

Interview with the Chollima

 quetzal.bitso.com/p/interview-with-the-chollima

Share this post



[Bitso Quetzal Team](#)

[Interview with the Chollima](#)



Another year, another Chollima added to our trophy wall.

February came and went once again, this time without a peep from our dear friends behind the [Great Firewall](#), nor from those under the menacing guise of the [Great Leader](#). Not that I missed them, but something felt... off.

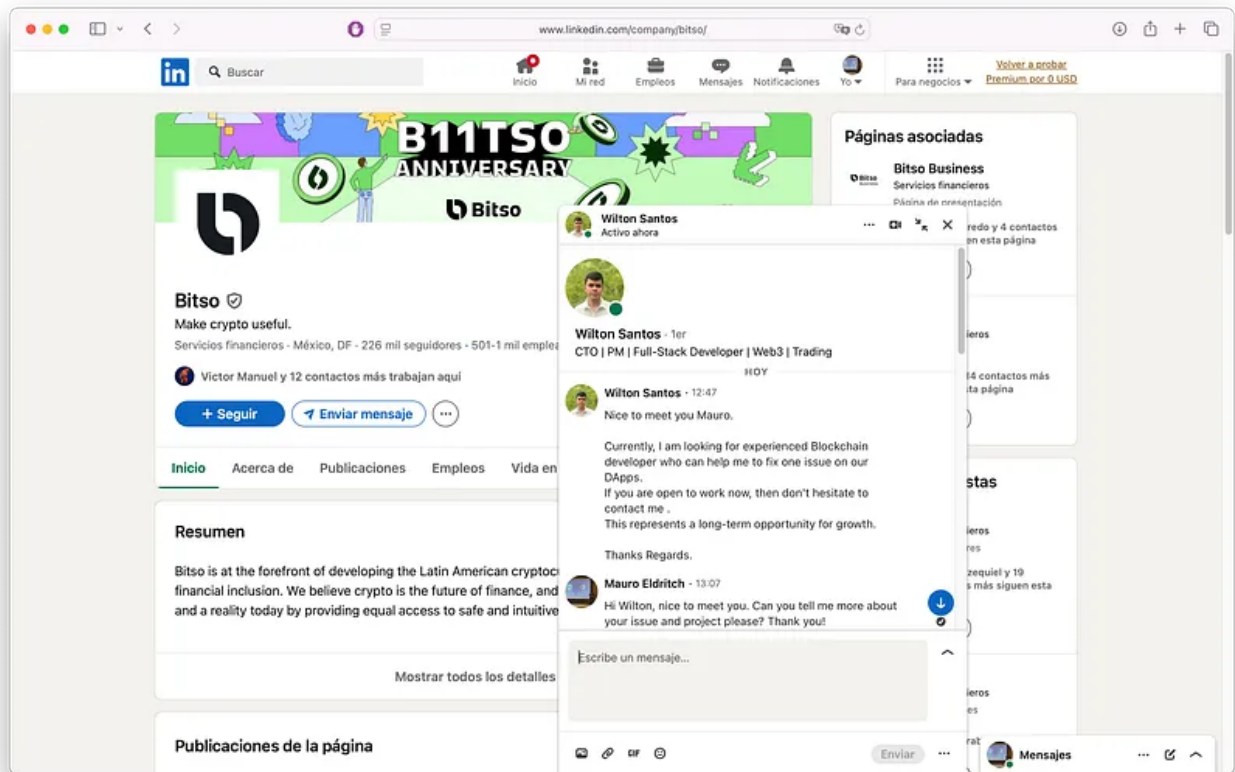
Had they forgotten about us? Are we no longer *that* important of a target? Did they simply decide to move on and forgive us every time we mocked them publicly—when their **ACME**-branded malware blew up in their faces, giving us the chance to weaponize it into [talks](#) and [articles](#) at the best conferences and magazines in the world?

No, I don't think they're the type to turn the other cheek. They waited until April to fine-tune the stockade after planning something highly targeted. At us.

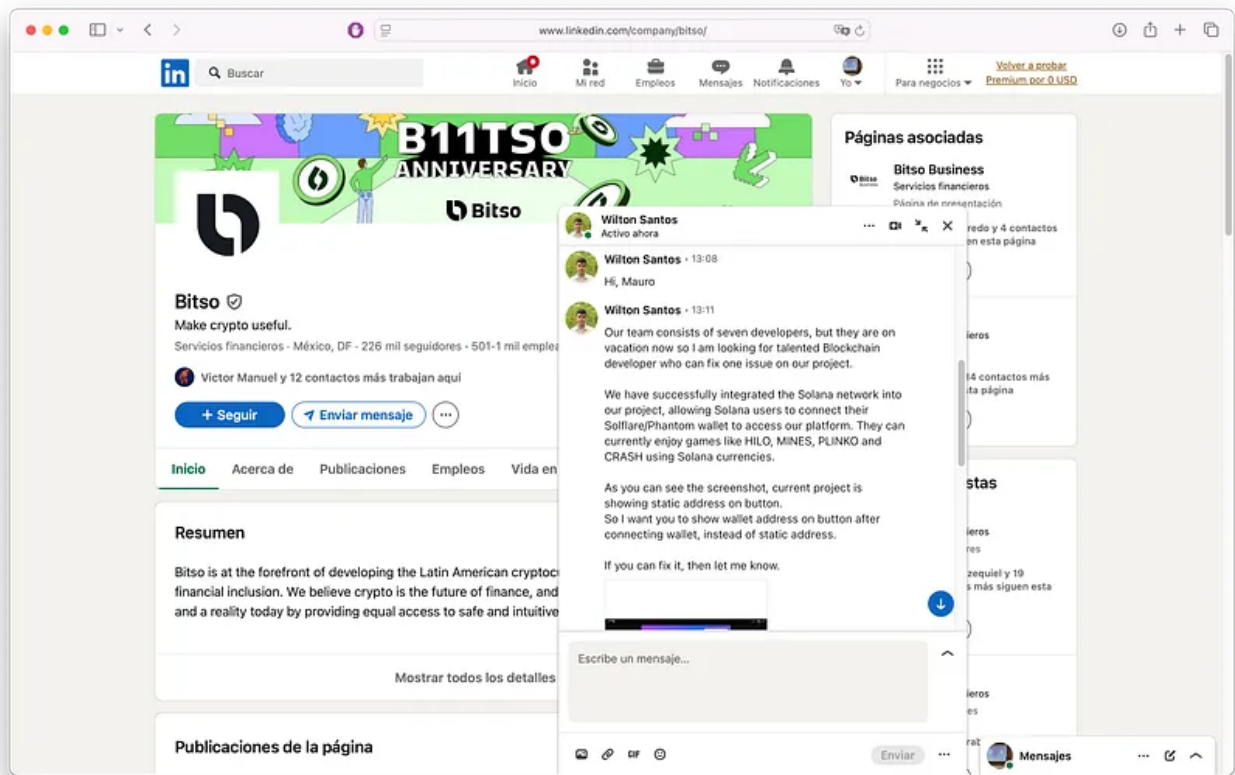
Well, at me.

The North Korean Job

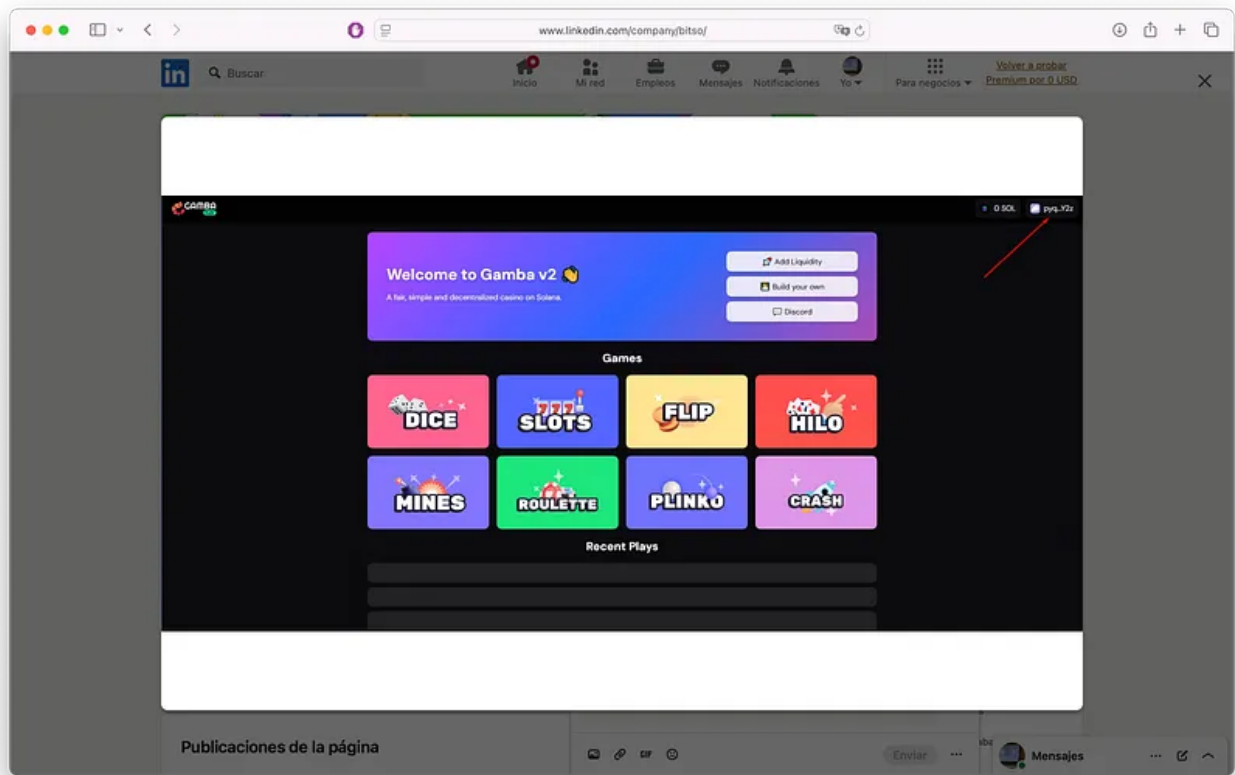
It all started in the [most vicious hunting ground for Threat Actors](#), when a muppet well-respected [Lazarus](#) agent approached me carelessly under the name "Wilton Santos", asking if I was open to working on a fix for its [DApp](#).



I asked for further clarification, and the sad story rolled in: they were a team of seven developers, but **all of them were on vacation** (bad human resources planning there), and needed a trivial but urgent fix on its UI.



The DApp was “**Gamba v2**”, a gaming platform where users could join by connecting with their wallets (you might think this is the strike point, but that would be too obvious). The *UI problem* was that they wanted to dynamically display the user’s wallet in a specific profile button. This is pretty trivial, and many JS libraries can do it in two or three lines of code.

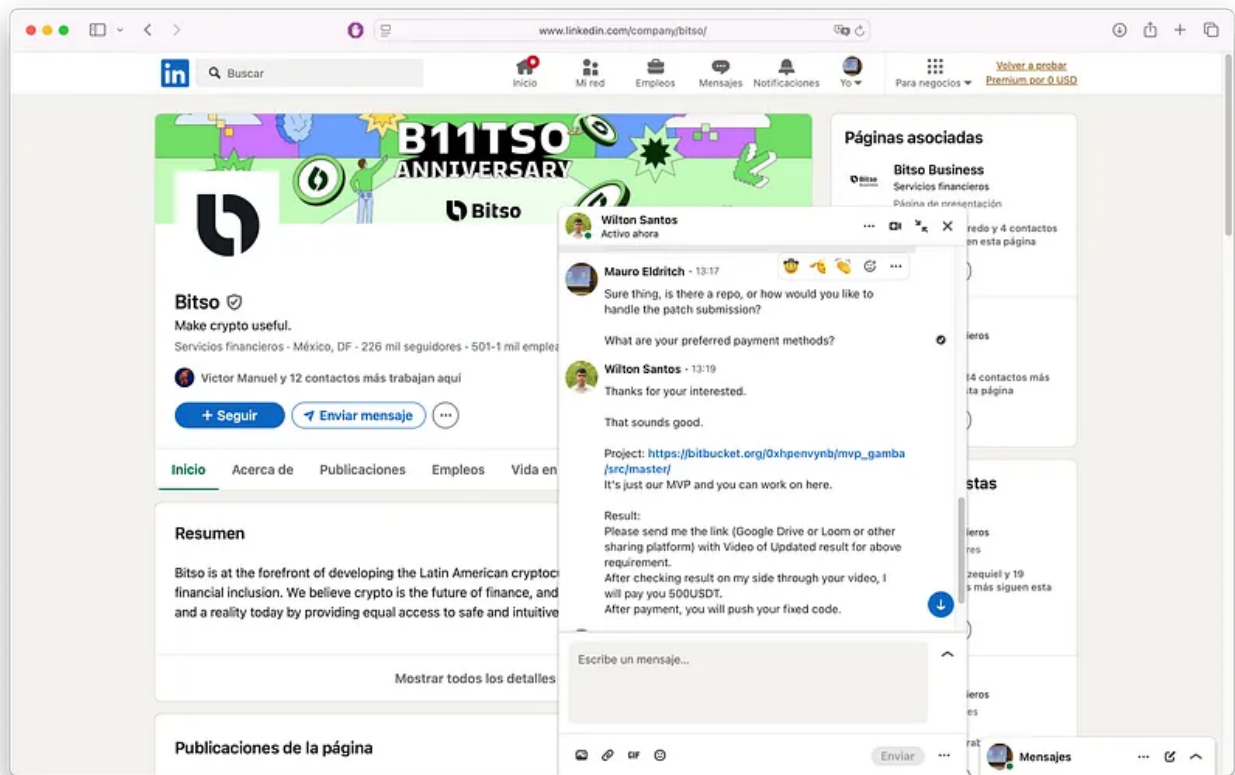


After successfully patching the issue, I just needed to send a video of the fix working, and I would be paid the astronomical amount of **500 USDT** for a *3-line patch*.

<sarcasm>

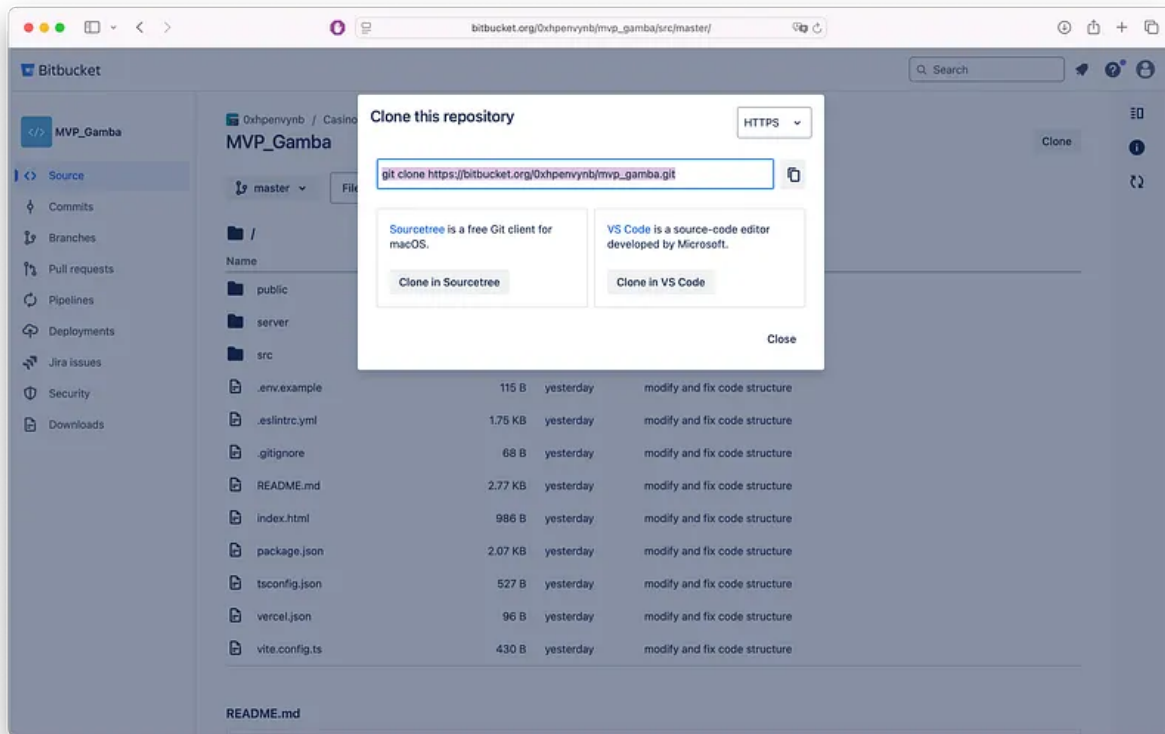
I can't believe there are [BSD kernel developers](#) out there submitting patches for free...

</sarcasm>



I accepted, because I can smell a good old-fashioned [APT](#) campaign miles away.

And then, our friend “Wilton” shared a **BitBucket** repository.



See the background panel. Creation date: yesterday.

Time to get the work done, I guess.

Trying to Catch an Otter

There's an important context to add here. The [Chollimas](#) (North Korean state-sponsored actors) are running a campaign **targeting engineers and executives** in the fintech and crypto sectors.

They attempt to trick engineers with **fake job interviews** or postings, coaxing them into running **infected coding challenges**.

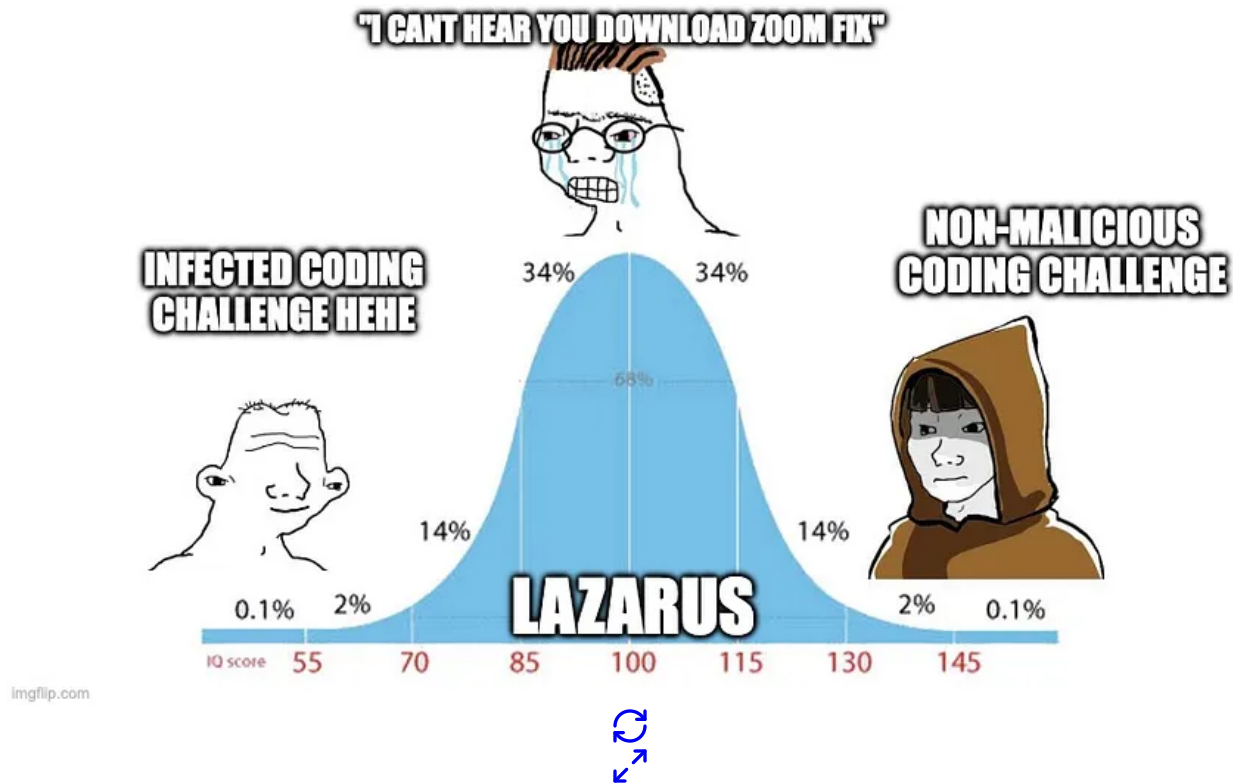
They also target executives by luring them into **Zoom calls with fake VCs or business partners**. Once in, the attackers **feign not hearing the victim speaking** and act angry about it. Then, one of them shares a **fake Zoom fix or update** to solve the issue. Over the fear of the deal going sour, the victim usually runs it and gets infected. This effort is being tracked as **DevPopper** or [ContagiousInterview](#).

But this time, it's different. As noted in the closure of the last section, this repository was created just a day ago, with no public mentions of it, the person who shared it with me, or the endpoints and infrastructure it attempts to reach.

Everything is **brand new**.

But there's still something more sinister about it: **the code is completely clean**. There's no malicious payload, fake packages, infected libraries, or dependencies.

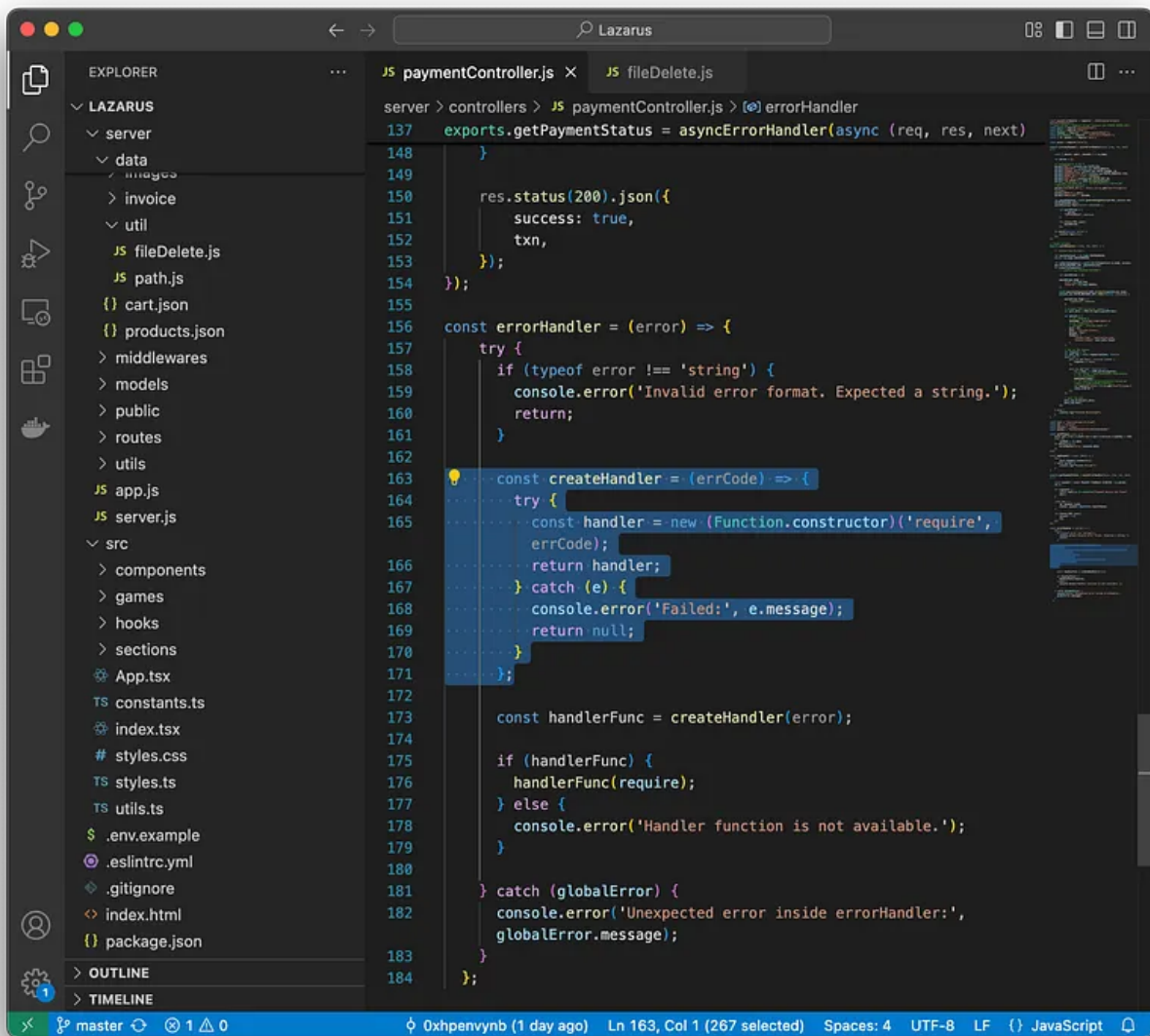
In [Gibson's](#) words: *It's clear as ethanol*.



Different Lazarus divisions have different approaches.

But then, after carefully reviewing the code, I found their *entry point*, and I must say, it **is the most creative one I've seen in a long time**. While the code itself isn't malicious, there's a specific bootstrap function **that will always fail**. However, it's contained within a [Try/Catch](#) block—a special construct where the language will **Try** to do something and **Catch** the workflow if something goes wrong, preventing it from crashing, **and doing something** to remediate the error.

That **Catch** block, in this case, will invoke a function called **errorHandler**, which will receive an error code **directly from an external API**... and execute it.



```
server > controllers > JS paymentController.js > [0] errorHandler
137 exports.getPaymentStatus = asyncErrorHandler(async (req, res, next)
148   }
149
150   res.status(200).json({
151     success: true,
152     txn,
153   });
154 });
155
156 const errorHandler = (error) => {
157   try {
158     if (typeof error !== 'string') {
159       console.error('Invalid error format. Expected a string.');
```



You read it right: the error code comes from a distant server in **Finland** (not a Gibson reference), and it is then executed via a *require* statement.

This is our implant!

Now, we could just burn it in an intelligence pulse... or better yet, use what we know to strike back.

The server has two open ports: port 80, running the API on [Node.js Express](#), and port 7777, identified as running **RDP (Remote Desktop Protocol for Windows)**.

```
~
→ ping chainlink-api-v3.cloud                                04/10/25 - 7:59 PM
PING chainlink-api-v3.cloud (135.181.123.177): 56 data bytes
64 bytes from 135.181.123.177: icmp_seq=0 ttl=90 time=186.182 ms
^C
--- chainlink-api-v3.cloud ping statistics ---
1 packets transmitted, 1 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 186.182/186.182/186.182/nan ms

~
→ nmap -sV 135.181.123.177                                    04/10/25 - 7:59 PM
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-10 19:59 -03
Nmap scan report for static.177.123.181.135.clients.your-server.de (135.181.123.177)
Host is up (0.19s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
80/tcp    open  http        Node.js Express framework
7777/tcp  open  ms-wbt-server Microsoft Terminal Services
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 139.71 seconds

~
→                                                                    04/10/25 - 9:00 PM
```



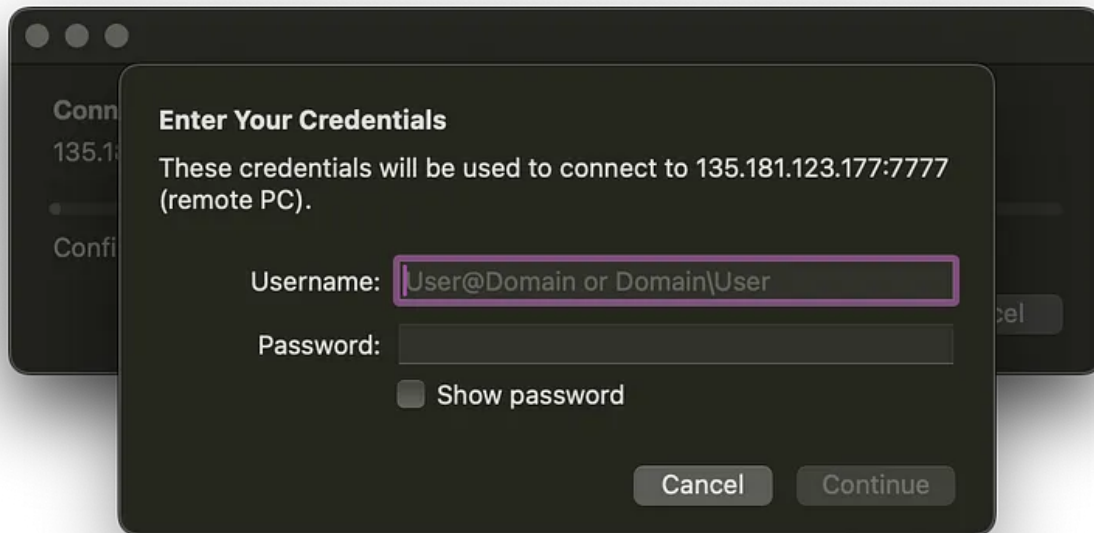
Reaching out to the API and triggering a fabricated error reveals an interesting **internal** output, where we can see that our friends are indeed using **Windows... with the Administrator user**.

```
bash-3.2$ curl http://chainlink-api-v3.cloud/api/service/token/56e15ef3b5e5f169fc063f8d3e88288e -H "Content-Type: application/json" --data '{"..."}'
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>SyntaxError: Unexpected token &#39;&quot;&#39;, &quot;&#39;&quot; is not valid JSON<br>    at JSON.parse (<anonymous>:1:1)
    at createStrictS
yntaxError (C:\Users\Administrator\Videos\script-server-main\node_modules\body-parser\lib\types\json.js:160:10)
    at parse (C:\Users\Administrator\Videos\script-server-main\node_modules\body-parser\lib\types\json.js:83:15)
    at C:\Users\Administrator\Videos\script-server-main\node_modules\body-parser\lib\read.js:128:18
    at AsyncResource.runInAsyncScope (node:async_hooks:211:14)
    at invokeCallback (C:\Users\Administrator\Videos\script-server-main\node_modules\raw-body\index.js:231:16)
    at done (C:\Users\Administrator\Videos\script-server-main\node_modules\raw-body\index.js:220:7)
    at IncomingMessage.onEnd (C:\Users\Administrator\Videos\script-server-main\node_modules\raw-body\index.js:280:7)
    at IncomingMessage.emit (node:events:518:28)
    at endReadableNT (node:internal/streams/readable:1698:12)
</pre>
</body>
</html>
bash-3.2$
```



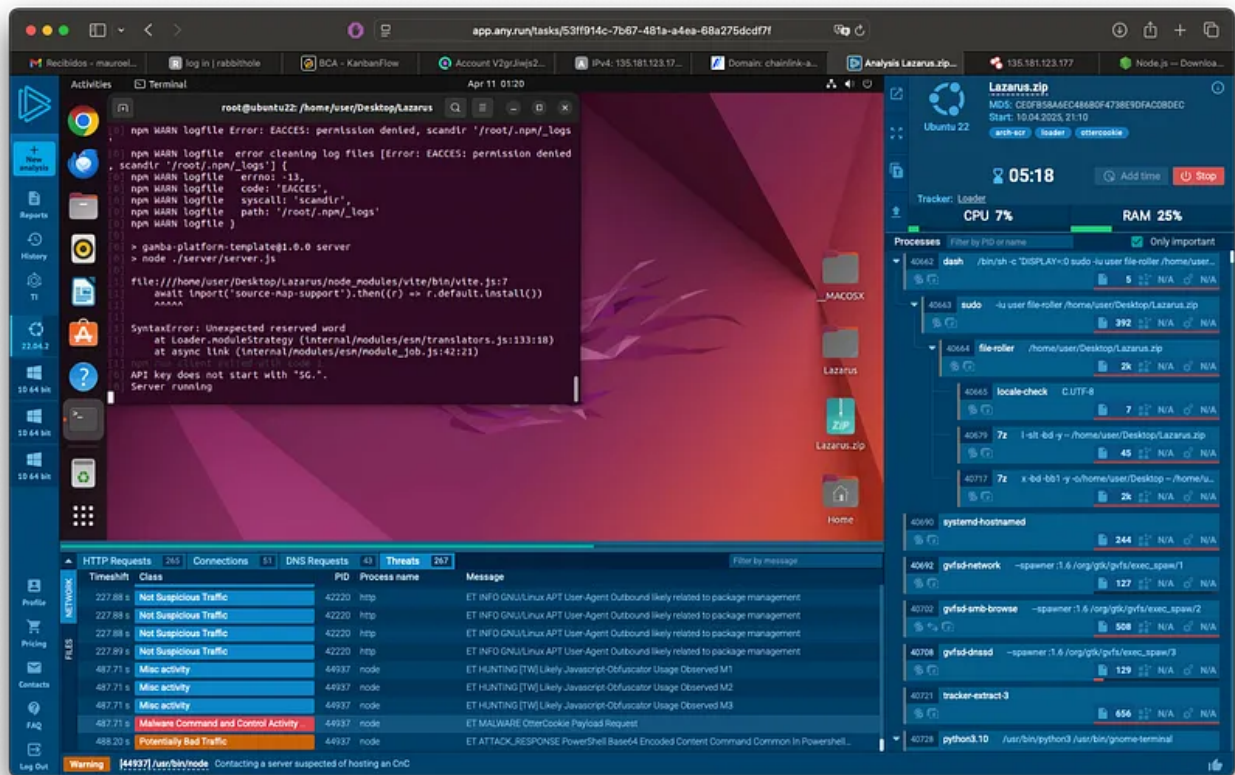
Running internet-facing services as a privileged user—worse yet, as **Administrator**—is a **bad idea** and Wilton will find out about it pretty soon.

The other service running on port 7777 is **Remote Desktop Protocol**, which allows us to authenticate—if only we had the necessary credentials.



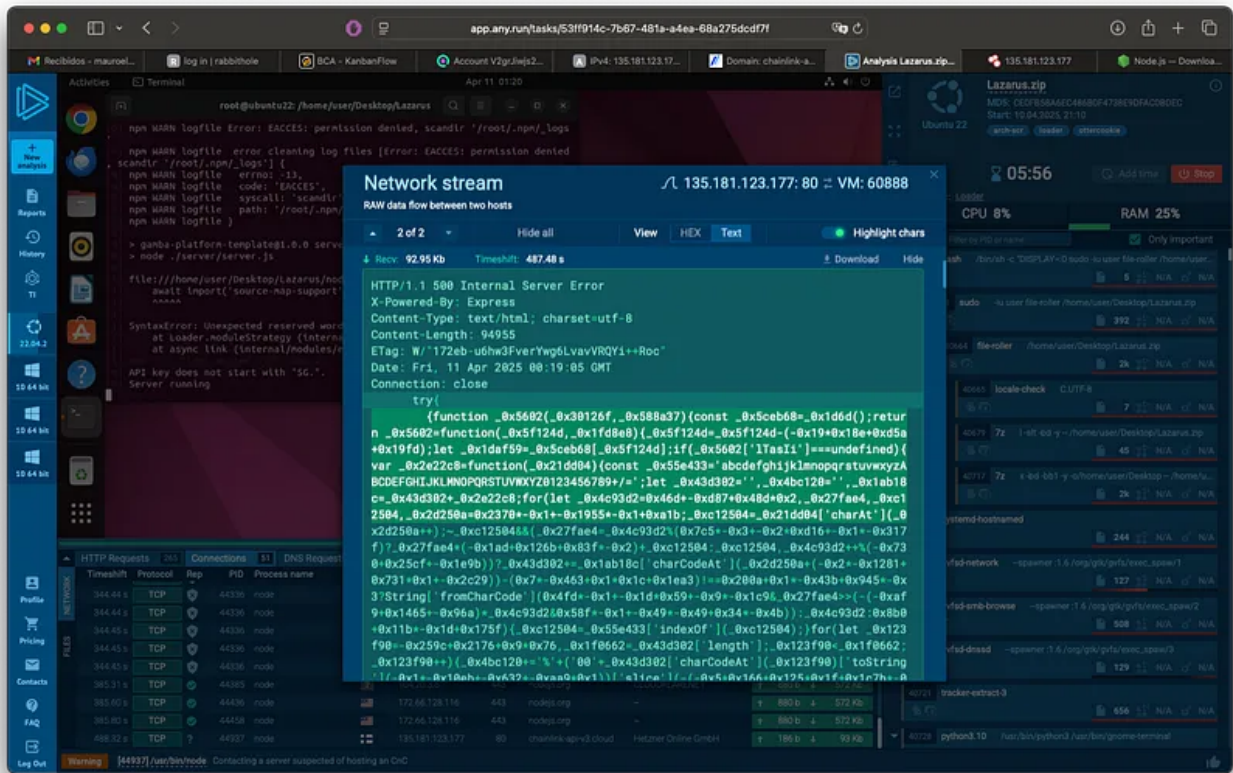
But that's a story *for another day*. They don't know we know, so let's use that to our tactical advantage.

Using [ANY.RUN](#)'s sandbox, we spin up a disposable [Ubuntu](#) machine, install [NodeJs](#), and run the code as if we were an unsuspecting victim trying to make a quick **500 USDT** (remember we're here for that reason after all?).

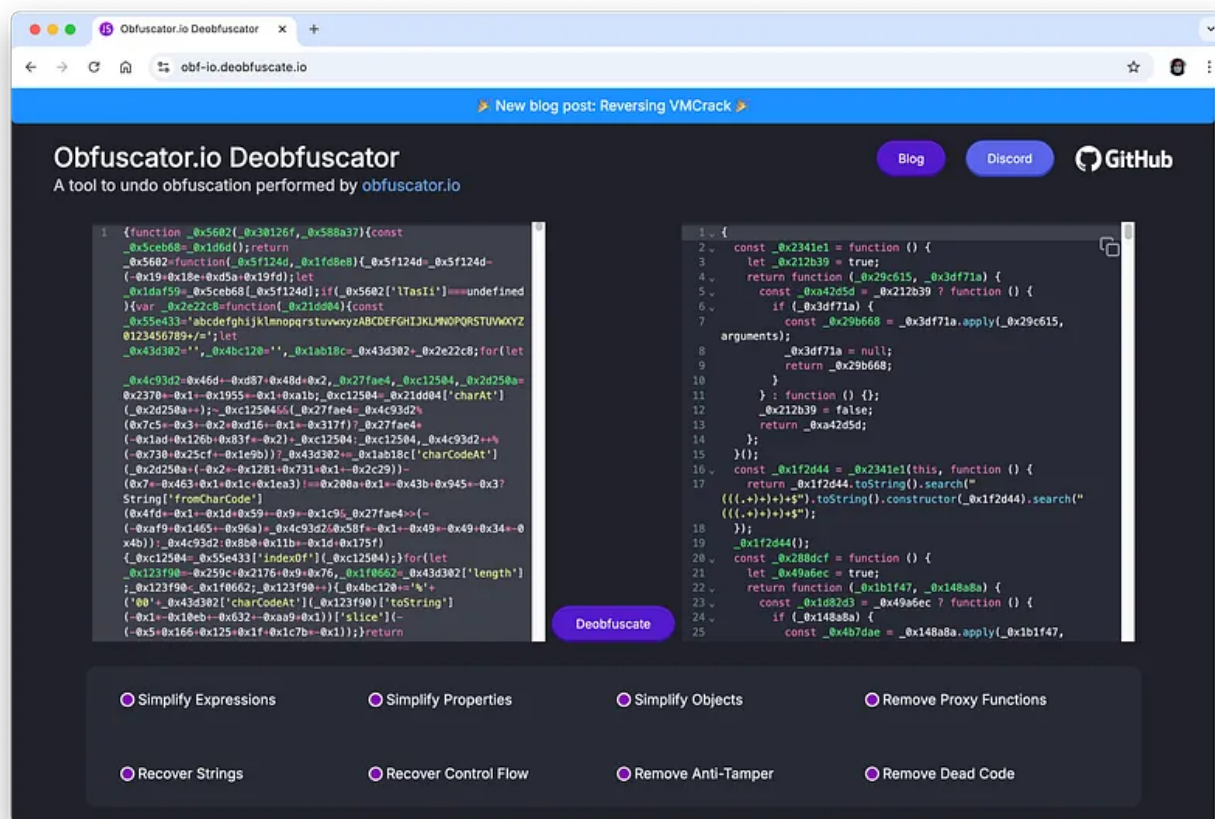


Running the code, of course, triggers the **mandatory failure**, which bumps into the **Try/Catch** block, which receives the error from the **Finnish API**.

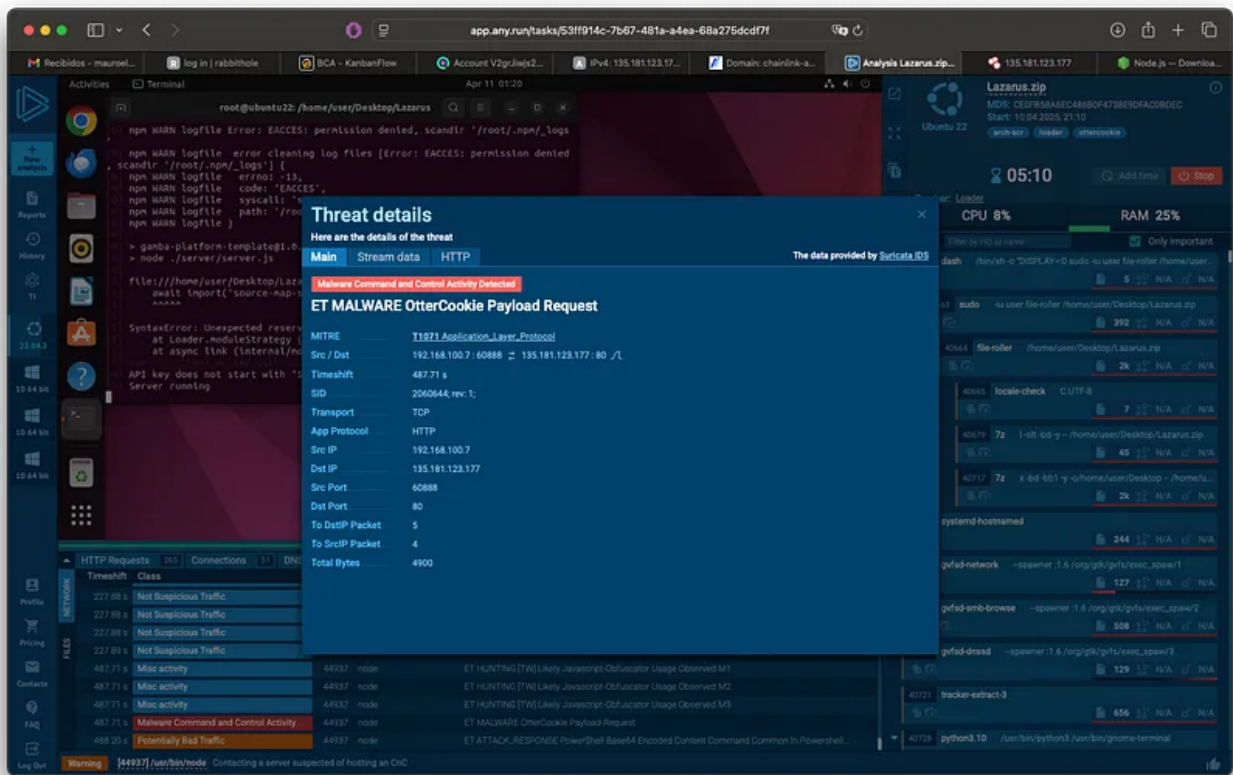
But what does that error say? It's an [obfuscated](#) JavaScript snippet.



This would be executed **directly** on our local machine while we're distracted trying to submit a patch for a visual bug. **Lazarus loves obfuscating JavaScript code with a [popular online tool](#)**, but since they don't know we know, we'll use it **against** them.



At first glance, it looks like [BeaverTail](#)—one of **Lazarus**’ latest cyberweapons, commonly paired with [InvisibleFerret](#)—but on closer inspection, it turns out to be another animal we still didn’t have in our personal collection: [OtterCookie](#).

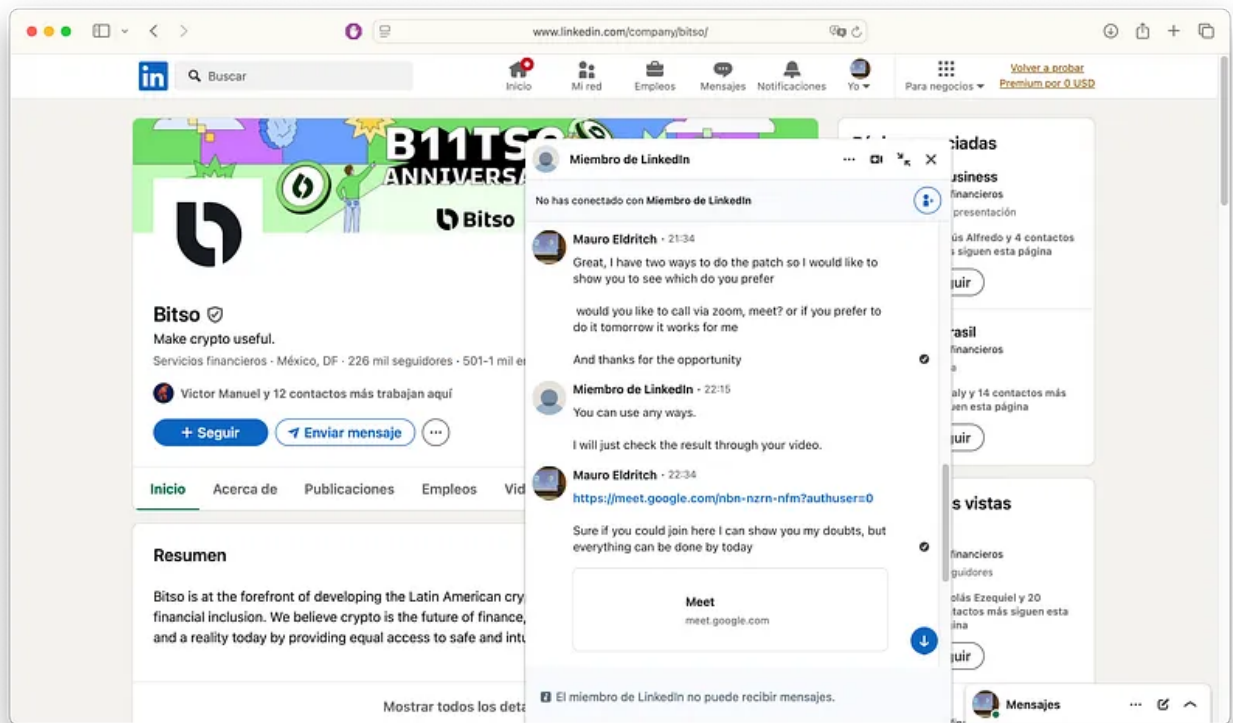


This not-so-playful otter works as a **Stealer-type malware**, targeting **browser password managers and extensions** (specifically **crypto wallets**), and is also deployed in the **ContagiousInterview** campaign.

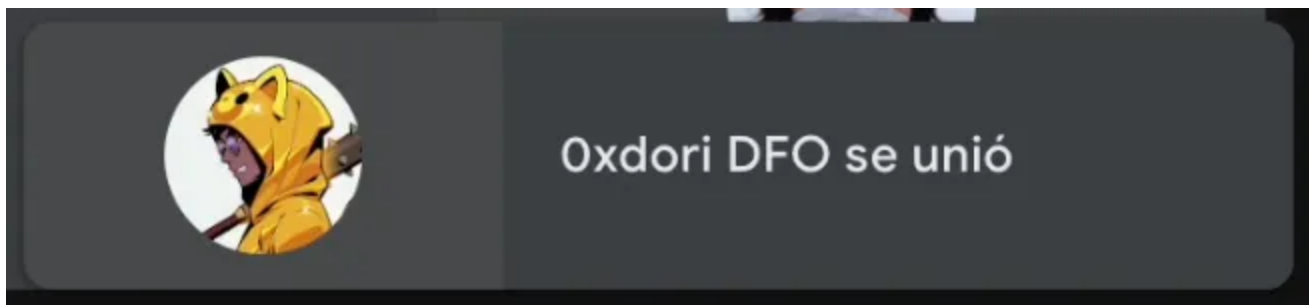
We *tried catching* the **Otter**, and we did. We have the infrastructure, the sample, the involved accounts, and their communication script.

It was time to contact our ‘employer’ to update them on the progress of the job.

I went back to LinkedIn and told my employer that I had two ways of implementing the patch and would like to have a short meeting to discuss them. **He bought it** and wanted me to **run the sample during the call** to see the results.



And so, I started recording my screen, waiting in silence for the wild **Chollima** to set foot in the trap—and it did, with an anime profile picture and under the name of “**0xdori DFO**”.



I expected to see a *majestic winged horse* stomping bravely over the scene, but instead, I ended up with a *scared little pony fleeing*.

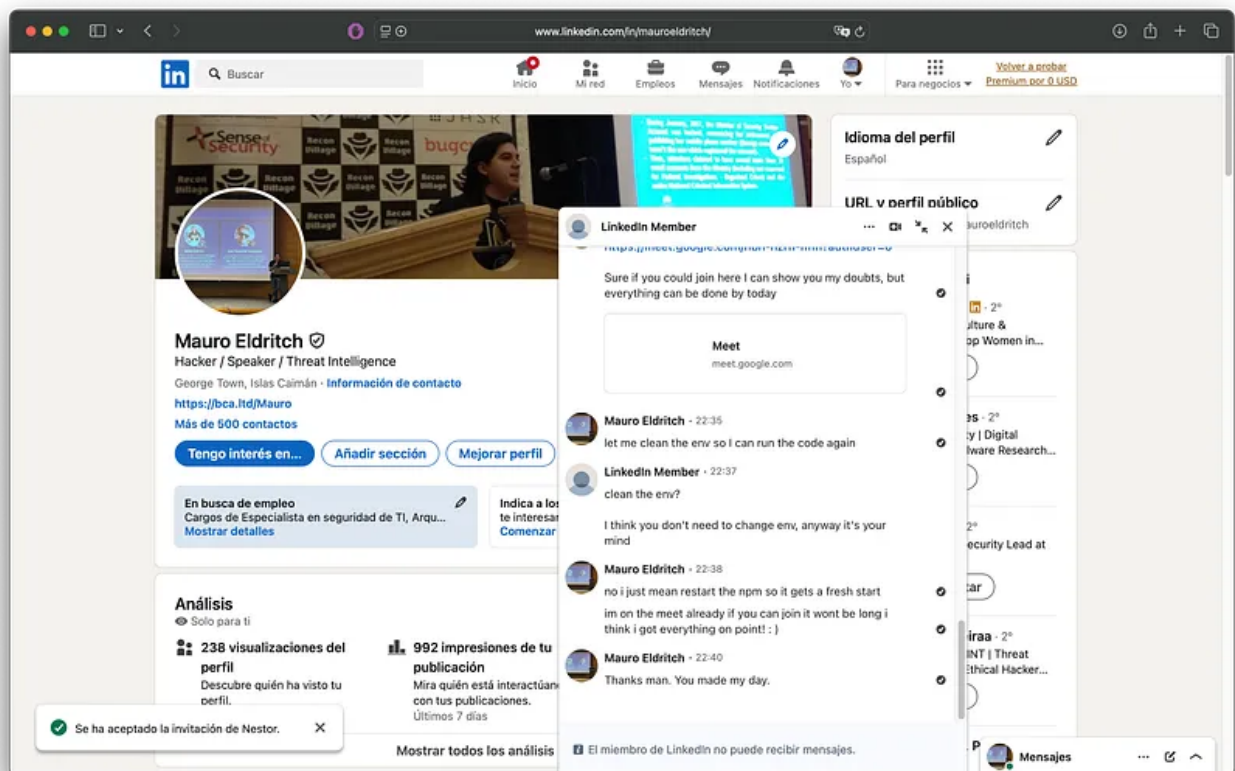
But, I'll let you judge that for yourself...

The following video has been edited (and the Threat Actor muted) to avoid disclosing certain indicators.

We may upload a full version in the future.

The pony pranced away in panic, without saying another word.

Being gentlemen, I thought we could at least exchange [a proper farewell](#), but I was promptly blocked.



We claimed the game and called it a day.

But, my friends, **the hunting season never ends...**

Indicators of Compromise

```
IPv4:135.181.123.177
Domain:chainlink-api-v3.cloud
URL:http[:]//chainlink-api-v3[.]cloud/api/service/token/56e15ef3b5e5f169fc063f8d3e88288e
URL:http[:]//chainlink-api-v3[.]cloud/api/
URL:https[:]//bitbucket.org/0xhpenvynb/mvp_gamba/downloads/
SHA256:aa0d64c39680027d56a32ffd4ceb7870b05bdd497a3a7c902f23639cb3b43ba1
SHA256:071aff6941dc388516d8ca0215b757f9bee7584dea6c27c4c6993da192df1ab9
SHA256:486f305bdd09a3ef6636e92c6a9e01689b8fa977ed7ffb898453c43d47b5386d
SHA256:ec234419fc512bade05f7b29fefbf12f898a505f62c43d3481aed90fef33687
FileName:0xhpenvynb-mvp_gamba-6b10f2e9dd85.zip
SOLWallet:V2grJiwjs25iJYqumbHyKo5MTK7SFqZSdmoRaj8QWb9
```

Resources & References

- [Original Intelligence Pulse on LevelBlue OTX](#)
 - [OtterCookie source code \(obfuscated\)](#)
 - [OtterCookie source code \(deobfuscated\)](#)
 - [Fake project \(loader\) source code](#)
-

Acknowledgements & Contributors

This work would not be possible without the contribution from dedicated **Lazarus Agents** who surrendered their cyber-weapons to the **Quetzal Team**. We honor them here:

✖ **Edward** from **Labyrinth Chollima**. Fell during the [QRLog](#) campaign, in which we discovered the [QRLog malware](#).

✖ **Nargis** from **Velvet Chollima**. Fell during the **DreamJob** campaign, in which we captured the [Docks malware](#).

✖ **Artyom** from **Velvet Chollima**. Fell during the **ContagiousInterview** campaign, in which -alongside another team- we discovered the [ChaoticCapybara](#) malware.

✖ **Wilton** from **Famous Chollima**. Fell during the **ContagiousInterview** campaign, in which we captured the **OtterCookie** malware.

Comment **[F]** to pay respects and don't cry for them: **they fell bravely against the best.**



[7](#)

Share this post



[Bitso Quetzal Team](#)

[Interview with the Chollima](#)



1

[Share](#)