Contagious Interview Actors Now Utilize JSON Storage Services for Malware Delivery

By Bart Parys

Published: 2025-11-13 · Archived: 2026-04-05 20:19:10 UTC

NVISO reports a new development in the Contagious Interview campaign. The threat actors have recently resorted to utilizing legitimate JSON storage services like JSON Keeper, JSONsilo, and npoint.io to host and deliver malware from trojanized code projects, with the lure being a use case or demo project as part of an interview process.

Background

Contagious Interview is a campaign aligned with Democratic People's Republic of Korea (DPRK) actors that has been active since at least 2023, primarily aimed at financial gain to generate revenue for the regime. The campaign targets software developers across all major operating systems, including Windows, Linux, and macOS, with a particular focus on those involved in cryptocurrency and Web3 projects. Initial access is gained through social engineering tactics, such as ClickFix and fake recruiter profiles, delivering trojanized code during staged job interviews. The most common payloads deployed by this campaign are the BeaverTail and OtterCookie infostealers, along with the InvisibleFerret modular Remote Access Tool (RAT).

An overview of the campaign discussed in this blog post can be seen in Figure 1:

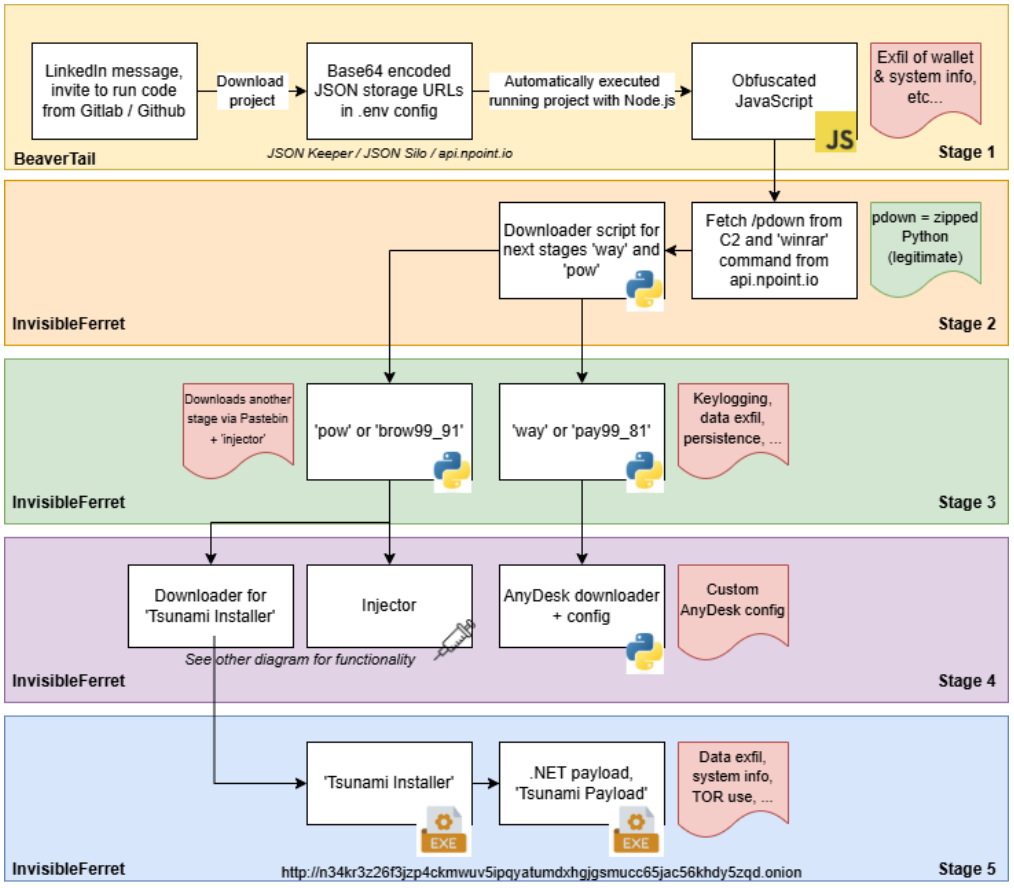

Figure 1 – Overview of the Contagious Interview malware campaign, illustrating the stages from initial contact to malware delivery.

Social Engineering

The campaign typically starts with a fake recruiter reaching out to a potential victim on job searching platforms like LinkedIn. In this identified case, a seemingly medical director reaches out as seen in Figure 2:

Dr. med. Hubert Buschmann

Personal Assistant to Chief Executive Officer | Director Healthcare

Dr. med. Hubert Buschmann

Dear [redacted]

Hope you are doing well. 😁

🚀 I am contacting you because I happened to see your profile and believe that your skills and experience would be useful to our team. Currently, we're developing a next-generation Realtor Platform that will integrate property listings, virtual tours, AI-powered recommendations, secure payment systems, and other key features to streamline real estate transactions.

👉 To support this project, we're looking to hire:

1-2 Blockchain Developers

1-2 Backend Developers

1-2 Frontend Developers

1-2 Backend Developers

1-2 Mobile Developers

Figure 2 – "Dr. med Hubert Buschmann" reaching out to potential hires for a next-generation Realtor Platform development.

From this first message, alarm bells should go off: a medical doctor would not send a message typically sent by recruiters, but even if this were the case, surely it would not be for a new Realtor Platform.

After a few messages back and forth, the good doctor sends over a demo project hosted on GitLab as shown in the next figures.

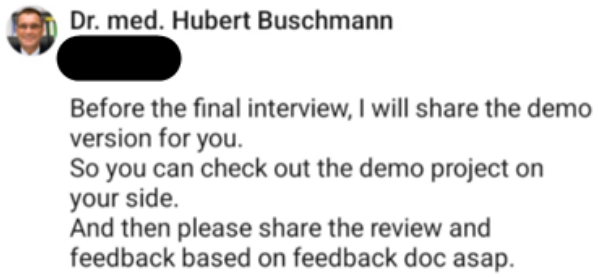

Figure 3 – Time for the demo project?

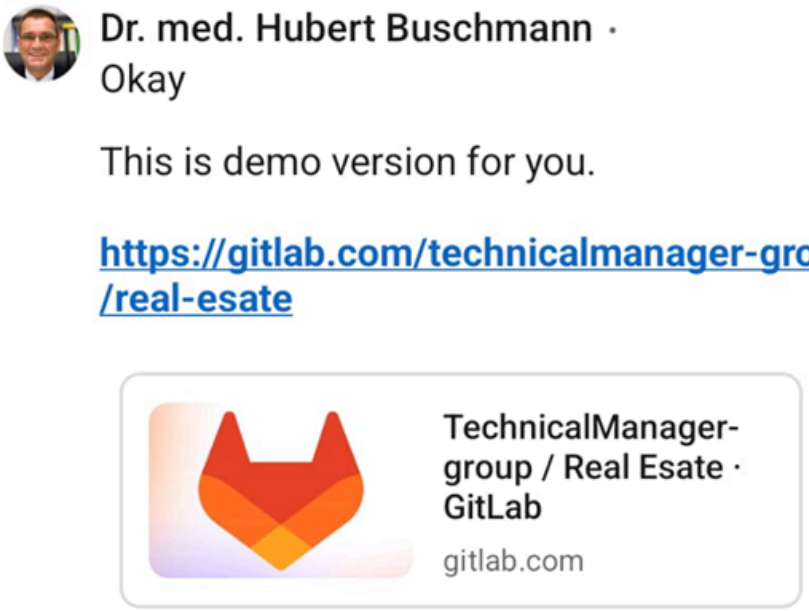

Figure 4 – Link to the ‘demo’ hosted on Gitlab.

Then, the target is intended to run the interview code tasks using Node.JS. A few examples of such demo projects are shown in Figures 5 through 8.

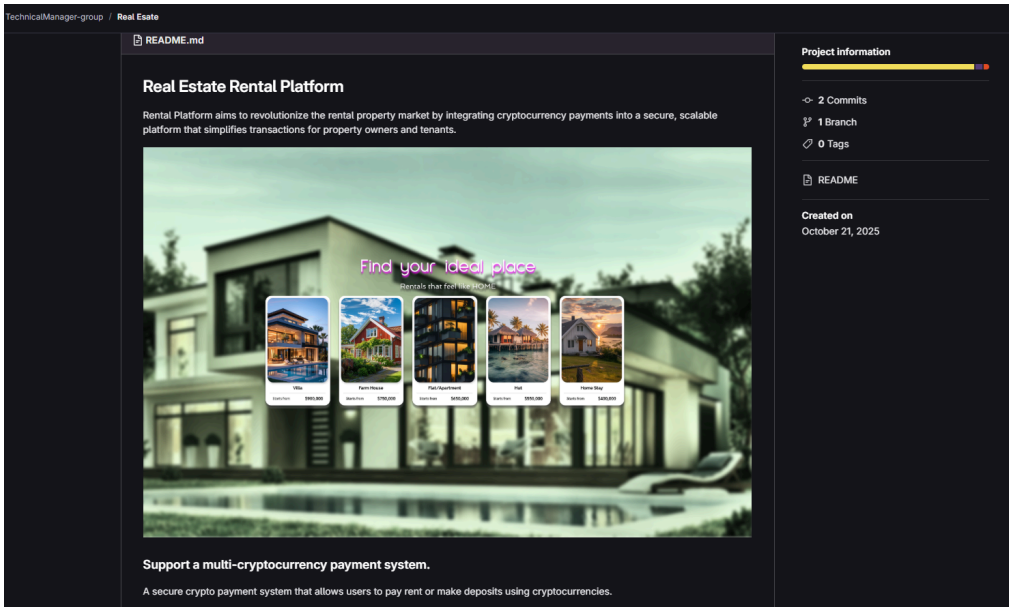

Figure 5 – “Real Estate Rental Platform project”.

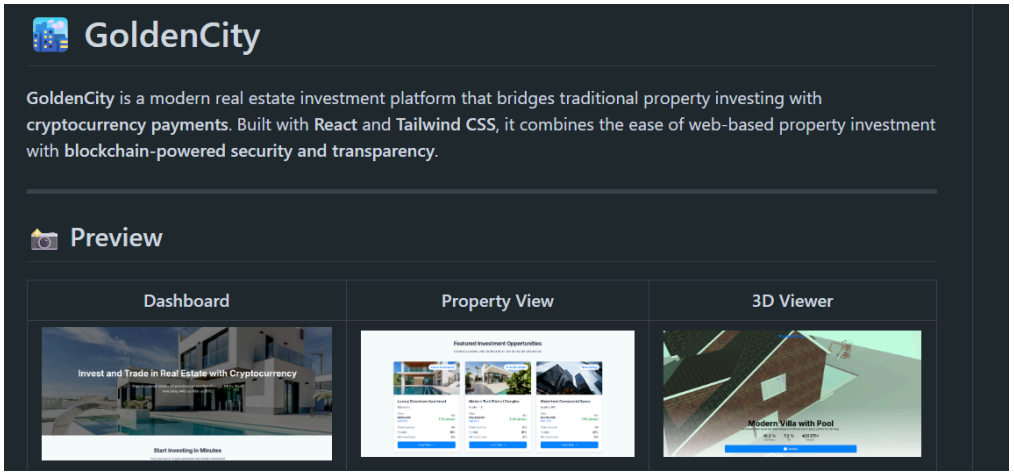

Figure 6 – Another real estate project, this time named ‘GoldenCity’.

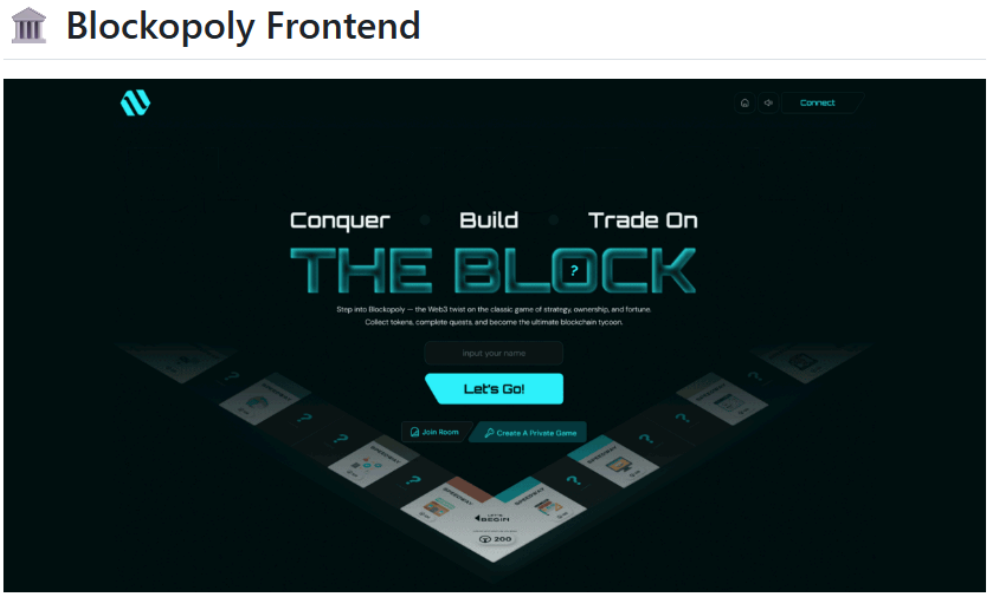

Figure 7 – A Web3 project that’s a twist on the well-known board game, Monopoly.

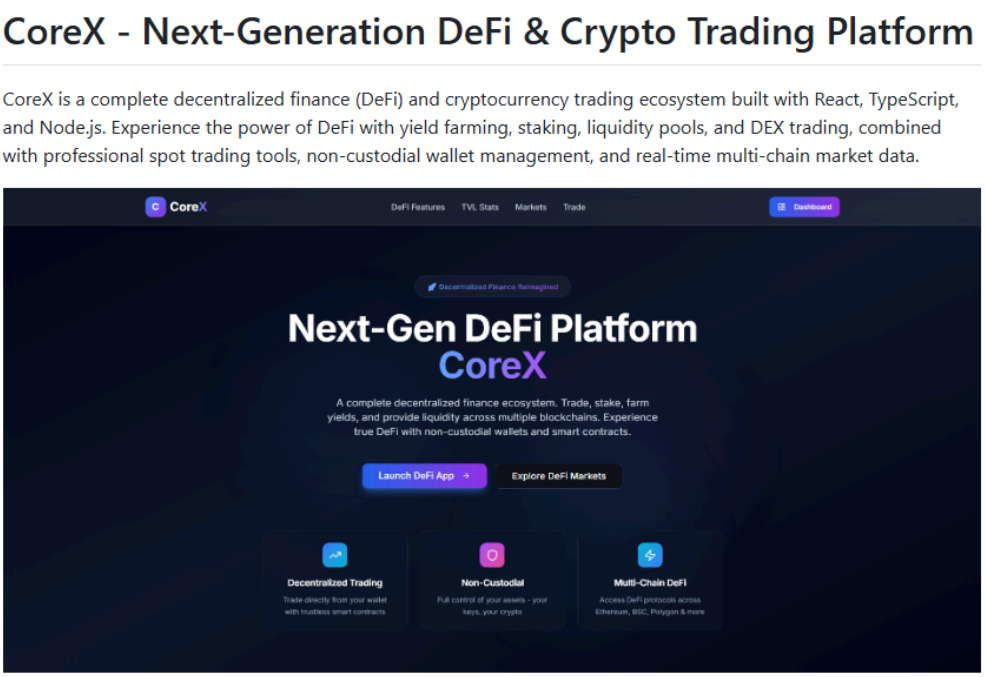

Figure 8 – A decentralized finance ecosystem.

The file `server/config/.config.env` contains a base64-encoded variable that is masqueraded as an API key, which is actually a JSON storage service URL hosting obfuscated code:

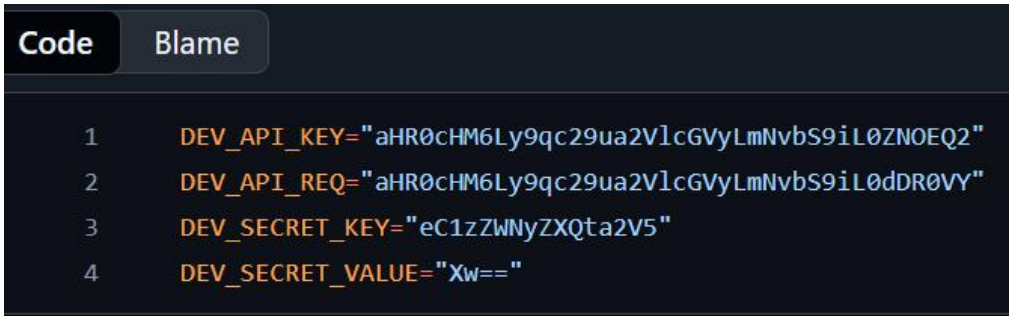

Figure 9 – Base64 encoded variable masquerading as an API key in the config file.

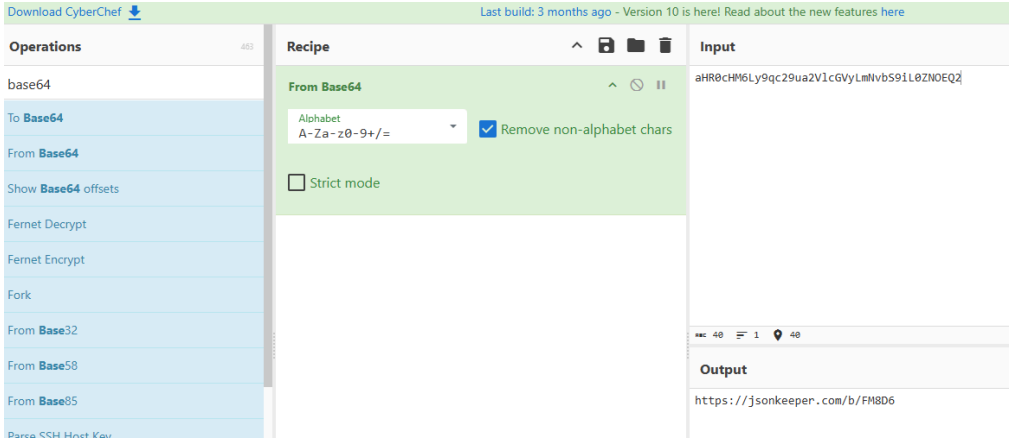

Figure 10 – The variable decodes to a JSON Keeper URL.

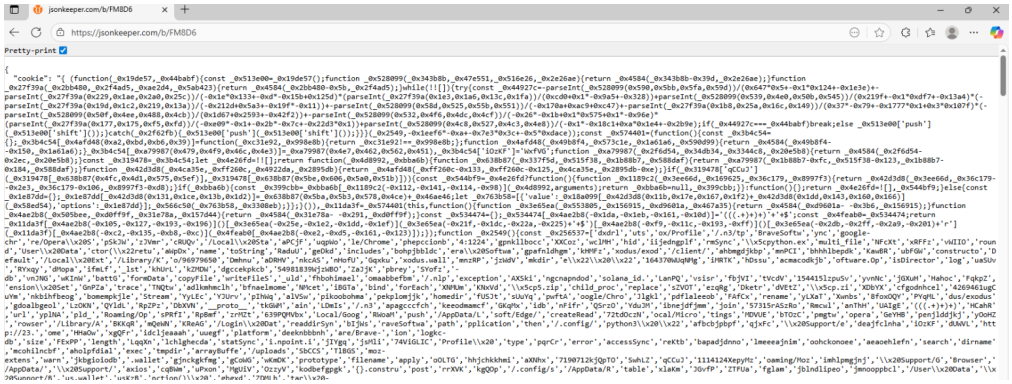

Figure 11 – Obfuscated JavaScript code hosted on JSON Keeper.

This code is fetched from JSON Keeper and imported in `server/controllers/userController.js` :

Figure 12 – Loading the encoded data from config.env via userController.js.

The JavaScript code hosted on JSON Keeper is heavily obfuscated with techniques such as packing, array and string obfuscation, and other common techniques such as concatenation.

After several layers of deobfuscation, the final payload is a variant of **BeaverTail** with the same capabilities described in Veracode's article:

- Usage of Axios as 'embedded' HTTP client.
- Enumeration and exfiltration of system information.
- Searching browser profiles and extension directories for sensitive data (wallets, logs and extensions such as MetaMask, Phantom and TronLink) and trying to exfiltrate them.
- Searching for and exfiltrating Word documents, PDF files, screenshots, secret files, files containing environment variables, and other sensitive files such as the logged-in user's Keychain (macOS password database).

BeaverTail fetches and executes the next stage which is **InvisibleFerret**:

- hxxp[:]//146[.]70[.]253[.]107:1224/pdown (ZIP file with Python)
- hxxp[:]//146[.]70[.]253[.]107:1224/client/99/81 (Malicious Python script – InvisibleFerret)

Interestingly, the campaign used here specifically mentions it has been "Updated on 5th of August". The additional components are outlined in Figure 1 from the introduction and also described further.

InvisibleFerret

InvisibleFerret, a modular malware framework written in Python, has been described as well in several publications like the one from Palo Alto. The sample discussed in this blog does not differ significantly from those specific reports, with the exception of gathering an additional payload from Pastebin.

A high-level view of this additional Pastebin functionality is shown in Figure 13 below:

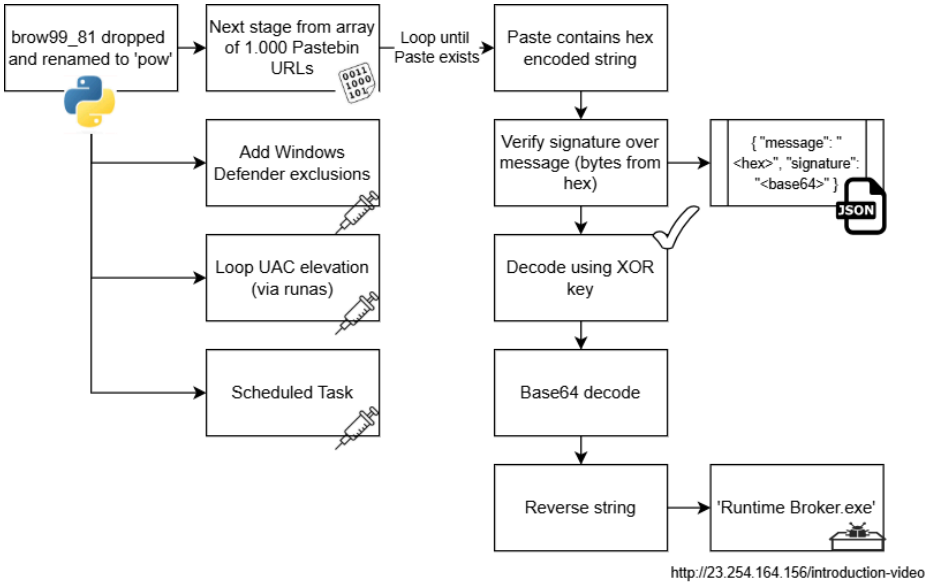

Figure 13 – InvisibleFerret's Pastebin functionality

Do note that this functionality is part of the capability in `'pow` ' as seen in stage 4 of this campaign (refer to Figure 1 for the campaign overview). It has 3 main components:

1. **Tsunami Payload**: Adds exceptions to Windows Defender, creates scheduled tasks, and downloads the next stage from Pastebin.
2. **Tsunami Injector**: Ensures persistence and installs required Python packages, such as `cryptography`, and obfuscates the injector script.
3. **Tsunami Infector**: Validates if Python is installed on the system. If not, it downloads and installs Python silently, using a User Account Control (UAC) prompt to gain administrative privileges. This ensures both the payload and injector component can be executed.

Since the threat actors have given those names to these components, they are slightly confusing. The component responsible for retrieving the next stage from Pastebin, sits within ‘Tsunami Payload’. The ‘pow’ script further embeds a hardcoded public SSL that will verify the integrity of the ‘message’ (explained further below). Exactly 1.000 encoded URLs are embedded in the script, which get converted from hex, decoded with the hardcoded XOR key “**!!!HappyPenguin1950!!!**” and finally base64-decoded.

```
##### URL Downloader #####

def xor_encrypt(text: bytes):
    XOR_KEY = b"!!!HappyPenguin1950!!!"

    encrypted_text = bytearray()
    for i in range(len(text)):
        encrypted_text.append(text[i] ^ XOR_KEY[i % len(XOR_KEY)])
    return bytes(encrypted_text)

def xor_decrypt(text: bytes):
    return xor_encrypt(text)

def decode(encoded: str) -> str:
    encoded_bytes = binascii.unhexlify(encoded)
    encoded_bytes = xor_decrypt(encoded_bytes)
    encoded = base64.b64decode(encoded_bytes).decode()

    return encoded[::-1]

def download_installer_url() -> str:
    URLS = [
        "6f5b783f2e283e4c31232c57165b2860614f7913
```

Figure 14 – URL Downloader functionality

A few examples of decoded Pastebin URLs:

Pastebin URL	**Profile**	**File**
hxxps[://]pastebin[.]com/u/NotingRobe2871_FranzStill8494	hxxps[://]pastebin[.]com/u/NotingRobe2871	FranzStill8
hxxps[://]pastebin[.]com/u/ShadowGates1462_PastPhys9067	hxxps[://]pastebin[.]com/u/ShadowGates1462	PastPhys9
hxxps[://]pastebin[.]com/u/AmendMinds7934_LoverTumor2853	hxxps[://]pastebin[.]com/u/AmendMinds7934	LoverTum

The list goes on. In this campaign, the first Paste listed in the table, *hxxps[://]pastebin[.]com/u/NotingRobe2871_FranzStill8494*, is the one that exists and will therefore go through the decoding routine. Interestingly, this paste has been ‘viewed’ over 400 times, which implies the campaign has had a reasonable amount of success. The content of the paste is hex encoded and can be observed in Figure 15.

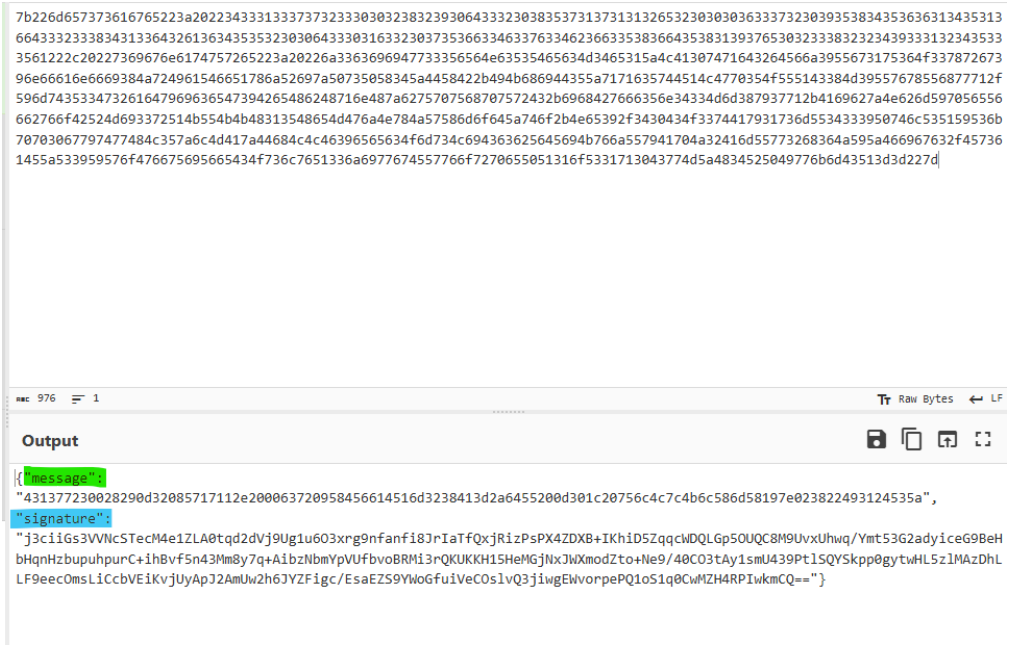

Figure 15 – The content of Pastebin, hex decoded.

The signature (highlighted in blue) is an RSA signature over the ‘message’. The malware uses an embedded public key to verify if the signature matches, and if so, it will perform the decoding of the data contained in ‘message’.

This message (highlighted in green) in the JSON data will be hex decoded, XORed with the same key as before, base64 decoded and finally reversed to end up with the next stage downloader URL, which is: hxxp[://]23[.]254[.]164[.]156/introduction-video. This CyberChef recipe displays the decoding functionality.

At the time of writing this post, the next stage or ‘introduction-video’, was offline. However, we assess with medium confidence, this would be the **TsunamiInstaller**. Through further hunting, we discovered a binary recently uploaded to VirusTotal, which incorporates a .NET payload (9d9a25482e7e40e8e27fdb5a1d87a1c12839226c85d00c6605036bd1f4235b21) with capabilities as seen in Figure 16.

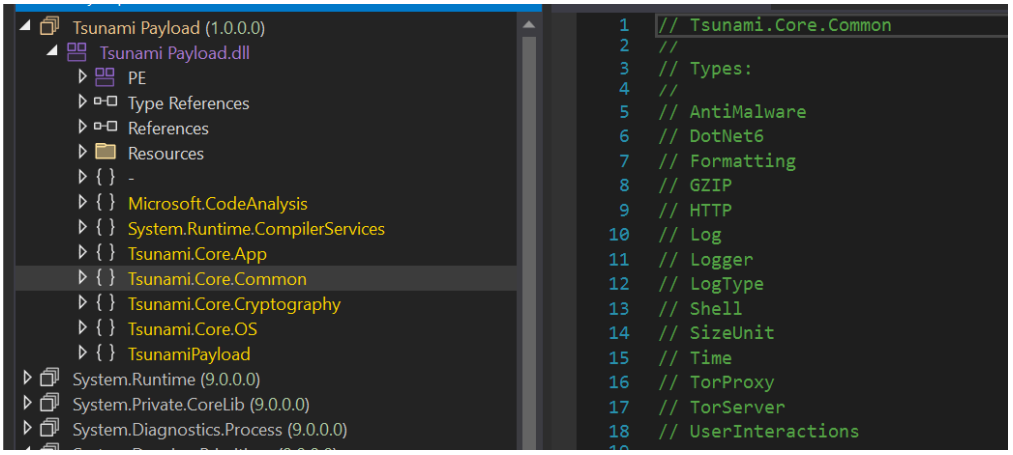

Figure 16 – Tsunami Payload showing key components such as anti-malware and logging.

This Tsunami ‘payload’ (again a slight misnomer as it is also an installer/dropper) has multiple capabilities such as additional system fingerprinting, data collection, and embedding TOR for downloading additional payloads from a hardcoded .onion address (offline at the time of writing this blog):

- hxxp[://]n34kr3z26f3jzp4ckmwuv5ipqyatumdxhgjgsmucc65jac56khdy5zqd[.]onion.

Infrastructure

The attack chain discussed in this blog uses JSON storage services for payload hosting. By pivoting on identified indicators, we were able to identify many additional repositories, payloads hosted on JSON storage services and IP addresses. This includes for example the use of Railway to host and serve additional payloads.

We have listed the identified IOCs at the end of the blog. Representatives of the JSON storage services were contacted and briefed regarding the abuse and are working on removing all malicious content. We'd like to thank them for their swift response and actions.

Conclusion

It's clear that the actors behind Contagious Interview are not lagging behind and are trying to cast a very wide net to compromise any (software) developer that might seem interesting to them, resulting in exfiltration of sensitive data and crypto wallet information.

The use of legitimate websites such as JSON Keeper, JSON Silo and npoint.io, along with code repositories such as Gitlab and GitHub, underlines the actor's motivation and sustained attempts to operate stealthily and blend in with normal traffic.

Never run code from an unknown repository or from a 'recruiter' as part of any first interview, especially when contact has been recently established. If needed, inspect the configuration files for any signs of malicious activity.

Indicators of Compromise (IOCs)

The following IOCs list all indicators seen in this campaign, as well as additional indicators uncovered via threat hunting and other methods.

Email addresses used to upload the malware to JSON storage services

```
 ahmadbahai07[@]gmail.com
drgru854[@]gmail.com
jack.murray.tf7[@]gmail.com
jackhill2765[@]gmail.com
reichenausteve[@]gmail.com
magalhaesbruno236[@]gmail.com
stromdev712418[@]gmail.com
trungtrinh0818[@]gmail.com
```

Repositories hosting malicious code

```
 hxxps[://]gitlab[.]com/technicalmanager-group/real-esate
hxxps[://]gitlab[.]com/real-world-assest-tokenization/goldencity
hxxps[://]gitlab[.]com/goldencity-group/goldencity-demo
hxxps[://]github[.]com/meta-stake/RealEstateVC
hxxps[://]github[.]com/meta-stake/RaceStake
hxxps[://]github[.]com/parth5805/iGuru-Task
hxxps[://]github[.]com/adammajoros250-creator/alex111
hxxps[://]github[.]com/adammajoros250-creator/123456ddd
hxxps[://]github[.]com/adammajoros250-creator/corex-arc-fork
hxxps[://]github[.]com/adammajoros250-creator/demotest
hxxps[://]github[.]com/adammajoros250-creator/bot111
hxxps[://]github[.]com/adammajoros250-creator/Apexora-test
hxxps[://]github[.]com/adammajoros250-creator/123456ddd
hxxps[://]github[.]com/harrypotter060327-netizen/test_project
hxxps[://]github[.]com/harrypotter060327-netizen/Harry-Potter
hxxps[://]github[.]com/harrypotter060327-netizen/David-test
hxxps[://]github[.]com/harrypotter060327-netizen/Test_Estoken
hxxps[://]github[.]com/harrypotter060327-netizen/eeeee
hxxps[://]github[.]com/harrypotter060327-netizen/TEST_LORD
```

```
hxxps[://]github[.]com/edwardtam919/staking-platform-main
hxxps[://]github[.]com/TommyMinion/DeFi-Market
hxxps[://]github[.]com/0x3ca54/arena-world
hxxps[://]github[.]com/InfiniGods-Tech/rei
```

GitHub account identified making initial commits to other repositories containing malicious code

```
hxxps[://]github[.]com/carlotalentengine-sketch
```

JSON Storage URLs

```
hxxps[://]jsonkeeper[.]com/b/GNOX4
hxxps[://]jsonkeeper[.]com/b/IARGW
hxxps[://]jsonkeeper[.]com/b/FM8D6
hxxps[://]jsonkeeper[.]com/b/GCGEX
hxxps[://]jsonkeeper[.]com/b/IXHS4
hxxps[://]jsonkeeper[.]com/b/86H03
hxxps[://]jsonkeeper[.]com/b/6OCFY
hxxps[://]jsonkeeper[.]com/b/E4YPZ
hxxps[://]jsonkeeper[.]com/b/8RLOV
hxxps[://]jsonkeeper[.]com/b/BADWN
hxxps[://]jsonkeeper[.]com/b/4NAKK
hxxps[://]jsonkeeper[.]com/b/JV43N
hxxps[://]www[.]jsonkeeper[.]com/b/VBFK7
hxxps[://]www[.]jsonkeeper[.]com/b/JNGUQ
hxxps[://]www[.]jsonkeeper[.]com/b/O2QKK
hxxps[://]www[.]jsonkeeper[.]com/b/RZATI
hxxps[://]www[.]jsonkeeper[.]com/b/T7Q4V
hxxps[://]api[.]jsonsilo[.]com/public/0048f102-336f-45dd-aef6-3641158a4c5d
hxxps[://]api[.]jsonsilo[.]com/public/942acd98-8c8c-47d8-8648-0456b740ef8b
hxxps[://]api[.]npoint[.]io/e6a6bfb97a294115677d
hxxps[://]api[.]npoint[.]io/8df659fd009b5af90d35
hxxps[://]api[.]npoint[.]io/f4be0f7713a6fcdaac8b
hxxps[://]api[.]npoint[.]io/148984729e1384cbe212
hxxps[://]api[.]npoint[.]io/2169940221e8b67d2312
hxxps[://]api[.]npoint[.]io/a1dbf5a9d5d0636edf76
hxxps[://]api[.]npoint[.]io/62755a9b33836b5a6c28
hxxps[://]api[.]npoint[.]io/336c17cbc9abf234d423
hxxps[://]api[.]npoint[.]io/832d58932fcfb3065bc7
hxxps[://]api[.]npoint[.]io/cb0f9d0d03f50a5e1ebe
hxxps[://]api[.]npoint[.]io/f6dd89c1dd59234873cb
hxxps[://]api[.]npoint[.]io/03f98fa639fa37675526
hxxps[://]api[.]npoint[.]io/38acf86b6eb42b51b9c2
```

BeaverTail/InvisibleFerret Command & Control servers

```
107[.]189[.]25[.]109
144[.]172[.]100[.]142
144[.]172[.]103[.]97
144[.]172[.]95[.]226
144[.]172[.]97[.]7
146[.]70[.]253[.]10
146[.]70[.]253[.]107
147[.]124[.]197[.]138
147[.]124[.]197[.]149
147[.]124[.]212[.]146
```

```
147[.]124[.]212[.]89
147[.]124[.]214[.]129
147[.]124[.]214[.]131
147[.]124[.]214[.]237
165[.]140[.]86[.]227
172[.]86[.]84[.]38
172[.]86[.]98[.]240
185[.]153[.]182[.]241
185[.]235[.]241[.]208
216[.]126[.]229[.]166
23[.]106[.]253[.]194
23[.]106[.]253[.]215
23[.]106[.]253[.]221
23[.]106[.]253[.]242
23[.]106[.]70[.]154
23[.]227[.]202[.]242
23[.]227[.]202[.]244
23[.]254[.]164[.]156
38[.]92[.]47[.]151
38[.]92[.]47[.]85
38[.]92[.]47[.]91
45[.]128[.]52[.]14
45[.]137[.]213[.]30
45[.]43[.]11[.]201
45[.]61[.]133[.]110
45[.]61[.]150[.]30
45[.]61[.]150[.]31
45[.]61[.]151[.]71
45[.]76[.]160[.]53
5[.]253[.]43[.]122
66[.]235[.]168[.]232
66[.]235[.]175[.]109
67[.]203[.]7[.]163
67[.]203[.]7[.]171
86[.]104[.]74[.]51
88[.]218[.]0[.]78
94[.]131[.]97[.]195
95[.]164[.]17[.]24
```

Tsunami Indicators

```
 hxxp[://]23[.]254[.]164[.]156/introduction-video
hxxps[://]pastebin[.]com/u/NotingRobe2871_FranzStill8494
hxxp[://]n34kr3z26f3jzp4ckmwuv5ipqyatumdxhgjgsmucc65jac56khdy5zqd[.]onion
```

Anydesk Command & Control server

```
 95[.]164[.]17[.]24
```

About the authors

Bart Parys

Bart is a senior manager at NVISO where he mainly focuses on SOC maturity & assessments, Threat Intelligence, Incident Response and Malware Analysis.

Stef Collart

Stef Collart is a Threat Hunter & CTI Analyst within NVISO's CSIRT, combining information gained from CTI to perform threat hunts for various customers.

Efstratios Lontzetidis

Efstratios Lontzetidis is a CTI analyst within NVISO’s CSIRT and is mainly involved in Intelligence Production.
