 document.getElementById("status").innerHTML = arg;
33 })
34 ipcRenderer.on('detail', (event, arg) => {
35 document.getElementById("detail").innerHTML = arg;
36 })
37
38 ipcRenderer.on('showform', (event, arg) => {
39 document.getElementById("form").style.display = arg;
40 })
41
42 document.getElementById("form").onsubmit = function() {
43 code = document.getElementById("code").value;
44 ipcRenderer.send('code', code);
45 }
46 </script>
47 </body>
48 </html>

ADMINISTRATIVE REVIEW TRIBUNAL
MELBOURNE REGISTRY
INTELLIGENCE AND SECURITY JURISDICTIONAL AREA

2025/4916

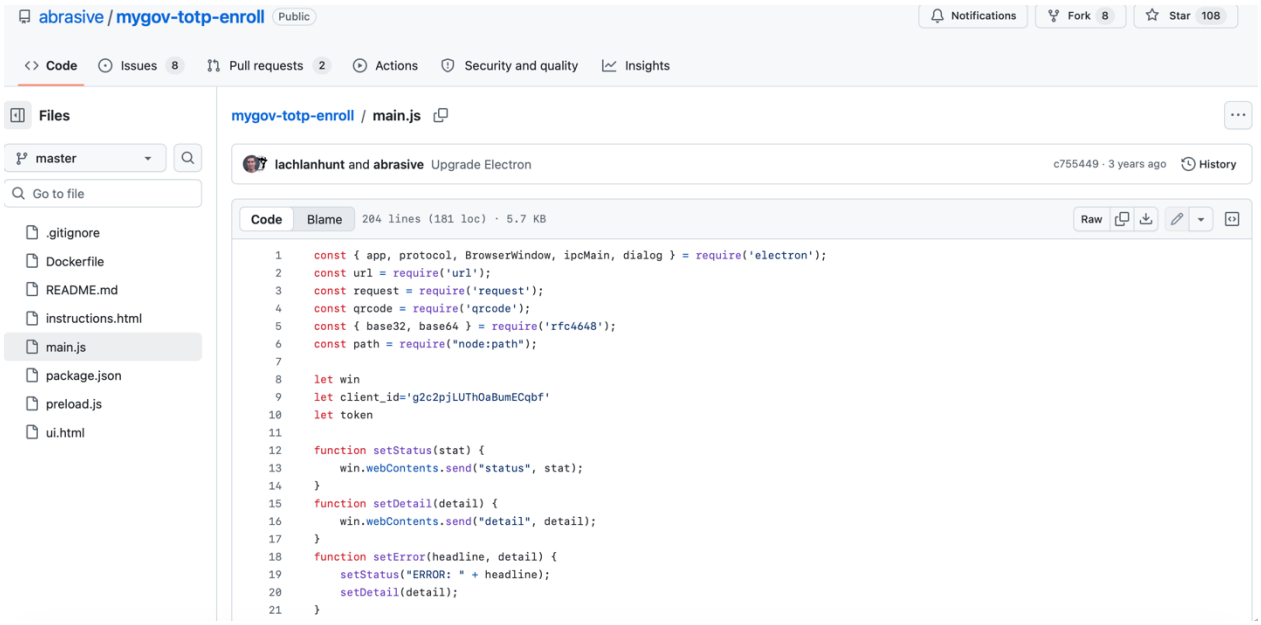
FRASER TWEEDALE
Applicant

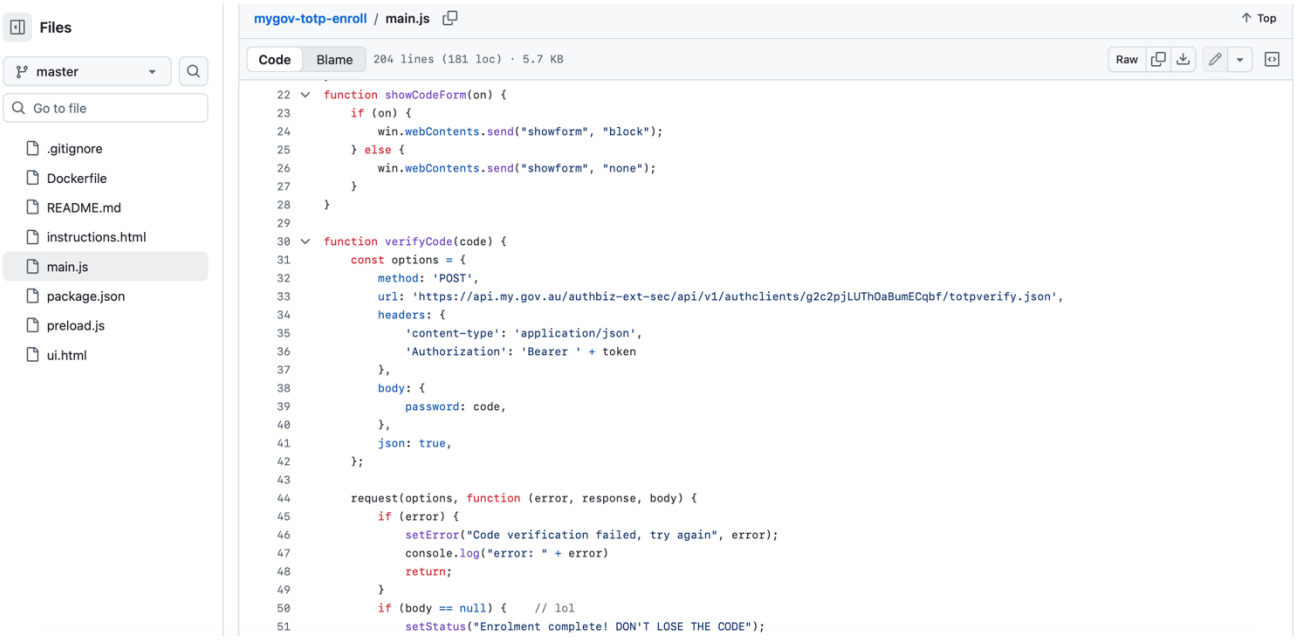
CEO, SERVICES AUSTRALIA
Respondent

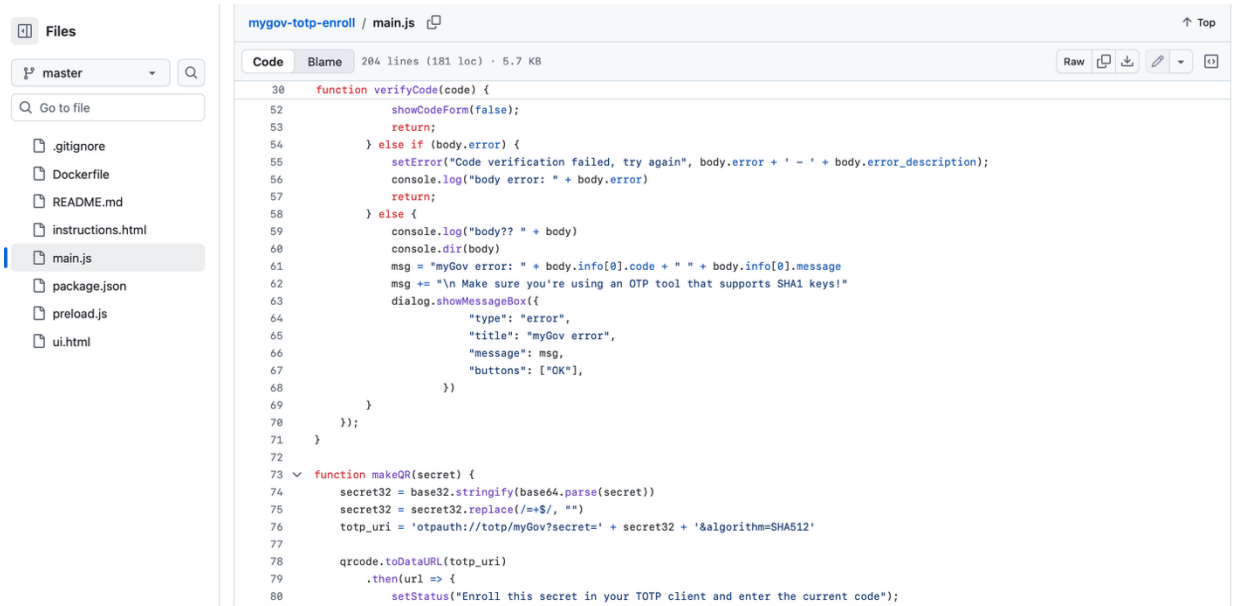
ANNEXURE FT-3

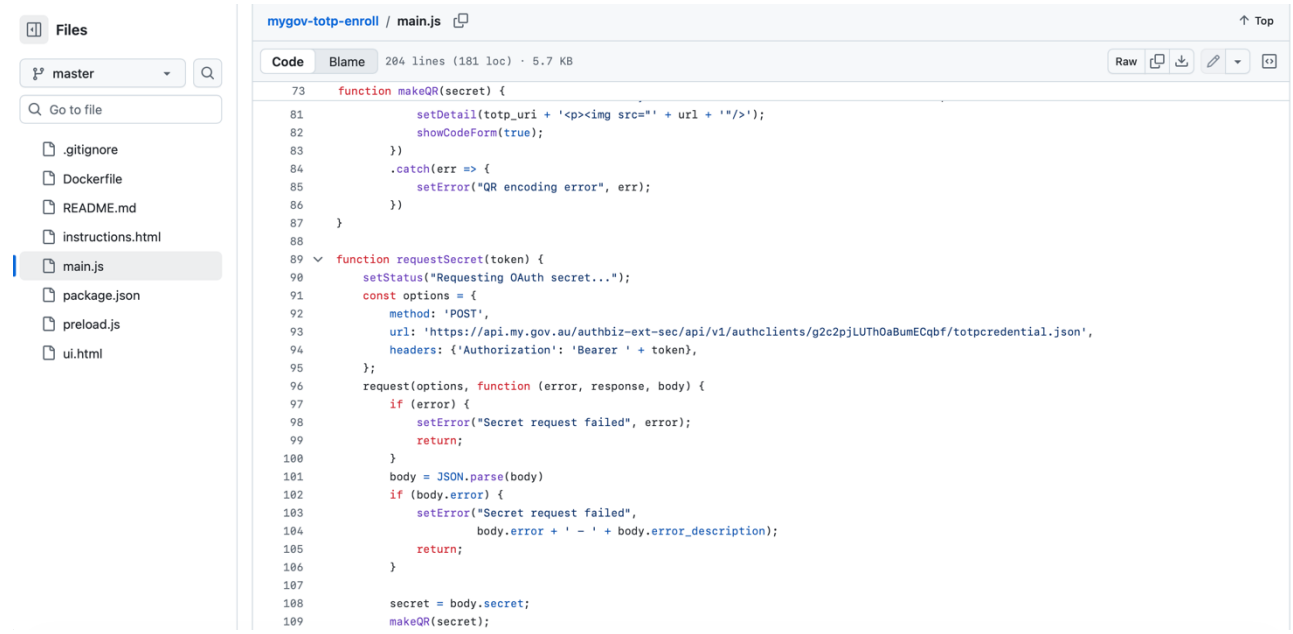
The following is the annexure marked **FT-3** referred to in the affidavit of Fraser Tweedale, affirmed at Brisbane on 2 July 2026. This annexure includes screenshots of the application code written in the JavaScript programming language published in the GitHub repository, 'mygov-totp-enroll', as at July 1 2026: https://github.com/abrasive/mygov-totp-enroll/blob/master/main.js.

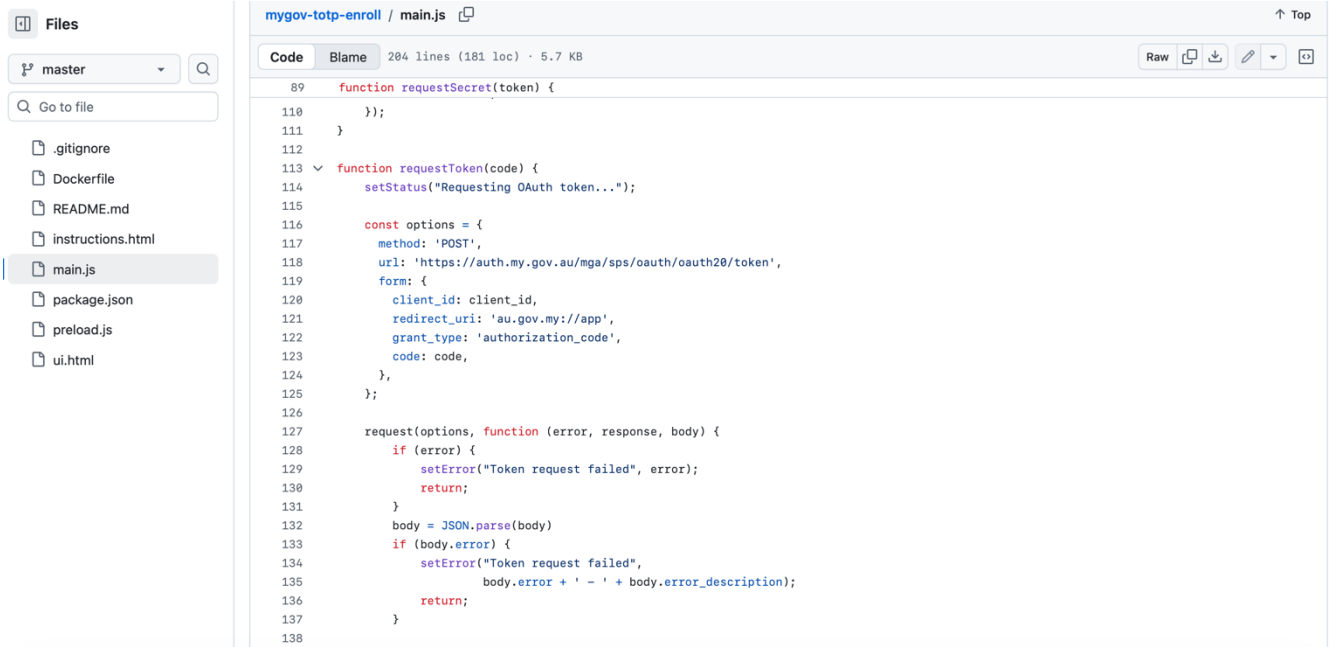
.....
Jason Bosland
Australia Legal Practitioner
Melbourne

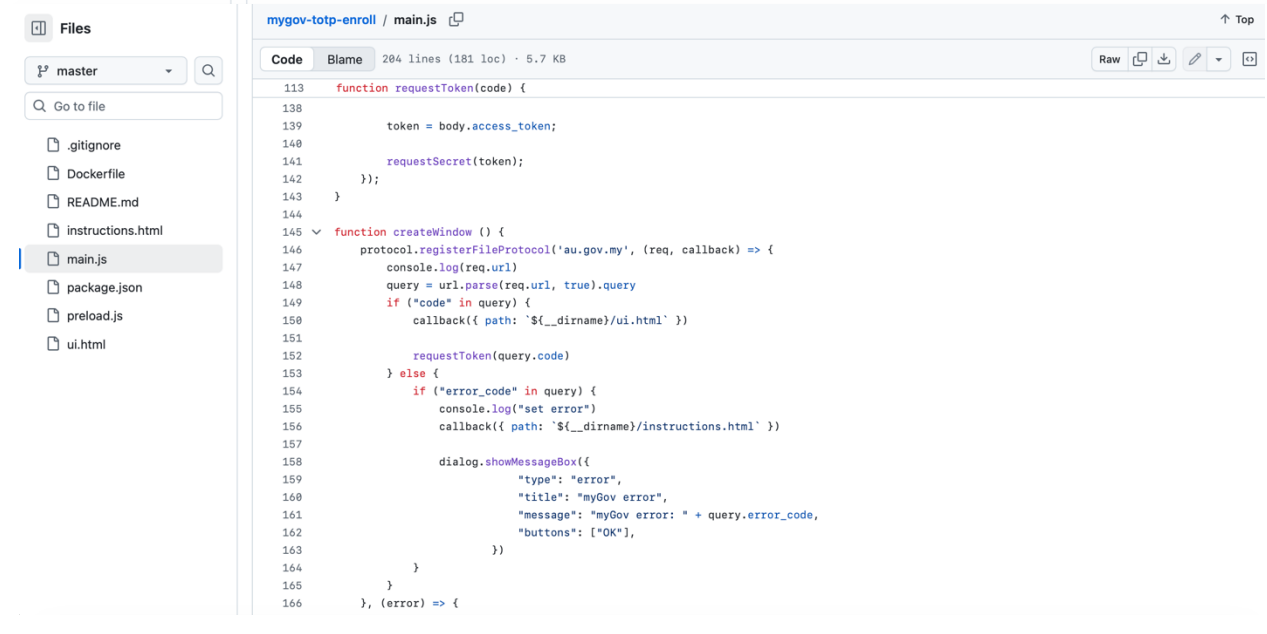

abrasive / mygov-totp-enroll Public
Notifications
Fork 8
Star 108
Code
Issues 8
Pull requests 2
Actions
Security and quality
Insights
Files
master
Go to file
.gitignore
Dockerfile
README.md
instructions.html
main.js
package.json
preload.js
ui.html
mygov-totp-enroll / main.js
lachlanhunt and abrasive Upgrade Electron
c755449 · 3 years ago
History
Code
Blame
204 lines (181 loc) · 5.7 KB
Raw
1 const { app, protocol, BrowserWindow, ipcMain, dialog } = require('electron');
2 const url = require('url');
3 const request = require('request');
4 const qrcode = require('qrcode');
5 const { base32, base64 } = require('rfc4648');
6 const path = require("node:path");
7
8 let win
9 let client_id='g2c2pjLUThOaBumECqbf'
10 let token
11
12 function setStatus(stat) {
13 win.webContents.send("status", stat);
14 }
15 function setDetail(detail) {
16 win.webContents.send("detail", detail);
17 }
18 function setError(headline, detail) {
19 setStatus("ERROR: " + headline);
20 setDetail(detail);
21 }

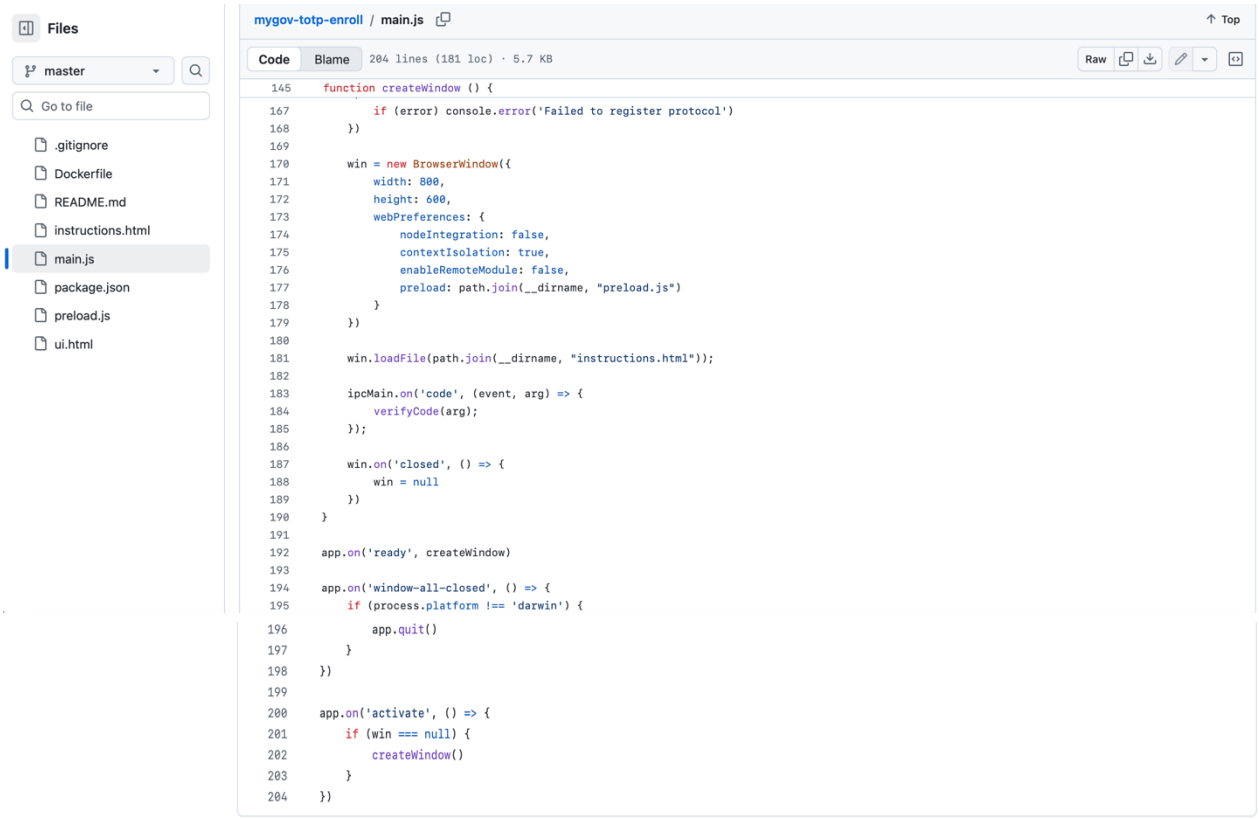

Files
master
Go to file
.gitignore
Dockerfile
README.md
instructions.html
main.js
package.json
preload.js
ui.html
mygov-totp-enroll / main.js
Top
Code
Blame
204 lines (181 loc) · 5.7 KB
Raw
22 function showCodeForm(on) {
23 if (on) {
24 win.webContents.send("showform", "block");
25 } else {
26 win.webContents.send("showform", "none");
27 }
28 }
29
30 function verifyCode(code) {
31 const options = {
32 method: 'POST',
33 url: 'https://api.my.gov.au/authbiz-ext-sec/api/v1/authclients/g2c2pjLUThOaBumECqbf/totpverify.json',
34 headers: {
35 'content-type': 'application/json',
36 'Authorization': 'Bearer ' + token
37 },
38 body: {
39 password: code,
40 },
41 json: true,
42 };
43
44 request(options, function (error, response, body) {
45 if (error) {
46 setError("Code verification failed, try again", error);
47 console.log("error: " + error)
48 return;
49 }
50 if (body == null) { // lol
51 setStatus("Enrolment complete! DON'T LOSE THE CODE");


Files
master
Go to file
.gitignore
Dockerfile
README.md
instructions.html
main.js
package.json
preload.js
ui.html
mygov-totp-enroll / main.js
Top
Code
Blame
204 lines (181 loc) · 5.7 KB
Raw
30 function verifyCode(code) {
52 showCodeForm(false);
53 return;
54 } else if (body.error) {
55 setError("Code verification failed, try again", body.error + ' - ' + body.error_description);
56 console.log("body error: " + body.error)
57 return;
58 } else {
59 console.log("body?? " + body)
60 console.dir(body)
61 msg = "myGov error: " + body.info[0].code + " " + body.info[0].message
62 msg += "\n Make sure you're using an OTP tool that supports SHA1 keys!"
63 dialog.showMessageBox({
64 "type": "error",
65 "title": "myGov error",
66 "message": msg,
67 "buttons": ["OK"],
68 })
69 }
70 });
71 }
72
73 function makeQR(secret) {
74 secret32 = base32.stringify(base64.parse(secret))
75 secret32 = secret32.replace(/=+$/, "")
76 totp_uri = 'otpauth://totp/myGov?secret=' + secret32 + '&algorithm=SHA512'
77
78 qrcode.toDataURL(totp_uri)
79 .then(url => {
80 setStatus("Enroll this secret in your TOTP client and enter the current code");


Files
master
Go to file
.gitignore
Dockerfile
README.md
instructions.html
main.js
package.json
preload.js
ui.html
mygov-totp-enroll / main.js
Top
Code
Blame
204 lines (181 loc) · 5.7 KB
Raw
73 function makeQR(secret) {
81 setDetail(totp_uri + '<p>');
82 showCodeForm(true);
83 })
84 .catch(err => {
85 setError("QR encoding error", err);
86 })
87 }
88
89 function requestSecret(token) {
90 setStatus("Requesting OAuth secret...");
91 const options = {
92 method: 'POST',
93 url: 'https://api.my.gov.au/authbiz-ext-sec/api/v1/authclients/g2c2pjLUThOaBumECqbf/totpcredential.json',
94 headers: {'Authorization': 'Bearer ' + token},
95 };
96 request(options, function (error, response, body) {
97 if (error) {
98 setError("Secret request failed", error);
99 return;
100 }
101 body = JSON.parse(body)
102 if (body.error) {
103 setError("Secret request failed",
104 body.error + ' - ' + body.error_description);
105 return;
106 }
107
108 secret = body.secret;
109 makeQR(secret);


Files
master
Go to file
.gitignore
Dockerfile
README.md
instructions.html
main.js
package.json
preload.js
ui.html
mygov-totp-enroll / main.js
Top
Code
Blame
204 lines (181 loc) · 5.7 KB
Raw
89 function requestSecret(token) {
110 });
111 }
112
113 function requestToken(code) {
114 setStatus("Requesting OAuth token...");
115
116 const options = {
117 method: 'POST',
118 url: 'https://auth.my.gov.au/mga/sps/oauth/oauth20/token',
119 form: {
120 client_id: client_id,
121 redirect_uri: 'au.gov.my://app',
122 grant_type: 'authorization_code',
123 code: code,
124 },
125 };
126
127 request(options, function (error, response, body) {
128 if (error) {
129 setError("Token request failed", error);
130 return;
131 }
132 body = JSON.parse(body)
133 if (body.error) {
134 setError("Token request failed",
135 body.error + ' - ' + body.error_description);
136 return;
137 }
138


Files
master
Go to file
.gitignore
Dockerfile
README.md
instructions.html
main.js
package.json
preload.js
ui.html
mygov-totp-enroll / main.js
Top
Code
Blame
204 lines (181 loc) · 5.7 KB
Raw
113 function requestToken(code) {
138
139 token = body.access_token;
140
141 requestSecret(token);
142 });
143 }
144
145 function createWindow () {
146 protocol.registerFileProtocol('au.gov.my', (req, callback) => {
147 console.log(req.url)
148 query = url.parse(req.url, true).query
149 if ("code" in query) {
150 callback({ path: `${__dirname}/ui.html` })
151
152 requestToken(query.code)
153 } else {
154 if ("error_code" in query) {
155 console.log("set error")
156 callback({ path: `${__dirname}/instructions.html` })
157
158 dialog.showMessageBox({
159 "type": "error",
160 "title": "myGov error",
161 "message": "myGov error: " + query.error_code,
162 "buttons": ["OK"],
163 })
164 }
165 }
166 }, (error) => {



Lodged on behalf of the Applicant

Applicant contact: Jason Bosland

Prepared by: Jason Bosland

Wise Law
141A Como Parade East
Parkdale Vic 3195

Telephone: 0421 193 711

Email: JB@wiselaw.com.au

ADMINISTRATIVE REVIEW TRIBUNAL
MELBOURNE REGISTRY
INTELLIGENCE AND SECURITY JURISDICTIONAL AREA

2025/4916

FRASER TWEEDALE
Applicant

CEO, SERVICES AUSTRALIA
Respondent

ANNEXURE FT-4

The following is the annexure marked **FT-4** referred to in the affidavit of Fraser Tweedale, affirmed at Brisbane on 2 July 2026. This annexure contains screenshots of the 'issues' and 'pull request' entries on the GitHub repository, 'mygov-totp-enroll', in which users discuss how to use the repository to enable TOTP authentication in the App, as at 1 July 2026: https://github.com/abrasive/mygov-totp-enroll/issues/9; https://github.com/abrasive/mygov-totp-enroll/issues/11; https://github.com/abrasive/mygov-totp-enroll/pull/5; https://github.com/abrasive/mygov-totp-enroll/issues/1

.....
Jason Bosland
Australian Legal Practitioner
Melbourne

abrasive / mygov-totp-enroll Public

Code | Issues 8 | Pull requests 2 | Actions | Security and quality | ...

New issue

Not an issue: just thanks! #9

Open

ramebd opened on Jun 17, 2021

Sorry, couldn't leave a comment on the reddit thread I found, but this is just plain awesome, and thanks for all your work!
Worked flawlessly, got myGov all set up with andOTP and no more relying on sms 2fa.

Thanks! 😄

👍 6 ❤️ 2

VirtualWolf on Jun 21, 2021

Seconding this!

DuBistKomisch on Jul 8, 2021

thirding

n3ih7 on Jul 30, 2021

Still works two years after

Lord-Evil on Sep 29, 2022

Found it today, awesome!

abrasive / mygov-totp-enroll Public

Code | Issues 8 | Pull requests 2 | Actions | Security and quality

New issue

Doesn't seem to work any longer? #11

Open

gsklee opened on Sep 15, 2021

This is what I am seeing in the console when clicking on the "Register" button on the "Finish registration" screen:

```
au.gov.my://app?error=invalid_request&error_description=FBTOAU202E+The+required+parame
```

And nothing happens. Looks like it's broken?

jimtsao on Oct 3, 2021

Just set mine up today. Still works for me.

phik on Jun 6, 2022

Also worked for me today. Big thanks for this. You're a legend!

Scrub000 on Oct 1, 2024 — Last edited by Scrub000

Can confirm; no longer working. Patched by myGov.

midnightveil on Jul 18, 2025

Works for me, still. Needed this patch to the code due to electron version changes but otherwise fine:

```
diff --git a/main.js b/main.js
index 6e71652..fabb42d 100644
--- a/main.js
+++ b/main.js
@@ -171,9 +171,11 @@ function createWindow () {
         width: 800,
         height: 600,
         webPreferences: {
-            nodeIntegration: false,
-            contextIsolation: true,
-            enableRemoteModule: false,
+            nodeIntegration: true,
+            nodeIntegrationInWorker: true,
+            nodeIntegrationInSubFrames: true,
+            enableRemoteModule: true,
+            contextIsolation: false,
             preload: path.join(__dirname, "preload.js")
         }
     })
```

abrasive / mygov-totp-enroll Public

Code | Issues 8 | Pull requests 2 | Actions | Security and quality

Add Dockerfile #5

Merged abrasive merged 1 commit into `abrasive:master` from `alexlance:add-dockerfile` on Dec 30, 2019

Conversation 1 | Commits 1 | Checks 0 | Files changed 2

alexlance commented on Dec 27, 2019 — Contributor

Fantastic project, super happy to have found a way around installing the mygov app and stick to normal, reliable totp/mfa. THANK YOU.

No worries if you don't want to merge this PR, but thought a Dockerfile which installs the dependencies (and node) might be useful.

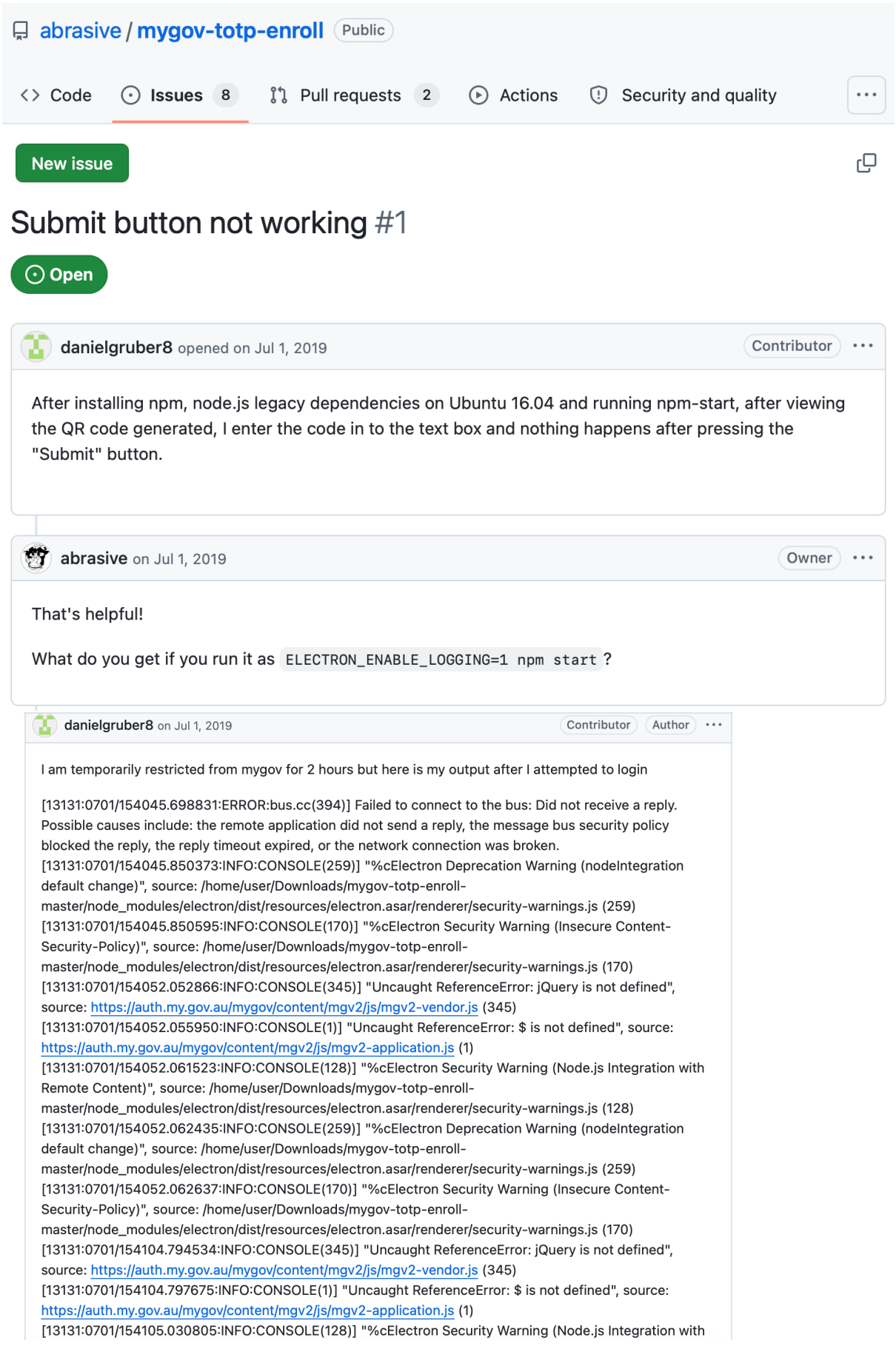
Cheers!

`Add Dockerfile` 63dcdf6

abrasive commented on Dec 30, 2019 — Owner

Nice! Cheers :D

abrasive merged commit **ebf1b78** into `abrasive:master` on Dec 30, 2019


abrasive / mygov-totp-enroll
Public
Code
Issues 8
Pull requests 2
Actions
Security and quality
New issue
Submit button not working #1
Open
danielgruber8 opened on Jul 1, 2019
Contributor
After installing npm, node.js legacy dependencies on Ubuntu 16.04 and running npm-start, after viewing the QR code generated, I enter the code in to the text box and nothing happens after pressing the "Submit" button.
abrasive on Jul 1, 2019
Owner
That's helpful!
What do you get if you run it as ELECTRON_ENABLE_LOGGING=1 npm start?
danielgruber8 on Jul 1, 2019
Contributor
Author
I am temporarily restricted from mygov for 2 hours but here is my output after I attempted to login
[13131:0701/154045.698831:ERROR:bus.cc(394)] Failed to connect to the bus: Did not receive a reply. Possible causes include: the remote application did not send a reply, the message bus security policy blocked the reply, the reply timeout expired, or the network connection was broken.
[13131:0701/154045.850373:INFO:CONSOLE(259)] "%cElectron Deprecation Warning (nodeIntegration default change)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (259)
[13131:0701/154045.850595:INFO:CONSOLE(170)] "%cElectron Security Warning (Insecure Content-Security-Policy)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (170)
[13131:0701/154052.052866:INFO:CONSOLE(345)] "Uncaught ReferenceError: jQuery is not defined", source: https://auth.my.gov.au/mygov/content/mgv2/js/mgv2-vendor.js (345)
[13131:0701/154052.055950:INFO:CONSOLE(1)] "Uncaught ReferenceError: $ is not defined", source: https://auth.my.gov.au/mygov/content/mgv2/js/mgv2-application.js (1)
[13131:0701/154052.061523:INFO:CONSOLE(128)] "%cElectron Security Warning (Node.js Integration with Remote Content)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (128)
[13131:0701/154052.062435:INFO:CONSOLE(259)] "%cElectron Deprecation Warning (nodeIntegration default change)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (259)
[13131:0701/154052.062637:INFO:CONSOLE(170)] "%cElectron Security Warning (Insecure Content-Security-Policy)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (170)
[13131:0701/154104.794534:INFO:CONSOLE(345)] "Uncaught ReferenceError: jQuery is not defined", source: https://auth.my.gov.au/mygov/content/mgv2/js/mgv2-vendor.js (345)
[13131:0701/154104.797675:INFO:CONSOLE(1)] "Uncaught ReferenceError: $ is not defined", source: https://auth.my.gov.au/mygov/content/mgv2/js/mgv2-application.js (1)
[13131:0701/154105.030805:INFO:CONSOLE(128)] "%cElectron Security Warning (Node.js Integration with

Security-Policy)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (170)
[13131:0701/154104.794534:INFO:CONSOLE(345)] "Uncaught ReferenceError: jQuery is not defined", source: https://auth.my.gov.au/mygov/content/mgv2/js/mgv2-vendor.js (345)
[13131:0701/154104.797675:INFO:CONSOLE(1)] "Uncaught ReferenceError: $ is not defined", source: https://auth.my.gov.au/mygov/content/mgv2/js/mgv2-application.js (1)
[13131:0701/154105.030805:INFO:CONSOLE(128)] "%cElectron Security Warning (Node.js Integration with Remote Content)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (128)
[13131:0701/154105.033220:INFO:CONSOLE(259)] "%cElectron Deprecation Warning (nodeIntegration default change)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (259)
[13131:0701/154105.033682:INFO:CONSOLE(170)] "%cElectron Security Warning (Insecure Content-Security-Policy)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (170)

danielgruber8 on Jul 2, 2019 Contributor Author

Here is the full log since unlocked from myGov

user@tux:~/Downloads/mygov-totp-enroll-master$ ELECTRON_ENABLE_LOGGING=1 npm start

> mygov-totp-enroll@0.1.0 start /home/user/Downloads/mygov-totp-enroll-master
> electron .

[5009:0702/105149.612664:INFO:CONSOLE(259)] "%cElectron Deprecation Warning (nodeIntegration default change)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (259)
[5009:0702/105149.612907:INFO:CONSOLE(170)] "%cElectron Security Warning (Insecure Content-Security-Policy)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (170)
[5009:0702/105153.295575:INFO:CONSOLE(345)] "Uncaught ReferenceError: jQuery is not defined", source: https://auth.my.gov.au/mygov/content/mgv2/js/mgv2-vendor.js (345)
[5009:0702/105153.298488:INFO:CONSOLE(1)] "Uncaught ReferenceError: $ is not defined", source: https://auth.my.gov.au/mygov/content/mgv2/js/mgv2-application.js (1)
[5009:0702/105154.027935:INFO:CONSOLE(128)] "%cElectron Security Warning (Node.js Integration with Remote Content)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (128)
[5009:0702/105154.028858:INFO:CONSOLE(259)] "%cElectron Deprecation Warning (nodeIntegration default change)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (259)
[5009:0702/105154.029018:INFO:CONSOLE(170)] "%cElectron Security Warning (Insecure Content-Security-Policy)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (170)

[5009:0702/105204.585167:INFO:CONSOLE(345)] "Uncaught ReferenceError: jQuery is not defined", source: https://auth.my.gov.au/mygov/content/mgv2/js/mgv2-vendor.js (345)
[5009:0702/105204.588237:INFO:CONSOLE(1)] "Uncaught ReferenceError: $ is not defined", source: https://auth.my.gov.au/mygov/content/mgv2/js/mgv2-application.js (1)
[5009:0702/105205.843714:INFO:CONSOLE(128)] "%cElectron Security Warning (Node.js Integration with Remote Content)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (128)
[5009:0702/105205.845080:INFO:CONSOLE(259)] "%cElectron Deprecation Warning (nodeIntegration default change)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (259)
[5009:0702/105205.845265:INFO:CONSOLE(170)] "%cElectron Security Warning (Insecure Content-Security-Policy)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (170)
[5009:0702/105214.135939:INFO:CONSOLE(345)] "Uncaught ReferenceError: jQuery is not defined", source: https://auth.my.gov.au/mygov/content/mgv2/js/mgv2-vendor.js (345)
[5009:0702/105214.138827:INFO:CONSOLE(1)] "Uncaught ReferenceError: $ is not defined", source: https://auth.my.gov.au/mygov/content/mgv2/js/mgv2-application.js (1)
[5009:0702/105214.144003:INFO:CONSOLE(128)] "%cElectron Security Warning (Node.js Integration with Remote Content)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (128)
[5009:0702/105214.144999:INFO:CONSOLE(259)] "%cElectron Deprecation Warning (nodeIntegration default change)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (259)
[5009:0702/105214.145121:INFO:CONSOLE(170)] "%cElectron Security Warning (Insecure Content-Security-Policy)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (170)
[5009:0702/105218.441059:INFO:CONSOLE(259)] "%cElectron Deprecation Warning (nodeIntegration default change)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (259)

[5009:0702/105218.441110:INFO:CONSOLE(170)] "%cElectron Security Warning (Insecure Content-Security-Policy)", source: /home/user/Downloads/mygov-totp-enroll-master/node_modules/electron/dist/resources/electron.asar/renderer/security-warnings.js (170)

abrasive mentioned this on Jul 10, 2019

windows asking for an app to open au.gov.my #2

BeauGiles on Jul 28, 2019

Experiencing the same thing here on macOS - nothing happens after hitting submit.

abrasive on Jul 29, 2019 Owner

@BeauGiles can you try the latest master?

BeauGiles on Aug 1, 2020

@abrasive figured I should probably reply so this can be closed 👍
Can confirm that the latest master is fine on the latest version of macOS Catalina - I was able to add the myGov OTP to 1Password - as both Authy & Microsoft Authenticator *do not* support the SHA512 algorithm.

Might be worth making it more obvious what authenticator apps do or do not support SHA512 required by myGov

BeauGiles on Jun 22, 2021

This works with the built in iOS 15 / iCloud Keychain code generator.

ADMINISTRATIVE REVIEW TRIBUNAL
MELBOURNE REGISTRY
INTELLIGENCE AND SECURITY JURISDICTIONAL AREA

2025/4916

FRASER TWEEDALE
Applicant

CEO, SERVICES AUSTRALIA
Respondent

ANNEXURE FT-5

The following is the annexure marked **FT-5** referred to in the affidavit of Fraser Tweedale, affirmed at Brisbane on 2 July 2026. This annexure contains screenshots of the GitHub gist posted by hacker1024, including comments from other users as at July 1 2026: https://gist.github.com/hacker1024/5d0845863e2dced27fd5eebc4ac95a39.

.....
Jason Bosland
Australian Legal Practitioner
Melbourne

hacker1024 / **mygov_totp_storage.md**

Last active last year

Information about TOTP token storage in the myGov Code Generator app, as well as token extraction instructions.

mygov_totp_storage.md

This gist contains information about TOTP token storage in the myGov Code Generator app, along with instructions on how to extract tokens.

App data

Structure

```
/data/user/0/au.gov.dhs.centrelink.mygovauthenticator:
├──files
│       myGov.ks
│       sharedSecret
│
└──shared_prefs
        au.gov.dhs.centrelink.mygovauthenticator.prefs_file.xml
```

Details

- `myGov.ks` : A BKS keystore containing a private RSA-256 key used to decrypt the contents of `sharedSecret` (after decoding the base64 data). The key is called `sharedSecret`, and uses the hard-coded password of `km5QzJJ5NhfGymfp`.
- `sharedSecret` : The encrypted TOTP token. The encrypted data is stored in base64 form, and decrypts to more base64.
- `au.gov.dhs.centrelink.mygovauthenticator.prefs_file.xml` : An XML file containing the IV used to decrypt `sharedSecret` along with the key in `myGov.ks`, as well as the `myGov.ks` keystore password.

TOTP format

The TOTP token must be used with the SHA512 algorithm, and the standard 6-digit length and 30 second period.

Example URI: `otpauth://totp/myGov?secret=<BASE32_ENCODED_SECRET>&algorithm=SHA512`

Note that some apps like Google Authenticator and Authy do not handle SHA512 properly. BitWarden, for example, does.

Manual instructions

1. Gain access to the files shown above
2. Use the `keyStorePwd` in `au.gov.dhs.centrelink.mygovauthenticator.prefs_file.xml` to open `myGov.ks` with a tool like KeyStore Explorer
3. Use the password `km5QzJJ5NhfGymfp` to access the `sharedSecret` key

At this point, you can use this CyberChef recipe to generate a URI, or continue manually:

5. Decode the base64 data in the `sharedSecret` file
6. Use the `sharedSecret` key, along with the `sharedSecret_iv` in `au.gov.dhs.centrelink.mygovauthenticator.prefs_file.xml`, to decrypt the decoded `sharedSecret` file contents with AES-256-CBC
7. Convert the decrypted `sharedSecret` file contents from base64 to base32, removing any `=` padding from the end.
8. Generate a URI with the properties specified above

soraxas commented on Sep 6, 2021

Thank you SO MUCH for the instruction!! The instructions are very detailed and the recommended tools had been great as well. Specifically, you had created the CyberChef to help automate the process, which had worked faultlessly. Thanks heaps and it helps to consolidate all my TOTP :)

kylemd commented on Feb 2, 2022

Thank you for this! The myGov Authenticator app is awful

K1ngJony commented on Jul 20, 2022

Any advice on how to get the files in the first place? I've tried adb backup but it comes up empty

hacker1024 commented on Aug 31, 2022 Author

> Any advice on how to get the files in the first place? I've tried adb backup but it comes up empty

@K1ngJony You'll need a rooted device (or one with file access from a recovery environment). Perhaps an Android emulator would work as well - I'm not sure if the app uses SafetyNet.

Jarodwr commented on Jul 7, 2023

Attempted to do this from bluestacks and didn't have any success, couldn't get access to the app data at that path (I assume it's to do with that safetynet thing you talked about)

rogerkeays commented on Aug 17, 2023 • edited

Sublime.

I only hit a couple of obstacles. Firstly, I found the files in `/data/data/au.gov.dhs.centrelink.mygovauthenticator` on my device. Secondly, I couldn't cut and paste the secret directly into my authenticator because the textbox truncated the data. I converted the URI to a QR code using https://qr-creator.com/url.php and it worked perfectly.

Thanks for this. Love your work!

ADMINISTRATIVE REVIEW TRIBUNAL
MELBOURNE REGISTRY
INTELLIGENCE AND SECURITY JURISDICTIONAL AREA

2025/4916

FRASER TWEEDALE
Applicant

CEO, SERVICES AUSTRALIA
Respondent

ANNEXURE FT-6

The following is the annexure marked **FT-6** referred to in the affidavit of Fraser Tweedale, affirmed at Brisbane on 2 July 2026. This annexure contains screenshots of the subsequent revisions made to the original gist published by hacker1024 as at July 1 2026: https://gist.github.com/hacker1024/5d0845863e2dced27fd5eebc4ac95a39/revisions.

.....
Jason Bosland
Australian Legal Practitioner
Melbourne

Instantly share code, notes, and snippets.

hacker1024 / mygov_totp_storage.md

Last active last year

Code Revisions 2 Stars 8 Forks 1 Embed Download ZIP

Revisions

Split Unified

hacker1024 **renamed** this gist on Apr 27, 2021. **1 changed file** with **0 additions** and **0 deletions.**

0 instructions.md → mygov_totp_storage.md

File renamed without changes.

hacker1024 **created** this gist on Apr 27, 2021.

54 instructions.md

```
@@ -0,0 +1,54 @@
+ This gist contains information about TOTP token storage in the
+ [myGov Code Generator](https://play.google.com/store/apps/details?id=au.gov.dhs.centrelink.mygovauthenticator)
+ app, along with instructions on how to extract tokens.
+
+ ### App data
+ #### Structure
+ ```
+ /data/user/0/au.gov.dhs.centrelink.mygovauthenticator:
+ ├───files
+ │       myGov.ks
+ │       sharedSecret
+ │
+ └───shared_prefs
+         au.gov.dhs.centrelink.mygovauthenticator.prefs_file.xml
+ ```
+ #### Details
+ - `myGov.ks`: A BKS keystore containing a private RSA-256 key used to decrypt
+   the contents of `sharedSecret` (after decoding the base64 data).
```

```
+   The key is called `sharedSecret`, and uses the hard-coded password of
+   `km5QzJJ5NhfGymfp`.
+ - `sharedSecret`: The encrypted TOTP token. The encrypted data is stored in
+   base64 form, and decrypts to more base64.
+ - `au.gov.dhs.centrelink.mygovauthenticator.prefs_file.xml`: An XML file
+   containing the IV used to decrypt `sharedSecret` along with the key in
+   `myGov.ks`, as well as the `myGov.ks` keystore password.
+
+ ### TOTP format
+ The TOTP token must be used with the SHA512 algorithm, and the standard 6-
  digit
+ length and 30 second period.
+
+ Example URI:
+ `otpauth://totp/myGov?secret=<BASE32_ENCODED_SECRET>&algorithm=SHA512`
+
+ Note that some apps like Google Authenticator and Authy do not handle SHA512
  properly.
+ BitWarden, for example, does.
+
+ ### Manual instructions
+ 1) Gain access to the files shown above
+ 2) Use the `keyStorePwd` in
  `au.gov.dhs.centrelink.mygovauthenticator.prefs_file.xml` to open `myGov.ks`
  with a tool like [KeyStore Explorer]
+ 3) Use the password `km5QzJJ5NhfGymfp` to access the `sharedSecret` key
+
+ At this point, you can use this
+ [CyberChef recipe]
  (https://gchq.github.io/CyberChef/#recipe=Comment('Put%20the%20contents%20of
  %20the%20sharedSecret%20file%20in%20the%20input%20box.')From_Base64('A-Za-
  z0-
  9%2B/%3D',false)Comment('Add%20the%20sharedSecret%20RSA%20key%20from%20myGov
  .ks%20in%20the%20%22Key%22%20field,%20and%20the%20sharedSecret_iv%20string%2
  0from%20au.gov.dhs.centrelink.mygovauthenticator.prefs_file.xml%20in%20the%2
  0%22IV%22%20field.')AES_Decrypt(%7B'option':'Hex','string':''%7D,%7B'option'
  :'Base64','string':''%7D,'CBC','Raw','Raw',%7B'option':'Hex','string':''%7D,
  %7B'option':'Hex','string':''%7D)From_Base64('A-Za-z0-
  9%2B/%3D',false)To_Base32('A-Z2-
  7%3D')Find_/_Replace(%7B'option':'Regex','string':'(.*?)%3D*$'%7D,'otpauth:/
  /totp/myGov?
  secret%3D$1%26algorithm%3DSHA512',false,false,false,false)&input=Rm9sbG93IHR
  oZSBpbnN0cnVjdGlvbnMgaW4gdGhlIHJlY2lwZSBjb21tZW50cywgYW5kIHJlcGxhY2UgdGhpcyB
  0ZXh0IHdpdGggeW91ciBpbnB1dCE)
+ to generate a URI, or continue manually:
+
```

```
+ 5) Decode the base64 data in the `sharedSecret` file
+ 6) Use the `sharedSecret` key, along with the `sharedSecret_iv` in
+    `au.gov.dhs.centrelink.mygovauthenticator.prefs_file.xml`, to decrypt the
+    decoded `sharedSecret` file contents with AES-256-CBC
+ 7) Convert the decrypted `sharedSecret` file contents from base64 to base32,
+    removing any `=` padding from the end.
+ 8) Generate a URI with the properties specified above
+
+ [KeyStore Explorer]: https://keystore-explorer.org/
```

ADMINISTRATIVE REVIEW TRIBUNAL
MELBOURNE REGISTRY
INTELLIGENCE AND SECURITY JURISDICTIONAL AREA

2025/4916

FRASER TWEEDALE
Applicant

CEO, SERVICES AUSTRALIA
Respondent

ANNEXURE FT-7

The following is the annexure marked **FT-7** referred to in the affidavit of Fraser Tweedale, affirmed at Brisbane on 2 July 2026. This annexure contains a screenshot of the initial Reddit post by user 'j_l-w' in the r/Australia forum as at 1 July 2026:
https://www.reddit.com/r/australia/comments/bc6gmv/for_all_the_mygov_fake_2fa_haters/

.....
Jason Bosland
Australian Legal Practitioner
Melbourne

r/australia • 7y ago
j_l-w

for all the myGov fake 2FA haters

no politics

myGov actually supports TOTP, but only via their own app, so I got really grumpy and made this:

https://github.com/abrasive/mygov-totp-enroll

It allows you to enroll for their TOTP and displays your credentials as a QR code in the traditional way. (It's a stupid long key, you don't want to type that in a phone by hand.)

It's not distributed as a ready-to-run app because how could you trust it? - so you'll need npm to run it. Sorry. It's short enough that it should be trivial to audit, but good luck with the dependencies –^

Archived post. New comments cannot be posted and votes cannot be cast.

22 39

ADMINISTRATIVE REVIEW TRIBUNAL
MELBOURNE REGISTRY
INTELLIGENCE AND SECURITY JURISDICTIONAL AREA

2025/4916

FRASER TWEEDALE
Applicant

CEO, SERVICES AUSTRALIA
Respondent

ANNEXURE FT-8

The following is the annexure marked **FT-8** referred to in the affidavit of Fraser Tweedale, affirmed at Brisbane on 2 July 2026. This annexure contains a screenshot of the comments made by user, 'Thingface', on the Reddit post of 'j_l-w' as at 1 July 2026:
https://www.reddit.com/r/australia/comments/bc6gmv/for_all_the_mygov_fake_2fa_haters/

.....
Jason Bosland
Australian Legal Practitioner
Melbourne

Thingface • 7y ago • Edited 7y ago

I had been using 2FA with andOTP for ages. I was thinking about writing an app to do this, in Electron too. Now you've done it u/j_l-w I won't have to. On a side note this is what I discovered independently back in December 2018.

The reason I use andOTP and not Authy is because that is proprietary software and not open-source. I also don't install Gapps and I run LineageOS on my phone.

How

I looked at `/data/data/au.gov.dhs.centrelink.mygovauthenticator/files/sharedSecret` however that file is encrypted. When I did an internet search for the filename of the other file in there, `myGov.ks` I found a de-compilation of the app was returned.

To get around this I had to get a copy of the encrypted secret **before** it landed on the phone and the app encrypted it. To do this:

1. I performed a Man-in-the-middle attack on myself in order to strip the encryption off the TLS session (ie peek inside the https). I used Charles Proxy. Video here and configured a HTTP proxy on my phone. This is a HTTP proxy which issues a different certificate that I hold the private key for. That means I can look inside the https encrypted requests. The HTTP proxy handles the connection to MyGov.

2. I was able to add a new root certificate on an old phone which I plan to erase afterwards. It ran Android 6.0 so it was easy. If you're not using an old phone you should make sure you **remove the certificate authority afterwards**.
    - I could have used the Android Emulator instead of a phone if I didn't have an old one. Note in Android 7+ developers have the option of preventing you from using user certificates but adding the certificates to the root store will bypass that. Another method included Intercepting HTTPS Traffic from Apps on Android 7+ using Magisk & Burp fortunately I didn't need to do that either.
    - The MyGov app at this time doesn't employ a network security configuration. They also aren't using certificate pinning either.
    - There are ways around that such as SSLUnpinning_Xposed. This will work as long as they don't decide to get their app to 'check for root', although there are ways to hide root using Magisk should they do that in the future.
    - The system that MyGov uses on their back end seems to be IBM's api_client_totp.
    - I observed that `api.my.gov.au` returns a json file ie `https://api.my.gov.au/authbiz-ext-sec/api/v1/authclients/{{client_id}}/totpcredential.json` which contains the base64 encoded secret.
    - I then converted this to base32 (what andOTP expects) and added it using SHA512 hash. The command I used for converting to base32 was (on my Linux system). The sed portion simply removes the new line:

```
echo "base64 encoded SECRET" | base64 -d | base32 | sed '/^/{N;s/\n//;}'`
```

It works great, now I don't need to use the MyGov app and I am free to use andOTP or whatever I want on my new phone that isn't Android or iOS. It would have been better if MyGov had implemented this with a QR Code.

What I discovered

However the "myGov Access - code creator" will **call home every time you open it** as I said previously. You might be curious to know **what is actually in this transmission**.

This occurs after you've paired the phone with the MyGov account and received the shared secret. The app makes two HTTP GET requests:

- The first is to `https://api.my.gov.au/authbiz-ext-unsec/api/v1/meta/time.json` which returns the current server time in ISO 8601 format ie: `{ "serverTime": "YYYY-MM-DDTHH:MM:SS.sss+0000" }` which I wouldn't have thought was necessary as Android keeps the time up to date with it's service `NetworkTimeUpdateService`, aka Network Time Protocol the way any computer does. There's also NITZ which is a way of doing the same thing over the cellular network without the internet. That's the way that old dumb-phones did it.
- The second is to `https://api.my.gov.au/authbiz-ext-unsec/api/v1/authclients/{{client_id}}/authappmetadata.json` your `{{client_id}}` is unique to the device you signed up with when you logged into your MyGov account with the "myGov Access - code creator" This appears to serve no purpose as the `authappmetadata.json` only returns `{}` ie a .json file with no data.

Thingface • 7y ago

> As my phone isn't rooted, and the default network configuration (for that API level) won't allow user CAs, I had to decompile the apk and add a network configuration to allow it to use my mitmproxy cert. (One can also flip the backupEnabled flag to allow retrieving its stored data on an unrooted phone.)

Ah yes, my phone was rooted and running LineageOS so, simply just installed the certificate into my root store.

> Do you still have your dumps? The client_id in OAuth flows is supposed to represent the client app rather than the end-user, I'd be curious if they are actually using it that way.

- `{{ client_id }}` = `bCrzfdVmkZ` (used the same values as IBM did)
- `code&state` = Appears to be a SHA1 value. I have put in a dummy value `printf $(echo -e code\&state | sha1sum) | sed 's/./\U&/g'` ie `26069F833E2E28347E47C05B2BA4FC57CA85EE6B`
- `{{ device_name }}` = ie `ro.product.model`
- `{{ device_type }}` = `ro.product.manufacturer` + `ro.product.model`
- `response_type` = Appears to be a SHA1 value. I have put in a dummy value `printf $(echo -e response_type | sha1sum) | sed 's/./\U&/g'` ie `3A7DB7703A076241984BD3275115E7FDDBA73045`
- {{ client_secret }} = `DSuCczsEVH` (used the same values as IBM did
- `{{ access_token }}` = `wj0LQ4tPUDTz5LIleVLQkgUc1punUKUfFnBmWy1p` (used the same values as IBM did

Some requests are made:

```
GET /authbiz-ext-unsec/api/v1/meta/time.json HTTP/1.1
GET /authbiz-ext-unsec/api/v1/authclients/bCrzfdVmkZ/authappmetadata.json HTT
GET /mga/sps/oauth/oauth20/authorize?response_type=code&state=26069F833E2E283
GET /las/login?TAM_OP=login&USERNAME=unauthenticated&ERROR_CODE=0x00000000&UR
```

Then there appears to be a few requests GET/POST to

```
 /las/login?execution=e1s1
 /las/login?execution=e1s2
 /las/login?execution=e1s3
```

Few requests with favicon, css, JavaScript. I think for the login part).

Finally:

```
GET /mga/sps/oauth/oauth20/authorize?device_name={{ device_name }}&scope=totp
```

Then a cookie is set.

```
POST /mga/sps/oauth/oauth20/token HTTP/1.1

HTTP/1.1 200 OK
Date: {{ CENSORED }}
Server: HTTPD/1.0.0
X-Frame-Options: SAMEORIGIN
content-language: en-US
content-type: application/json;charset=UTF-8
p3p: CP="NON CUR OTPi OUR NOR UNI"
x-frame-options: DENY
x-content-type-options: nosniff
cache-control: no-cache, no-store, must-revalidate
inst: mygov-auth_1_a
expires: Thu, 01 Dec 1994 16:00:00 GMT
x-xss-protection: 1; mode=block
content-security-policy: frame-ancestors 'none'
strict-transport-security: max-age=2592000
pragma: no-cache
Set-Cookie: JSESSIONID={{ CENSORED }}; Path=/mga/;Secure
Set-Cookie: PD_STATEFUL_{{ CENSORED }}=%2Fmga; Path=/;Secure
Set-Cookie: MYGOVAUTH= {{ CENSORED }}; path=/; Httponly;Secure
Keep-Alive: timeout=15, max=100
Transfer-Encoding: chunked
Connection: keep-alive
```

Then we get the return value

```
--EOF-{{CENSORED TIME}}-
Request-Body:<<--EOF-{{CENSORED TIME}}-
client_id=bCrzfdVmkZ&redirect_uri=au.gov.my%3A%2F%2Fapp&grant_type=authorizat
--EOF-{{CENSORED TIME}}-
Response-Body:<<--EOF-{{CENSORED TIME}}-
{"access_token":"wj0LQ4tPUDTz5LIleVLQkgUc1punUKUfFnBmWy1p","refresh_token":"
--EOF-{{CENSORED TIME}}-
```

Make a request with our bearer token

```
POST /authbiz-ext-sec/api/v1/authclients/bCrzfdVmkZ/totpcredential.json HTTP/
Authorization: Bearer wj0LQ4tPUDTz5LIleVLQkgUc1punUKUfFnBmWy1p


HTTP HEADER
...
...
...

{
  "secret" : "{{ VERY SECRET TOKEN FOR YOUR TOTP APP IN BASE64 }}"
}
```

> The whole thing with using a (modified?) Java KeyStore plus an encrypted blob is weird, what's up with that? Why not just use a normal-sized key and put it straight in the KeyStore?

I'm not sure what they've done with that. The keystore is the correct place for secrets, particularly as that makes the correct API calls. If in the future Google/OEMs make more use of ARM's TrustZone/SecurCore that's the correct way. AndOTP has the option of using the KeyStore 😀.

> After all this it was amusing that putting together a half-usable Electron app took longer than working out the enrollment flow. I remain frustrated that that's the only way (purely because of the quite sensibly implemented restriction on redirect URLs).

I have to admit CharlesProxy is quit good for determining that. I ran it in a VM.

> Also, is that ostorlab link still live? I just get a redirect to their homepage.

No it isn't, you'll have to resubmit, static analysis is free though by the looks of it.
https://ostorlab.co/scan/mobile/

⇧ 1 ⇩

ADMINISTRATIVE REVIEW TRIBUNAL
MELBOURNE REGISTRY
INTELLIGENCE AND SECURITY JURISDICTIONAL AREA

2025/4916

FRASER TWEEDALE
Applicant

CEO, SERVICES AUSTRALIA
Respondent

ANNEXURE FT-9

The following is the annexure marked **FT-9** referred to in the affidavit of Fraser Tweedale, affirmed at Brisbane on 2 July 2026. This annexure contains a screenshot of the follow-up comment made by Reddit user, 'j_l-w', on his post as at 1 July 2026:
https://www.reddit.com/r/australia/comments/bc6gmv/for_all_the_mygov_fake_2fa_haters/

.....
Jason Bosland
Australian Legal Practitioner
Melbourne

j_l-w OP • 7y ago

Thanks for the write-up! I see we took similar approaches, which I suppose should not surprise. ProGuard is an effective nuisance - it's enough to make me think twice before spending the time required to reverse an app statically.

As my phone isn't rooted, and the default network configuration (for that API level) won't allow user CAs, I had to decompile the apk and *add* a network configuration to allow it to use my `mitmproxy` cert. (One can also flip the backupEnabled flag to allow retrieving its stored data on an unrooted phone.)

Do you still have your dumps? The `client_id` in OAuth flows is supposed to represent the client app rather than the end-user, I'd be curious if they are actually using it that way.

The whole thing with using a (modified?) Java KeyStore plus an encrypted blob is weird, what's up with that? Why not just use a normal-sized key and put it straight in the KeyStore?

After all this it was amusing that putting together a half-usable Electron app took longer than working out the enrollment flow. I remain frustrated that that's the only way (purely because of the quite sensibly implemented restriction on redirect URLs).