Initial access via PDF file silently

A technique for gaining initial access to internal infrastructure without requiring any interaction from employees or users by attacking wkhtmlTOpdf v0.12.6.

Introduction

Hi Infosec community,

It's been a long time since I wrote a blog. I'm back again with a new technique used by Red Teamers to get initial access to the infrastructure without requiring any user interaction. Before we start, this is a research I made in wkhtmlTOpdf 0.12.6 library and assigned as a CVE with ID: CVE-2022-35583, through this zero-day the attacker can get initial acess to the internal assets and gain unauthotized access to functions and informations of the compromized assets. This means, you can gain access and surf the internal infrastructure anonymously without getting caught. In this blog, our method to gain initial access to that infected PDF will be through SSRF (Server Side Request Forgery) but with the PDF file. Before you exit this blog after knowing that the initial access will be with SSRF, which the web is full of these techniques, let me tell you that our method today will be different. We will print the content and the assets access into the PDF, and the PDF will be rendered to us again with additional papers that will have the internal infrastructure screens.

Affected Environment

This attack technique works in any environment working with 'wkhtmltopdf' tools. These 'wkhtmltopdf' tools are command line tools used to convert the HTML page to PDF. In the system, the insecure handling of user input causes a lot of attacks that could affect the internal infrastructure. I was in an environment where HTML-based invoices were converted to PDF using the 'wkhtmltopdf' tools; I did a thorough analysis of this library and then simulated an attack that could occur using this set of tools.

Building The Methodology

Before I start any assessment, if I find something attractive that could help me gain initial

access to an infrastructure, I'd like to plan and put together an organised path starting from the payload crafting to the final results of the attack. I drew the plan in my mind like that:

- As we know, the 'wkhtmltopdf' takes the HTML and renders it to PDF, and all of this is being performed initially in the system.
- Then I could look for a method that makes me exploit the functionality of the HTML to gain initial access to the internal assets.
- To do so, I need an HTML tag that helps me retrieve information and data from a specified source.
- After that, I need to see what assets I want to access. I wanted to access the web based assets internally. When thinking of accessing internal infrastructure, you should put into consideration the IP classes. So, for example:
  - IP Class A: 10.10.10.1
  - IP Class C: 192.1.1.1
- Finally, if we were able to integerate the HTML tag that makes us retrieve or access resources with a set of IP calasses to internally gain access through an SSRF attack, we would get initial access to the systems in the target infrastructure.

Validating Exploitation

Before starting the exploitation, I decided to validate the possibility of exploiting the flaw and gaining initial access. I wanted to see what the IP address was that would interact with the source of the asset that I wanted to access. I know that the IP address will be the server's IP address because the whole generation and rendering process will be fully on the server-side, but I wanted to double check that.

First, let's analyze the request that handles the parameter which contains the PDF structure that will be sent to the back-end:

As you can see, the HTML structure will be sent to the back-end in order to render it to PDF. I'm going to add the payload:

```
<iframe src="http://yourHiddenCollab/" width="1000" height="2000">
```

You will also add that the collaborator you will add should be unintendent, black-listed, or even familiar with the SOC team to avoid being caught. In this case, I'm going to use the burp collaborator just because I've already done the engagement, but I highly recommend using your own domain that's connected to your server's IP address to monitor the traffic that could come from the target.

Some people may argue why we just didn't obfuscate the payload. Then the SOC will not be able to identify the threat! However, not all techniques can be used in all cases; if you do something unusual or not ordinary here, such as obfuscation, the activity will be immediately identified as malicious. Instead, you can host a JQuery script file on your server with your own domain and inject the following:

```
<script src="https://securedomain.com/scripts/jquery.min.js"></script>
```

Then, when the application comes to crawl/import the JavaScript, the IP of the request origin will be logged into the server's logs. Because of the restrictions on executing and importing JavaScript files, the JavaScript import can be quite dangerous at times.In that case, you can use the following:

```
<img src="http://securedomain.com/assets/images/puppy.png">
```

Anyway, there are non-countable ways to do it. All you need to do is to think deeply about your goal and the move you are going to make, and then you will simply do it. I received the interaction from the server's IP address. Then we are sure that the whole process is fully implemented in the back-end and we can start attacking:

Interaction after injection

Attack launching

Now, and as usual in these cases, I tried to get the localhost address, but unfortunately, when the rendering is done, I get that there are no resources available at the local host like that:

No resources avaliable in localhost

Then, I decided to move to other assets. I started with the IP Class A assets. It's one of the most common classes used internally in companies, and I was able to access the Cisco Miraki panel through the IP Class A like that:

And after some digging, I was able to gain access to the Cloudflare panel they are using, but I cannot add this here.

Final Touches

Now, I made a simple diagram that explains how it works to make the mission easier for the developers team. It would benefit the non-technical people too to understand what's really happening in this attack:

Previous
Blogs

Next
PHP Backdoor Obfuscation

Last modified 3mo ago