A Golden SAML Journey: SolarWinds Continued

splunk.com/en_us/blog/security/a-golden-saml-journey-solarwinds-continued.html

January 8, 2021

By Marcus LaFerrera January 08, 2021

TL;DR: In this blog post we will review what SAML is, how what is old is new again, and how you can start detecting and mitigating SAML attacks. Our focus for detection is intended as scaffolding to get you started, rather than a solution that will work for everyone and all installations. We won't spend much time going over Sunburst or Supernova, instead, we will focus on some lessons learned and how we can better position ourselves against future attacks.

What is SAML?

Security Assertion Markup Language (SAML) is a method for exchanging authentication and authorization between trusted parties. It's essentially an XML schema that allows for federated Single Sign-On (SSO) to work. At least, this is how it's most commonly used today. The basic construct is when a client attempts to authenticate with a service provider, they are redirected to an authentication server. Once authenticated, they are provided a cryptographically signed response that the client then provides back to the service provider. Once received, the response is validated thanks to the magic of cryptography. At least, that's how it's supposed to work and keep us (and our data) safe.

If this is confusing, take a look at this flow chart from Sygnia:

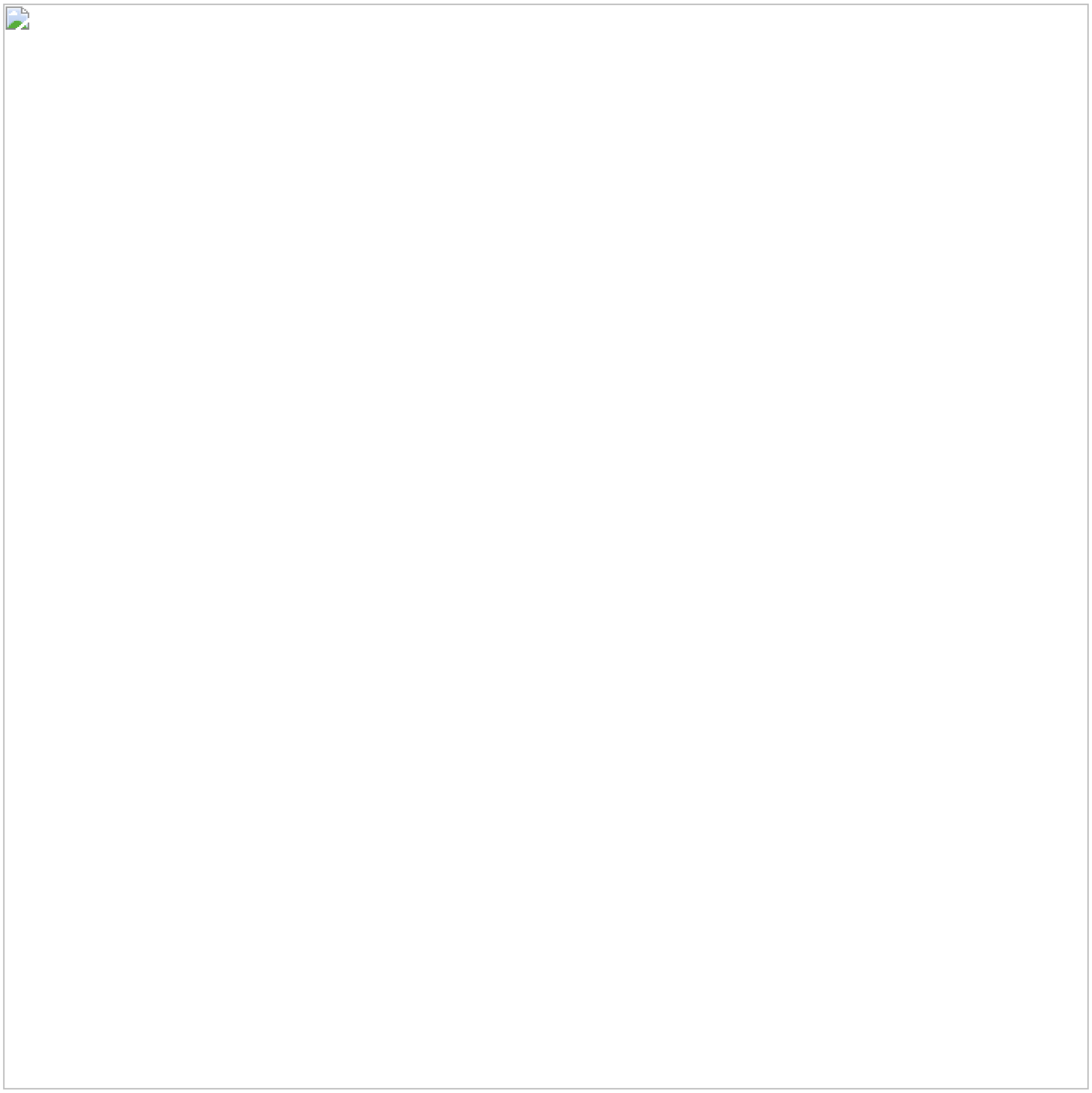
(Image credited to Sygnia)

Without SAML, our system administrators would have to create countless new accounts for each service we all need to do our jobs. That means we'd have to remember multiple usernames and passwords for each service (because, you never reuse passwords, right?) and also have to ensure they are periodically updated. It would be an administrative nightmare for system admins and users alike. Thankfully, technology has saved the day and allowed us all to rest easy at night.

A Brief History of Golden SAML

Well, that was nice while it lasted. As with any technology, there are weaknesses that our adversaries will work very hard to find and exploit. Way back in 2017, CyberArk published a blog post detailing a new attack technique called Golden SAML. In this blog post, they detailed a technique that allows an adversary to generate their own SAML response with the content and authorizations they deem necessary. Now, there are a few gotchas that the adversary needs to figure out first. Namely, they need access to the certificates used to sign the SAML objects. This generally means they need a foothold into the network and privileged access to extract the certificates. Multiple tools are available that will help the adversary extract the needed certificates to include certutil.exe, PowerShell, ADFSDump, and the ever-popular Mimikatz (don't worry, we detail how to detect this in a bit). All the adversary needs to do is run their preferred tool to extract the certificates and they are well on their way to Golden SAML bliss.

Sygnia has another great visual representation of what a Golden SAML attack would look like.



(Image credited to Sygnia)

Why is this such a big deal? Now that the adversary has access to the extracted certificates, they can impersonate just about any user and privilege in the organization. Not only that, but they can do it from anywhere in the world. Additionally, the adversary will be able to bypass Multi-Factor Authentication (MFA) protections. Why, you ask? Because the actual authentication server is being removed from the process entirely. The adversary has forged a valid response from the authentication server, completely bypassing MFA.

Golden SAML and the SolarWinds Cyberattack

You may be wondering what any of this has to do with the SolarWinds attack. Well, the SolarWinds Orion compromise resulted in the first recorded use of Golden SAML in the wild. Nearly three years after it was first documented as a viable post-exploitation technique, this was the first time anyone had detected it. That either speaks to the effectiveness, flexibility, and stealthiness of this technique, or the determination and tradecraft of this actor. Probably both.

What Data Do I Need?

In our examples, we will be focusing on a Windows environment that leverages Active Directory Federation Services (ADFS) for SAML. ADFS security and audit logs will help provide additional insight into potential Golden SAML attacks. More on that in a bit though. Additionally, some queries may need to be customized to your environment, depending on the Splunk TA's that are installed and configured. One thing to keep in mind, just because our examples are limited to a Windows environment and ADFS, doesn't mean this attack vector is. Golden SAML attacks are possible with just about any SAML provider.

But first, a few caveats. This is not a simple endeavor. Detecting this activity can be extremely difficult and require data from multiple sources across an enclave, both internal and external. We will work towards providing examples that can help you detect this type of attack.

Now that we have that out of the way, let's go over some Windows Events that we'll need to find this activity.

Windows Event Codes

In order to detect SAML attacks against Microsoft infrastructure, you will need (not surprisingly) Microsoft logs. Now we are going to post some links/ideas below on how to enable and collect the events you need to detect Golden SAML excitement. Think of them as "Splunkspiration". It may not work for you out of the box, but it should get you along the right path. Depending on the detection methodology, there are several Windows Events that need to be collected for detection to be feasible. Keep in mind that you'll probably want some adult supervision (SysAdmin) around before making any system changes. YMMV and there are some requirements (like Windows 2016 or later and sysmon installed) to actually be able to get them!

Oh, and since this is a Splunk blog, we'll assume you're using the Splunk Universal Forwarder to ingest Windows events into Splunk.

ADFS Event IDs

These are perhaps the most important Windows Events to collect, and unfortunately, several are not enabled by default. You can find additional information on enabling these events from this blog post and PowerShell Gallery, and documentation for them here and here.

- 1200 (AD FS-Admin): The Federation Service validated a new credential
- 1202 (AD FS-Admin): The Federation Service issued a valid token
- 307 (AD FS-Admin): The Federation Service configuration was changed
- 510 (AD FS-Admin): Additional information
- 1007 (Microsoft-Windows-CertificateServicesClient-Lifecycle-System): Certificate Exported

Now, we don’t by default collect the event logs that these show up in, so we need to add them to a Splunk TA configuration — for example, you could add them to the inputs.conf in the Splunk Add-on for Windows. First, add a stanza to enable the proper Certificate Services log (this one only brings in 1007 — remove the whitelist directive if you want the other Certificate Services goodness):

```
[WinEventLog://CertificateServicesClient-Lifecycle-System/Operational]
disabled = 0
start_from = oldest
current_only = 0
checkpointInterval = 5
renderXml=true
whitelist = $XmlRegex=’(?:1007).+’
```

And just in case you thought that was the only tweak, think again. Active Directory Federation Services has its own log for events. So we need to add another stanza to inputs.conf (on our forwarder):

```
[WinEventLog://AD FS/Admin]
disabled = 0
start_from = oldest
current_only = 0
checkpointInterval = 5
renderXml=true
```

Both of these inputs.conf examples were tested on a Windows 2019 Server.

Domain Controller Event IDs

To gather the needed events from the Domain Controller(s), we will have to enable them via the Local Security Policy snap-in, or via Group Policy. More information on the settings can be found in Microsoft’s documentation.

> 4769 (Security): A Kerberos service ticket was requested

Endpoint Event IDs

All of the below Event IDs will help identify command line execution or powershell for certification extraction. Need help getting the PowerShell events into Splunk? We’ve got you covered there (the documentation is from UBA, but don’t worry it works for any logging of PowerShell.)

- 4688 (Security) (and make sure you enable process arguments.)
- 4103 (PowerShell) (slide 79ish, here)
- 4104 (PowerShell) (see above)
- 18 (Sysmon)

How Can Golden SAML Be Detected?

I’m glad you asked! We have a few opportunities to detect related activity. Let’s take a look at how we can find some in Splunk using the methodology that Sygnia discussed in a recent advisory. Also, keep in mind these are example queries to help you along. Detecting this activity in an enterprise can be challenging, so YMMV with the following queries and they will probably need some massaging to get working correctly in your environment. Note that they all assume you are using XML-formatted Windows logs (but can be adjusted for Classic format.)

1. Search for certificate exports in ADFS event logs
In order for this attack to be successful, the adversary must export the appropriate certificates from an ADFS server. This should be fairly rare as there are few reasons to export certificates legitimately. We can search for this activity in the Windows Event Logs for the ADFS server for the Event ID 1007.

```
sourcetype=xmlwineventlog
source="xmlwineventlog:CertificateServicesClient-Lifecycle-System/Operational"
EventCode="1007"
```

Additionally, we can search for indications that ADFS certificates were exported from the command-line or PowerShell. These commands can be executed from anywhere, to include servers or clients. So be sure your queries aren’t limited to just the ADFS server.

First, let’s look for certificate exports via PowerShell

```
index=main sourcetype=xmlwineventlog
source="xmlwineventlog:Microsoft-Windows-PowerShell/Operational"
EventCode IN (4104, 4103) ScriptBlockText IN
("*Export-PfxCertificate*", "*certutil -exportPFX*" )
```

We can also look for any indication by the use of certutil.exe

```
index=main sourcetype=xmlwineventlog
source="xmlwineventlog:Security" EventCode=4688
CommandLine="*certutil.exe -exportPFX*"
```

Last, let’s look for the methods used by Mimikatz and ADFSDump. Please note that this pipe is popular, so we will want to see which process (images) are not commonly used and dig into them further.

```
index=main sourcetype="xmlwineventlog:microsoft-windows-sysmon/operational"
EventCode=18 PipeName="\microsoft##wid\tsql\query"
| stats count by Image
| sort count
```

2. Identify abnormal authentication events
Note: This technique will only work on Windows Server 2016 or greater. Event IDs 1200 and 1202 did not exist in prior versions of Windows.

Valid SAML authentication events will generate multiple security events on the ADFS and DC servers, as well as the service we are authenticating to. We will look for all three of the following Event IDs on our ADFS and DC in the Windows Security Events logs.

- 1200 - The Federation Service issued a valid token
- 1202 - The Federation Service validated a new credential
- 4769 - A Kerberos service ticket was requested

We will then look to correlate these events with authentication events from the service the user attempted to authenticate to (the service provider). In this use case, we will check for authentication attempts to Azure AD Sign-in logs.

Golden SAML attacks will lack ADFS (and most likely DC) authentication logs, since the adversary is conveniently forging the SAML response, effectively removing them from the authentication process. We should however still see a “successful” authentication from the service provider since they did receive a signed SAML response, even though it was forged by the adversary.

```
sourcetype=xmlwineventlog source="xmlwineventlog:Security"
EventCode IN (1200, 1202, 4769)
`comment("Look for these three event codes in Windows Security Event Logs")`

   [ search sourcetype=ms:aad:signin status.errorCode=0

   | eval Account_Name=mvindex(split(userPrincipalName,"@"),0)
`comment("Drop the @domain part from the userPrincipalName")`

   | fields Account_Name ]
`comment("Build a list of account names from log in attempts through Azure AD to correlate
against Windows Security Event Logs")`

| eval Account_Name=mvindex(Account_Name,-1)
`comment("Only use the last Account_Name value from the Windows Event Log, which should filter
out situations where Account_Name is - or a host value")`

| eval Account_Name=mvindex(split(Account_Name,"@"),0)
`comment("Drop the @domain part from the Account_Name")`

| transaction dest Account_Name maxspan=2s maxevents=3
`comment("Create a single event with the same dest and Account_Name that occur within 2
seconds and limited to three events")`

| eval mcount=mvcount(EventCode)
`comment("Count the number of unique event codes for each account name")`

| where mcount<3
`comment("Look for events where there is an absence of all three event codes")`

| table _time Account_Name EventCode
```

3. Monitor for ADFS Trust Modifications
Rather than extract the required certificates from a trusted ADFS server, the adversary may prefer to add a new trusted ADFS server that they control. We will look for Event ID 307 (The Federation Service configuration was changed) and correlate with Event ID 510 with the same instance id. Configuration changes for federations don’t frequently occur in many organizations so use this as a starting point and adjust as you see fit.

```
sourcetype=xmlwineventlog source="xmlwineventlog:AD FS/Admin"
EventCode IN (307, 510)
```

Mitigation Recommendations

As with any technology, following best practice guides and recommendations are always the best path forward. If you are using Active Directory Federated Services (ADFS), Microsoft provides an excellent resource for securing it. Considering how critical ADFS is to many organizations, implementing as many security measures as possible is always highly recommended, perhaps none more so than leveraging a Hardware Security Module (HSM) for generating and storing certificates. HSM has the added benefit of securely storing your keys, as well as all cryptographic functions, on a physical device. Why is this so important? It negates the ability for an adversary to extract that ever-important private key, effectively mitigating (or at least raising the bar and cost of) the adversary's ability to conduct a Golden SAML attack.

Conclusion

Today we went over what SAML is, how a Golden SAML attack works, and some example detection methodologies as well as pertinent data sources. We also briefly went over high-level recommendations to help mitigate this type of attack in the future. All of this content is meant to help you forge a path forward and improve your overall security posture. This won't be the last time we see an adversary leveraging Golden SAML to ensure persistence in a victims network. If anything, the prominence of this attack has almost certainly highlighted the effectiveness of this technique which may result in more adversaries bringing it into their toolbox.

For more information on this and other content related to Solarwinds, check out our SolarWinds Cyberattack response site.

I'd also like to make a special mention and thank John Stoner, Lily Lee, James Brodsky, and Ryan Kovar here at Splunk. They were instrumental in pulling this blog post together, creating the queries, and ensuring logic prevailed.

Posted by

Marcus LaFerrera

US | JP | Pentagon | DARPA | Splunk