Setting up Single Sign-On (SSO) between G Suite and Office 365 with G Suite as identity provider (IdP)

James Winegar
Jan 2 · 7 min read

What are we trying to accomplish?

We've been working with one of our G Suite accounts and manually syncing members to Office 365 who have requirements for Microsoft Office licenses. This is a huge pain and we wanted to tighten up our security with SSO everywhere and enforcing 2FA.

Traditionally organizations have used Microsoft's Active Directory (AD) and its federation solution, Active Directory Federation Services (ADFS), to bridge authentication and authorization. However, we're using G Suite as our identity provider since it has nice integrations with a lot of the tools we're using.

Here's where the fun begins. Since we're taking the road less traveled and not integrating with ADFS, we will be using a Google maintained app for integrating with Office 365. It's not directly plug and play since Google needs to know which user is which through an immutable ID. This leads to some interesting things for us later on.

Most SSO solutions leverage SAML 2.0 (Security Assertion Markup Language 2.0) in order to provide authentication and authorization. IBM actually has a great definition here: SAML 2.0 is a version of the SAML standard for exchanging authentication and authorization data between security domains. SAML 2.0 is an XML-based protocol that uses security tokens that contain assertions to pass information about a user between a SAML identity provider (IdP) and a SAML service provider (SP). SAML 2.0 enables web-based authentication and authorization scenarios that include cross-domain SSO, and thus helps to eliminate the distribution of multiple authentication tokens to the user.

The biggest benefit of SAML is that a user authenticates with their normal workflow and the SAML integration provides a streamlined user experience, increased security, and less administrative configuration and management.

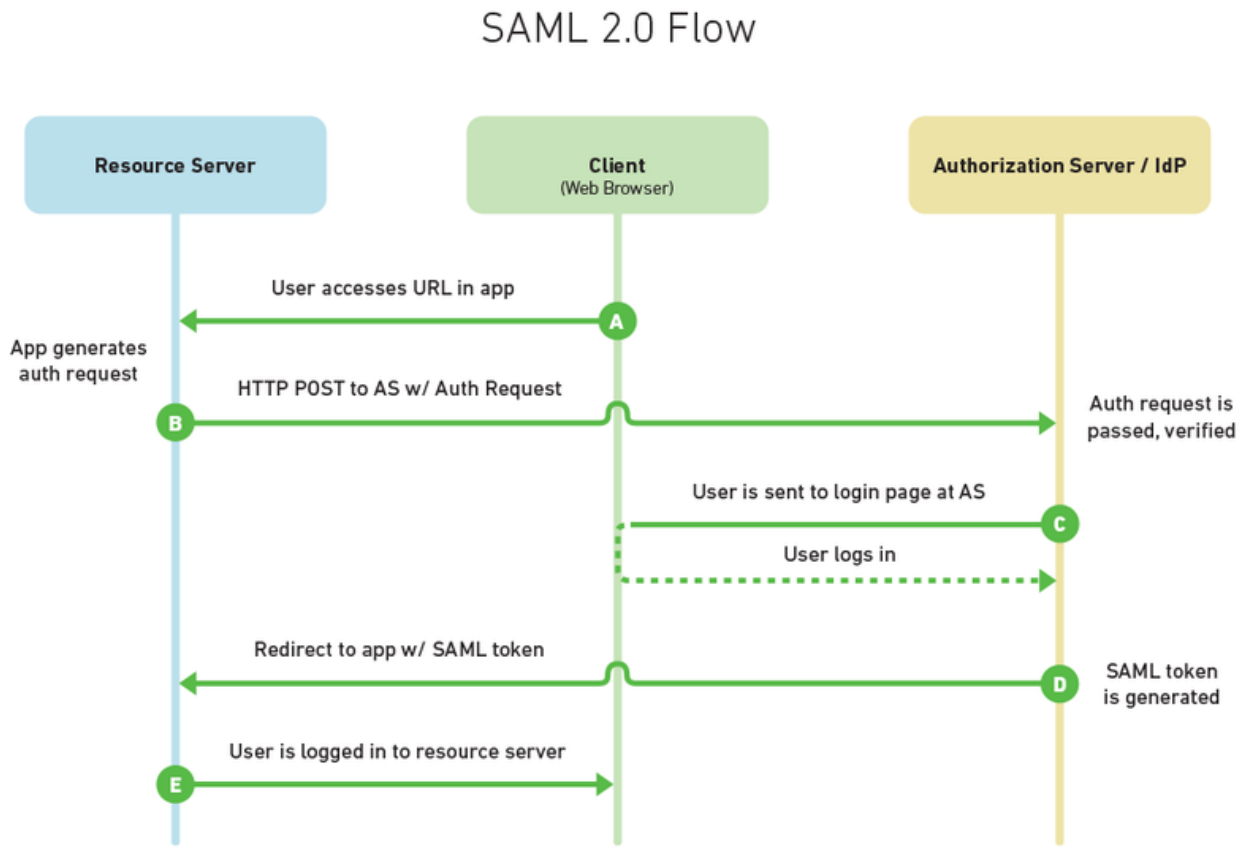

Figure 1. Outline of SAML 2.0 Flow. Reprinted from "An Interoperability Framework for Identity Federation in Multi-Clouds" A.M. Saied, International Journal of Computer Networks and Communications 7(5):67–82 2015

Plan of Action

1. Create a new Office 365 Tenant
2. Add our domain to Office 365
3. Verify our domain with Office 365
4. Setup SAML SSO certificate on G Suite
5. Setup SAML App for Office 365 on G Suite
6. Configure Office 365 to leverage SAML with G Suite's parameters
7. Set up automated user provisioning

During the course of this blog, we'll try to use the command line as much as possible. We'll be using PowerShell because of its Office 365 client libaries in conjunction with gcloud SDK for when we update DNS records. We believe you could automate most of this work with API-based workflows but that was outside the time window we gave ourselves to finish this project. If people are interested in automated provisioning of this let us know!

Step 1 — Create a new Office 365 Tenant

We tried very hard to make this work for an existing tenant; five solid attempts with various degrees of success. However, there were several dependencies the G Suite integration requires that just didn't play nice with the existing tenant.

Some example issues include

- Users unable to login because their immutable ID doesn't match between the two systems.
- Pre-existing users not generating immutable IDs in G Suite from their existing ID.

It's probably the case that since we're using G Suite as our IdP it likely has all our important information and Office 365 is just a licensing portal. Luckily in our case this was true and we could rip the existing tenant down and start fresh. In our case that meant quite a bit of headache with dealing with Azure Active Directory and Office 365, but a little bit of PowerShell later and we were good to go.

If you don't know how to setup an Office 365 tenant, here is some documentation to get you started.

Step 2 — Add our domain to Office 365

Rightfully so, Microsoft won't let you use accounts for domains you don't own. So for our domain we're going to need to validate it with a DNS records. Let's automate this whole process!

If you've never used PowerShell before, I recommend a little bit of reading. Here is a decent tutorial.

Step 2.1 — Connect to Office 365 in Powershell

```
1  Install-Module MSOnline
2  Import-Module MSOnline
```

```
3  $msolCred = get-credential
4  Connect-MsolService -credential $msolCred
```

Office365-login.ps1 hosted with ❤️ by GitHub view raw

Step 2.2 — Add our domain to Office 365

Here we initially create our domain using Azure Active Directory (Office 365's version of IdP).

```
1  $domainName = "example.com"
2  # We'll start with a managed domain so that we can get Authentication clearence easily.
3  New-MsolDomain -Name $domainName -Authentication Managed
```

Office365-new-domain.ps1 hosted with ❤️ by GitHub view raw

Step 3 —Verify our domain with Office 365

We need to store our output of what type of record so we need to use its contents later. Office 365 allows a couple ways to validate. One of the simplest is to add a TXT record to your DNS to show that you have ownership of the DNS and thus can control all aspects of the domain.

Step 3.1 — Tell Office 365 to give us the text record to add

```
1  $recordToAdd = Get-MsolDomainVerificationDns -DomainName $domainName -Mode DnsTxtRecord
```

Office365-domain-verify-txt.ps1 hosted with ❤️ by GitHub view raw

This record has three properties

```
Label : example.com
Text  : MS=ms84927192
Ttl   : 3600
```

Step 3.2— Add DNS using gcloud

Since we're focused on G Suite you might be using GCP for your DNS, let's use gcloud command line to update our DNS record in a transaction.

If Google is not your DNS provider, just update it with whichever tools you are comfortable.

Here we add a TXT record of $recordToAdd.Text with a TTL of $recordToAdd.Ttl and name of $recordToAdd.Label

```
$managedZone = "example"

# We're making an assumption that I don't already have a TXT record with a label of our
# You can update this however you need it to be.
gcloud dns record-sets transaction start --zone $managedZone
gcloud dns record-sets transaction add --zone $managedZone `
      --name $recordToAdd.Label `
      --ttl $recordToAdd.Ttl `
      --type TXT $recordToAdd.Text
gcloud dns record-sets transaction describe --zone $managedZone
gcloud dns record-sets transaction execute --zone $managedZone
```

gcloud-add-text.ps1 hosted with ❤️ by GitHub view raw

Step 3.3 — Wait for verification

DNS is not an instantaneous action and has side-effects and consequences. This is why we did our DNS updates in a transaction with the gcloud command line. If you don't know what we mean by this I highly recommend reading up more on it since it will save you a lot of frustration in the future. Here's a few short guides to get you going. [An Introduction to DNS Terminology, Components, and Concepts] [How DNS Works] [What is a domain name?]

Typically, you can expect to wait a few minutes for DNS to update across the board. Office 365 asks for 5–10 minutes, so we built in a waiter of 15 minutes before it tells us something might be up with our DNS.

The code below checks if our domain is verified every 30 seconds, and notifies us of a problem if it has not been verified after 15 minutes. This will keep running until we're verified so if you have an issue you can keep it running while you're debugging.

```
$sleepSeconds = 30
$TimeStart = Get-Date
$TimeEnd = $timeStart.addminutes(15)
Write-Host "Start Time: $TimeStart"
write-host "End Time:   $TimeEnd"
Do {
```

```
Do {
  $TimeNow = Get-Date
  if ($TimeNow -ge $TimeEnd) {
    Write-host "Something is taking a while. Are you sure you updated your DNS?"
    Write-host "Check if you have the following TXT record in your DNS:" $recordToAdd.T
  } else {
    Write-Host "Domain not available yet"
  }
  Start-Sleep -Seconds $sleepSeconds
} while ((Get-MsolDomain -DomainName $domainName).Status -ne "Verified")
```

Office365-wait-for-domain-verification.ps1 hosted with ❤️ by GitHub view raw

Step 4 — Setup SAML SSO integration on G Suite

Sadly we did this in the web console for G Suite so this isn't scripted. I'm linking to the official documentation here since things have a habit of changing over time. [Documentation]

The jist of it in its current state is

1. Login to G Suite Admin Portal
2. Go to Security
3. Go to Set up single sign-on (SSO)
4. Generate a X.509 certificate
5. Download IdP metadata

After we have our IdP metadata download (XML file). Let's verify the contents since we'll use this later in our script. Below is an example with a fake certificate. Never share your real certificates; however, we should note that this certificate is pretty easy to pick up once we setup SSO since it's how we verify the sender. It's still best practices to only share what you need, this is similar to the idea of Principle of least privilege, but slightly different context.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" entityID="https://
  <md:IDPSSODescriptor WantAuthnRequestsSigned="false" protocolSupportEnumeration="urn:
    <md:KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>MIID2jCCA0MCAg39MA0GCSqGSIb3DQEBBQUAMIGbMQswCQYDVQQGEwJKU
```

```
A1UECBMFVG9reW8xEDAOBgNVBAcTB0NodW8ta3UxETAPBgNVBAoTCEZyYW5rNERE
MRgwFgYDVQQLEw9XZWJDZXJ0IFN1cHBvcnQxGDAWBgNVBAMTD0ZyYW5rNEREIFdl
YiBDQTEjMCEGCSqGSIb3DQEJARYUc3VwcG9ydEBmcmFuazRkZC5jb20wHhcNMTIw
ODIyMDUyODAwWhcNMTcwODIxMDUyODAwWjBKMQswCQYDVQQGEwJKUDEOMAwGA1UE
CAwFVG9reW8xETAPBgNVBAoMCEZyYW5rNEREMRgwFgYDVQQDDA93d3cuZXhhbXBs
ZS5jb20wggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAw</ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDF
    <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect
    <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Lo
  </md:IDPSSODescriptor>
</md:EntityDescriptor>
```

GoogleIDPMetadata-$domainName.xml hosted with ❤️ by GitHub view raw

We see our certificate, as well as a few links. These links tell the SSO process how to redirect requests/responses.

Step 5 — Setup SAML App for Office 365 on G Suite

I'll make a call out to the documentation for G Suite here again with a couple call outs about the current state. We're working through step 3 of this right now.

The jist once again is

1. Login to G Suite Admin Portal
2. Go to Apps > SAML apps.
3. Click Add
4. Search for Microsoft Office 365
5. Click Next
6. Check Signed Response
7. Set NameID to Basic Information:Primary Email
8. Set NameID format to "persistent".
9. Click Next
10. Set IDPEmail Attribute to Basic Information:Primary Email

11. Click Finish

We're now almost done setting up SSO for Office 365, we need to configure Office 365 to leverage that XML file we used earlier.

Step 6 — Configure Office 365 to leverage SAML with G Suite's parameters

Office 365 is kind enough to provide another PowerShell cmdlet called Set-MsolDomainAuthentication which let's us switch from Managed to SAML federated authentication.

Here we just parse the XML using a native XML parser in PowerShell (very nice) and pull out the data we need.

```
# If you want to automate getting the domain SSO metadata see https://developers.google

[xml]$idp = Get-Content C:\Path\to\xml\GoogleIDPMetadata-$domainName.xml

$activeLogonUri = "https://login.microsoftonline.com/login.srf"
$signingCertificate = ($idp.EntityDescriptor.IDPSSODescriptor.KeyDescriptor.KeyInfo.X50
$issuerUri = $idp.EntityDescriptor.entityID
$logOffUri = $idp.EntityDescriptor.IDPSSODescriptor.SingleSignOnService.Location[0]
$passiveLogOnUri = $idp.EntityDescriptor.IDPSSODescriptor.SingleSignOnService.Location|

Set-MsolDomainAuthentication `
  -DomainName $domainName `
  -FederationBrandName $domainName `
  -Authentication Federated `
  -PassiveLogOnUri $passiveLogOnUri `
  -ActiveLogOnUri $activeLogonUri `
  -SigningCertificate $signingCertificate `
  -IssuerUri $issuerUri `
  -LogOffUri $logOffUri `
  -PreferredAuthenticationProtocol "SAMLP"
```

Office365-parse-gsuite-idp-metadata.ps1 hosted with ❤️ by GitHub view raw

This has a lot of handling going on, but it's quite readable. We're just reading from the XML and creating a few variables to update our authentication for our domain.

We ran into a few race conditions here and decided to wait a while before we went on for all of Microsoft's internal systems to update.

A useful command for checking what your current authentication is

```
Get-MsolDomainFederationSettings -DomainName $domainName
```

Step 7 — Set up automated user provisioning

At this point we just need to map our users over to Office 365 from G Suite. Google provides a nice way to do this with “User Provisioning”.

The documentation is once again pretty good and hopefully updated so that this article ages well. We found that the default provisioning configuration did not work to our standards, but luckily there are many configuration options based on your security needs.

Once you follow all of the instructions, wait 5 minutes and in your audit logs you should see user provisioning start to happen. The admin audit logs are at a link similar to https://admin.google.com/example.com/AdminHome?fral=1#Reports:subtab=admin-audit

If you check in Office 365 you should now see your user with no licences as well. Another way would be to run a PowerShell command to return the users.

```
Get-MsolUser -DomainName $domainName
```

Conclusion

Setting up Office 365 as a SAML SP for use with G Suite as an IdP was a unique challenge that required us to create a new Office 365 tenant. Overall we felt like the process was fairly seamless and we spent around 8 hours implementing this.

We hope you find this useful and let us know if you have questions!

Authentication Sso G Suite Office 365 Saml