The first step is to create your free developer account with Microsoft which you can do [here](https://developer.microsoft.com/en-us/microsoft-365/dev-program) and then create your free azure account [here](https://azure.microsoft.com/en-us/free/search/) with the same email you used for your developer account so that they link together (this will be essential for this to work). Next you need to create a source of logs for sentinel to fire alerts from, you can choose what you want to ingest for logs in the data connectors and for this project the ones that are necessary are the Azure active directory sign-in logs. After setting that up you can test that you are ingesting logs correctly by going into Azure active directory and signing into any account you have under this free created account (the developer account assigns users under your admin account by default). If you are seeing sign-in logs then your good to move on to the next step.

\*Ignore the previous step if you already use Sentinel as your SEIM

Next step is choosing the account you want to use for your honey credentials, you can pick any account that is a user in your AAD, if you are a company that uses Azure Sentinel as your SEIM you’re going to have to create an additional account and I recommend disabling the account after creating it, an additional recommendation with this is to make sure that the account you create follows the same username pattern as the other accounts in your environment because some threat actors won’t even check the credentials submitted unless it follows certain parameters. Next, you need to create an analytic rule under the analytics tab in Sentinel, when you select create choose the option that says, “Scheduled query rule”. Enter what you want the rules description to be for mine I put “Create alert every time \*\*\*\*\*\*\*\*\*\*.onmicrosoft.com is attempted to be signed into” but what you want the description to be is up to you. Under entity mapping you want to make sure you add the IP field so that you can see the IP address associated with the event when it fires and then have that feed into the email you configure to send within the logic app. For the rule frequency you want this query to be run every 5 minutes which is the least amount of time you can set it to, that way a new incident will be created within 5 minutes of the honey account being signed into. For the rule query I used the following KQL:

SigninLogs

| where UserPrincipalName contains "\*\*\*\*\*\*\*\*\*\*.onmicrosoft.com"

| where ResultType contains "50126"

The bottom line in the KQL signifies a sign in event that fails, so for this honey account, when entering the credentials into the credential harvester website the password does not have to be the actual password for the user since all we are using this account for is to get the IP address used to attempt the sign-in. You want to set this rule to trigger for every event and make sure that the “Create incidents from this rule” is set to enabled. For the automated response tab, you are going to come back to this part later and add your logic app playbook to get the logic app to fire when an incident is created. After this analytic rule is created you can test it by signing into the account that you want to act as your honey account Graphical user interface, text, application, email

Description automatically generatedand checking if an incident is created in the incidents tab in Sentinel.

The next step is to go ahead and create your logic app. You can use most of the defaults that are present in the logic app when you create it, it is important that you create it as a workflow and not a docker container. Add any preferences you want for your logic app and create it. Once you have your logic app created you can go into the logic app and you will need to add some permissions and enable some features to be able to authenticate your playbook in the logic app. [This document](https://docs.microsoft.com/en-us/azure/sentinel/authenticate-playbooks-to-sentinel#authenticate-as-an-azure-ad-user) does a great job of outlining everything you need to do to set up the managed identity with Sentinel and get all the correct configurations needed for this logic app to work correctly (note: you do not need to do the service principal portion of this document). After going through this document and configuring everything you will create a new API Graphical user interface, text, application, email

Description automatically generatedconnection and it should look like this:

Now that you have the permissions required for the logic app you can begin to design the logic app. The trigger for this logic app is going to be the Microsoft Sentinel Incident which will fire on any sign-in attempt from the user that you specified in the analytic rule on the previous step, it is essential that for this trigger you are connected to the connector that you created for Azure Sentinel and not just your admin account within Azure. The next step is to get the IP’s associated with the incident so that we can feed that information into the email that is sent out when a login attempt is made, for this step you also need to be connected to the same connector as the previous step for the logic app to have the correct permissions needed to output these steps. The last step in designing your logic app is to use the “for each” action and populate the select an output from the previous step field with IP’s so that the logic app references the entities table and gets the IP’s from the incident created from your analytic rule. The last step is to use the send an email (V2) function to get the logic app to send you (or your ticketing system’s email address) an email that says what IP address was used to attempt the login to the account.

Graphical user interface, application, Word

Description automatically generatedNow that you have your logic app created, designed, and have all the permissions needed to successfully run the logic app, the last step is to go back to your analytic rule and add the playbook for your Logic app that you created. This will allow the logic app to run every time an incident is created from your analytic rule.

Graphical user interface, application

Description automatically generated

Now you have correctly configured your logic app to run when you get a sign in attempt from the honey account you chose to use! You can now use this information to do additional threat hunting in your environment based on the IP addresses that you get from the alerts. I hope this information has been useful for everyone who has followed along and implemented this, as it took me a great deal of time to get this working! I also hope you use this in your environment regardless of what SEIM provider you use because this method is FREE to do! Now all you need to do is wait for a credential harvester phish to come in and enter the account email and any password into the page and wait for them to take the bait!