Recent AI-Assisted Cyber Attacks in Japan: Investigating Attack Infrastructure and Techniques - JPCERT/CC Eyes

By JPCERT/CC

Published: 2026-10-20 · Archived: 2026-11-21 02:00:24 UTC

October 21, 2026

On October 8, 2026, JPCERT/CC issued an alert regarding a series of unauthorized access incidents involving the leakage of personal and other information reported by multiple organizations in Japan. In cooperation with multiple organizations, JPCERT/CC has been investigating infrastructure that may have been used in these attacks. We confirmed that the attackers had used AI to scan for known vulnerabilities and carry out attacks via APIs. We identified several servers used for these activities and confirmed that the attackers had also targeted web services intended for users outside Japan.
This article presents our findings from an investigation of attack servers involved in attacks targeting web services and other systems in Japan and abroad.

Investigation of Attack Servers: Part 1

One notable characteristic of these attacks was that the attackers analyzed Android application packages (APK files) to identify APIs and exploit them[1]. On one of the attack servers, we found evidence that the attackers had analyzed APK files and tested the associated APIs for vulnerabilities such as server-side request forgery (SSRF). Figure 1 shows a partial list of files found on this server. The server also contained applications intended for users outside Japan.



Directory listing for /

- %2Ftext.txt
- .acme.sh/
- .bash_history
- .bashrc
- .bun/
- .cache/
- .claude/
- .claude.json
- .config/
- .docker/
- .java/
- .local/
- .npm/
- .npm-global/
- .npmrc
- .profile
- .python_history
- .qwen/
- .ssh/
- .viminfo
- .xscan/
- 10
- 11.speex
- 11.xlsx
- _vps_redir.py
- action_params.json
- add_path.out
- admin_api.sh
- admin_index.html
- admin_index_full.txt
- [redacted]_oob/
- apk_dex/
- apk_status.txt
- apk_url.txt
- auth_home.html
- auto_login.py
- [redacted].apk
- [redacted]js
- bc/
- [redacted].apk
- [redacted]ssrf.log
- blob-1.2.4.bin
- brute3.err
- brute3.out
- brute3.py

Figure 1: Files found on the server under investigation

Figure 2 shows a Python script that appears to have been used to test for SSRF.

```python
import http.server, ssl, socketserver
class H(http.server.SimpleHTTPRequestHandler):
    def log_message(self, fmt, *args):
        line = "HTTPS %s - %s" % (self.address_string(), fmt%args)
        print(line, flush=True)
        with open("/root/ssrf/access.log","a") as f: f.write(line+"\n")
    def do_GET(self):
        if self.path.startswith("/r302"):
            import urllib.parse
            q = urllib.parse.urlparse(self.path)
            to = urllib.parse.parse_qs(q.query).get("u", [""])[0]
            with open("/root/ssrf/access.log","a") as f: f.write("302-to %s\n" % to)
            self.send_response(302)
            self.send_header("Location", to)
            self.end_headers()
            return
        super().do_GET()
socketserver.TCPServer.allow_reuse_address = True
httpd = socketserver.TCPServer(("0.0.0.0", 443), H)
ctx = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
ctx.load_cert_chain(certfile="/root/ssrf/cert.pem", keyfile="/root/ssrf/key.pem")
httpd.socket = ctx.wrap_socket(httpd.socket, server_side=True)
httpd.serve_forever()
```

Figure 2: Python script related to SSRF testing

The script functions as an HTTPS server and returns an HTTP 302 redirect response to specific requests.

Investigation of Attack Servers: Part 2

On another server under investigation, we found several APK files, along with files associated with two tools: Blutter, used for reverse engineering Flutter applications, and jadx, used for analyzing Android applications. For more information on Blutter, see Appendix A.
Figure 3 shows a partial list of files found on the server.

Directory listing for /

- .ansible/
- .bashrc
- .cache/
- .config/
- .local/
- .profile
- .ssh/
- apk_extract/
- blutter/
- blutter.log
- [redacted].py
- [redacted].py
- [redacted].py
- [redacted].py
- [redacted].py
- [redacted].py
- [redacted].py
- [redacted]_all.sh
- [redacted]_ssh.log
- [redacted]ssh.sh
- chk.py
- dl_sweep.sh
- [redacted]_brute.log
- [redacted]_brute.py
- [redacted]_brute.sh
- [redacted]_rot.sh
- [redacted]ssh.log
- [redacted]ssh.sh
- gcc13.log
- gcc13deb/
- gql_catch.py
- jadx-out/
- jadx.log
- jadx.zip
- [redacted]_ips.txt
- padding_oracle_poc.py
- poc.log
- pw.txt
- pw2.py
- pw3.py
- pw3000.txt
- pw_chunks.py
- pw_probe.py
- [redacted].py
- [redacted]
- [redacted]
- [redacted]
- [redacted].apk
- [redacted].apk
- reg_newuser.png
- RuoYi-src/
- ruoyi4.tar.gz
- RuoYi40/
- s17s_src.tgz
- setup13.log



Figure 3: Files related to Android application analysis and vulnerability testing

The server also contained directories such as **apk_extract/**, which may have been associated with APK extraction. Filenames such as **strings_libapp.txt** suggest that the files may have been used to store strings extracted from native libraries.

Investigation of Attack Servers: Part 3

On a separate server not directly related to these attacks, we found files associated with various attack techniques targeting web applications, MySQL, Jenkins servers, and email gateways.
Figure 4 shows a partial list of files found on this server. We also found evidence of attempts to evade EDR systems, as well as web shells believed to have been deployed within the targeted network. These findings suggest that the attackers were attempting to gain access to internal networks.



Figure 4: Files associated with attempts to access internal networks

In conclusion

JPCERT/CC will continue working with relevant parties to respond to these attacks. We would appreciate the assistance of security researchers in hunting for servers used in these attacks. If you uncover any information related to attacks targeting Japanese organizations or incidents affecting them during your investigations, please share your findings with JPCERT/CC at **info@jpcert.or.jp** so that we can help notify the affected organizations.

JPCERT/CC Attack Infrastructure Investigation Team
(This article was machine-translated and manually reviewed.)

References

[1] NICT：【注意喚起】相次ぐ情報漏洩、その原因の一つは「アプリ解析からのサーバ攻撃」
https://cynex.nict.go.jp/library/261013-01.html

Appendix A: Reverse Engineering Flutter Applications

Flutter is a framework for building applications for multiple platforms from a single codebase. Blutter, identified on one of the attack servers, is an open-source tool for analyzing ahead-of-time (AOT)-compiled Dart code in Flutter applications. Reverse engineering with such tools can reveal an application's internal structure and how it communicates with APIs.
A technical article published by IQCrafter in June 2026 describes an approach to analyzing a real-world Flutter application using Frida, Blutter, LIEF, Capstone, and other tools. Static and dynamic analysis of libapp.so revealed an embedded RSA private key, API authentication mechanisms, and the logic used to generate request signatures. The article also reports that the request-signing process was successfully reproduced, enabling a standalone Python client to access the production API without using the legitimate application.
A technical article by NerdzLab describes techniques for analyzing APK files and AOT-compiled libapp.so files to examine API keys, secrets, class structures, and function addresses. These examples demonstrate that reverse engineering may reveal embedded secrets and API authentication mechanisms, even in compiled applications.
If a shared private key or other secrets used for API authentication are embedded in an application, attackers may be able to extract them and generate API requests without using the legitimate application. To address these risks, we recommend the following measures:

- **Protect secrets**: Do not embed private keys used for API authentication or long-lived credentials in applications.
- **Enforce server-side authentication and authorization**: Do not rely solely on request-signing mechanisms or code obfuscation within the application. Authenticate users on the server and verify that they are authorized to access the requested resources and perform the requested operations.
- **Prevent API abuse**: Implement rate limiting and monitor for anomalous requests. Account for the possibility that the API may be accessed by clients other than the legitimate application.
- **Prevent SSRF**: For APIs that retrieve content from external URLs, restrict the destinations they can access and validate redirect destinations as well.

References：

- Reverse Engineering a Flutter Mobile App: A Complete Security Research Methodology（IQCrafter）
- Reverse Engineering a Flutter App: How Secure Is Your Data, Really?（NerdzLab）
- I Reverse Engineered a Banking App With a Million Downloads（Medium）
- Blutter（GitHub）

Appendix B: IoC

- 210.149.87[.]120
- 118.26.36[.]16
- 69.10.51[.]162

JPCERT/CC

Please use the below contact form for any inquiries about the article.

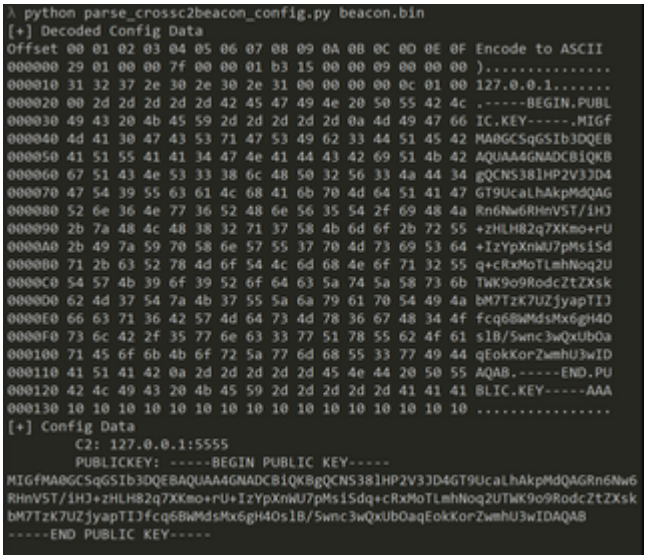
Related articles

- 

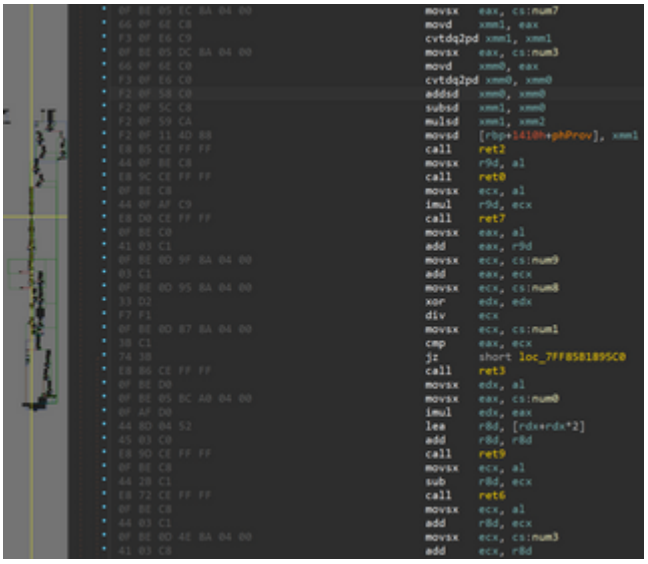
  Multiple Threat Actors Rapidly Exploit React2Shell: A Case Study of Active Compromise

- 

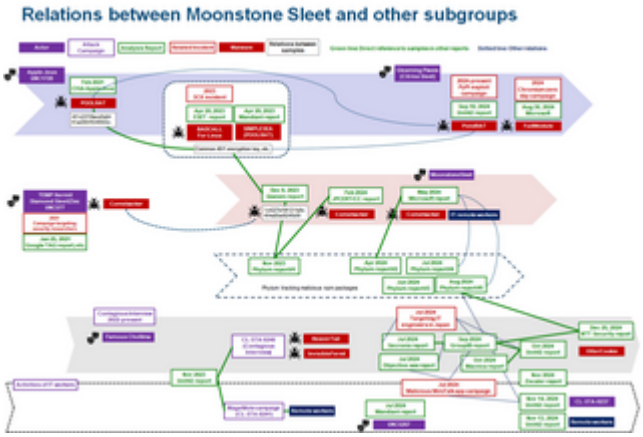
  Update on Attacks by Threat Group APT-C-60

- 

  CrossC2 Expanding Cobalt Strike Beacon to Cross-Platform Attacks

- 

  [Malware Identified in Attacks Exploiting Ivanti Connect Secure Vulnerabilities]

- 

  [Tempted to Classifying APT Actors: Practical Challenges of Attribution in the Case of Lazarus’s Subgroup]

Source: https://blogs.jpcert.or.jp/en/2026/10/web_api_attacks.html