

# Microsoft 365 Security

Everything about Microsoft Security

MENU



## What I Have Learned From Doing A Year Of Cloud Forensics In Azure AD

Posted on [July 13, 2021](#) | by [m365guy](#) | [One comment](#)

Today I would like to share my experience with doing Cloud forensics in Azure AD. I've been working for over a year with Azure Active Directory, and have primarily focused on the different security aspects of it. One of my main focus has been doing Cloud forensics, which I will tell more about. I was always interested in understanding, where logs are stored and what kind of information it contains.

During this blog post, I will share some of my experience. This includes the challenges that I've faced, but also the things I have learned. Last, but not least. I will share my methodology on doing Cloud forensics in Azure AD. An important note, but to clarify it first. This does not mean that it should be done this way. It's how I have done it, so I understand that there's always room for improvements.

I have blogged about a similar topic, which can be found [here](#). However, I would like to keep this blog post more detailed and share some in-depth examples. Which includes the challenges as well when doing Cloud forensics.

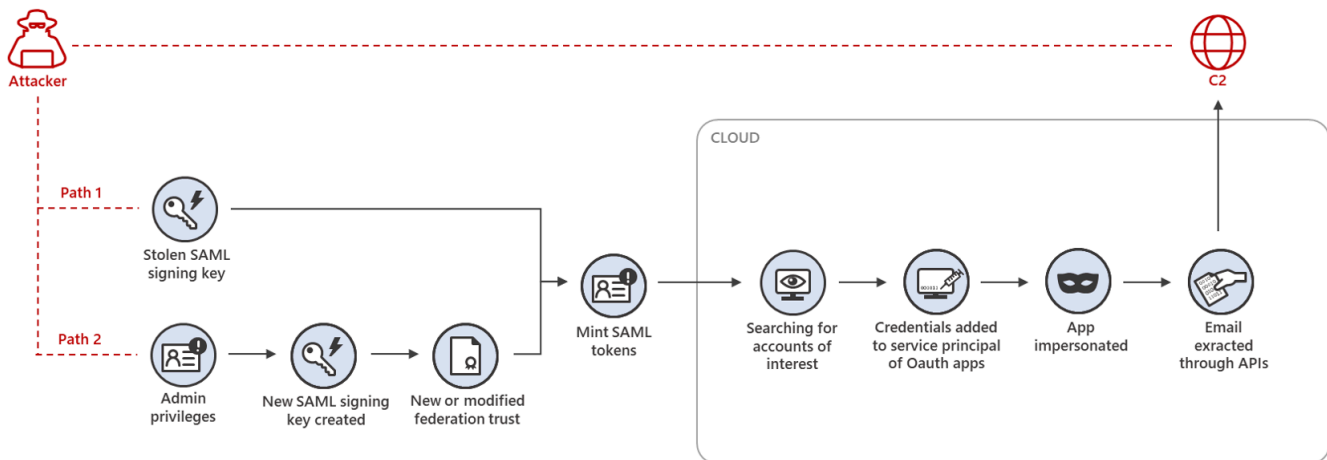
## What are the challenges?

A lot of organizations don't collect the relevant logs in the first place, so monitoring on certain activities won't be possible at all. Being pro-active is the recommended approach, so if you don't log and monitor anything at all. Your organization will have blind spots, and that will make it incredible hard for both security & IR teams.

Let's give an example about a real-case. During the SolarWinds Post-Compromise breach, we have seen adversaries pivot around in Azure AD and were using the following techniques:

### SOLORIGATE ATTACK

Stage 3: Hands-on-keyboard attack in the cloud



They were adding credentials to an existing application and later on were using the Service Principal to access emails of users through the available APIs.

An organization could have visibility in all of this, if they were collecting and monitoring the right logs, which can be done with something like **Log Analytics**.

**Log Analytics** allows us to ingest data, so we could query all the generated logs and write custom detection rules for it, based on the configured data telemetry via

## diagnostic settings.

Diagnostic setting name	RelevantLogs
Category details	Destination details
<div>log</div> <div><div><input checked="" type="checkbox"/> AuditLogs</div><div><input checked="" type="checkbox"/> SignInLogs</div><div><div><div><div></div><div></div></div><div>In order to export Sign-in data, your organization needs Azure AD P1 or P2 license. If you don't have a P1 or P2, <a href="#">start a free trial</a>.</div></div></div><div><input checked="" type="checkbox"/> NonInteractiveUserSignInLogs</div><div><input checked="" type="checkbox"/> ServicePrincipalSignInLogs</div><div><input checked="" type="checkbox"/> ManagedIdentitySignInLogs</div></div>	<div><input checked="" type="checkbox"/> Send to Log Analytics workspace</div> <div>Subscription</div> <div></div> <div>Log Analytics workspace</div> <div>LogAnalytics ( centralus )</div> <div><input type="checkbox"/> Archive to a storage account</div> <div><input type="checkbox"/> Stream to an event hub</div> <div><input type="checkbox"/> Send to partner solution</div>

Another example is that organizations forget to turn on **Unified audit logs**. **Unified audit logs** contains user, group, application, domain, and directory activities performed in the Microsoft 365 admin center or in the Azure management portal. This is an very important setting, which needs to be turned on. Leaving this disabled will give the IR team a very tough challenge.

## Audit log search

! To use this feature, turn on auditing so we can start recording user and admin activity in your organization. When you turn this on, activity will be recorded to the Office 365 audit log and available to view in a report.

Turn on auditing

### Search

Clear

Activities

Show results for all activities

Start date

2019-09-23



00:00

### Results

Date

IP address

User

Activity

Item

Last, but not least. Organizations with the appropriate license for Microsoft Cloud App Security (MCAS) may not have connected the Office 365 App to MCAS or perhaps they did, but forgot to mark the different check boxes. This will leave some

blind spots in the O365 activities, which means that you wouldn't be able to find out which user was added to an O365 Admin Role or who created an new inbox-rule in Exchange Online for instance.

## Connect Office 365

Before you connect Office 365, we highly recommend reviewing the [Office 365 connection guide](#). Follow these steps in order to connect Office 365.

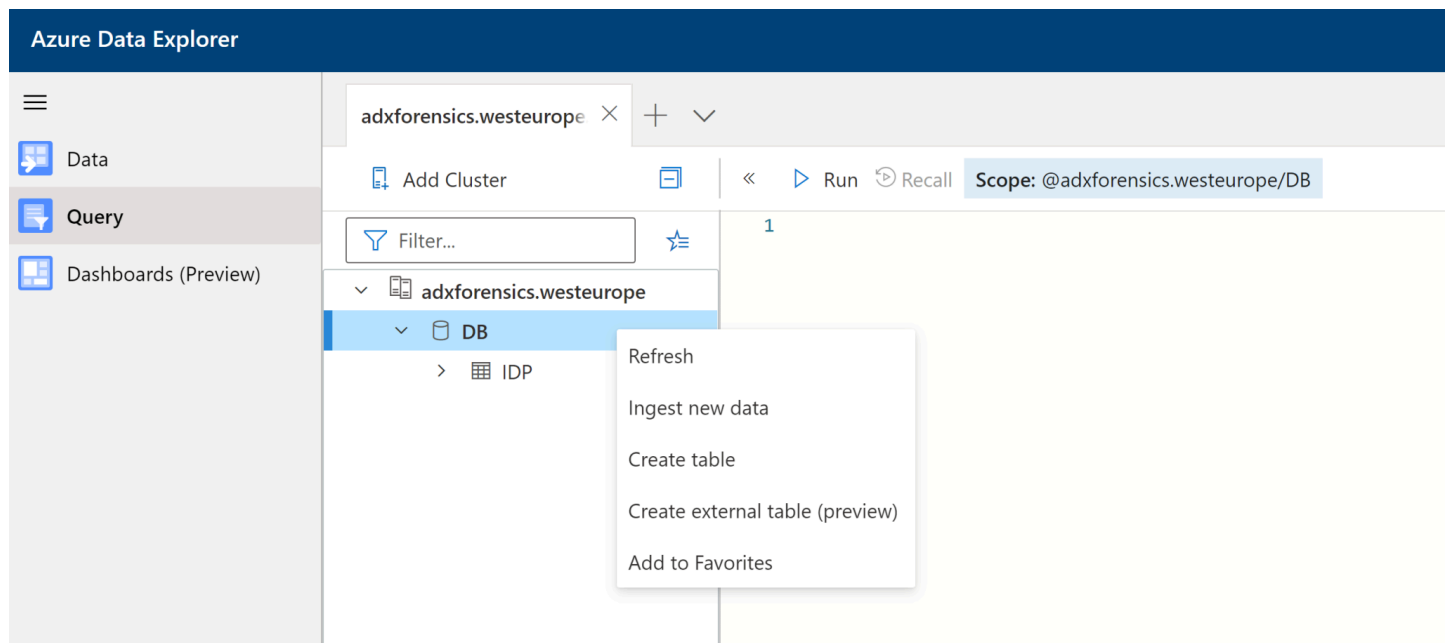
## 1 Select Office 365 components

- ☒ Azure AD Users and groups ⓘ
  - ☒ Azure AD Management events ⓘ
  - ☒ Azure AD Sign-in events ⓘ
  - ☒ Azure AD Apps ⓘ
- 
- ☒ Office 365 activities ⓘ
  - ☐ Office 365 files ⓘ
- Enable file monitoring before enabling Office 365 files.

## Setup Azure Data Explorer

Azure Data Explorer is an analytic engine that is similar to Log Analytics. ADX is preferred when it comes down to analyzing large amount of data, so the first thing we have to do is setup an ADX cluster and create a new database.

Once we have created a database in ADX, we can start ingesting data. This will primary be an export of Unified Audit Logs, MCAS, Azure AD Identity Protection, etc. The reason that we are ingesting the data into ADX is, because it allows us to use Kusto to query the data.



## What is next?

The next step is going to explore logs at different places in a tenant and export the data, which will be CSV files. Once that's finished, we will ingest all of it in Azure Data Explorer. We will show examples on querying the data to discover attacker activities.

## Azure Identity Protection

Azure AD Identity Protection include any identified suspicious actions related to user accounts in the directory. Identity Protection generates risk detection's only when the correct credentials are used. This is a good place to start, because it may give us an starting point of an identity being compromised.

We are going to export all the users that are marked as risky, and will ingest it into ADX. Here is how the exported result looks like.

Before making any assumptions. It is good to be aware of how an organization has setup their Azure Identity Protection. We should for sure look at the **user** and **sign-in** risk remediation policy, as well as the **MFA** registration policy, and the **Trusted locations** in Conditional Access.

An very important thing to look at first is the **Sign-in** risk remediation policy. We can see in this example, that this policy has been enforced on all the users in the tenant. It is supposed to block access once a **Sign-In** risk detection was marked as '**High**' risk. This means that accounts with an '**Medium**' risk for example shouldn't get blocked, so it all depends on how Azure Identity Protection has been configured.

Another thing that has been overlooked a lot is the **Trusted Locations** in Conditional Access. Admins can designate locations defined by IP address ranges to be trusted named locations. Sign-ins from trusted named locations improve the accuracy of Azure AD Identity Protection's risk calculation, lowering a user's sign-in risk when they authenticate from a location marked as trusted.

Here is an example of a trusted location that someone can configure. Where you can exclude a certain country from getting blocked by Azure Identity Protection. This can be done for a specific IP range as well.

Since we now have a better understanding of Azure Identity Protection. We can be more efficient in our investigation. First, we will start with looking at both **‘Medium’** and **‘Risk’** detection’s. Yes, despite that accounts with a **‘High’** risk are getting blocked. We should include them as well in our query, because despite that they were blocked. We should consider investigating those accounts later, since they may be under attack.

## Query

```
IDP
| where ['Risk level'] == 'Medium'
      or ['Risk level'] == 'High'
| where Source == 'IdentityProtection'
| project DetectionType = ['Detection type'], RiskState = ['Risk state'],
RiskLevel = ['Risk level'], RiskDetail = ['Risk detail'], Source, UPN,
IPAddress = ['IP address'], Location
| sort by RiskLevel asc
```



## Result

At the sample result, we can see the risky users that have been detected by Identity Protection. We see a couple of High risks and a few Mediums. As we mentioned earlier. Users with a 'High' risks will get blocked, so those accounts can perhaps be investigated later.

All the 'Medium' users won't get blocked by the **Sign-In** risk remediation policy, because it was configured to only block the 'High' risks users.

As we may have notice, there is one account that has been highlighted, which has been marked in yellow. The account is still 'at risk' and we can't find any information in the 'RiskDetail' column. We know that the account has NOT been blocked either, because it's a 'Medium' risk. Why is the 'RiskDetail' column empty?

Well, the user that signed in from an 'Anonymous IP address' is based in a specific country, which has been marked as trusted in the **named locations**. Yes, you may encounter such cases. Which is why the 'RiskDetail' column is empty.

## Lessons learned

Hunting in the data of Azure Identity Protection can be useful, but there are things that we need to be aware of. Every organization may configure their Azure Identity Protection different than others. Be aware of how the **User** and **Sign-In** Risk

Remediation Policy, as well as the **MFA** registration policy and the **trusted locations** have been configured.

In overall, Azure Identity Protection does a pretty solid job in blocking access (if configured). The important part is to ensure that once Identity Protection has fired an alert. We need to be sure that MFA is enabled on the account. If we can't confirm that MFA is enabled on the account, that's where further investigation needs to happen.

## Unified Audit Logs

Unified Audit Logs contains Azure AD & O365 activities. This is an important feature to enable, because this allows you to record user & admin activities. Unified Audit Logs can be exported through the **Search-UnifiedAuditLog** cmdlet, as well as the **Security & Compliance center**. You can export up to 50k entries of records in Unified Audit Logs, and it will generate a CSV file, which we will ingest in Azure Data Explorer.

I've spent a lot of time digging into the data of Unified Audit Logs. There are challenges for Incident Responders, which I want to highlight. First, we can see that the 'AuditData' contains a lot of unstructured data. It is stored in a JSON format. For those that are familiar with Kusto. You will realize that it becomes a bit complex to "parse" the different fields, due to how the properties are stored in the first place. This doesn't mean that it's impossible of course.

Here is a closer look at how the properties are stored in the 'AuditData' column. Good luck with parsing all the different fields into columns!

The second thing is that Microsoft has documented a list of different operations. All of these operations contains the activities for the users and admins. Here we can see a few examples:

**Source:**

<https://docs.microsoft.com/en-us/microsoft-365/compliance/search-the-audit-log-in-security-and-compliance?view=o365-worldwide#application-administration-activities>

There is a list of operations, but the list of Microsoft is not complete. Unfortunately, there are different operations that are pretty useful to look at, but aren't documented. We will cover some examples more on this later.

Last thing we have to be aware of is, that Unified Audit Logs may have a delay up to 24 hours or longer. This means that once an admin adds a new owner to an application for example. You won't immediately see it in the logs.

## Hunting in Unified Audit Logs

This section will cover in-depth examples of different techniques that adversaries have been using in the wild, to attack organization's their Azure AD tenant. An great example was the SolarWinds Post-Compromise breach. Microsoft did a great job in sharing hunting queries, which could be used in Azure Sentinel to detect different TTPs. The blog post can be found [here](#). However, the thing is. There are a lot of organizations that don't use Azure Sentinel in the first place.

During this section, we will cover how we could hunt for the described techniques, while relying on Unified Audit Logs. I've found a nice [blog post](#) from Microsoft that described all the techniques in a simple way, so I will use it as a reference.

The first thing we have to do is export the relevant Unified Audit Logs and ingest it in Azure Data Explorer. Once we have done that, we can start hunting in our data.

- **Modified application and service principal credentials/authentication methods**

*“One of the most common ways for attackers to gain persistence in the environment is by adding new credentials to existing applications and service principals. This allows the attacker to authenticate as the target application or service principal, granting them access to all resources to which it has permissions.”*

In practice, this would look similar like this. Where someone is adding new credentials to an existing application. As we can see here:

## Query

```
UAL2
| where Operations has ("Certificates and secrets management")
| extend ParsedFields = parse_json(AuditData)
| evaluate bag_unpack(ParsedFields)
```

```
| extend value_set = tostring(ModifiedProperties[0].NewValue)
| extend AppName = tostring(Target[3].ID)
| extend new_keyIdentifier = tostring(split(value_set, ",")[4])
| extend old_displayName = tostring(split(value_set, ",")[3])
| extend new_keyDisplayName = tostring(split(value_set, ",")[7])
| extend new_keyDisplayName = tostring(split(new_keyDisplayName,
"DisplayName=")[1])
| extend new_keyDisplayName = tostring(split(new_keyDisplayName, " ")[0])
| extend new_keyIdentifier = tostring(split(new_keyIdentifier,
"KeyIdentifier=")[1])
| parse value_set with * "KeyIdentifier=" keyIdentifier:string ",KeyType="
keyType:string ",KeyUsage=" keyUsage:string ",DisplayName="
keyDisplayName:string "]" *
| where isnotempty(new_keyDisplayName)
| project CreationDate, UserIds, Operations, AppName, keyDisplayName,
new_keyDisplayName, keyIdentifier, new_keyIdentifier, keyType, keyUsage
```

## Result

At the sample result, we can see that a new credential has been added to the 'TestApp' application. The two columns 'new\_KeyDisplayName' and 'new\_KeyIdentifier' indicates, that it is actually the case. We can also see which user principal performed this action, but I have excluded it from the result out of privacy reasons.

## Hunting tip

When we want to find out, whether a (new) secret or certificate has been added to an application. We need to filter on the following operation:

Operation	Description
Update application – Certificates and secrets management	An new client secret or certificate has been added to an application

**Tip:** Look when a new credential has been added to an existing application that contains one of the following sensitive permissions:

1. *Mail.\* (including Mail.Send\*, but not Mail.ReadBasic\*)*
2. *Contacts.\**
3. *MailboxSettings.\**
4. *People.\**
5. *Files.\**
6. *Notes.\**
7. *Directory.AccessAsUser.All*
8. *User\_Impersonation*

Delegated and AppOnly versions of the following permissions:

- *Application.ReadWrite.All*
- *Directory.ReadWrite.All*
- *Domain.ReadWrite.All\**
- *EduRoster.ReadWrite.All\**
- *Group.ReadWrite.All*
- *Member.Read.Hidden\**
- *RoleManagement.ReadWrite.Directory*
- *User.ReadWrite.All\**
- *User.ManageCreds.All*
- All other AppOnly permissions that allow write access

Adding an existing credential to a new application is an normal operation, but it does not happen often though. An credential may have been expired, so an admin can add a new credential to an existing application. This is something to keep in mind, when doing an investigation.

- **New permissions granted to service principals**

*"In cases where the attacker cannot find a service principal or an application with a high privilege set of permissions through which to gain access, they will often attempt to add the permissions to another service principal or app."*

API permissions may look like the following:

We can see in this example that we have granted an application to read all the mails from every mailbox.

## Query

```
UAL2
| where Operations == "Add delegated permission grant."
| extend ParsedFields = parse_json(AuditData)
| evaluate bag_unpack(ParsedFields)
| extend AppPermissions = tostring(ModifiedProperties[0].NewValue)
| where AppPermissions has_any ("Mail.Read","Mail.ReadWrite","Mail.Send")
```



```
| where ResultStatus == "Success"
| join kind=leftouter(
    UAL2
    | where Operations == "Consent to application."
    | extend ParsedFields = parse_json(AuditData)
    | evaluate bag_unpack(ParsedFields)
    | extend IsAdminConsent = tostring(ModifiedProperties[0].NewValue)
    | extend Permissions = tostring(ModifiedProperties[4].NewValue)
    | where Permissions has "Mail.Read"
        or Permissions has "Mail.ReadWrite"
        or Permissions has "Mail.Send"
    | extend AppName = tostring(Target[3].ID)
    ) on ActorContextId
| project CreationDate, UserIds, OperationName = Operations1, AppName,
AppPermissions, IsAdminConsent
| sort by AppName desc
```

## Result

At the sample result, we can see that a user has added an sensitive permission to an application and later consented it.

## Hunting tip

When we want to find out when someone has added permissions to an application and has consented it. We need to look at the following operations:

Operation	Description
-----------	-------------

Add delegated permission grant	API permissions have been delegated to an application
Consent to application	Consent is the process of a user granting authorization to an application to access protected resources on their behalf

**Tip:** Look when one of the following API permission have been granted to an application

1. *Mail.\* (including Mail.Send\*, but not Mail.ReadBasic\*)*
2. *Contacts.\**
3. *MailboxSettings.\**
4. *People.\**
5. *Files.\**
6. *Notes.\**
7. *Directory.AccessAsUser.All*
8. *User\_Impersonation*

Delegated and AppOnly versions of the following permissions:

- *Application.ReadWrite.All*
- *Directory.ReadWrite.All*
- *Domain.ReadWrite.All\**
- *EduRoster.ReadWrite.All\**
- *Group.ReadWrite.All*
- *Member.Read.Hidden\**
- *RoleManagement.ReadWrite.Directory*

- *User.ReadWrite.All\**
  - *User.ManageCreds.All*
  - All other AppOnly permissions that allow write access
- 
- **Directory role and group membership updates for service principals**

*"Following the logic of the attacker adding new permissions to existing service principals and applications, another approach is adding them to existing directory roles or groups."*

## Query

```
UAL5
| where Operations == 'Add member to role.'
| extend ParsedFields = parse_json(AuditData)
| evaluate bag_unpack(ParsedFields)
| extend Role = tostring(ModifiedProperties[1].NewValue)
| extend UserType = tostring(Target[0].ID)
| extend UserType = tostring(split(UserType, "_")[0])
| extend Principal = tostring(Target[3].ID)
| where UserType == 'ServicePrincipal'
| project CreationDate, UserIds, Operations, Principal, Role, UserType
```

## Result

At the sample result, we can see that a Service Principal has being added to two directory roles.

## O365 Activities

We have covered the techniques that were used during the Solarigate attack. However, as we may know. There are other things that we should look after, which might not have been used during the Solarigate attack.








A while ago, Microsoft has shared an article about a real world experience. Where they have encountered six different threat groups having access to a corporate network. This article can be read [here](#).

While reading the attached PDF, they had. I've noticed something that was worth to mention.

- **Using Content Search to obtain data from mailboxes and more**

I've found out that the Content search **eDiscovery** tool in the Microsoft 365 compliance center allows you to search for in-place content such as email, documents, and instant messaging conversations in your organization.

This feature looks like the following:

Locations			
Status	Location	Included	Excluded
<input checked="" type="checkbox"/> On	 Exchange mailboxes 	All <a href="#">Choose users, groups, or teams</a>	None
<input checked="" type="checkbox"/> On	 SharePoint sites 	All <a href="#">Choose sites</a>	None
<input checked="" type="checkbox"/> On	 Exchange public folders 	All	None
<input checked="" type="checkbox"/> Add App Content for On-Premises Users. <a href="#">Learn more</a> 			

## Query

```
UAL2
| where Operations == "SearchStarted"
| extend ParsedFields = parse_json(AuditData)
| evaluate bag_unpack(ParsedFields)
| project CreationDate, UserIds, Operations, CreationTime, ExchangeLocations,
SharepointLocations, PublicFolderLocations, ObjectId, Query
```

## Result

At the sample result, we can see that a user has made two content searches. Since the 'ExchangeLocations' column is not empty, we know that the user was searching in all the mailboxes. The 'Query' column contains the keyword that was filled in the search. An great pivot point is to look at if the query was targeted on a specific mailbox for instance.

## Hunting tip

When we want to find out when someone has started a Content Search. We have to filter on the following operation:

Operation	Description
SearchStarted	An Content Search has been started

- **Using eDiscovery Case to obtain data**

eDiscovery Case is very similar to Content Search. However, there is a slight difference. eDiscovery allows you to put content *on hold*, as well as scope searches to content. eDiscovery case has it's own security boundary, meaning that only a user that has been granted access to a case, is able to see it.

In this example, we are going to find out if someone has created an eDiscovery Case.

## Query

```
UAL6
| where Operations == "CaseAdded"
| extend ParsedFields = parse_json(AuditData)
```

```
| evaluate bag_unpack(ParsedFields)
| join kind=leftouter(
    UAL6
    | where Operations == "New-ComplianceSearch"
    | extend ParsedFields = parse_json(AuditData)
    | evaluate bag_unpack(ParsedFields)
  ) on UserIds
| project CreationDate, UserIds, Operations, Case, SearchKeywords =
Parameters1, Workload
| sort by CreationDate desc
```

## Result

At the sample result, we can see that a user has made three different Compliance Searches with the eDiscovery Case feature.

## Hunting tip

When we want to find out when someone has an eDiscovery Case. We have to filter on the following operations:

Operation	Description
CaseAdded	An new eDiscovery Case was added
New-ComplianceSearch	An compliance search was started to obtain data across different sources (e.g. Exchange, SharePoint, etc)

- **Content Search was exported**

It is possible to export the results of a Content Search that was being created. When speaking from my own experience. This is a very rare activity, which I haven't encountered yet. Exporting the results of a Content Search may contain a lot of sensitive information.

## Query

```
UAL6
| where Operations == "SearchExportDownloaded"
| extend ParsedFields = parse_json(AuditData)
| evaluate bag_unpack(ParsedFields)
| project CreationDate, UserIds, Operations, ObjectId, Query,
ExchangeLocations, PublicFolderLocations, SharepointLocations, Workload
```

## Result

At the sample result, we can see that a Content Search export has been downloaded. This includes information on the keyword that was used during the search, and what



mailboxes were included during the search.

## Hunting tip

When we want to find out when someone has downloaded an Content Search. We have to filter on the following operation:

Operation	Description
SearchExportDownloaded	An Content Search has been downloaded

- **Compliance Search of eDiscovery Case has been downloaded**

Once an eDiscovery Case has been created. We are able to create a new search, which we have discussed before. All the compliance searches can be downloaded as well. I haven't experienced this operation a lot, so it's good to keep an eye on this one.

## Query

```
UAL6
| where Operations == "ViewedSearchExported"
| extend ParsedFields = parse_json(AuditData)
| evaluate bag_unpack(ParsedFields)
| where isnotempty(Case)
| project CreationDate, UserIds, Operations, Case, ObjectId, Workload, Query
```

## Result

At the sample result, we can see that an Compliance Search has been downloaded twice.

## Hunting tip

When we want to find out when someone has downloaded an Compliance Search from an eDiscovery Case. We have to filter on the following operation:

Operation	Description
ViewedSearchExported	An search was exported, which can be both from Content Search and eDiscovery Case

- **User was added to eDiscovery role**

Before a user can download Content Searches. It needs to have certain permissions first, before it is possible. Even when you have Global Admin rights. You can't download the searches, because you need to be at least a member of the 'eDiscovery Manager' role in the Security & Compliance center.

In this example, we are going to look when and which member was added to this role.

## Query

```
UAL6
| where Operations == "Set-RoleGroup"
| extend ParsedFields = parse_json(AuditData)
| evaluate bag_unpack(ParsedFields)
| extend Role = tostring(Parameters)
| extend Role = tostring(split(Role, "-DisplayName")[1])
| project CreationTime, Operations, Workload, Role, UserIds
```

## Result

At the sample result, we can see that two members have been added to this role.

An eDiscovery Administrator is a member of the eDiscovery Manager role group, and can perform the same content search and case management-related tasks that an eDiscovery Manager can perform. This is something that we should not overlooked!

## Query

```
UAL3
| where Operations == "CaseAdminUpdated"
| extend ParsedFields = parse_json(AuditData)
| evaluate bag_unpack(ParsedFields)
| extend Members = tostring(split(ExtendedProperties, "Value")[1])
| extend Members = tostring(split(ExtendedProperties, "Name")[1])
| extend Members = tostring(split(ExtendedProperties, "Value")[1])
| project CreationDate, UserIds, Operations, ObjectId, ObjectType, Members
```

## Result

At the sample result, we can see that this role has been updated. It doesn't show exactly that a new member was added, but it will display the entire list of members that are currently a member of the 'eDiscovery Administrator' role.

## Hunting tip

When we want to find out when someone has updated the eDiscovery Manager role. We have to filter on the following operation:

Operation	Description
CaseAdminUpdated	A new user has been added or removed from the eDiscovery Manager role

- **Mail Items accessed**

MailItemsAccessed is an audit event in Office 365 that records when email data is accessed by mail protocols and clients. This activity is only logged for users with an Office 365 or Microsoft 365 E5 license. Analyzing audit records for this activity is useful when investigating compromised email account.

**Source:**

<https://www.microsoft.com/security/blog/2020/12/28/using-microsoft-365-defender-to-coordinate-protection-against-solorigate/>

## Query

```
UAL
| where Operations == "MailItemsAccessed"
| extend ParsedFields = parse_json(AuditData)
| evaluate bag_unpack(ParsedFields)
| extend FolderItems = tostring(split(Folders, "Path")[1])
| extend FolderItems = tostring(split(FolderItems, "}}")[0])
```

```
| extend InternetMessageId = tostring(split(Folders, "InternetMessageId")[1])  
| extend InternetMessageId = tostring(split(InternetMessageId, ">")[0])  
| extend InternetMessageId = tostring(split(InternetMessageId, "<")[1])  
| project CreationDate, UserIds, Operations, AppId, InternetMessageId,  
ClientIPAddress, ExternalAccess, FolderItems, MailboxOwnerUPN
```

## Result

At the sample result, we can see two results. There are enterprise applications accessing email content from Exchange Online. Unfortunately we aren't able to see directly which specific mail(s) were read for example, but we can see what folder were accessed (e.g. Inbox, Junk, Spam, etc).

In order to find out what kind of mails have been accessed, we have to do extra investigation. This starts with looking at the 'InternetMessageId' value, because it contains the metadata of the mails that were accessed. However, keep it mind that it is very limited.

## Hunting tip

When we want to find out when someone has accessed email content from Exchange Online. We have to filter on the following operation:

Operation	Description
-----------	-------------

MailItemsAccessed	Messages were read or accessed in mailbox. Audit records for this activity are triggered in one of two ways: when a mail client (such as Outlook) performs a bind operation on messages or when mail protocols (such as Exchange ActiveSync or IMAP) sync items in a mail folder. This activity is only logged for users with an Office 365 or Microsoft 365 E5 license.
-------------------	--

- **Organization Management Role in EXO**

All the directory roles in Azure AD can be managed with Privileged Identity Management (PIM), so organizations can avoid granting permanent privileges. However, there are certain roles in Exchange Online that are often overlooked and not monitored well, and yes. These roles can't unfortunately be managed with PIM. I'm talking about the 'Organization Management' role in EXO.

All the members of 'Organization Management' have administrative access to the entire Exchange Online organization and can perform almost any task in Exchange Online. An interesting thing use-case would be to look if a user has been added to this role.

## Query

```
UAL6
| where Operations == "Update-RoleGroupMember"
| extend ParsedFields = parse_json(AuditData)
```

```
| evaluate bag_unpack(ParsedFields)
| where Workload == "Exchange"
| extend Members = tostring(split(Parameters, "Value")[1])
| where ObjectId == "Organization Management"
| project CreationDate, UserIds, Operations, Role = ObjectId, Members,
Workload
| sort by CreationDate desc
```

## Result

At the sample result, we can see that a user has been added to the 'Organization Management' role.

## Hunting tip

When we want to find out when someone has been added to a role in Exchange Online, Compliance Center, and so on. We have to filter on the following operation:

Operation	Description
Update-RoleGroupMember	An user was added or removed from an role in Exchange Online or Compliance Security.

- **File was accessed in SharePoint Online / OneDrive for Business**

This operation is interesting if we have determined which account was compromised. It will tell you what files were accessed in SharePoint / OneDrive by the compromised account.



## Query

```
UAL2
| where Operations == "FileAccessed"
| extend ParsedFields = parse_json(AuditData)
| evaluate bag_unpack(ParsedFields)
| project CreationDate, UserIds, Operations, ClientIP, EventSource, ObjectId,
SourceFileName
```

## Result

At the sample result, we can see two results.

## Hunting tip

When we want to find out what file has been accessed in SharePoint & OneDrive. We have to filter on the following operation:

Operation	Description
FileAccessed	An file was accessed in SharePoint / OneDrive

## Microsoft Cloud App Security (MCAS)

Microsoft Cloud App Security is a Cloud Access Security Broker (CASB) that operates on multiple clouds. It provides rich visibility, control over data travel, and

sophisticated analytics to identify and combat cyber threats across all different cloud services. The data that MCAS gets from different apps is saved according to retention policies for 180 days (for activities).

I have once discovered an interesting case. Where an organization had connected the Office 365 app to MCAS, which will look like this:

However, they forgot to turn on the relevant logging. This means that an organization would only receive Sign-In activities across different O365 services, such as Teams, EXO, SPO, and so on. All the other operations would not be logged.

MCAS allows you to export up to 5k records of different Cloud services. In this example, we have exported some logs from EXO, SPO, MCAS Admin activities, and so on. This section will show how the logs are looking, and see if it can be useful.

- **MCAS Admin activities**

All the admin activities around MCAS are logged, which includes when someone logs into the MCAS portal to disabling a policy, and so on.

Let's say that we were interested in finding out who disabled a policy in MCAS.

## Query

```
MCAS01
| where Category == "Disable policy"
| where App == "Microsoft Cloud App Security"
| project Date, Category, Description, App, User, IPAddress = ['IP address']
```

## Result

At the sample result, we can see that three MCAS rules have been disabled.

- **Exchange Online activities**

Exchange Online activities are logged in MCAS and after looking at the logs for the first time. I did find some value in it, because it helps me to discover if someone has downloaded a file for example. Other examples are activities such as, granting permissions on a mailbox. Looking in MCAS is way more easier than the Unified Audit Logs.

The first example will be querying to see if someone has downloaded a file in Exchange Online. This activity is generated, when a user has received an e-mail with an attachment. Once the attachment has been downloaded, the activity will be logged.

## Query

```
EXO2
| where Category == "Download file"
| where App == "Microsoft Exchange Online"
| project Date, Category, Description, App, Device, User, IP = ['IP address']
```

## Result

At the sample result, we can see different users downloading a file. This may be helpful to look who has downloaded a file that perhaps was malicious, etc.

The second example is to look when permissions have been granted to an mailbox.

## Query

```
EXO2
| where App == "Microsoft Exchange Online"
| where Category == "Add permission to mailbox"
| project Date, Category, Description, User, IPAddress = ['IP address']
| sort by Date desc
```

## Result

At the sample result, we can see who has granted permissions to mailboxes.

- **SharePoint & OneDrive activities**

MCAS logs different activities of SharePoint & OneDrive. This includes file being accessed to being modified, and so on. I would rather look at such logs in MCAS, instead of how it stored in Unified Audit Logs.

We are looking when a user has accessed a file in SharePoint / OneDrive.

## Query

```
SPO
| where App == "Microsoft OneDrive for Business"
      or App == "Microsoft SharePoint Online"
```

```
| where Category == "Access file"  
| project Date, Category, Description, User, IP = ['IP address']
```

## Result

At the sample result, we can see which user has accessed a file in OneDrive for example.

## Tips

A few important tips from my side. Once you export the Unified Audit Logs. Be aware that you can only export up to 50k records. An recommendation that I can give is, to filter on specific users in a certain time frame. You can have multiple Unified Audit Logs exports for example. I like to make different exports, so one export will filter on users A, B, and C, while the other export will filter on users X, Y, and Z, and so on.

Find what's best for you, but don't expect that one export of the UAL logs will be enough, especially when you have to deal with larger tenants. Once you ingest the CSV files in Azure Data Explorer. Start with giving the tables a name that makes sense to you.

Ok, so in this example. I didn't do that, but what you can see is that I have different tables, that contains different UAL logs. This allows me for example to correlate logs

from different tables with each other, but in overall. It is used to have a better overview.

## Overall learning experience

The overall experience of doing Cloud forensics in Azure AD has been fun. There are challenges, as we all know. Not every organization has setup everything correctly, which means that there's no proper auditing in place.

Diving in Unified Audit Logs has been a challenging from the beginning, because of how the schema of UAL has been made up. All the unstructured data, ugh! It took some time to find a way to parse the different fields and query the data properly. Great thing is that we can use Kusto with Azure Data Explorer to solve this challenge.

Microsoft Cloud App Security (MCAS) allows you to connect the O365 app, so you will receive different activity logs across different services with the likes of EXO, SPO, Teams, and so on. I did have overlooked this in the first place, but later on. I realized that it does provide some value, because it's much easier to find out who has accessed a file in SharePoint or OneDrive at the logs in MCAS.

Understanding the different techniques that are attackers are using to attack an Azure AD tenant is not something you'll see that much in the news. During the SolarWinds Post-Compromise, we have learned that attackers are not just compromising an On-Premises environment anymore. You may not see it often in the news, but organizations their Azure AD tenant do get breached as well. I'm expecting to see this more in the future, due to the rise of the Cloud. The majority of organizations are already in the Cloud, which the attackers have realized it as well. They go where the data is, and that's for sure not only On-Premises.

## Reference

- Search the audit log in the compliance center: <https://docs.microsoft.com/en-us/microsoft-365/compliance/search-the-audit-log-in-security-and-compliance?view=o365-worldwide>
- Using Microsoft 365 Defender to protect against Solorigate: <https://www.microsoft.com/security/blog/2020/12/28/using-microsoft-365-defender-to-coordinate-protection-against-solorigate/>
- Azure AD workbook to help you assess Solorigate risk: <https://techcommunity.microsoft.com/t5/azure-active-directory-identity/azure-ad-workbook-to-help-you-assess-solorigate-risk/ba-p/2010718>
- Incident response playbooks: <https://docs.microsoft.com/en-us/security/compass/incident-response-playbooks>
- What is Azure Data Explorer?: <https://docs.microsoft.com/en-us/azure/data-explorer/data-explorer-overview>
- What is Cloud App Security?: <https://docs.microsoft.com/en-us/cloud-app-security/what-is-cloud-app-security>
- AND THEN THERE WERE SIX: <https://mssecurity.wpengine.com/wp-content/uploads/2020/03/then-there-were-six.pdf>



---

Share this:



---

#### Related

Incident Response Series:  
Collecting and analyzing logs in  
azure ad  
March 8, 2021  
In "Azure Active Directory"

Exfiltrating data by transferring it  
to the cloud with Azcopy  
June 11, 2021  
In "Azure Active Directory"

Incident Response Series:  
Reviewing data in Azure AD  
for investigation  
March 16, 2021  
In "Azure Active Directory"

## ONE COMMENT

---

**Henning Rauch**

July 14, 2021 7:15 pm

Great work!

★ Like

Reply

## LEAVE A COMMENT

---

---

Search ...



BUY ME A COFFEE



POPULAR THIS WEEK

- Investigating Certificate Template Enrollment Attacks - (ADCS)
- Hunting in On-Premises Exchange Server logs
- Everything about Service Principals, Applications, and API Permissions
- How to roll out Microsoft LAPS via GPO and why you should do it?
- Using Microsoft Defender for Endpoint during investigation

CATEGORIES

- M365 Advanced Hunting
- Azure Sentinel
- Azure Active Directory
- KQL
- Microsoft Identity
- Windows OS

- [Jupyter Notebooks](#)

## CONTACT



## RECENT POSTS

- [Investigating Certificate Template Enrollment Attacks – \(ADCS\)](#)
- [How one misconfiguration in ADCS can lead to full AD Forest compromise](#)
- [Investigating Ransomware Deployments that happened via Group Policy](#)
- [Hunting and Responding to ProxyShell Attacks](#)
- [Investigating ProxyLogon Attacks and how to mitigate it](#)

## TAGS

[Jupyter Notebooks](#)