Bypassing MPX Node Authentication - Firmware analysis

A full Red Team engagement scenario to get initial access to the Omnia MPX Node.

Introduction

Hi again!

I'm back once more with a new Red Team engagement snippet that delves deeper into the firmware and combines Black Box testing without source code, credentials, or even any information about the assets to simulate the malicious outsider actor (Black Hat), then turns it into a White Box through firmware analysis and research techniques.

When you are in a Red Team engagement, all you need to think about is the initial access to the protected and prohibited resources from any outsiders or unauthorized actors. This means you may face a slew of vulnerabilities, but no one will give you anything or even assist you in your goal. So, you are going to skip it easily, and today's scenario will be a chain between multiple vulnerabilities, along with some firmware analysis and some other techniques that could help you gain initial access to an asset.

Finally, this is a zero-day that has been discovered by me, and this is the first advisory about it. It's now under assignment and the vendor of this product will be notified about it very soon. The reason for sharing the details about it is to help developers mitigate it as soon as possible and, at the same time, the companies understand the business and security impacts that could affect them while using this affected firmware.

Discovering the asset

First of all, when I entered the affected asset after deep port scanning, I faced the following login panel for a product called "Omnia MPX Node":

MPX Node Login

As usual, I searched for the default credentials, AKA backdoor accounta related to this product, the weak credentials, along with some credential stuffing, and I was able to login through the credentials:

```
demo:demo
```

low privilege demo account

Usually, the demo account is only for what the name refers to. It's just a demo for the product without any permission, privilege, or access to anything, so we are still in the beginning. Then I decided to forget the demo account and start acting like an outsider who wants to get inside but has no credentials or accounts. The first thing I decided to do was the fuzzing. Improper access controls are very common in such products, so for example, if I found a broken access control to something that could be a backdoor for me, it's an easy win and maybe chained with something else to produce something bigger. So, I started with the common wordlist in SecLists and got the following result:

logs endpoint can be easily accessed without any type of authentication

Chaining vulnerabilities

Now, let's try access the log endpoint to see what kind of logs exists there and how can we get benifit from this endpoint to get initial access to the MPX Node panel with high privilege:

logs endpoint unauthenticated access

As you can see, we are unauthenticated actors but have the permission to access the log endpoint and download the logs too. So, this is an improper access control but we still don't get initial access, so we are going to continue. When I tried to download any log file, I found the following information in the logs:

the logs shows the username that loggged in/out and interesting fname parameter

Now we have two interesting points here: The first is the username that is being shown in the logs when the user logs in or out of the panel; the second point is the fname, which includes the log file name; so, we can try getting Local File Disclosure here to get access to any point that may contain credentials:

passwd with only system user is root

Based on my experience, this passwd file shows that the only existing users here are root for the system and www-data for the web server. So, we have a 99% possibility of showing files with the root privilege. To see if this is right or not, I'm going to show the shadow file contents like this:

shadow file is accessible with root privilege

Firmware downloading

For your information, any credentials obtained from the shadow, service configurations, or even passwd file in some cases, could be used for password spraying or reusing, but this was

not possible in my case, so I decided to analyze the firmware to see where the credentials

firmware update file

I know that we don't even know what the product version is that we are working on. I tried to enumerate the version with multiple techniques, but I wasn't able to get the product's version. So, I decided to download the firmware directly from the website. Then I'm going to understand the workflow, paths, and technology that this product works with, then implement and plan for the attacking vectors that will be simulated against this product. So, after downloading and extracting this firmware, I found the pkg file inside it like that:

firmware pkg

Analyzing the firmware

Now, let's unpack and extract the firmware files to see the whole content of the firmare, including the system and web files. To do so, I used the command:

```
binwalk -Me 1601-00585-XXX-mpxnode-1.3.35.pkg
```

extracted image

Now, let's unpack and extract the firmware files to see the whole content of the firmare, including the system and web files. To do so, I used the command:

1. The Kernel
2. The System, Configurations, Data files and everything that supports a full Linux System (rootfs)

Our goal is to gain initial access to the panel, so we should look deep into the system files; we won't even touch the kernel, so we'll look deep into the rootfs; but first, we need to know where we were when we were trying to get the logfiles. This could help us in getting into the product's configuration or even source code files, so I decided to go back to the product again and try to put in an unexisting file to see what was going to happen:

full path disclosure

Do you remember the full pathway disclosure? It was a misconfiguration with very low impact, but now it will be our entrance key. So, I'm going to seek out the `appLogs` path in our firmware. To do so, I used this simple command:

```
grep -i -r applogs
```

And I was able to detect where the applogs path is, and at the same time, I got some interesting scripts:

the applogs path in this firmware and at the same time installation script

You may ask why the installation script could be interesting. In most cases, it contains credentials for databases or services and even the paths to the files that contain these credentials. It also gives you a deeper look at the implementation of the software and how everything is working there. So, before I jumped into the path of the appLogs, I wanted to see where the version is in this firmware, so I surfed into the root path of this firmware and found this `info.txt` file:

info.txt is the version file, located in the root path

As you can see in the image above, the `info.txt` is the firmware version file and located in the root path of the system, it looks like outdated/old in this firmware even if the vendor's website said that it's up to date, let's simulate the same path in the product we have and see what is the version that the target's firmware works with:

the latest version is 1.5.0+r1

Credentails storage

So, as I expected, the firmware on the vendor's website is for hard-installing, but there are online updates that the product receives that include improvements and bug fixes, and the version in our hand seems to be the latest version. Now, I'm going to surf into the `appLogs` path and see where the configurations are located:

the start of appLog file path

If you remember, the starting path of the `appLog` was `/usr/mpxnode`. So, let's take the lead from here and start analyzingthe scripts. First, I started with the installation script:

configurations paths

I like databases a lot:), so I decided to look at `userDB.json` to see what it contained:

login usernames and passwords

Now, we are going to take this path and put it into the vulnerable application. As we know, we were located in the `appLog` and it's located along with the `appConfig` ,which contains the `userDB.json` :

they are located at the same path

Obtain credentials

So, what we are going to do is just go back one step in the application and then write the full path of the `userDB.json` like this:

all the users credentials

As you can see, we were able to get the whole users credentials, let's try login with the highest privilege then and see the difference:

accessed

So, now we have the highest privilege here. You can intercept the FM waves, edit the network, route the traffic, manipulate the DNS to launch internal attacks and do everything. I did a deep analysis for this product and found the following:

file uploading firmware

So ... do you think we done? The rest is explicted ;).

Previous
The art of vulnerability chaining (PyScript)

Next
Cheat Sheets

Last modified 3mo ago