Senior Project I & II

Microsoft Azure and Sentinel – HoneyPot SOC

Bruno Jimenez

Dr. Darwish

Table of Context

Abstract

This project focuses on the use of honeypot systems as a hands on approach to understanding and analyzing cybersecurity threats. By leveraging Microsoft Azure, a virtual machine (VM) will be intentionally configured with vulnerabilities to attract real-world attackers. This honeypot will serve as a tool for collecting and analyzing failed Remote Desktop Protocol (RDP) login attempts. A custom API will extract geographic data, mapping the origins of these attacks and providing valuable insights into global attack patterns.

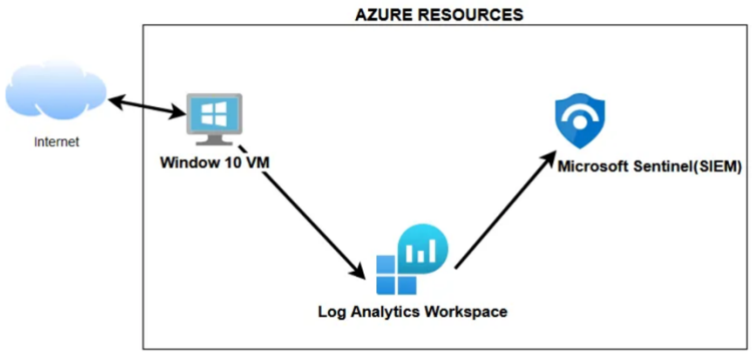
To complement the honeypot, Microsoft Azure Sentinel, a Security Information and Event Management (SIEM) system, will be implemented for real time analysis and visualization of security alerts. Sentinel's advanced capabilities will help detect, analyze, and respond to threats by monitoring logs and identifying trends in attack behavior, such as brute force attempts or credential stuffing techniques. The project will also focus on creating visualizations to showcase where attacks originate, using IP addresses to pinpoint longitude and latitude.

The primary goal of this project is to provide a practical learning environment for cybersecurity analysis while generating meaningful data about attacker behavior. This hands on approach will enhance skills in cloud security, logging, and threat analysis, particularly within the context of cloud environments. By identifying common attack vectors, usernames, and passwords used in these attempts, the project will contribute to a deeper understanding of the threat landscape.

Introduction

Security Operations Centers (SOCs) play a crucial role in monitoring, detecting, and responding to security incidents. One approach to enhancing security awareness and improving defenses is the use of honeypot systems designed to attract and analyze attacks. After extensive research, I believe Azure is the best platform for this project, especially considering the $200 free voucher that can help cover the costs of running a network however there are multiple clouding websites you can use like AWS, and Google Cloud. Running the network will approximately cost around $0.0144 per hour and it can vary depending on how many you create. Using Microsoft Azure, I will create a Virtual Machine (or multiple if possible) that is intentionally vulnerable to attract potential hackers in real life. This setup will allow me to capture and log data in real time.

With the data I collect, I will be creating a way to see where the attacks are coming from within the world along with a way to display it as well. For this we will be using a custom API that will take the IP address and give us the longitude/latitude and some basic geographic data. Additionally, I will implement a Security Information and Event Management (SIEM) system for real time analysis of security alerts. What a SIEM does is helps detect, analyze, and respond to threats. We will use Microsoft Azure as well to plot the data collected, focusing on failed login attempts that try to get into our VM honeypot.



Related Work & Research

Researching the cost of implementing my project on Microsoft Azure was a critical step in determining the financial feasibility of running a honeypot system. Because I knew I was going to be running this project through a long period of time by analyzing Azure's pricing for virtual machines and services like Azure Monitor and Sentinel, I was able to estimate hourly costs for small VMs, which start at around $0.0144 per hour, and account for other factors such as storage. The data transfer was a new cost as well, which was $15 per month. However, this process also helped identify cost saving opportunities, like using Azure's pay-as-you go, ensuring that my project remained budget friendly without compromising its objective. "We offer eligible customers $200 in Azure credits ("Credits") to be used within the first 30 days of sign-up and 12 months of select free services (services subject to change). This offer is limited to one Azure free account per eligible customer and cannot be combined with any other offer unless otherwise permitted by Microsoft." Additionally, utilizing tools such as Azure's Pricing Calculator allowed me to estimate expenses based on different configurations, ensuring that my project stays within the constraints of the $200 free credit.

Familiarizing myself with Microsoft Sentinel and its tools is essential for leveraging the full potential of my project. Microsoft states, "traditional security information and event management (SIEM) systems typically take a long time to set up and configure. They're also not necessarily designed with cloud workloads in mind. Microsoft Sentinel enables you to start getting valuable security insights from your cloud and on-premises data quickly." Sentinel provides advanced capabilities for monitoring, analyzing, and visualizing security data, enabling real time insights into attack patterns on the honeypot. Its integration with log analytics allows me to identify

trends in failed login attempts, such as commonly used usernames or IP locations, which are key to understanding threat landscapes. Additionally, learning how to configure custom detection rules and alerts ensures that I can respond effectively to security incidents. This knowledge not only enhances the functionality of the honeypot but also equips me with practical skills in using SIEM systems, which are critical in modern cybersecurity operations.

I also needed to know how to use PowerShell and API code to capture logs in the Azure Virtual Machine (VM), it was a crucial part of my project. I began by studying Azure's documentation on PowerShell cmdlets, which are specifically designed for managing and extracting data from VMs. Additionally, I explored how to integrate custom APIs to map IP addresses by retrieving geolocation data. Tutorials and code examples on implementing IPGeoLocation APIs were particularly useful, as they demonstrated how to query IP addresses and convert them into actionable geographic insights. These resources not only helped me understand the technical setup but also ensured that my methods aligned with cybersecurity best practices.

Finding and making sure I used the correct API was a huge challenge aswell. The API provided by ipgeolocation.io was a critical step for my project to integrate accurate IP geolocation capabilities. The API offered a wide range of features, including real time geolocation data, city specific information, and integration with popular tools like Google Maps. Their website state, "Our IP API provides geographical information about website visitors with any IPv4 or IPv6 address in JSON and XML format over a secure HTTPS connection even in our free plan. We provide data such as country name, country code, city, state, local currency, time zone, ISP, ASN, Company Details, device data from User-Agent String, VPN, Proxy, TOR and threat intelligence data. Our services are globally available with latency-based routing." The

documentation provided clear guidelines for using endpoints, handling input parameters, and parsing responses, which helped me understand how to retrieve geolocation data and map attack sources effectively.

Lastly, Learning Microsoft Azure query commands was an essential part of my project to effectively analyze the data captured from the honeypot. I had to familiarize myself with Kusto Query Language (KQL), which is the query language used in Azure Monitor Logs and Azure Sentinel. Microsoft likes to describe it as, “The language is expressive, easy to read and understand the query intent, and optimized for authoring experiences. Kusto Query Language is optimal for querying telemetry, metrics, and logs with deep support for text search and parsing, time-series operators and functions, analytics and aggregation, geospatial, vector similarity searches, and many other language constructs that provide the most optimal language for data analysis.” This involved understanding the syntax and structure of KQL, such as how to filter data, summarize events, and visualize results through charts and graphs. Many videos along with official Azure documentation, and examples of common queries provided. Helped me construct queries for tasks like extracting failed RDP attempts, identifying common attack patterns, and plotting geographic data. These skills allowed me to efficiently query the log data and transform it into actionable insights, which is crucial for both monitoring and presenting results in the project.

Core Project Overview

Running a honeypot VM on Microsoft Azure to collect failed RDP logs offers numerous benefits for us students learning cybersecurity or analyzing attack patterns. It provides a hand on learning experience, enhancing skills in cloud security and logging while offering real world data to study attack techniques such as brute force attempts or credential stuffing. By monitoring logs, we can identify trends, such as commonly used usernames and passwords, and gain insights into the threat landscape, including the geographical sources of attacks.

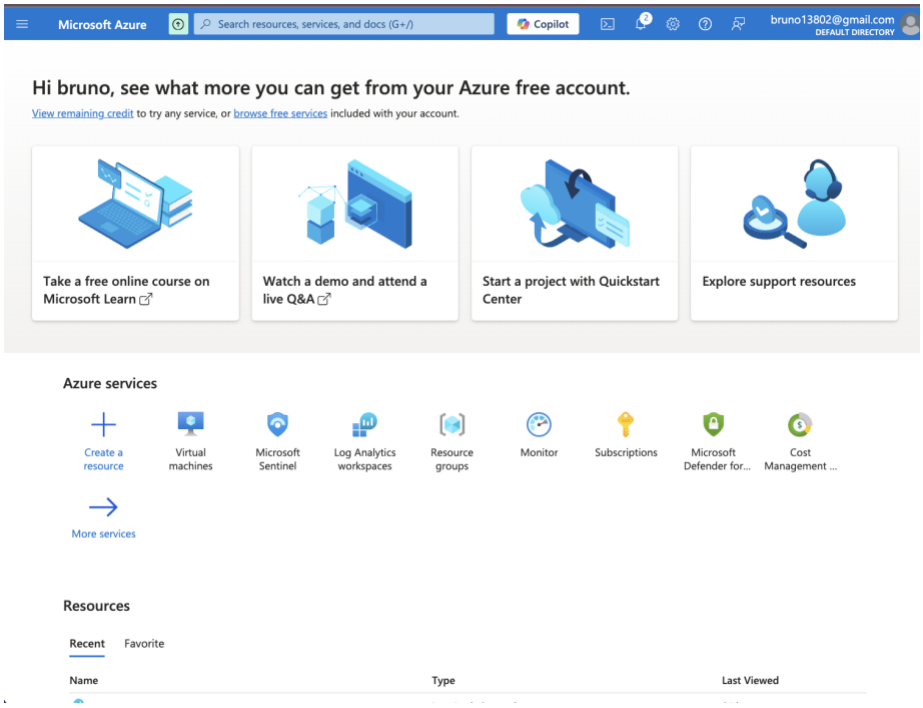
This setup supports security research by testing detection tools, developing custom rules, and potentially contributing to threat intelligence. Azure specific benefits include understanding attacker behavior in cloud environments, leveraging Azure Sentinel for monitoring and visualization, and maintaining cost efficiency with small VMs. Additionally, honeypots provide a safe, ethical environment to study attacker behavior, , and test incident response strategies without endangering real systems. Overall, it's a practical and impactful tool for advancing my cybersecurity knowledge and defenses.

In addition to its role in hosting honeypots, Microsoft Azure offers a wide range of tools and features that can significantly enhance cybersecurity learning and practice. Azure provides access to a large amount of security services, including Azure Security Center and Azure Defender, which allow for proactive monitoring and threat protection across virtual machines, databases, and networks. Azure's scalability ensures that students like me can experiment with different configurations and expand their projects without worrying about resource limitations.

Experimentation Section

For this project the first thing we need to do is create a Microsoft Azure account. This will need a debit card connected as well in order to fully complete the signup phase. I don't have an image however when creating the account you will have two options for the account: Pay-as-You-Go, or the Azure Subscription. Select the Azure Subscription and it will offer a $200 credit to use for your project and then it will automatically charge your card if you pass the limit.

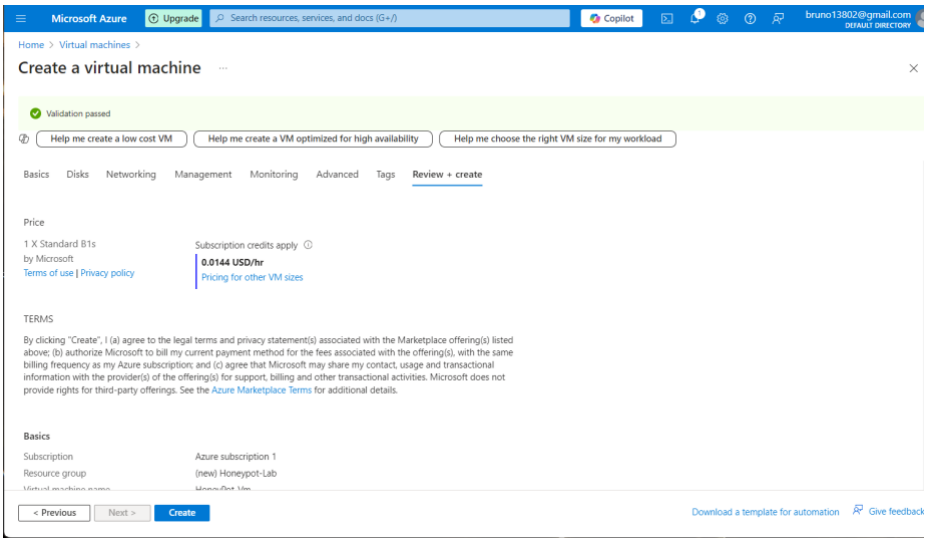
Figure 1



From here we will in the search bar above type "Virtual Machines" and we will select "Create New." Each Machine you create will cost $0.0144 an hour. We will select the "Windows 10 Pro" ISO disc for this project.
We will name it HoneyPot_VM
You will also need to add a USER and PASS to access it so write it down somewhere safe.

To make everything easier as well when creating your VM you can have the option to create a Resource Group to store all your VMs/Files together to make access simpler.

Figure 2



While creating the VM we are going to edit our NIC network security group. This is basically like a firewall. We will be creating our own so we delete the current existing one and make a new one.

First, for the destination port we will put a " * " to allow all destination port ranges available.
Then we will allow "**ANY**" protocols.
Lasty, for priority we will put "**100**." Basically what this does is this sets the order in which the rule is assessed in relation to other security rules, the lower the priority number the faster it is to be examined first.

This is all to make our VM discoverable to for hackers to see that it is online.

Figure 3

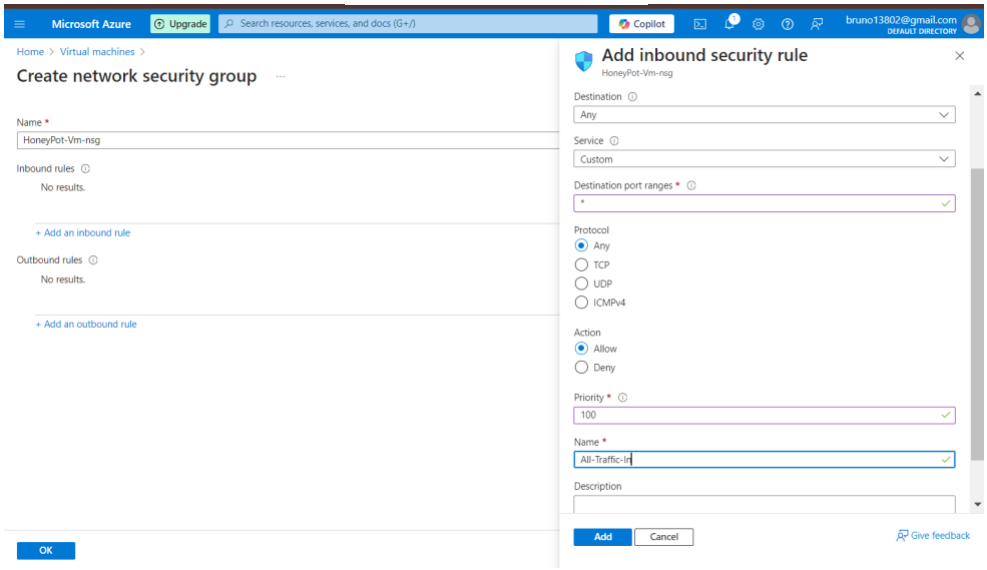
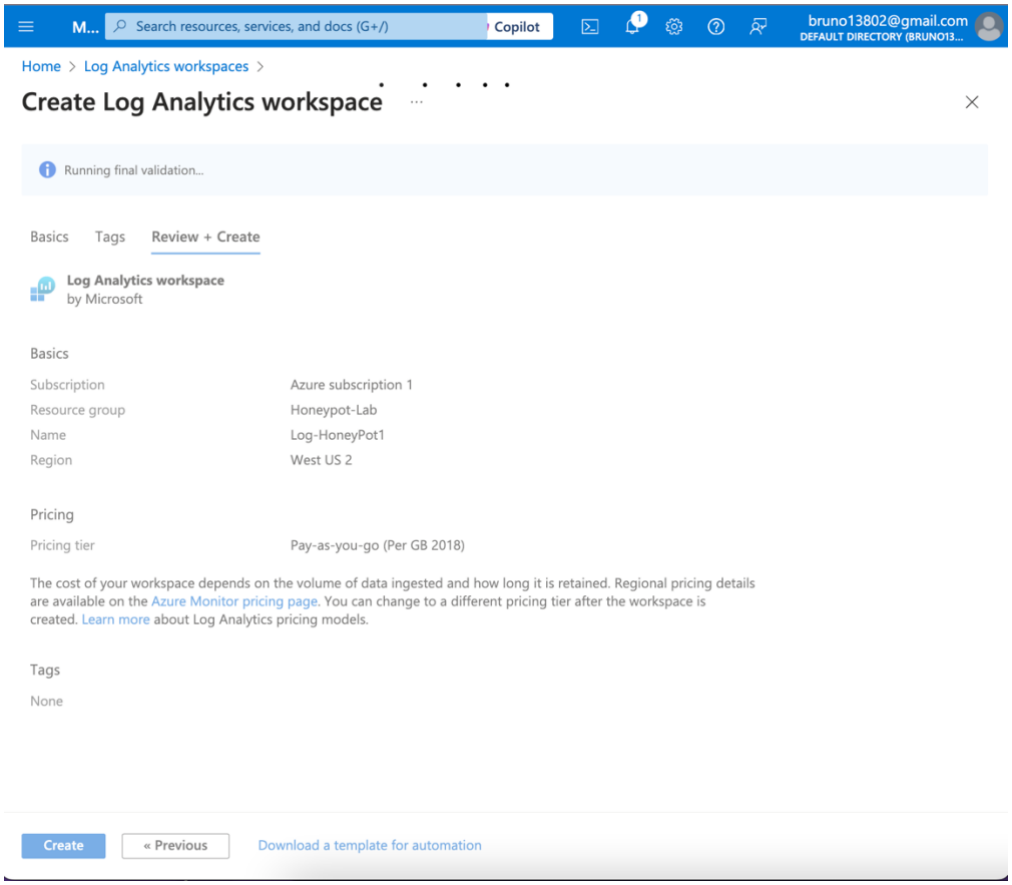


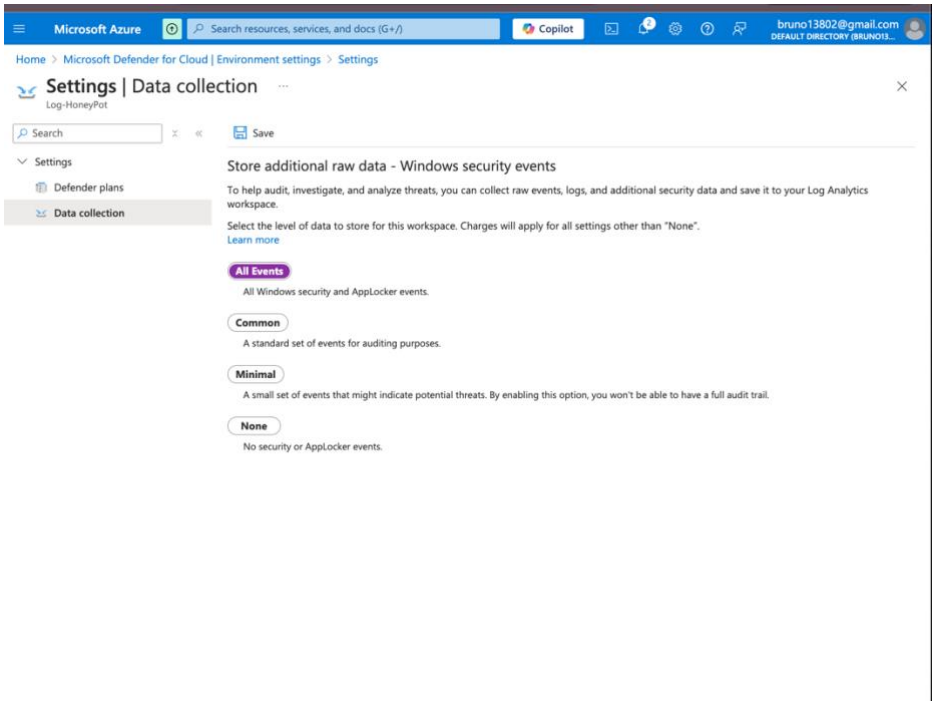
Figure 4



While the VM is being created, (takes around 5-10 mins)
We will create a Log analytics workspace. The purpose of this is to ingest logs from the VMs windows event logs and store them in our Log Analytic Workspace, then inside the actual VM we will also create a log in order to link the two together.
We will call the log Log-HoneyPot1
(The log will also take around 5 mins to be created.)

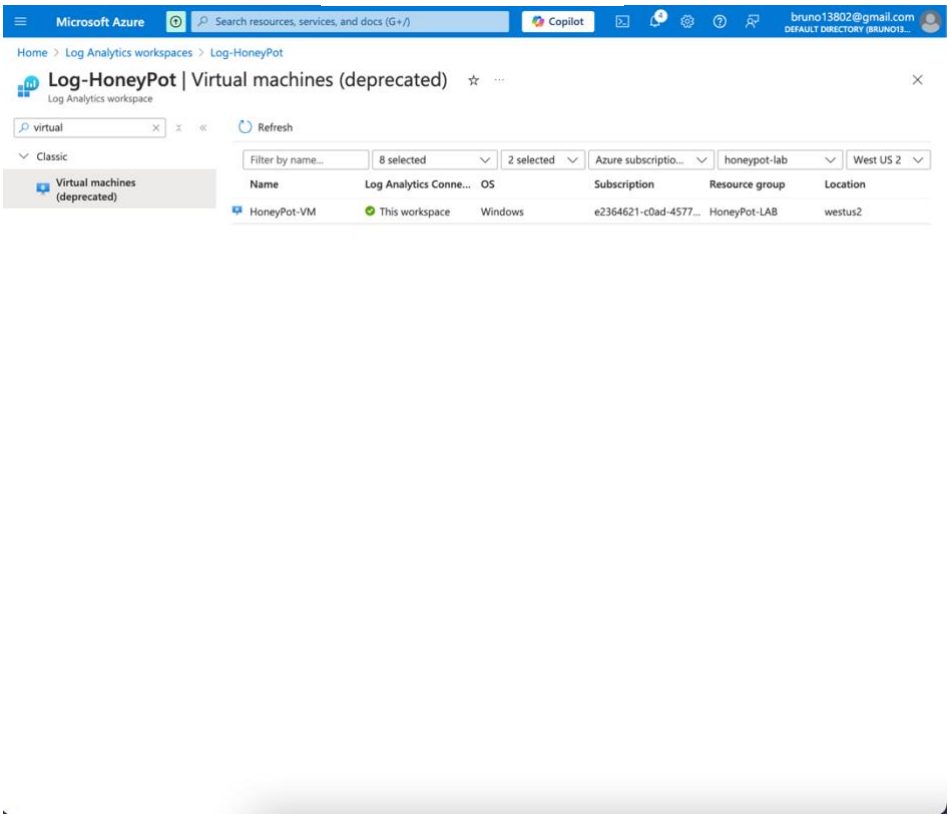
Next in the Search bar type “Microsoft Defender for Cloud” And select Data Collection. Before you get to this screen you will be presented with a $15 a month subscription needed in order to be able to collect the data from the VM and store it in our log that we created in the Log Analytic Workplace.
Once on the screen we will allow “ALL EVENTS” and save it. This will allow us to collect all the data from the Logs in the VM.

Figure 5



Once everything is saved we will return to our Log Analytic Workspace and our Log should be completed by now. On the side click Virtual Machines and an option next to the refresh will say “Connect” select the Log that we created and connect. Once connected this will link our VM to the log so we can get everything coming from the Windows Event Log.

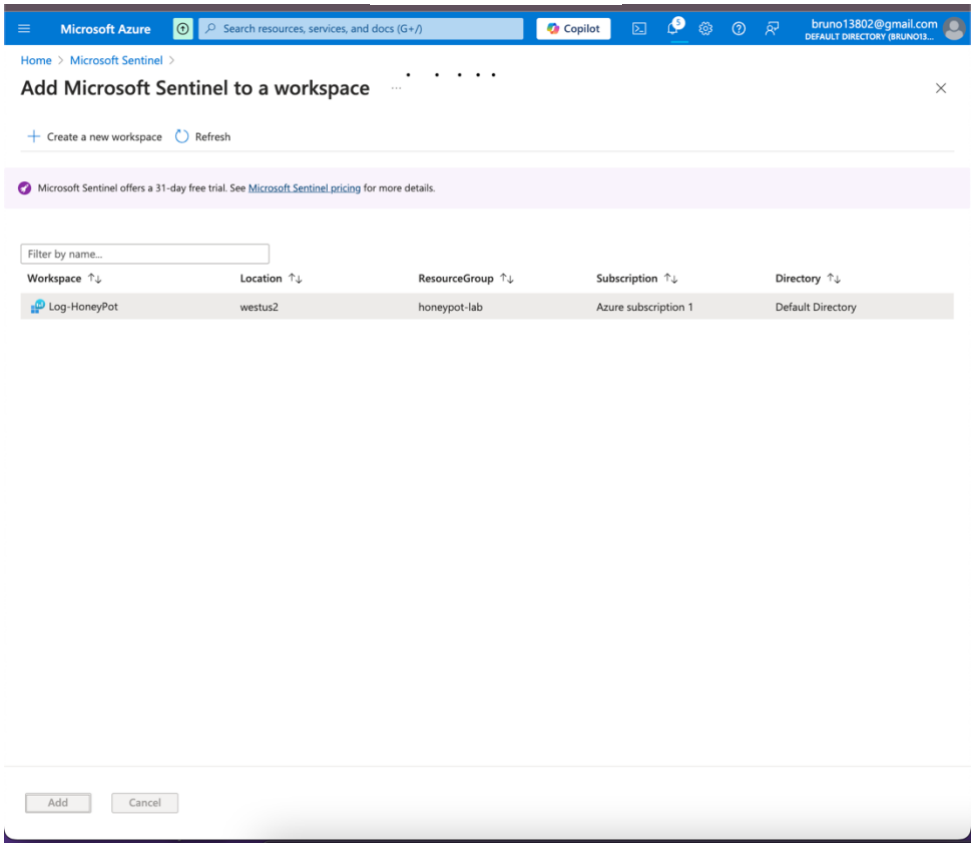
Figure 6



Now we need to set up Microsoft Sentinel. This is the SIEM we will be using to visualize the attack data. Here on Sentinel we will connect our log as well that we created. It is the only one that is showing select it.

To breakdown what we have done so far is create the VM , created a Log Analytics Workspace to connect to the VM to link data. Added Microsoft Defender to collect data from the VM. And lastly created the SIEM to visualize that data.

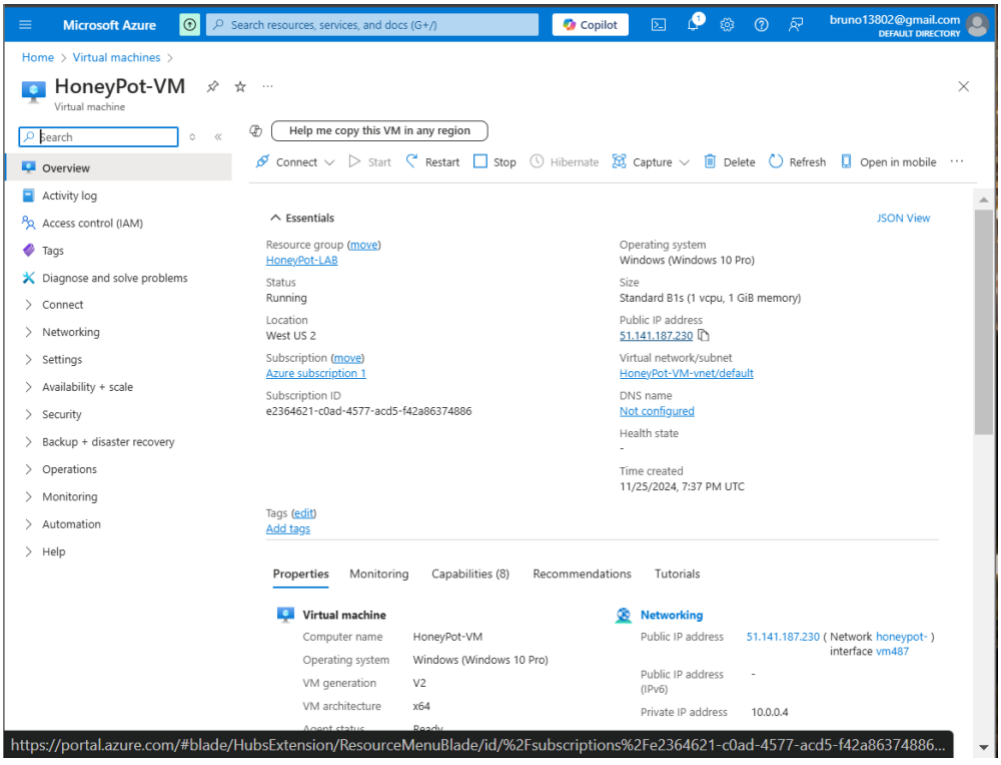
Figure 7



In the Search bar go back to Virtual Machines and select the one we created earlier. As you can see we were given a public IP address. Ours is 51.141.187.230

We will need to copy and save this in order to access our Virtual Machine. So far for setting up our network I have used a MacBook. But I will need to switch to a Windows Computer to access our VM since we will need an application only available on Windows PCs.

Figure 8



Now that I have switched Computers we need to go on our search and type “Remote Desktop Connection” This is what will let us connect to our VM. In the “Computer” Section type in your VMs IP address.

Next from here it will ask your permission to connect. Once access is given the USER and PASS that we made earlier must be typed in. After typing in your credentials correctly a new window will pop up with your VM ready.

Figure 9

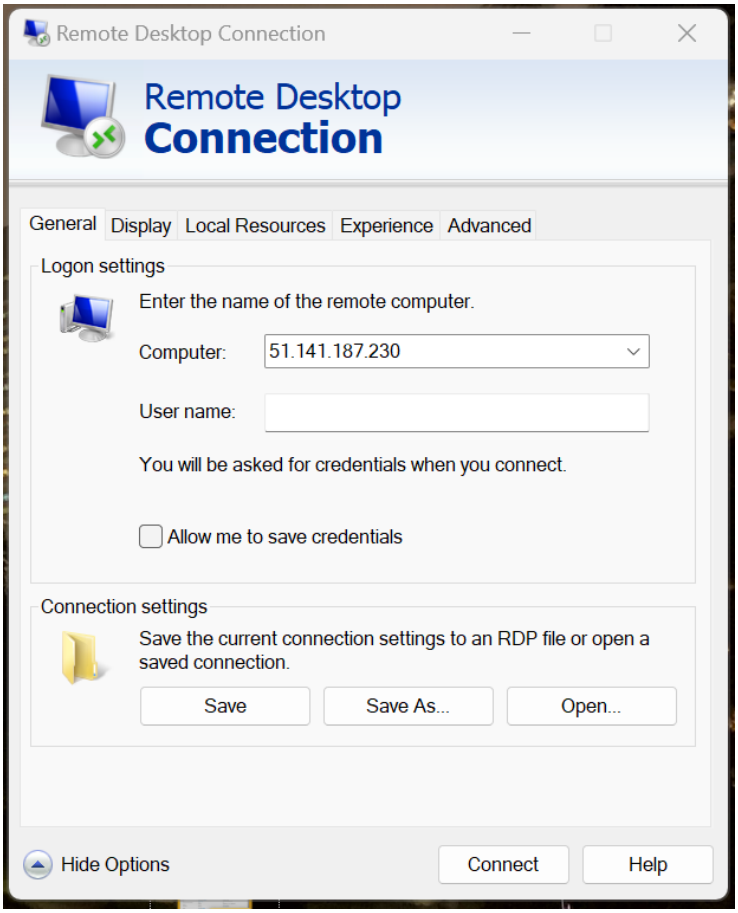


Figure 10

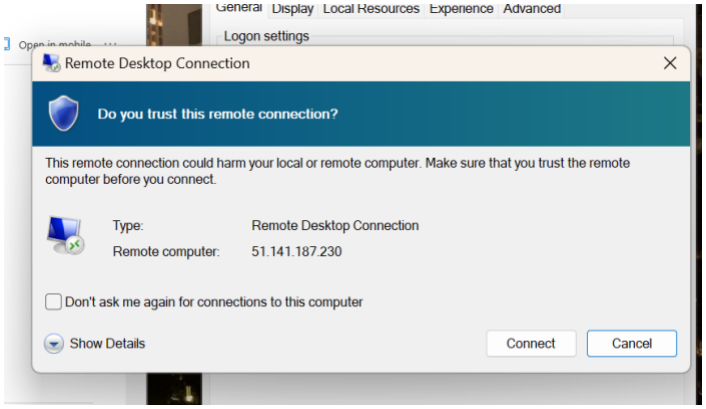


Figure 11

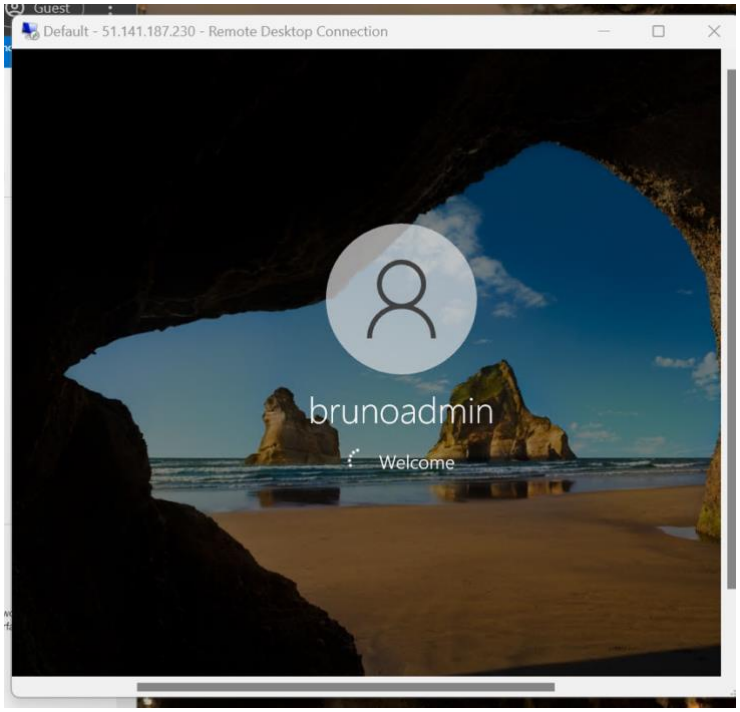


Figure 12

Once we are in the VM lets head to the start menu and type in “Event Viewer” on the side select “Security” and once inside we can see all the security events on the actual VM. We will be focusing on Event ID 4625 which are Audit Failures these will be the hackers trying to log in to my machine.

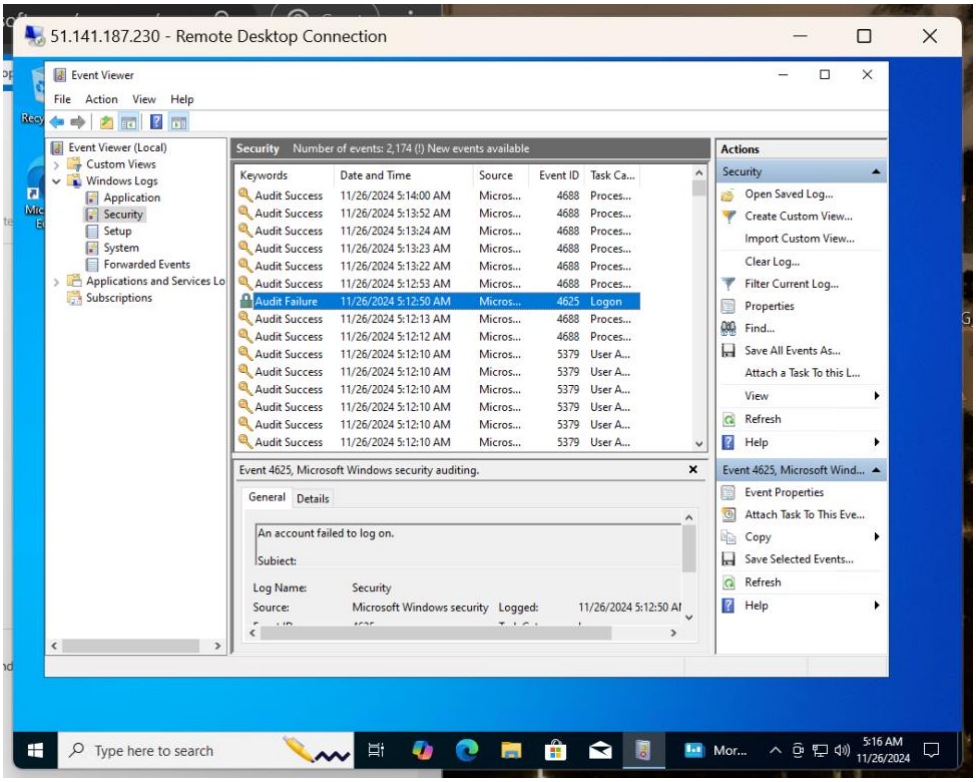


Figure 13

If we double click on the Audit Failure we will see a lot of Data. Now what we will do with this is try to get the IPs from these hackers trying to access my Machine. It will be under “Source Network Address.” However in this Event Viewer it doesn’t really give us any information on the IP. This is where the next step comes in handy.

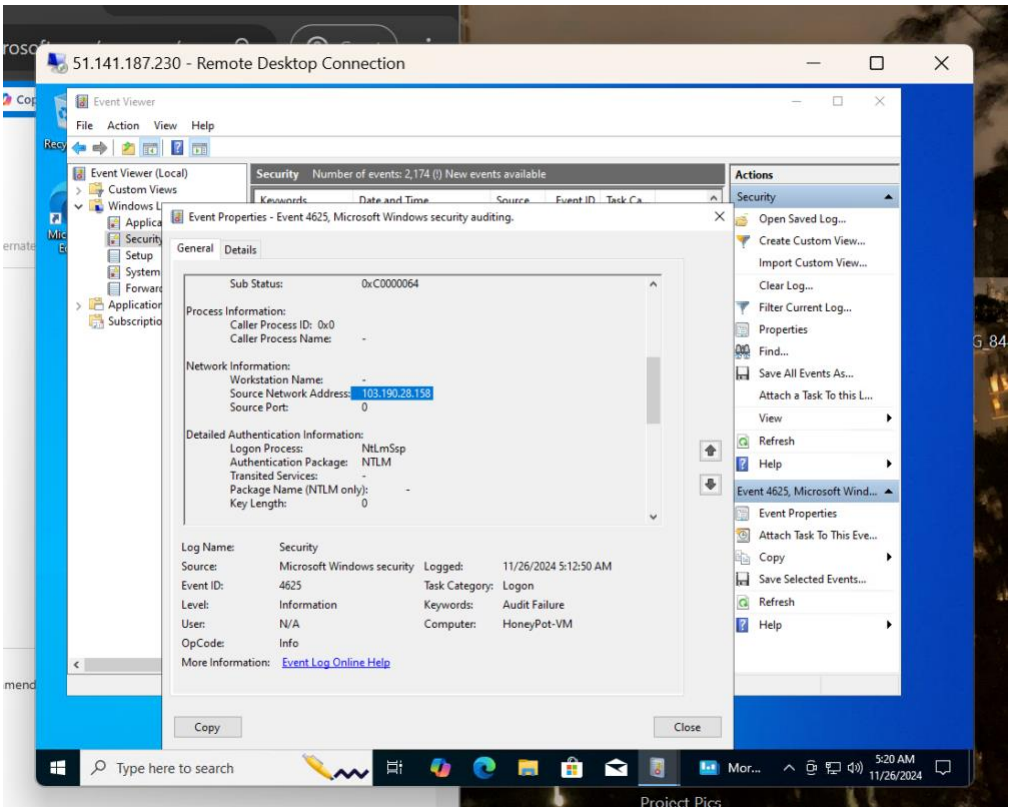


Figure 14

This is an API called “IpGeolocation” If we copy the IP from the one we saw above and paste it in here we can see that the attack actually came from Indonesia and as we can see it gave us the continent, state, hostname, etc.
We are going to use this to create our own custom log and send it to Log Analytics Workspace and use Sentinel to read and plot the data given from the API.
You must create an account for this and you will get a “Key” that we will need for the PowerShell script this will be using. The free version only allows to convert 1,000 IPs for free per day.

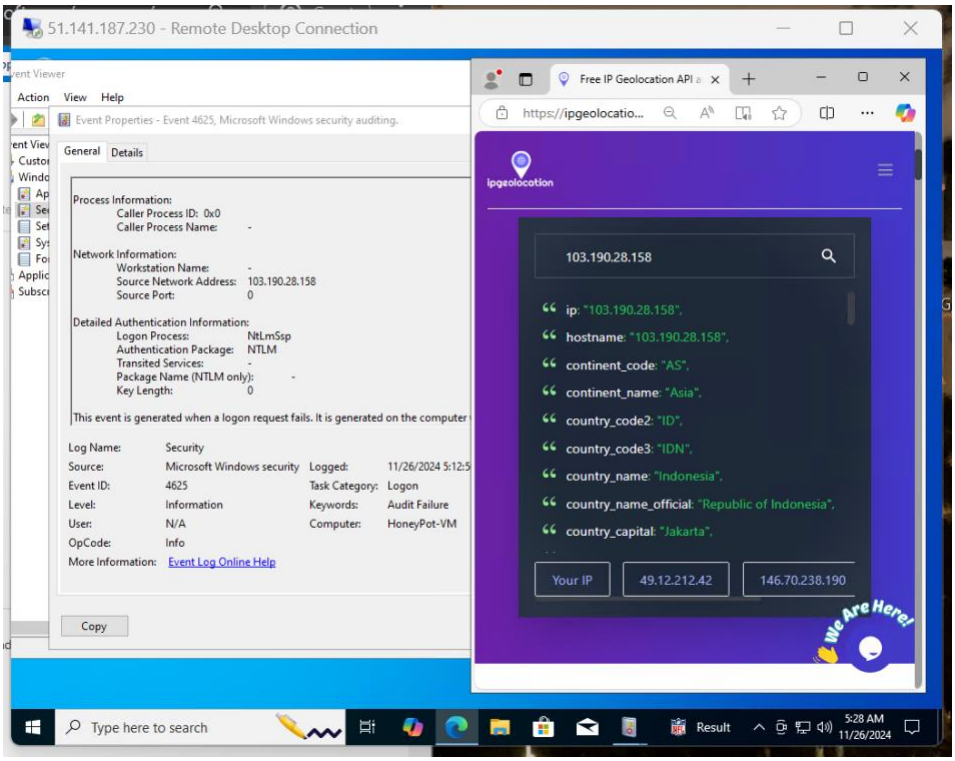


Figure 15

If we go on our actual Computer not the VM and try to ping ourselves it just timeout. Now we will go back to the VM and edit our firewall. We could just allow icmp however since it is a honeypot and is meant to be intentionally vulnerable we will remove a couple of things from the firewall just to allow hackers to see it more easily to get their attention.

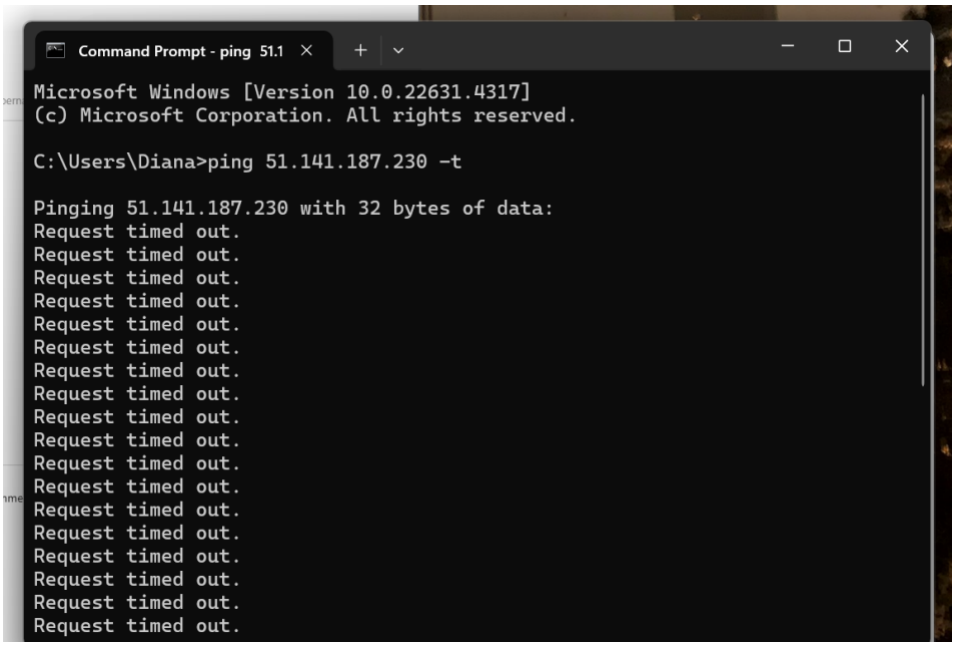


Figure 16

Head back into the VM and in the Start type in “Windows Defender Firewall with Advanced Security” select “Window Defender Firewall Properties” and disable the Public profile.

Now if we go back to our actual computer (Not VM) and ping our VM we can see that they are now going through.

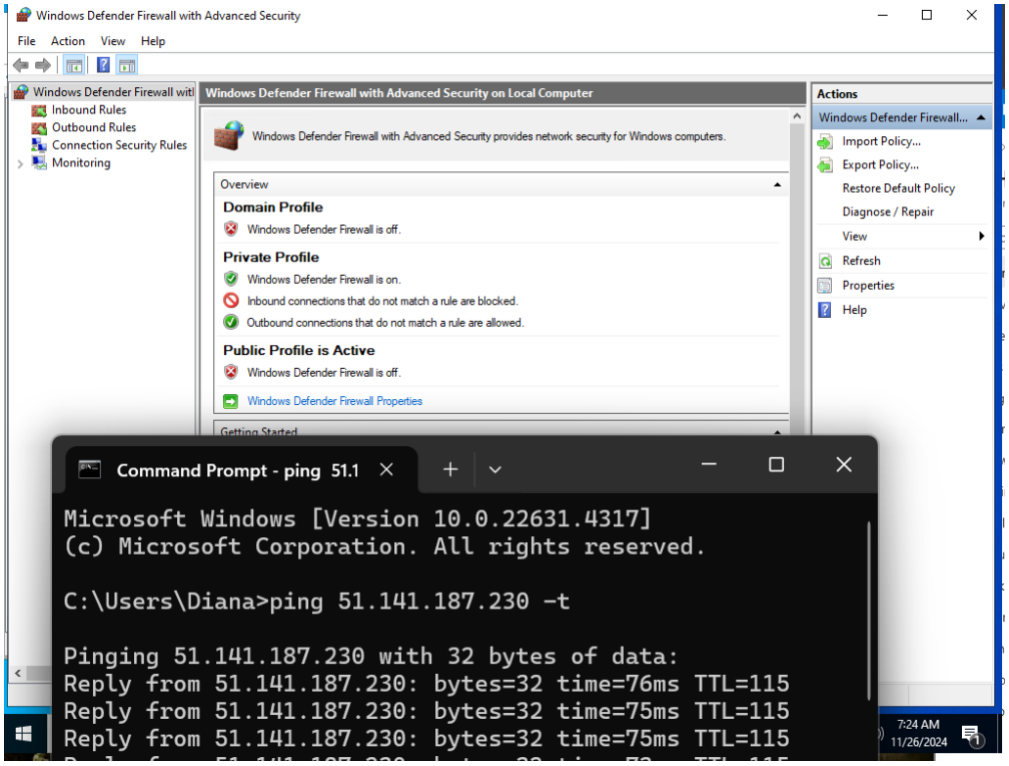
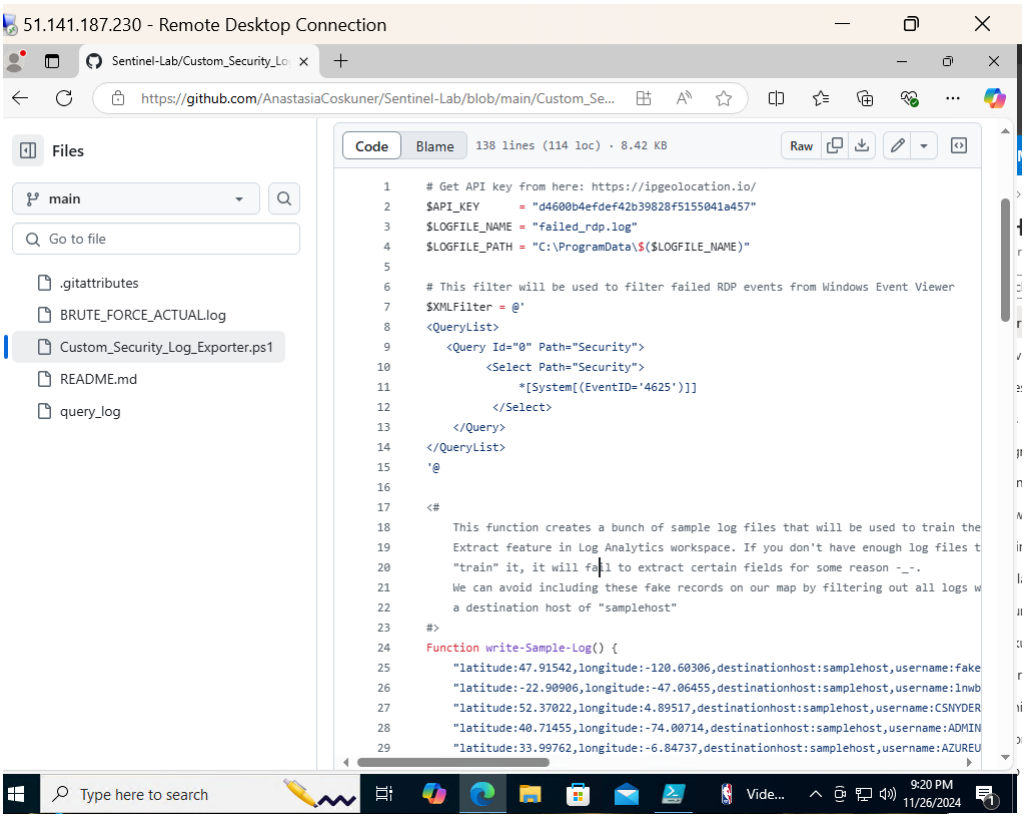


Figure 17

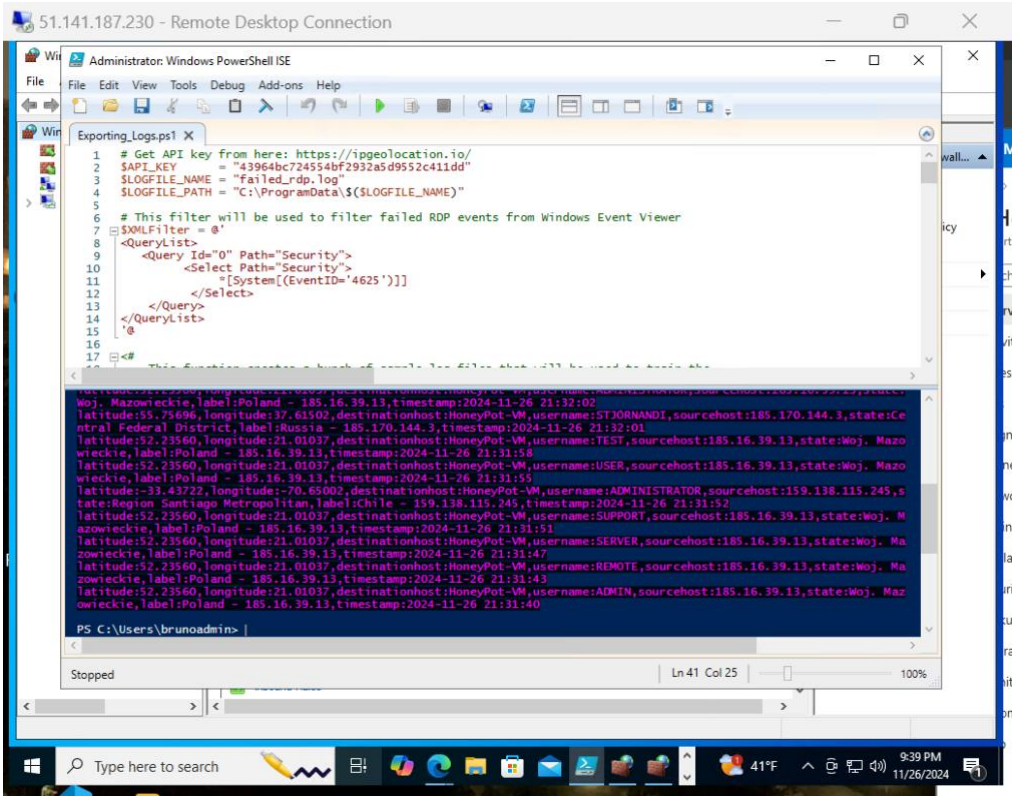
We want to automate the audit failures to be sent to the API. This is so that we can store it in a log within our VM to export to our Log Analytics Workspace. For that we will use a PowerShell script that automatically converts the IPs into the data we need using the API we found earlier.



In the VM go to the start and type “PowerShell ISE”
This will allow us to paste in our Script. Click create new, paste in the code, when creating an account for the API remember the key, paste that in where it says “API_KEY”, and then run the script.
As soon as I ran it I was getting flooded with Audit failures.

So basically this code looks through the Event Viewer and Security log and finds all the people who failed to login and it grabs their IP and processes the data for it.

Figure 18



All the data we collect from this script will be stored in a default file that will be automatically created once the script is ran. It is called “failed_rdp” This is the Log file that we will use to train our Log Analytics Workspace to show it what it will analyzing and print out our custom log for the visualization.

Once everything is completed Save the PowerShell Script.

Figure 19

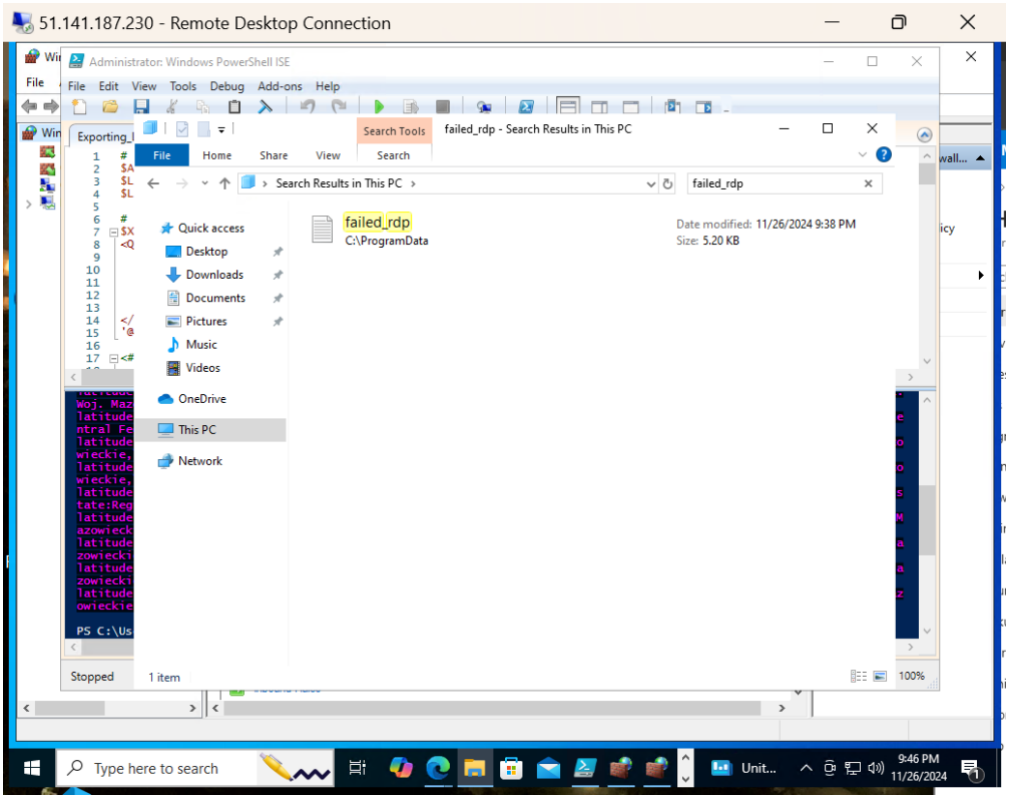


Figure 20

This is what is inside the actual Log, all the data that the script was able to extract from the event viewer.

Exit the VM for now and head back to Microsoft Azure. Go to our Log Analytics Work page and we are going to create a custom log. We will see a “Select Sample Log” and it will want us to add the logfile we created in the VM, For this we will just simply go to the VM and find the path, copy it and write it in the Sample log.

After that is complete your log data should be displayed in your Log Analytics Workspace.

Now save this and click next.

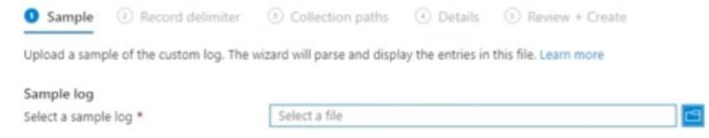
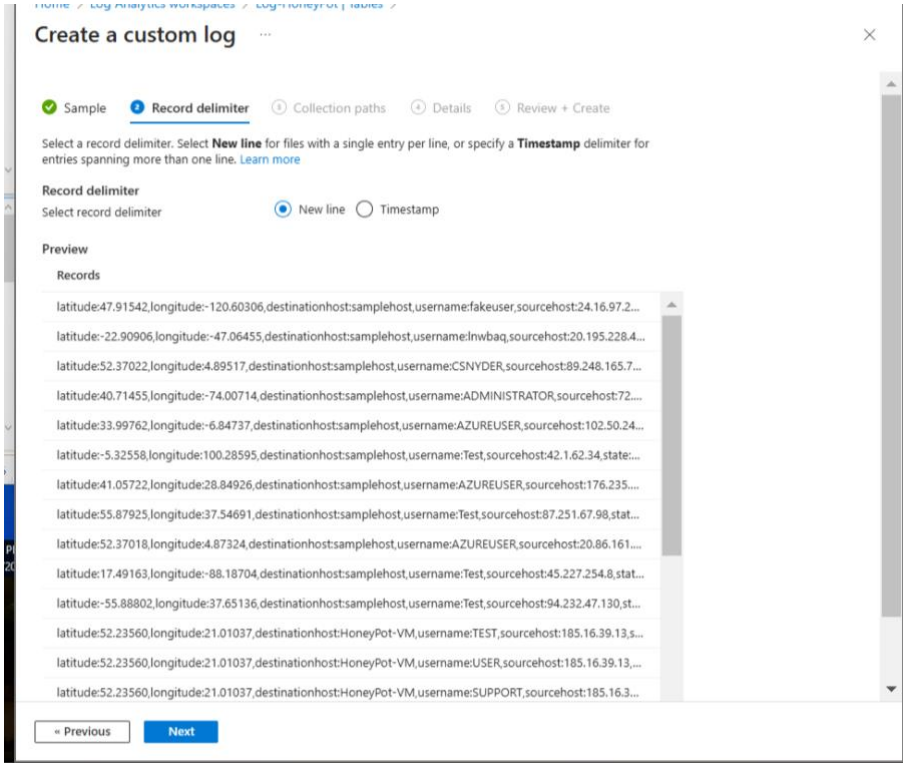
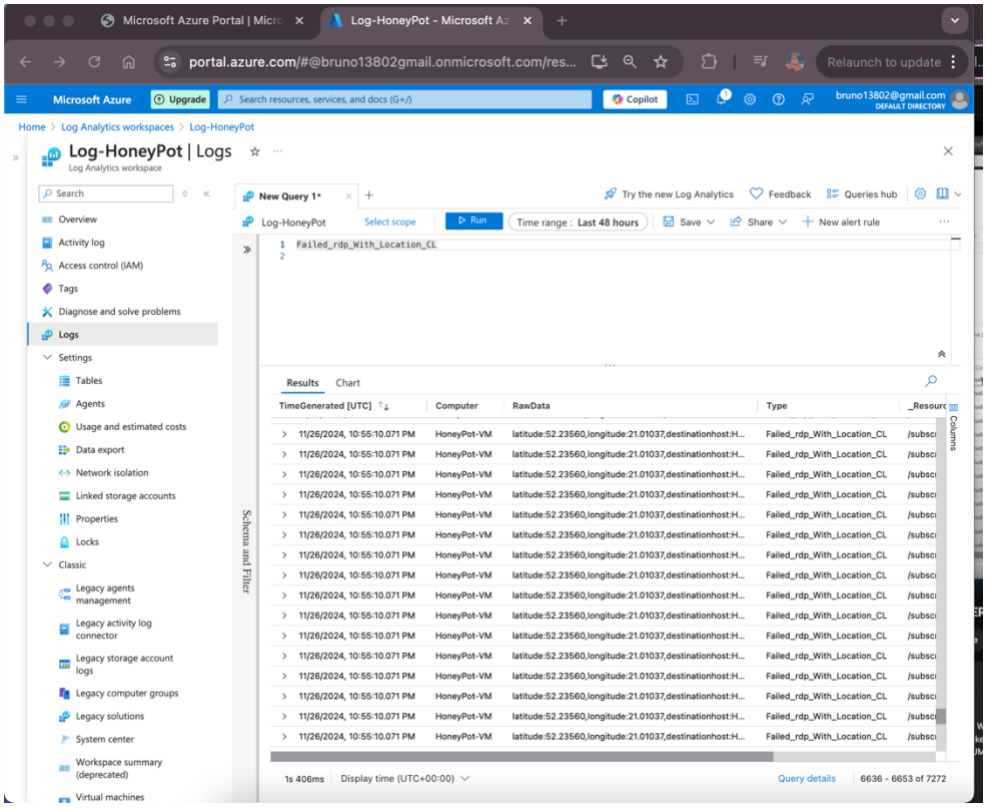


Figure 21 , 22



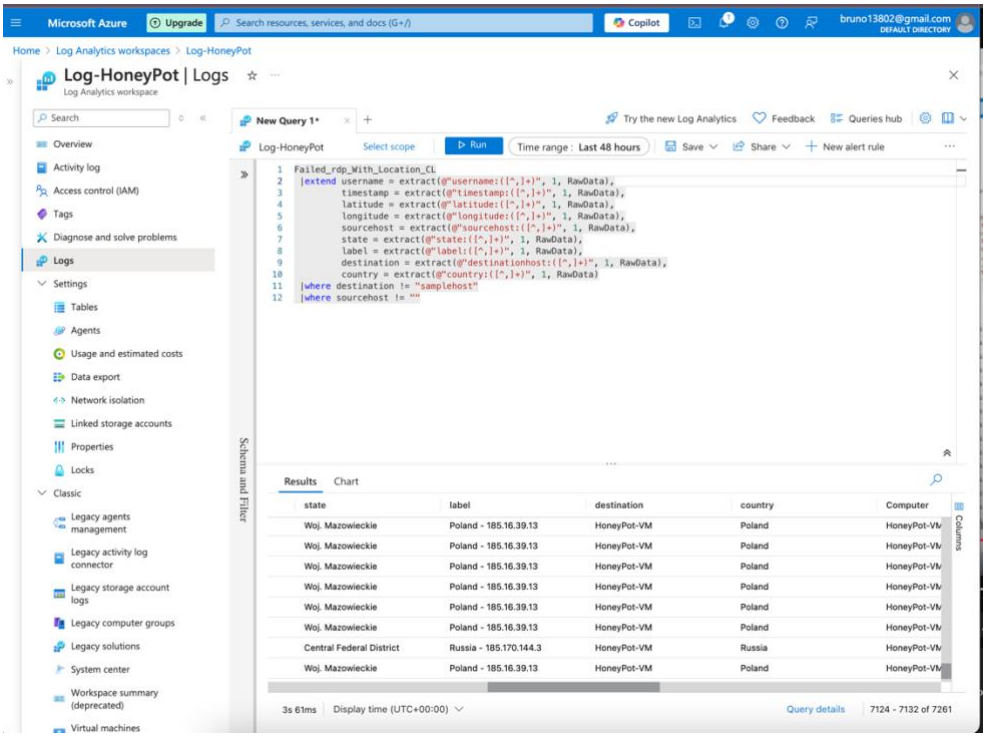
Once that is created Go back to Log Analytic Workspace select the log we just made and we will have a Query where it looks like we can write code. Type in “Failed_Rdp_with_location_CL ” and press “Run”
And as you can see we are getting Audit Failures Live from our Event Viewer being shown to our custom log we created for the visualization.

Figure 23



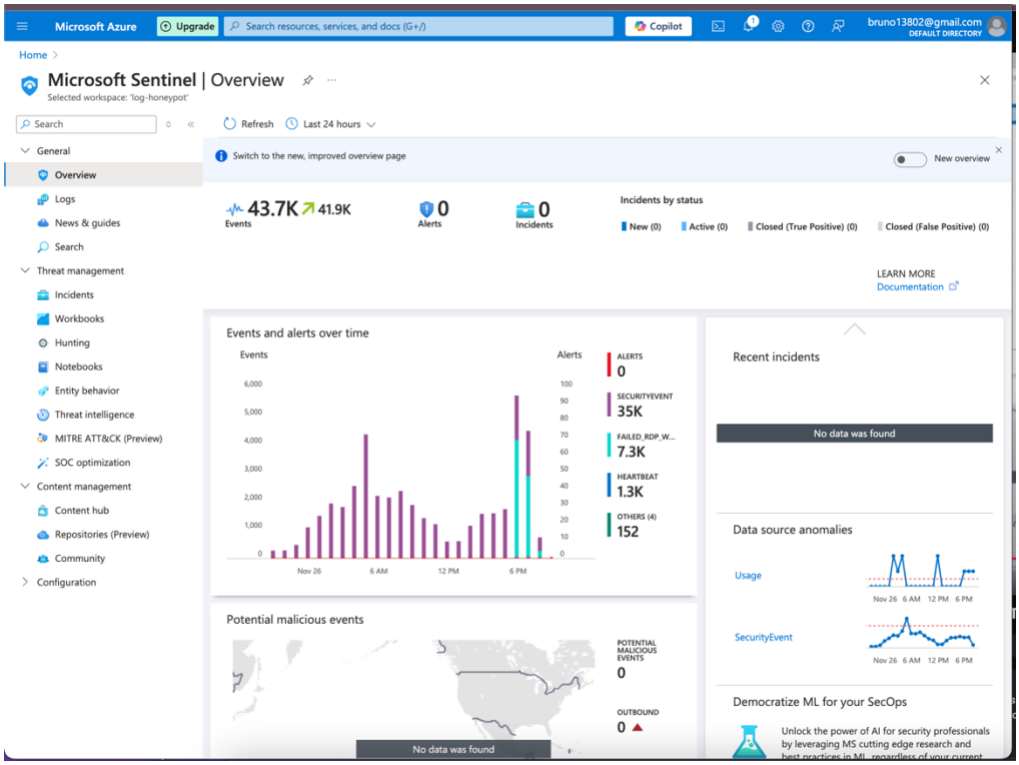
As we can see from the image above in the Raw data column, We have all this data but we got to extract it in order to get exactly what we are looking for, that is the Country, Continent, IP, Time Generated. The point of this is so everything can have its own field so that when it comes to plotting it on the map everything is organized. Once it is ran as we can see in the Raw Data Column it is now nice and organized so that we can plot it visually.

Figure 24



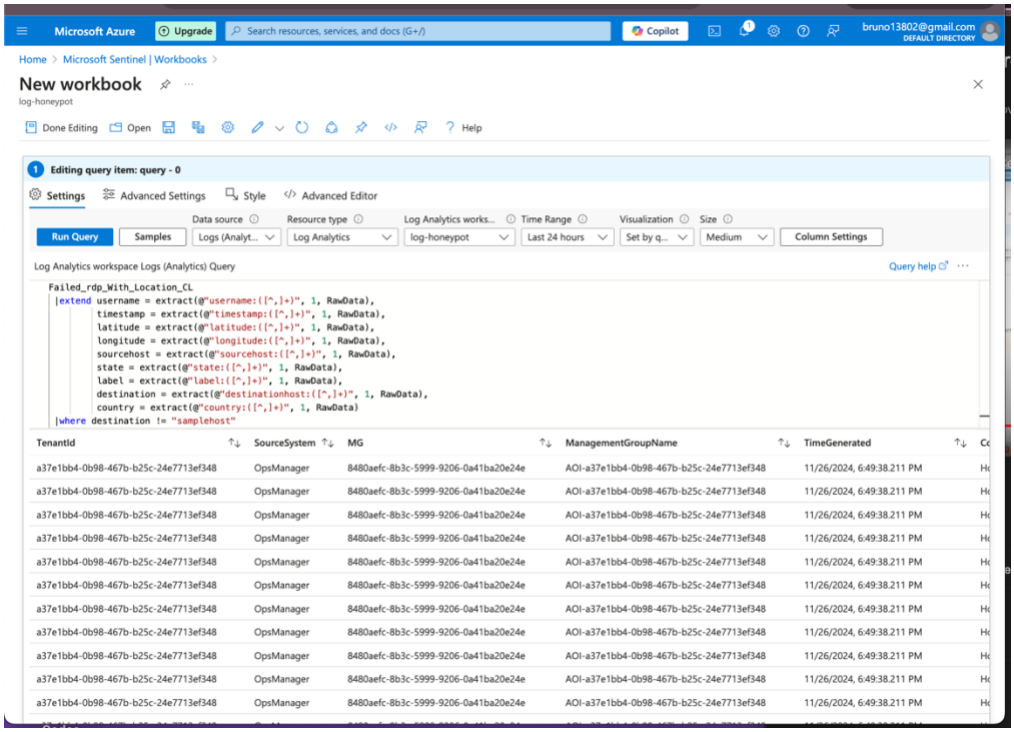
Once that is completed we must head back to Microsoft Sentinel, which is our SIEM. So far we have nothing to plot so we will head to “Workbook” on the side and select that and create a new workbook.

Figure 25



When creating the New workbook remember to select our Log Analytics Workspace log that we created called “Log-HoneyPot1”
Once selected the interface looks very similar to the Log Analytics Workspace. Use the same code as before and select “RUN” Then from here also where it says “Visualization” select “MAP”

Figure 26

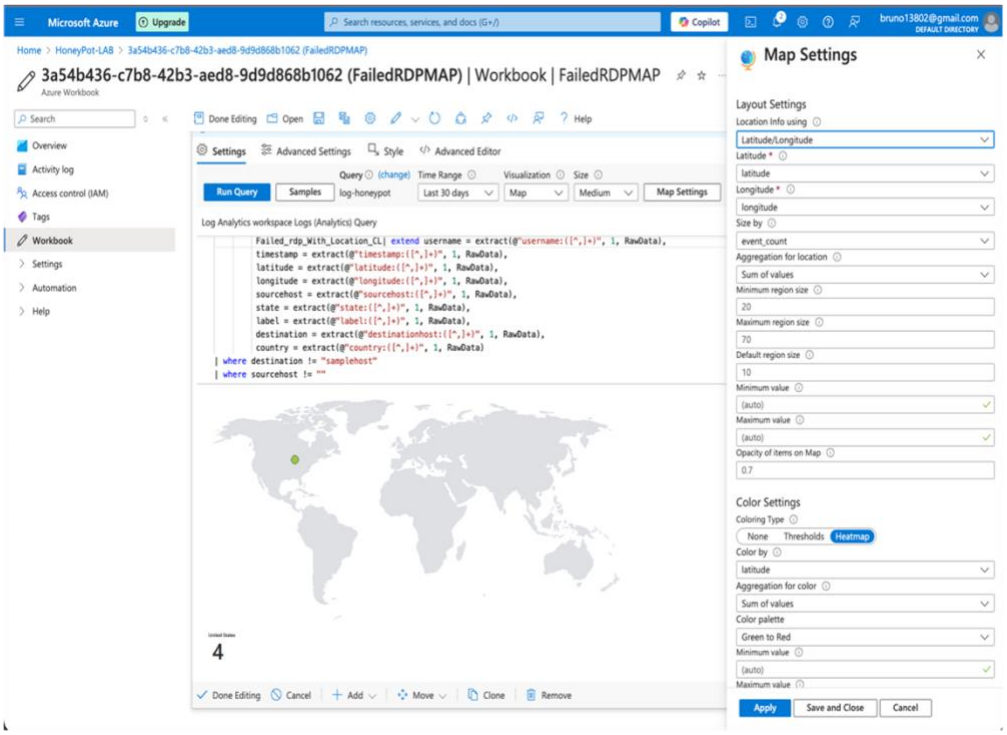


Now that we have selected map we can edit the setting to select what we want to see in the bottom, for example we can set the IP, Host County, and we get a little bubble that will grow as the more people attack from that place. We have to let this sit for a couple of days as we can only accept 1,000 API changes per day.

We will leave this running and return to it at a later date to see how the results are and where we are getting attacked from the most.

Now what I will do is repreat the process again so I have a second VM to grab further intel on the attackers.

Figure 27



Results

During my research project, I monitored two vulnerable VM's on Microsoft Azure to understand various attack methods.

One significant finding was the prevalence of brute force attacks, with repeated login attempts from IPs primarily originating in Russia, Poland, China, and the USA. Over a span of a couple of days, I detected 3,000 failed login attempts across two VM's using common usernames such as "admin," "test," and "administrator." By analyzing Windows Event Logs (Event ID 4625), I extracted these IPs and confirmed their origins using a geolocation API.

Another common attack observed was credential stuffing, where attackers used leaked username and password combinations like "admin:password123."

Figure 28

Map Of VM_1 After 12 Hours

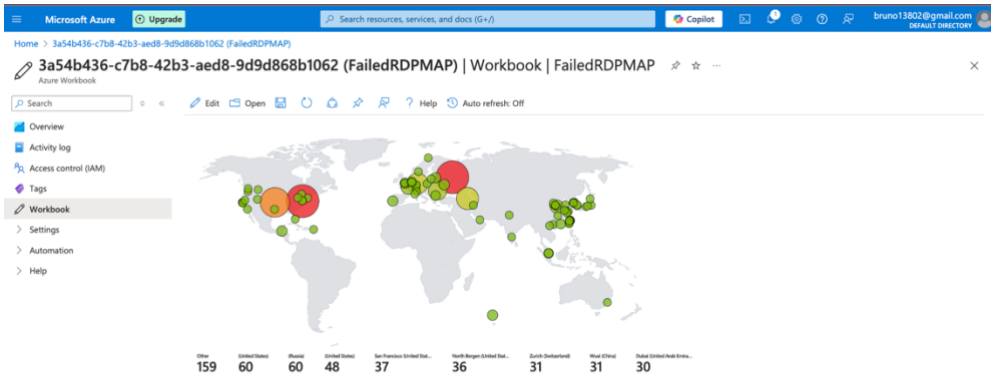
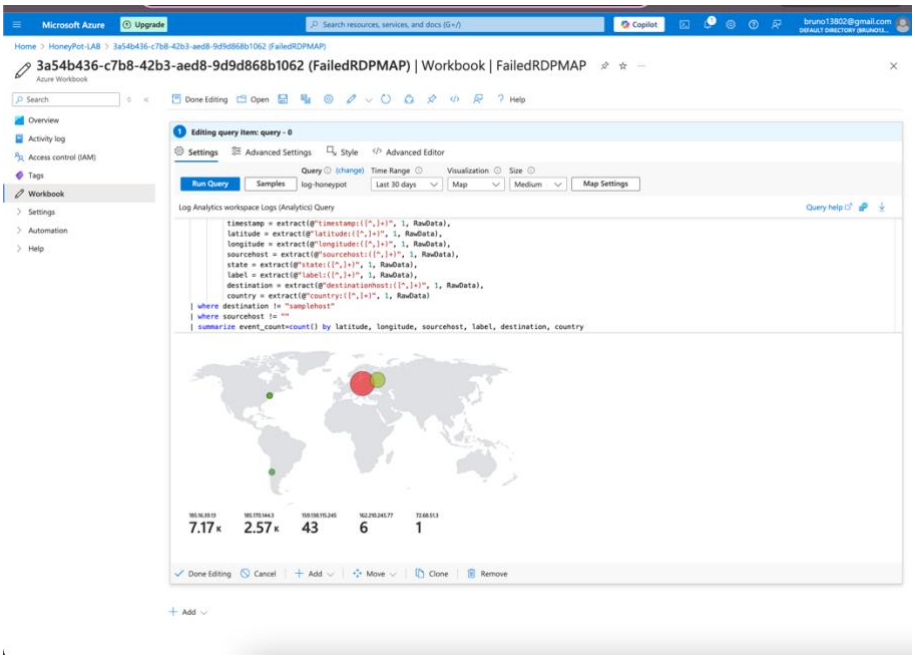


Figure 29

Map Of VM_2 After 3 Days



Additionally, leaving all ports open on the VM exposed frequent enumeration attempts, with attackers often from Chinese IPs scanning for open ports such as 22 (SSH), 80 (HTTP), and 3389 (RDP).

I also experienced denial-of-service (DoS) attacks, where excessive simultaneous login attempts caused resource exhaustion, temporarily rendering the VM unresponsive.

Luckily Azure's resource tools had allowed me to reveal significant CPU and memory spikes during these attack periods. These observations provided valuable insights into the nature of cyberattacks and the tools used by malicious actors.

Mitigation and Recommendations

Although the main objective of the honeypot setup in this project was to mimic an attack prone environment, a number of suggestions can be made to improve the system's overall effectiveness and strengthen the security posture.

First, isolating the honeypot using Virtual Networks and Network Security Groups is important to prevent attacks from spreading to other systems. Regular patching of any system exposed to the network, outside of the honeypot, minimizes the risk of real breaches through outdated services. Additionally, implementing an Intrusion Detection and Prevention System (IDPS) can bolster the security by monitoring and mitigating suspicious activities.

Comprehensive log management using tools like Azure Monitor and Sentinel allows for effective data collection and quick responses to threats, while integrating IP geolocation data helps in identifying the geographical sources of attacks. Moreover, adding a third party threat intelligence feeds can provide a deeper context for understanding emerging threats. Ethical and legal considerations should be observed, ensuring that data collection complies with privacy laws and does not encourage illegal activities. By following these recommendations, the honeypot setup becomes a more secure, functional, and valuable resource for cybersecurity research and learning

Another recommendation to enhance out honeypot setup is adding more Virtual Machines. This will be a great way to improve the realism and scale of our environment. By deploying multiple VMs with different operating systems and services, such as various Windows versions, Linux versions, or even IoT devices, we can simulate a more diverse network that reflects a real world scenario where attackers target a wide range of systems. This variety will

allow me to monitor and analyze different attack techniques and how attackers move across different devices and platforms. Additionally, leveraging Azure's auto scaling capabilities will help us environment scale based on traffic, ensuring that our honeypot can handle increased attack activity without overloading the system.

Conclusion

In this project, a mini honeynet was successfully constructed using Microsoft Azure, with log sources seamlessly integrated into a Log Analytics workspace. Microsoft Sentinel proved invaluable by triggering alerts, creating incidents based on ingested logs, and providing real time insights into attack patterns. During the project, metrics were systematically analyzed in the insecure environment to identify vulnerabilities and attack trends. Analyzing and recognizing attack patterns, as observed during this project, is crucial for understanding the tactics, techniques, and procedures used by malicious actors. By studying these patterns, I gained valuable insights into how attackers target vulnerable systems, enabling me to identify potential weaknesses in configurations and improve defensive strategies. For instance, identifying brute force attacks revealed the importance of implementing account lockout policies, strong password enforcement, and multi-factor authentication to mitigate such risks. Similarly, observing credential stuffing attempts underscored the significance of regularly auditing user credentials and ensuring they are not reused or compromised.

The enumeration of open ports and services highlighted the dangers of leaving unnecessary ports exposed, emphasizing the need for strict network segmentation, firewalls, and monitoring tools. Additionally, detecting denial of service attacks demonstrated how resource exhaustion could cripple a system, prompting me to explore rate limiting techniques and automated scaling solutions to maintain availability during high traffic events.

Beyond technical defenses, these patterns provided practical knowledge about attacker behavior, helping me anticipate their next moves and respond proactively. For example, correlating IP addresses with geographic locations can help identify hotspots for malicious activity, informing decisions about blocking certain regions or enhancing monitoring in those

areas. Ultimately, analyzing these attack patterns equips me with the skills and knowledge to design more robust systems, respond effectively to incidents, and stay ahead in the evolving cybersecurity landscape.

Once security controls were implemented, these metrics were re-evaluated, revealing a significant reduction in the number of security events and incidents. This outcome demonstrated the effectiveness of the applied security measures, showcasing the technical aspects of setting up and managing a honeypot, the project highlighted the importance of proactive monitoring and the role of Security Information and Event Management tools in modern cybersecurity. By leveraging Microsoft Sentinel, a deeper understanding of attacker behaviors, commonly exploited vulnerabilities, and defensive strategies was gained. This hands on experience not only advanced technical skills but also reinforced the importance of ethical and safe environments for cybersecurity research.

Ultimately, the project gained me knowledge in how accessible platforms like Microsoft Azure can empower students like me and professionals to explore, learn, and contribute to the vast field of cybersecurity.

Citations

Microsoft. *Azure Virtual Machines Pricing - Windows*. Microsoft, https://azure.microsoft.com/en-us/pricing/details/virtual-machines/windows/**.**

Microsoft. "Pricing." *Azure*, Microsoft, https://azure.microsoft.com/en-us/pricing/**.**

Microsoft. *Connect to a Windows virtual machine using RDP*. Microsoft, 2024, https://learn.microsoft.com/en-us/azure/virtual-machines/windows/connect-rdp**.**

"Introduction to Azure Sentinel." *Microsoft Learn*, Microsoft, https://learn.microsoft.com/en-us/training/modules/intro-to-azure-sentinel/**.**

IP Geolocation. *IP Geolocation API*. IP Geolocation, https://ipgeolocation.io/ip-location-api.html. Accessed 10 Dec. 2024**.**

Microsoft. "Kusto Query Language." *Microsoft Learn*, Microsoft, https://learn.microsoft.com/en-us/kusto/query/?view=microsoft-fabric**.**

Madakor, Josh. *Josh Madakor Cyber Course*. LogN Pacific, 2024, https://joshmadakor.tech/cyber/**.**

Code & Programs

Query Log -

```
      (CUSTOM LOG NAME)
 |extend username = extract(@"username:([^,]+)", 1, RawData),
      timestamp = extract(@"timestamp:([^,]+)", 1, RawData),
      latitude = extract(@"latitude:([^,]+)", 1, RawData),
      longitude = extract(@"longitude:([^,]+)", 1, RawData),
      sourcehost = extract(@"sourcehost:([^,]+)", 1, RawData),
      state = extract(@"state:([^,]+)", 1, RawData),
      label = extract(@"label:([^,]+)", 1, RawData),
      destination = extract(@"destinationhost:([^,]+)", 1, RawData),
      country = extract(@"country:([^,]+)", 1, RawData)
 |where destination != "samplehost"
 |where sourcehost != ""
 |summarize event_count=count() by timestamp, label, country, state,
sourcehost, username, destination, longitude, latitude
```

PowerShell Log Exporter –

```
# Get API key from here: https://ipgeolocation.io/
$API_KEY      = "ENTER_KEY_HERE"
$LOGFILE_NAME = "failed_rdp.log"
$LOGFILE_PATH = "C:\ProgramData\$($LOGFILE_NAME)"

# This filter will be used to filter failed RDP events from Windows Event Viewer
$XMLFilter = @'
<QueryList>
  <Query Id="0" Path="Security">
      <Select Path="Security">
          *[System[(EventID='4625')]]
       </Select>
   </Query>
</QueryList>
'@

<#
   This function creates a bunch of sample log files that will be used to train the
```

```
    Extract feature in Log Analytics workspace. If you don't have enough log files to
    "train" it, it will fail to extract certain fields for some reason.
    We can avoid including these fake records on our map by filtering out all logs
with
    a destination host of "samplehost"
#>
Function write-Sample-Log() {
    "latitude:47.91542,longitude:-
120.60306,destinationhost:samplehost,username:fakeuser,sourcehost:24.16.97.222,
state:Washington,country:United States,label:United States -
24.16.97.222,timestamp:2021-10-26 03:28:29" | Out-File $LOGFILE_PATH -
Append -Encoding utf8
    "latitude:-22.90906,longitude:-
47.06455,destinationhost:samplehost,username:lnwbaq,sourcehost:20.195.228.49,s
tate:Sao Paulo,country:Brazil,label:Brazil - 20.195.228.49,timestamp:2021-10-26
05:46:20" | Out-File $LOGFILE_PATH -Append -Encoding utf8

"latitude:52.37022,longitude:4.89517,destinationhost:samplehost,username:CSNY
DER,sourcehost:89.248.165.74,state:North
Holland,country:Netherlands,label:Netherlands - 89.248.165.74,timestamp:2021-
10-26 06:12:56" | Out-File $LOGFILE_PATH -Append -Encoding utf8
    "latitude:40.71455,longitude:-
74.00714,destinationhost:samplehost,username:ADMINISTRATOR,sourcehost:72.
45.247.218,state:New York,country:United States,label:United States -
72.45.247.218,timestamp:2021-10-26 10:44:07" | Out-File $LOGFILE_PATH -
Append -Encoding utf8
    "latitude:33.99762,longitude:-
6.84737,destinationhost:samplehost,username:AZUREUSER,sourcehost:102.50.24
2.216,state:Rabat-Salé-Kénitra,country:Morocco,label:Morocco -
102.50.242.216,timestamp:2021-10-26 11:03:13" | Out-File $LOGFILE_PATH -
Append -Encoding utf8
    "latitude:-
5.32558,longitude:100.28595,destinationhost:samplehost,username:Test,sourcehost
:42.1.62.34,state:Penang,country:Malaysia,label:Malaysia -
42.1.62.34,timestamp:2021-10-26 11:04:45" | Out-File $LOGFILE_PATH -
Append -Encoding utf8

"latitude:41.05722,longitude:28.84926,destinationhost:samplehost,username:AZU
REUSER,sourcehost:176.235.196.111,state:Istanbul,country:Turkey,label:Turkey -
```

```
176.235.196.111,timestamp:2021-10-26 11:50:47" | Out-File $LOGFILE_PATH -
Append -Encoding utf8

"latitude:55.87925,longitude:37.54691,destinationhost:samplehost,username:Test,s
ourcehost:87.251.67.98,state:null,country:Russia,label:Russia -
87.251.67.98,timestamp:2021-10-26 12:13:45" | Out-File $LOGFILE_PATH -
Append -Encoding utf8

"latitude:52.37018,longitude:4.87324,destinationhost:samplehost,username:AZUR
EUSER,sourcehost:20.86.161.127,state:North
Holland,country:Netherlands,label:Netherlands - 20.86.161.127,timestamp:2021-
10-26 12:33:46" | Out-File $LOGFILE_PATH -Append -Encoding utf8
   "latitude:17.49163,longitude:-
88.18704,destinationhost:samplehost,username:Test,sourcehost:45.227.254.8,state:
null,country:Belize,label:Belize - 45.227.254.8,timestamp:2021-10-26 13:13:25" |
Out-File $LOGFILE_PATH -Append -Encoding utf8
   "latitude:-
55.88802,longitude:37.65136,destinationhost:samplehost,username:Test,sourcehost
:94.232.47.130,state:Central Federal District,country:Russia,label:Russia -
94.232.47.130,timestamp:2021-10-26 14:25:33" | Out-File $LOGFILE_PATH -
Append -Encoding utf8
}

# This block of code will create the log file if it doesn't already exist
if ((Test-Path $LOGFILE_PATH) -eq $false) {
   New-Item -ItemType File -Path $LOGFILE_PATH
   write-Sample-Log
}

# Infinite Loop that keeps checking the Event Viewer logs.
while ($true)
{

   Start-Sleep -Seconds 1
   # This retrieves events from Windows EVent Viewer based on the filter
   $events = Get-WinEvent -FilterXml $XMLFilter -ErrorAction SilentlyContinue
   if ($Error) {
      #Write-Host "No Failed Logons found. Re-run script when a login has failed."
   }
```

```
    # Step through each event collected, get geolocation
    #    for the IP Address, and add new events to the custom log
    foreach ($event in $events) {


        # $event.properties[19] is the source IP address of the failed logon
        # This if-statement will proceed if the IP address exists (>= 5 is arbitrary, just
saying if it's not empty)
        if ($event.properties[19].Value.Length -ge 5) {

            # Pick out fields from the event. These will be inserted into our new custom
log
            $timestamp = $event.TimeCreated
            $year = $event.TimeCreated.Year

            $month = $event.TimeCreated.Month
            if ("$($event.TimeCreated.Month)".Length -eq 1) {
               $month = "0$($event.TimeCreated.Month)"
            }

            $day = $event.TimeCreated.Day
            if ("$($event.TimeCreated.Day)".Length -eq 1) {
               $day = "0$($event.TimeCreated.Day)"
            }

            $hour = $event.TimeCreated.Hour
            if ("$($event.TimeCreated.Hour)".Length -eq 1) {
               $hour = "0$($event.TimeCreated.Hour)"
            }

            $minute = $event.TimeCreated.Minute
            if ("$($event.TimeCreated.Minute)".Length -eq 1) {
               $minute = "0$($event.TimeCreated.Minute)"
            }


            $second = $event.TimeCreated.Second
            if ("$($event.TimeCreated.Second)".Length -eq 1) {
               $second = "0$($event.TimeCreated.Second)"
            }
```

```
        $timestamp = "$($year)-$($month)-$($day)
$($hour):$($minute):$($second)"
        $eventId = $event.Id
        $destinationHost = $event.MachineName# Workstation Name (Destination)
        $username = $event.properties[5].Value # Account Name (Attempted
Logon)
        $sourceHost = $event.properties[11].Value # Workstation Name (Source)
        $sourceIp = $event.properties[19].Value # IP Address


        # Get the current contents of the Log file!
        $log_contents = Get-Content -Path $LOGFILE_PATH

        # Do not write to the log file if the log already exists.
        if (-Not ($log_contents -match "$($timestamp)") -or ($log_contents.Length
-eq 0)) {

            # Announce the gathering of geolocation data and pause for a second as
to not rate-limit the API
            #Write-Host "Getting Latitude and Longitude from IP Address and
writing to log" -ForegroundColor Yellow -BackgroundColor Black
            Start-Sleep -Seconds 1

            # Make web request to the geolocation API
            # For more info: https://ipgeolocation.io/documentation/ip-geolocation-
api.html
            $API_ENDPOINT =
"https://api.ipgeolocation.io/ipgeo?apiKey=$($API_KEY)&ip=$($sourceIp)"
            $response = Invoke-WebRequest -UseBasicParsing -Uri
$API_ENDPOINT

            # Pull Data from the API response, and store them in variables
            $responseData = $response.Content | ConvertFrom-Json
            $latitude = $responseData.latitude
            $longitude = $responseData.longitude
            $state_prov = $responseData.state_prov
            if ($state_prov -eq "") { $state_prov = "null" }
            $country = $responseData.country_name
            if ($country -eq "") {$country -eq "null"}
```

```
            # Write all gathered data to the custom log file. It will look something
like this:
            #

"latitude:$($latitude),longitude:$($longitude),destinationhost:$($destinationHost),u
sername:$($username),sourcehost:$($sourceIp),state:$($state_prov),
country:$($country),label:$($country) - $($sourceIp),timestamp:$($timestamp)" |
Out-File $LOGFILE_PATH -Append -Encoding utf8

            Write-Host -BackgroundColor Black -ForegroundColor Magenta
"latitude:$($latitude),longitude:$($longitude),destinationhost:$($destinationHost),u
sername:$($username),sourcehost:$($sourceIp),state:$($state_prov),label:$($count
ry) - $($sourceIp),timestamp:$($timestamp)"
          }
          else {
            # Entry already exists in custom log file. Do nothing, optionally, remove
the # from the line below for output
            # Write-Host "Event already exists in the custom log. Skipping." -
ForegroundColor Gray -BackgroundColor Black
          }
        }
    }
}
```