From WordPress Vulnerability To Internal Access On Iranian Startup

This is a technical story with an educational moral.

I used to think that the world of hacking and security was purely fascinating and enjoyable. However, I've since realized it can be extremely risky. In this field, your first mistake could easily be your last…

Not long ago, I was invited to interview several interns in the field of hacking and security. During one interview, an intern presented a sample project as part of their response to a question. That's when I realized the severity of the situation: he had begun investigating the security of a well-known Iranian startup without obtaining permission or coordination. As you know, this is considered a crime. If the security team of that company had noticed his dangerous and suspicious behavior, it could have easily led to legal consequences.

Let's be honest with ourselves, most of us have made mistakes or acted mischievously when first entering this field. But the stakes are higher now. If you start working on a major project without proper authorization, the company's experienced technical and security team will

undoubtedly notice your behavior. If you push the boundaries too far, they'll take legal action against you, a simple act of mischief could leave a destructive and lasting mark on your professional record.

I spoke with that intern and discovered he was unaware of the legal ramifications. After a few days, I contacted the technical team of the startup and explained the situation in an attempt to amicably resolve the issue. Fortunately, they were professional and approachable. They welcomed my honesty and even invited me to collaborate further on the startup. And that's how this technical story began.

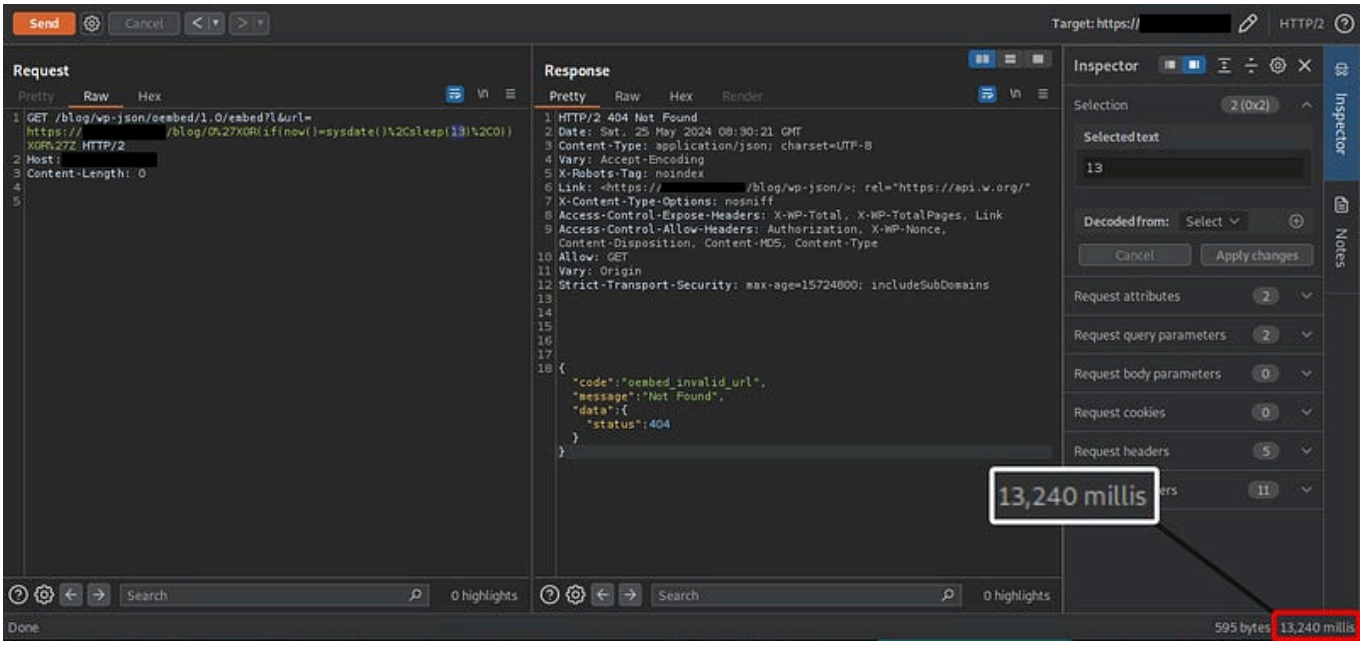
> Always remember that a million dollars in prison will not be useful for you…!

RECON PHASE

During the External Recon phase, I initially focused on vertical enumeration using my personal tools, which led to the successful discovery of some interesting subdomains. However, it's crucial to remember that the primary domain holds the most significance for the company, so I shifted my focus there.

The main domain was implemented using Node.js and had a robust security structure. For the first few days, I couldn't identify any sensitive vulnerabilities. However, I eventually discovered a `/blog` directory, which was hosted on a different server and implemented with WordPress.

Upon conducting a security analysis, I found that the WordPress instance was running an outdated and vulnerable version. I began researching CVEs and vulnerabilities associated with that version and successfully identified a CVE that led to an SQL injection vulnerability. This allowed me to extract certain data from the server. Although there was no existing exploitation method, I developed one myself.



Continuing my analysis, I discovered five additional critical vulnerabilities such as arbitrary file deletion and other within this WordPress setup.

Next, I proceeded to examine the other subdomains. One of them returned a 404 error, but the domain name seemed familiar, I had noticed it earlier in the BurpSuite HTTP History, which piqued my interest. After conducting some further investigation and performing directory fuzzing, I revisited the BurpSuite HTTP History for a more detailed analysis. This led me to discover that the subdomain was actually hosting a Django REST API system.

In response to a request made to the following endpoint:

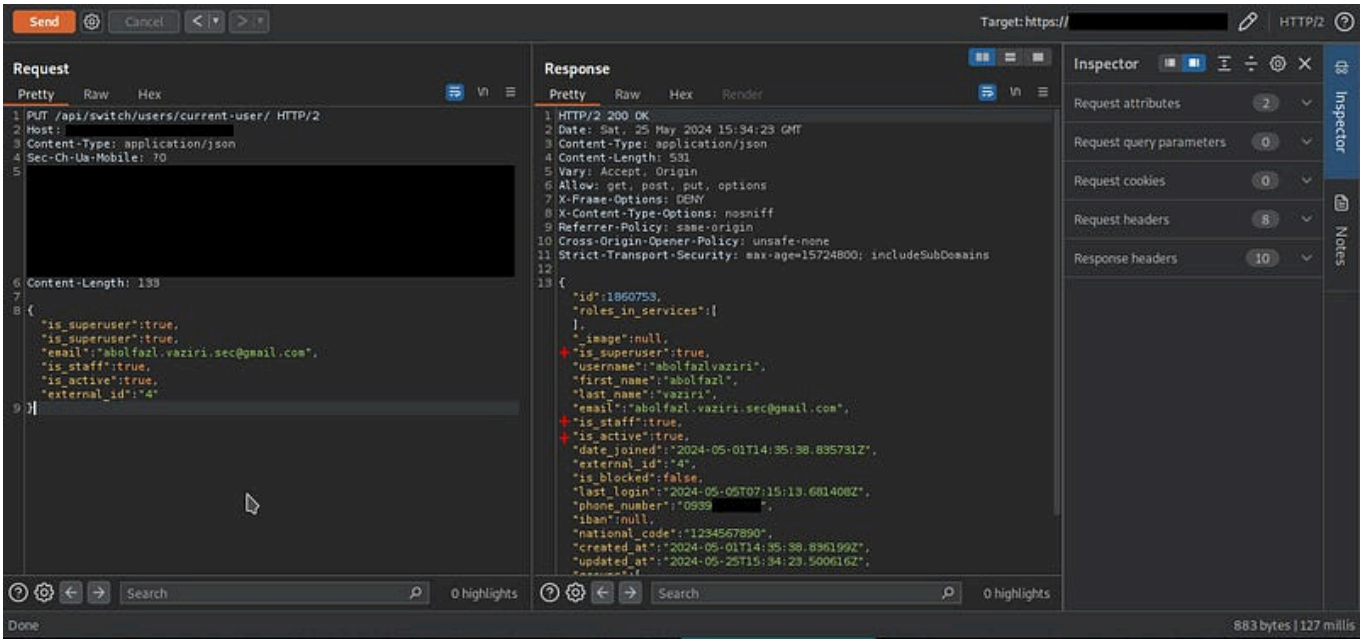
```
sub.startup.com/api/switch/users/current-user/
```

I received the following data:


```
{
 "id": 1860753,
 "roles_in_services": [],
 "_image": null,
 "is_superuser": false,
 "username": "abolfazlvaziri",
 "first_name": "abolfazl",
 "last_name": "vaziri",
 "email": "abolfazl.vaziri.sec@gmail.com",
 "is_staff": false,
 "is_active": false,
 "date_joined": "2024-05-01T14:35:38.835731Z",
 "external_id": "4",
 "is_blocked": false,
 "last_login": "2024-05-05T07:15:13.681408Z",
 "phone_number": "0939*******",
 "iban": null,
 "national_code": "1234567890",
 "created_at": "2024-05-01T14:35:38.836199Z",
 "updated_at": "2024-05-25T15:34:23.500616Z",
 "groups": [],
 "user_permissions": []
}
```


Here, I noticed that the data returned by the endpoint included detailed user information, including fields related to the user's access levels. I decided to attempt altering the access-related values in new request, and upon submitting the modified request, I observed that the

changes were successfully applied, converting a standard user's access to that of an administrator(BAC).



Interestingly, the changes were not reflected on the front-end of the website, and no new functionalities appeared to be available. This led me to suspect that the modifications might not have been properly registered, and that the response might have been erroneous. However, after reaching out to the technical team, I discovered that, in addition to the public website, this API was also being used by an internal system for the company's personnel. The access levels I modified were actually defined for that internal system, and the changes were being recorded there.

This may sound a bit confusing, but it's actually quite simple. They have a single API that serves multiple endpoints. Each endpoint utilizes only the specific parts of the API that it needs. In this scenario, I was able to escalate the access levels within the startup's internal system, ultimately gaining maximum access privileges.

> It's important to note that even if you don't find specific parameters in the response to your request, you should proceed with a blind approach by sending various parameters and values. This way, you can identify the correct parameters and use them to explore potential scenarios, such as the one described here.

GitHub: https://github.com/abolfazlvaziri
Instagram: https://instagram.com/abolfazlvaziriofficial
Telegram Channel: https://t.me/AVN_COMMUNITY
YouTube: https://www.youtube.com/@abolfazlvaziri