**Mastering Multi-Cloud**

**Labs: Security, Identity, and Compliance in Multi-Cloud**

## Lab 01: Azure Entra ID

**Case Study**

Washington County Public Schools (WCPS) is a large school district in Maryland, USA, managing 46 schools and serving more than 24,000 students with over 3,700 staff members. The district relied on multiple applications from different vendors, including Microsoft 365, Google Workspace, and classroom management tools, each with its own login system and identity store. This fragmented approach made staff authentication and application access difficult to manage and increased the administrative overhead of maintaining separate credentials and access permissions across platforms. To modernize its identity infrastructure, improve security, and simplify user access, WCPS partnered with Data Networks to migrate to Microsoft Entra ID, formerly Azure Active Directory. This implementation consolidated identity and access management in a centralized cloud platform, improving security, reducing password-related support calls, and enabling users to access multiple systems using a single identity.

**Business Challenge**

Before adopting Microsoft Entra ID, Washington County Public Schools (WCPS) struggled with multiple independent authentication systems. Staff and students managed separate credentials for different applications, confusing, frequent password resets, and reduced productivity.

IT teams faced high administrative overhead, as user provisioning and access management were handled manually across multiple systems, increasing the risk of errors and operational complexity.

Security was also a concern, with no centralized access control or Multifactor Authentication (MFA), leaving the district vulnerable to unauthorized access and potential data breaches. These challenges highlighted the need for a unified and secure identity management solution.

**Solution**

To address these challenges, WCPS implemented Microsoft Entra ID as a centralized identity and access management solution. Entra ID consolidated all user accounts into a single directory, providing a unified platform for authentication across applications and services. This allowed staff and students to access multiple systems with one set of credentials, improving productivity and reducing the number of password-related support requests.

The district also introduced Single Sign-On (SSO) and Multifactor Authentication (MFA) through Entra ID. SSO simplified the login process for users, while MFA strengthened security by requiring additional verification during sign-in. These measures ensured consistent identity policies and reduced the risk of unauthorized access.

In addition, Entra ID streamlined administrative tasks for the IT team. User provisioning, access updates, and deprovisioning could now be managed from a centralized dashboard, reducing manual effort and minimizing the risk of errors. Overall, the adoption of Microsoft Entra ID enhanced operational efficiency, strengthened security, and provided WCPS with a scalable and secure identity management platform for its staff and students.

The following steps will guide you through creating a user in Azure Entra ID and exploring its key features:

|  |
| --- |
| **Task 1: Create Azure Entra ID**   1. Log in to the **Microsoft Azure** portal and go to the portal menu.      1. Click on **Microsoft Entra ID** from the portal menu.      1. Now, click on **Users** from the left sidebar of the default directory.      1. The **Users** tab displays the list of all users currently in the tenant.      1. To add a new user to this active directory, click on the **+ New user** tab.      1. Write the username, first name, and last name of your choice.      1. After filling out the names section, click on **Auto-generated password** in the **Password** option.      1. Then, click **Create.**      1. After clicking **Create,** the notification of **Successfully created user** will appear.      1. Now, click on **New\_user** to see the entered details. |

**Lab 02: Interaction with KMS using the Command Line**

**Case Study**

Capital One is a major U.S.-based financial services company providing credit cards, banking, and loan services to millions of customers. As part of its digital transformation strategy, Capital One moved the majority of its infrastructure and applications to Amazon Web Services (AWS). This cloud-first approach allowed the company to improve scalability, accelerate innovation, and modernize legacy systems.

Because Capital One handles highly sensitive financial and personal data, security was a top priority throughout its cloud migration. Encryption of data at rest and in transit, along with centralized control of cryptographic keys, became a core requirement. To support secure cloud operations at scale, Capital One implemented AWS-native security services, including AWS Key Management Service (KMS), and integrated key management into automated workflows.

**Business Challenge**

Migrating critical banking workloads to the cloud introduced complex security and compliance challenges. Capital One needed to ensure that sensitive customer data remained protected under strict regulatory requirements while supporting a rapidly changing cloud environment. Managing encryption keys manually across numerous applications and services increases the risk of inconsistent security controls and potential misconfigurations.

Additionally, the organization relied heavily on automation and DevOps practices. Manual key creation and management through web consoles did not align with their infrastructure-as-code approach. Security teams required a repeatable, auditable, and programmatic method to create, rotate, and control encryption keys without slowing down development pipelines or increasing operational overhead.

Meeting regulatory compliance requirements also demanded detailed visibility into key usage and access. Without centralized logging and automated governance, producing audit evidence would have been time-consuming and error-prone.

**Solution**

To address these challenges, Capital One adopted AWS Key Management Service (KMS) as a centralized solution for encryption key lifecycle management. AWS KMS enabled the company to create and manage customer-managed encryption keys, enforce fine-grained access controls, and integrate encryption seamlessly with AWS services such as Amazon S3, Amazon EBS, and Amazon RDS.

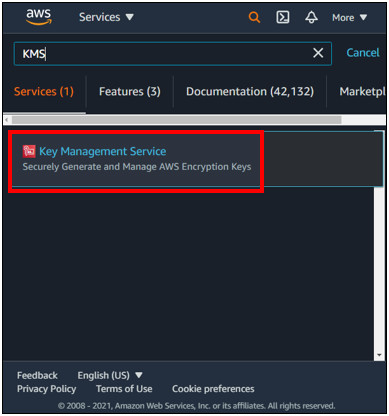
Capital One incorporated KMS operations into automated workflows using command-line tools and infrastructure-as-code practices. By interacting with KMS programmatically, the organization ensured consistent encryption policies across all environments. Automatic key rotation and strict IAM policies reduced the risk of key compromise and unauthorized access.

All key-related actions were logged through AWS CloudTrail, providing full audit visibility and supporting compliance reporting. This centralized, automated approach to encryption and key management strengthened Capital One’s overall security posture while enabling secure, scalable cloud operations.

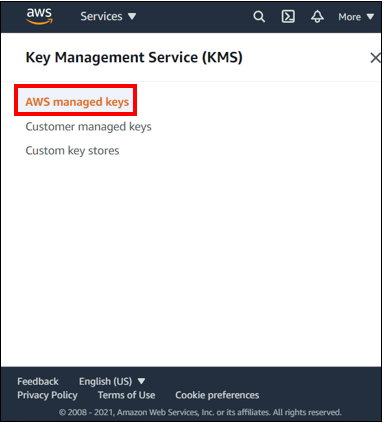
The following steps will guide you through implementing AWS KMS operations using the console and command line to create, manage, and use encryption keys securely.

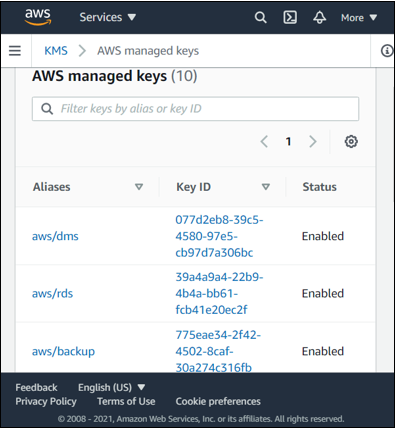
1. Log in to the **AWS Console.**

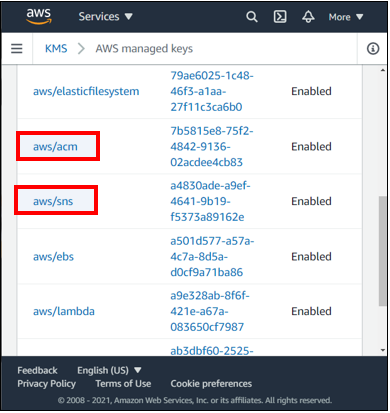
2. Go to **Services** and click **Key Management Service.**



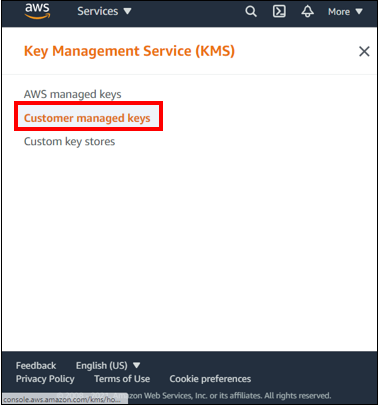
3. Click **AWS Managed Keys**.





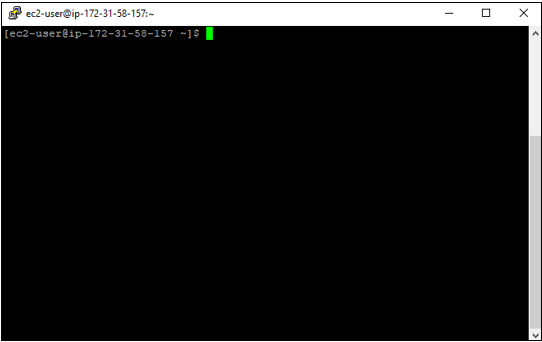


4. Click on **Customer Managed Keys**.

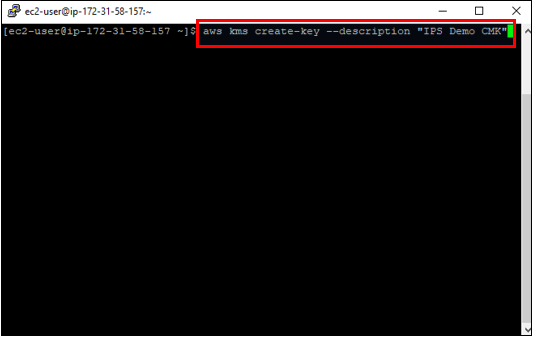


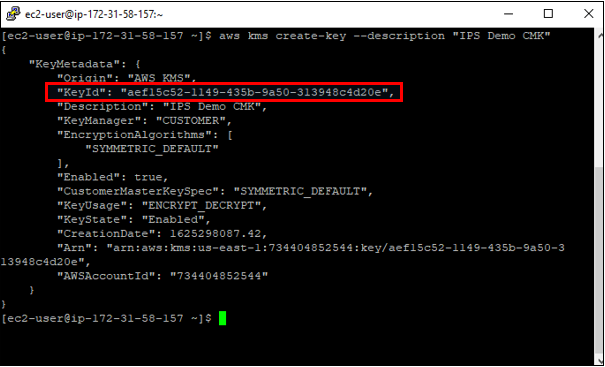
5. Since we do not have any customer-managed keys, we will go to the command line and create a key.

6. We have a terminal with an EC2 instance configured with the AWS configured command. This EC2 instance has full administrative access to the AWS account.



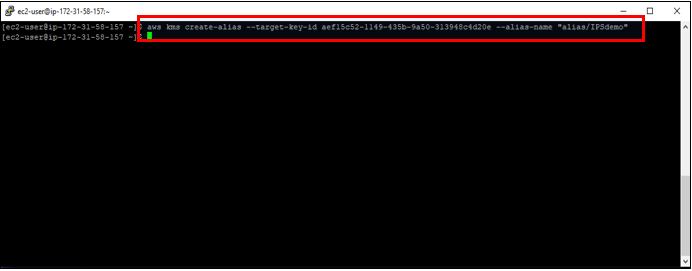
7. The first command is: **aws kms create-key --description “IPS Demo CMK.”** This command returns a number of values. Copy the **KeyId** into the clipboard.



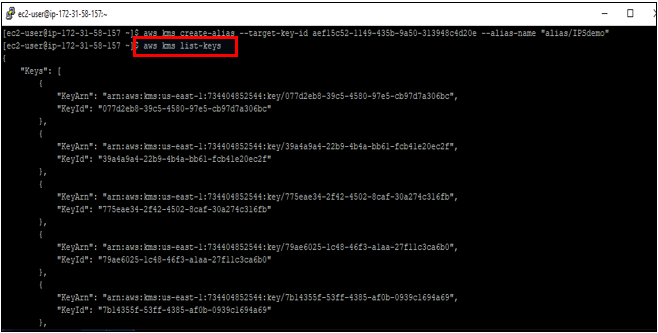


8. The command to create an alias is: **aws kms create-alias --target-key-id [KeyId] --alias-name “alias/IPSdemo.”**

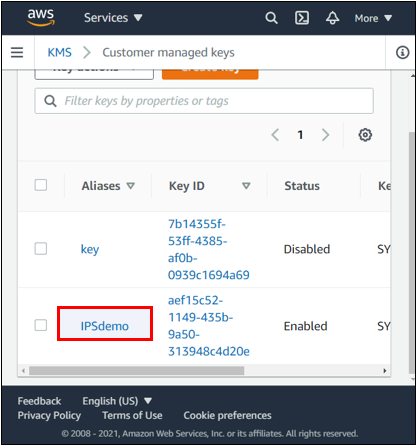
We need to specify a target key ID. This is the key ID returned by the previous command. This command also gives a name to the alias created.



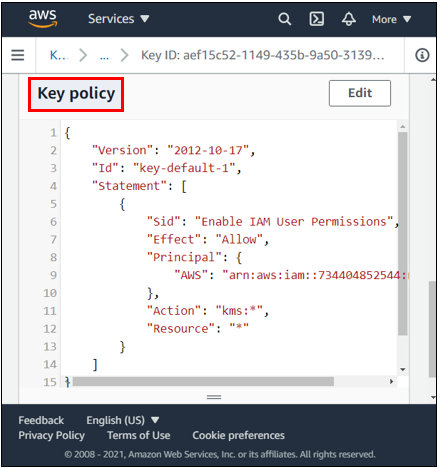
9. Next command is **aws kms list-keys.** This will list the keys and also the key we created.



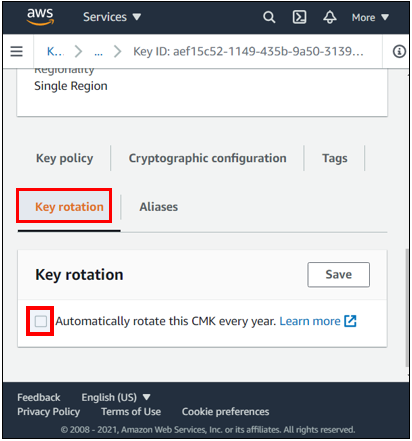
10. Go back to the AWS console. You can see the **IPSdemo** alias created here.



11. Click on the **IPSdemo** alias to view the **Key Policy**.



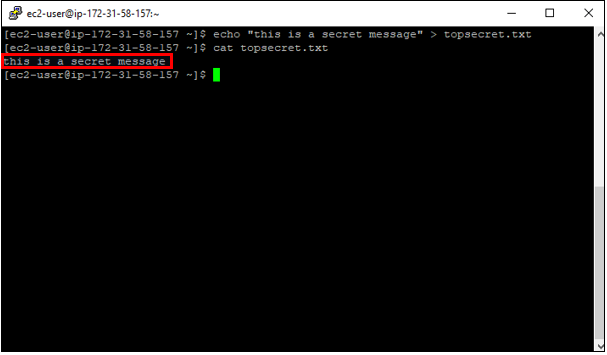
12. If you click on **key rotation**, you can also specify a rotation policy. If you check the box, this will automatically rotate the CMK yearly. AWS-managed keys, they are turned in by AWS every three years.



13. Go back to the command line. We will start using the key we have created. We will write out the string to encrypt a text file: **this is a secret message** to a file called top **secret.txt**.

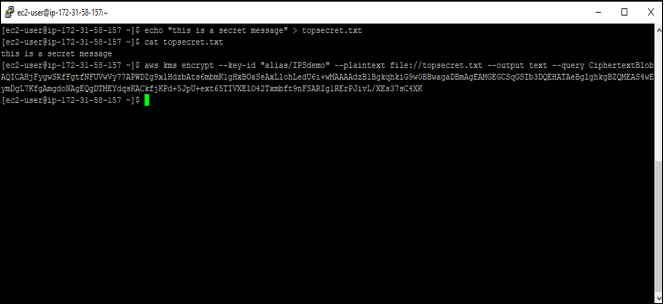
The command to do this is: **echo “this is a secret message” > topsecret.txt**

To view the contents of the file: **cat topsecret.txt**



14. Next command is: **aws kms encrypt --key-id “alias/IPSdemo” --plaintext file://topsecret.txt --output text --query CiphertextBlob**

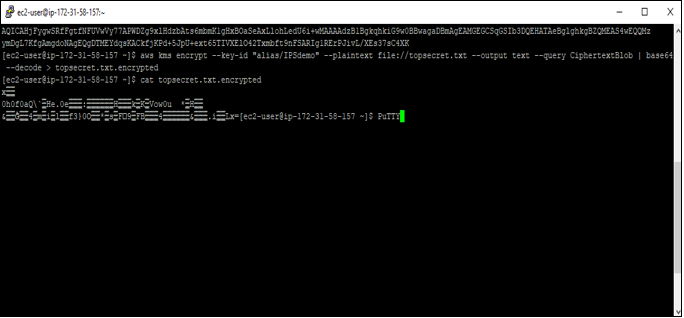
In this command, we specify plain text because we are using a local plain text file. We want our output in text format and to get back the value called ciphertext blog. This command will encrypt the topsecret.txt file using the IPSdemo CMK we created in the previous step, and it will output the ciphertext blob in base 64 encoded form.



15. Now, we will decode this encrypted blob base 64 and save the raw encrypted binary data to a local file.

The command to decode is: **aws kms encrypt --key-id “alias/IPSdemo” --plaintext file://topsecret.txt --output text --query CiphertextBlob | base64 --decode > topsecret.txt.encrypted**

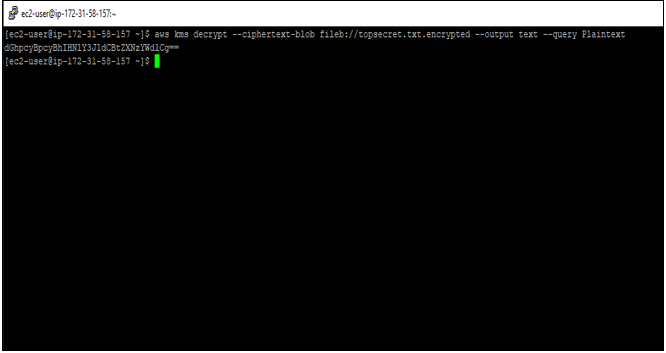
To view the contents of this file: **cat topsecret.txt. encrypted**



16. Next, we are going to decrypt this file using our CMK.

**aws kms decrypt --ciphertext-blob fileb://topsecret.txt.encrypted --output text --query Plaintext**

In this command, we are using **fileb** instead of a file.

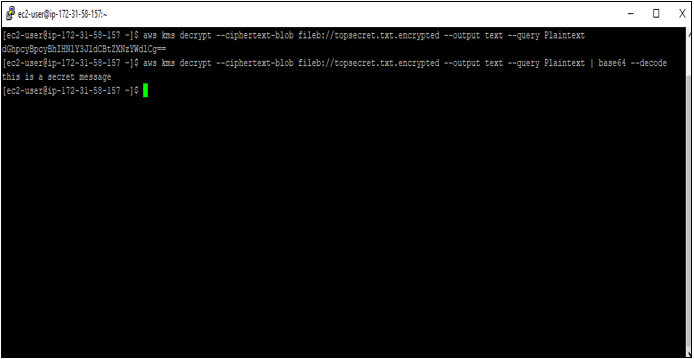


Since every time we work with KMS at the command line, we will be working with base 64 encoded data. Therefore, we need to decode it.

17. The command to decode is:

**aws kms decrypt --ciphertext-blob fileb://topsecret.txt.encrypted --output text --query Plaintext | base64 --decode**

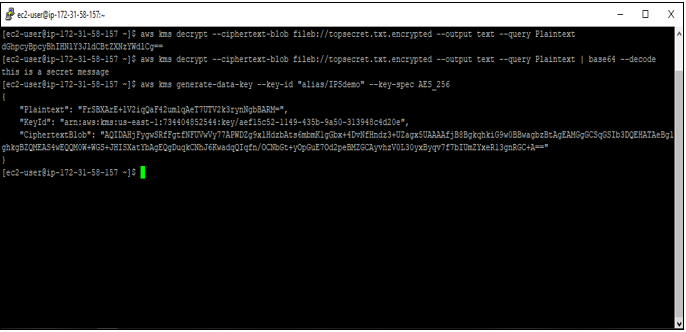
This gives us decrypted data.



CMKs are used to encrypt and decrypt data up to four kilobytes in size. If you want to encrypt a larger data file than four kilobytes, you can use a data encryption key or DEK.

**aws kms generate-data-key --key-id “alias/IPSdemo” --key-spec AES\_256**

This returns the plain text data key and an encrypted one with the specified CMK version of the data key. The encrypted version is referred to as a ciphertext blob.



The ciphertext blob has metadata that tells KMS which CMK was used to generate it.

While KMS supports sending data up to four kilobytes to be encrypted directly by KMS, envelope encryption can offer significant performance benefits.

**Lab 03: Threat Hunting**

**Case Study**

Shell is a global energy company operating large scale digital and industrial environments across more than seventy countries. The organization relies heavily on cloud platforms, enterprise applications, operational technology systems, and global networks to support its energy production, supply chain, and corporate operations. Due to the critical nature of its infrastructure, Shell places strong emphasis on cybersecurity, operational continuity, and risk management across its global technology environment.

As Shell’s digital footprint expanded, the company began collecting vast amounts of security telemetry generated from endpoints, servers, cloud workloads, and network devices. While traditional security monitoring tools provided alert-based detection, Shell recognized that many advanced threats could bypass automated alerts and remain hidden for long periods. To improve visibility and proactively identify suspicious activity, Shell adopted a threat hunting approach supported by cloud-based security analytics.

**Business Challenge**

Shell faced challenges related to limited visibility and reactive security monitoring. Existing security tools primarily relied on predefined alerts, which made it difficult to detect unknown threats, slow moving attacks, and attacker behavior that did not immediately trigger alarms. Security analysts were often overwhelmed by alert volume and could not efficiently investigate historical data across multiple systems.

Additionally, storing and analyzing large volumes of security logs on premises was costly and operationally complex. Shell required a scalable solution that could centralize log collection, support advanced search and correlation, and enable analysts to proactively hunt for threats across the environment. The organization needed security professionals who could use threat hunting techniques to investigate suspicious activity and identify indicators of compromise before incidents escalated.

Assume you have been recruited as a Security Analyst at Shell, responsible for conducting threat hunting activities using Microsoft Sentinel to identify suspicious behavior, investigate potential command and control activity, and improve the organization’s overall security posture.

**Solution**

In this lab, Microsoft Sentinel is used to demonstrate how Shell can perform proactive threat hunting using cloud-based security analytics. A Windows virtual machine is deployed in Azure to simulate an enterprise endpoint. Security logs from the virtual machine are collected using Microsoft Sentinel data connectors and stored in a Log Analytics workspace.

Simulated attack activities are performed on the virtual machine to generate security events such as persistence mechanisms, suspicious PowerShell execution, DNS-based command and control behavior, and unauthorized account creation. These activities generate telemetry that can later be analyzed during the threat hunting process.

Using Microsoft Sentinel, threat hunting queries are created with Kusto Query Language to identify suspicious PowerShell activity and potential command and control behavior. Hunting queries are then saved, bookmarked, and investigated using Microsoft Sentinel’s investigation graph and entity mapping features. This lab demonstrates how threat hunting enables security teams to move beyond reactive alerting and proactively identify hidden threats within the environment.

Microsoft Sentinel is a cloud native security information and event management and security orchestration platform that enables organizations to collect, analyze, and investigate security data at scale. It provides advanced hunting, investigation, and visualization capabilities that support modern security operations in large enterprises such as Shell.

The tasks performed in this lab include:

* Create a Windows virtual machine in Azure
* Connect Windows security logs to Microsoft Sentinel
* Simulate attack activity to generate security events
* Perform threat hunting using KQL queries
* Investigate suspicious activity using Microsoft Sentinel

|  |
| --- |
| **Step 1: Create a VM in Azure**  1. Click on the **Virtual Machines**.    2. Click on the **Creat**e.    3. Click on the **Azure Virtual Machines**.    4. Select **vanguardinvest** subscription. Create a new resource group, **Sentinel-Incident-Hunting**. Enter the name of virtual machine **Incident-Hunt-VM**. Select a region **(US) East US**. Select **Availability Zone** in availability options.    5. Scroll down. Select **Standard** in security type. Select **Windows Server 2022 Datacenter: Azure Edition – x64 Gen2** image. Select VM architecture **x64**.    6. Scroll down and select **Standard\_DS1\_v2 – 1 vcpu, 3.5 GiB memory ($86.87/month)** in size.    7. Scroll down and enter **username: azureuser**. Then, set your **Password** and **Confirm password**.    8. Select **Allow selected ports** radio button. Then select **HTTP (80), HTTPS (443), SSH (22), RDP (3389)** in inbound ports. Click on the **Review + Create** button.    9. After the validation has passed, click on the **Create** button.    10. It takes a few seconds to create a VM. After the VM is created, you should see that **Your deployment is complete**. |
| **Step 2: Connect Windows Devices to Microsoft Sentinel Using Data Connectors**  1. Click on **Microsoft Sentinel**.    2. Select **Sentinel-LAW** log analytics workspace.    3. Click on the **Data connectors** under **Configuration**.    4. In the search bar, type **Windows Security Event**. Then select **Windows Security Events via AMA**.    5. Click on the **Open connector page** button.    6. Click on the +**Create data collection rule** button.    7. Enter the rule name **Incident-Hunt-Rule**. Select **vanguardinvest** subscription. Select resource group **Sentinel-Incident-Hunting**.    8. Select the **Incident-Hunt-VM**. Then click on the **Next: Collect >**.    9. Click on the **Next: Review + create >** button.    10. Click on the **Create** button.    11. Click on the **Refresh**.    12. Hence, successfully connected Windows VM to Microsoft Sentinel using data connectors. Now, Microsoft Sentinel receives logs and data from Windows VM. |
| **Step 3: Conduct Attacks**  1. Go to the **Incident-Hunt-VM** virtual machine. Click on the **Connect** dropdown. Then click on the **Connect**.    2. Click on the **Download RDP file** to access this VM.    3. Click on the **Save** button to save this RDP file.    4. Double-click on the **Incident-Hunt-VM** RDP file to open.    5. Click on the **Connect** button.    6. Enter the **Password** you set. Then click on the **OK** button.    7. Click on the **Yes** button.    8. Hence, successfully access the VM via RDP.    9. In the search bar, type **Command Prompt**. Then click on **Run as administrator**.    10. Execute the commands below one by one. To create a Temp folder in the root directory and to simulate program persistence.   |  | | --- | | cd \  mkdir temp  cd temp  REG ADD "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /V "SOC Test" /t REG\_SZ /F /D "C:\temp\startup.bat" |     11. Execute the following command: **notepad c2.ps1** to create a script that will simulate a DNS query to a C2 server    12. Click on the **Yes** button to create a new file.    13. Copy and paste the PowerShell script provided below into **c2.ps1** file. Click on **File** from the toolbar and then click on **Save**.   |  | | --- | | param(  [string]$Domain = "microsoft.com",  [string]$Subdomain = "subdomain",  [string]$Sub2domain = "sub2domain",  [string]$Sub3domain = "sub3domain",  [string]$QueryType = "TXT",  [int]$C2Interval = 8,  [int]$C2Jitter = 20,  [int]$RunTime = 240  )  $RunStart = Get-Date  $RunEnd = $RunStart.addminutes($RunTime)  $x2 = 1  $x3 = 1  Do {  $TimeNow = Get-Date  Resolve-DnsName -type $QueryType $Subdomain".$(Get-Random -Minimum 1 -Maximum 999999)."$Domain -QuickTimeout  if ($x2 -eq 3 )  {  Resolve-DnsName -type $QueryType $Sub2domain".$(Get-Random -Minimum 1 -Maximum 999999)."$Domain -QuickTimeout  $x2 = 1  }  else  {  $x2 = $x2 + 1  }  if ($x3 -eq 7 )  {  Resolve-DnsName -type $QueryType $Sub3domain".$(Get-Random -Minimum 1 -Maximum 999999)."$Domain -QuickTimeout  $x3 = 1  }  else  {  $x3 = $x3 + 1  }  $Jitter = ((Get-Random -Minimum -$C2Jitter -Maximum $C2Jitter) / 100 + 1) +$C2Interval  Start-Sleep -Seconds $Jitter  }  Until ($TimeNow -ge $RunEnd) |     14. Return to the command prompt window and execute the following command: **Start PowerShell.exe -file c2.ps1.**    15. You will see DNS resolve errors. Do not close these windows. Let this PowerShell script run in the background. The command needs to generate log entries for some hours. The data created by this task will be used later in Threat Hunting. This process will not create substantial amounts of data or processing.    16. Execute the commands below one by one to simulate the creation of an Admin account.   |  | | --- | | net user theusernametoadd /add  net user theusernametoadd ThePassword1!  net localgroup administrators theusernametoadd /add | |
| **Step 4: Threat Hunting in Microsoft Sentinel**  1. Click on **Microsoft Sentinel**.    2. Click on the **Sentinel-LAB** log analytics workspace.    3. Click on the **Logs** under **General**.    4. Click on **X** icon to close the **Queries hub** window.    5. Copy and paste below-provided below KQL query in the code editor. Then click on the **Run** button.   |  | | --- | | let lookback = 2d;  SecurityEvent  | where TimeGenerated >= ago(lookback)  | where EventID == 4688 and Process =~ "powershell.exe"  | extend PwshParam = trim(@"[^/\\]\*powershell(.exe)+" , CommandLine)  | project TimeGenerated, Computer, SubjectUserName, PwshParam  | summarize min(TimeGenerated), count() by Computer, SubjectUserName, PwshParam  | order by count\_ desc nulls last |     6. Review the different results. You have now identified PowerShell requests that are running in your environment.    7. Click on **X** icon to close the **Logs** window.    8. Click on the **Hunting** under **Threat Management**.    9. Click on the **+ New query**.    10. Enter the name of query **C2 Hunt**. In the query box, copy and paste below-provided KQL query.   |  | | --- | | let lookback = 2d;  SecurityEvent  | where TimeGenerated >= ago(lookback)  | where EventID == 4688 and Process =~ "powershell.exe"  | extend PwshParam = trim(@"[^/\\]\*powershell(.exe)+" , CommandLine)  | project TimeGenerated, Computer, SubjectUserName, PwshParam  | summarize min(TimeGenerated), count() by Computer, SubjectUserName, PwshParam  | order by count\_ desc nulls last |       11. Scroll down, and under **Entity mapping,** select **Host** in the **Entity type** drop-down list. For the **Identifier** drop-down list, select **HostName**. For the **Value** drop-down list, select **Computer**. Under **Tactics & Techniques,** select **Command.** Then, click on the **Create** button to create the hunting query.    12. Successfully created a threat hunting query in Microsoft Sentinel. Now, we will create a bookmark to investigate this event further.    13. Select the **C2 Hunt** threat hunting query. Click on the **View Results** button.    14. Click on the **Add bookmark**.    15. Enter the bookmark name **C2 Hunt Bookmark**. Scroll down and, under **Entity mapping**, select Host from the **Entity type** drop-down list. For the **Identifier** drop-down list, select **HostName**. For the **Value** drop-down list, select **Host\_0\_HostName**. Under **Tactics & Techniques,** select **Command.** Then, click on the **Create** button to create the hunting query.    16. Click **X** icon to close the window.    17. Click on the **Bookmarks** tab under **Hunting**. Select the **C2 Hunt Bookmark.** Click on **<< icon** on the right-hand side to expand the window.    18. Click on the **Investigate** button.    19. It shows the investigation graph to investigate how this event occurred and take further actions in the future to mitigate this type of event or incident in Microsoft Sentinel. Hover over the **Incident-Hunt-VM**. Then, click on the **Parent Process** running on the host to check what processes are running on the VM.    20. You will investigate the many malicious executable files running on the host that generate incidents and events. You can see in the screenshot below that powershell.exe and cmd.exe are generating events as we created earlier in this lab.    21. On the right side, click the **Timeline** to see when an event occurred.    22. Click on the **Entities** to do further investigation.    23. Close all the windows. Click on the **Queries** tab under **Hunting**. Select the **C2 Hunt** query and then **Right-Click** on it. Click on the **Add to Livestream**.    24. Click on the Livestream tab. Review that the **Status** is now **Running**. This will run every 30 seconds in the background, and you will receive a notification on the Azure Portal (bell icon) when a new result is found. |

**Lab 04: SOC Incident Workflow Using Microsoft Sentinel**

**Case Study**

Icertis, a global leader in contract intelligence, supports customers in regulated industries and needed to further scale security resources to help protect sensitive contract data and generative AI workloads. The organization relies on a combination of cloud services and AI-driven applications to support its contract management platform, which captures relationships for almost every customer, supplier, and partner relationship worldwide. Icertis employs thousands of staff members and contractors who require secure access to sensitive systems and intellectual property. Due to the critical nature of contract data and the emergence of generative AI, maintaining strong cybersecurity controls and rapid incident response capabilities is essential for ensuring safety, compliance, and business continuity.

As Icertis expanded its digital transformation initiatives and launched the industry’s first generative AI applications for enterprise contracting, the volume of security data generated from endpoints, identity platforms, network devices, and cloud services increased significantly. While multiple security tools were already deployed, the organization lacked a unified platform to correlate alerts and manage incidents efficiently. To strengthen its Security Operations Center (SOC) and establish consistent incident workflows, Icertis adopted Microsoft Sentinel as its cloud-native Security Information and Event Management (SIEM) and Security Orchestration, Automation, and Response (SOAR) solution.

**Business Challenge**

Before implementing Microsoft Sentinel, Icertis faced fragmented security visibility across its global environment. Security alerts and logs were generated by different tools but remained isolated within separate systems. This lack of centralized visibility made it difficult for SOC analysts to identify relationships between events and understand the full scope of potential security incidents, particularly those involving sensitive contract data and AI workloads. As a result, analysts experienced alert fatigue, delayed investigations, and reduced confidence in incident outcomes.

Incident response processes were also highly manual and inconsistent. Analysts followed different investigation and response procedures depending on location and experience level. This inconsistency increased the risk of delayed containment and incomplete remediation. Additionally, repetitive tasks such as alert enrichment, stakeholder notification, and escalation consumed significant analyst time. These challenges increased Mean Time To Detect (MTTD) and Mean Time To Respond (MTTR), allowing threats to persist longer within the environment.

The organization required a centralized platform that could correlate alerts into meaningful incidents, automate repetitive response tasks, and enforce standardized incident workflows across the SOC to protect its generative AI applications and sensitive data.

**Solution**

In this lab, Microsoft Sentinel is used to demonstrate how Icertis can modernize its SOC incident workflow through centralized visibility and automation. Microsoft Sentinel is configured to ingest security telemetry from endpoints, identity services, and infrastructure components into a single Log Analytics workspace. This enables the SOC to correlate related alerts into incidents that represent real security events rather than isolated alerts.

Analytics rules are created to detect suspicious behavior, such as repeated failed remote desktop login attempts or unauthorized access to AI workloads. When these rules trigger, Microsoft Sentinel automatically generates incidents that can be triaged, investigated, and responded to within a unified interface. SOC analysts use the incident dashboard, entity mapping, and investigation graph to assess incident scope, affected assets, and potential attack paths.

Automation plays a key role in improving response efficiency. Playbooks built using Azure Logic Apps are integrated with Microsoft Sentinel to automate actions such as threat intelligence enrichment and virtual machine shutdown in response to high-risk incidents. By integrating Microsoft Sentinel with automation, collaboration tools, and ticketing systems, Icertis achieves faster response times, reduced analyst workload, and consistent incident handling practices.

Through centralized incident management, automated response, and standardized workflows, Microsoft Sentinel enables the SOC to improve operational efficiency, evidenced by a 50% reduction in SOC incident volume and 80% faster alert triage, and strengthen the overall security posture of the Icertis environment.

In this lab, you will perform the following tasks to simulate a real SOC incident workflow and understand how Microsoft Sentinel helps detect, investigate, and respond to security threats:

* Create an analytics rule
* Triage an incident
* Investigate an incident
* Respond using playbooks and automation
* Handle false positives

|  |  |  |
| --- | --- | --- |
| **Step 1: Create a Rule**  1. Go to the **Virtual Machine** and create an **Incident-Hunt-VM1** virtual machine. How to create a virtual machine, go to **Lab 3-03: Threat Hunting**    2. After the virtual machine is created, go to **Microsoft Sentinel**.    3. Click on the **Sentinel-LAW** log analytics workspace.    4. Click on the **Logs** under **Overview (Preview)**. Copy and paste the below-provided KQL query. Then, click on the **Run** button to see previous security events.   |  | | --- | | SecurityEvent  | summarize EventCount = count() by Computer  | sort by EventCount desc |       5. Click on **Analytics** under **Configuration**. Click on the **+ Create** dropdown. Then click on the **Schedule query rule**.    6. Enter the name of the analytics rule **RDP-Login-Failures.** Enter your **Description**. Select the severity **High**. Click on the **MITRE ATT&CK** dropdown to select tactics, techniques, and sub-techniques of the MITRE ATT&CK framework.    7. In the search bar, type **Credential Stuffing**. Click on the **Credential Access > T1110 – Brute Force**. Select the **T1110.004 – Credential Stuffing**.    8. Again, in the search bar, type **Remote Desktop**. Click on the **Lateral Movement > T1020 – Remote Services**. Select the **T1021.001 – Remote Desktop Protocol**.    9. Selected **6** MITRE ATT&CK tactics. Then click on the **Next : Set rule logic >** button.    10. Copy and paste the below-provided KQL query under **Rule query**.   |  | | --- | | SecurityEvent  | where EventID == 4625 and LogonType == 3  | summarize FailedAttempts = count() by Computer, Account, IpAddress  | where FailedAttempts >= 3 |       11. Scroll down. Click on the **Entity mapping** drop-down. Then click on the **+ Add new entity**.    12. Select **Host** in the **Entity type** drop-down list. For the **Identifier** drop-down list, select **HostName**. For the **Value** drop-down list, select **Computer**. Then click on the **+ Add new entity**.    13. Select **Account** in the **Entity type** drop-down list. For the **Identifier** drop-down list, select **Name**. For the **Value** drop-down list, select **Account**.    14. Scroll down under Query scheduling. In the **Run query every,** enter **5** and select **Minutes**. In the **Look data from the last,** enter **1** and select **Hours**. Then select start running **Automatically**.    15. Scroll down under the **Alert threshold**. In the **Generate alert when number of query results,** select **Is greater than** and enter **1**. Then select **Group all events into a single alert**. After that, click on the **Next : Incident settings >** button.    16. In the **Limit the group to alerts created within the selected time frame,** enter **5** and select **Minutes**. Then select **Grouping all alerts triggered by this rule into a single incident**. After that, click on the **Next : Automated response >** button.    17. Click on the **Next : Review + create >** button.    18. After validation passes, click on the **Save** button.    19. Successfully created the **RDP-Login-Failures** analytics rule in Microsoft Sentinel. When a rule is triggered, it will generate an alert on Microsoft Sentinel.    20. Go to the **Virtual machines** to generate a fake RDP brute force attack.    21. Click on the **Incident-Hunt-VM1**.    22. Click on the **Connect**.    23. Click on the **Download RDP file** button.    24. Click on the **Save** button to save the RDP file.    25. Double-click on the **Incident-Hunt-VM1** RDP file.    26. Click on the **Connect** button.    27. Enter a fake **password** and then click on the **OK** button.    28. After that, you should see a message that **The logon attempt failed**.    29. Repeat this process many times. This time, change the user’s name. In the User name, enter **Attacker** or any fake user name. Then, enter a fake password. Click on the **OK** button.    30. Again, you should see a message that **The logon attempt failed**. |
| **Step 2: Triage an Incident**  1. Click on the **Incidents** under **Threat Management**. You should observe that there are many incidents generated after **RDP-Login-Failures** analytics triggers.    2. Click on any **incident**. Click on the **New** dropdown. You can change the incident status to **New**, **Active,** and **Closed**.    3. Under the RDP-Login-Failures incident, you can observe and triage an incident. On the left-hand side, you should see incident **Description** and **Evidence**. Click on the **Overview** tab to examine the incident timeline and entities.    4. Scroll down. On the left-hand side, you should inspect **entities**, **tactics, and techniques** that a rule detects. Click on the **Entities** tab to observe entities in more depth. On the right-hand side, click on the **Incident actions** drop-down. Then, click on the **Run playbook** to explore the incident more deeply using the pre-built playbook.    5. Click the **Run** button to run a playbook on an incident. To create this pre-built Virus Total playbook, go to the following link: <https://learn.microsoft.com/en-us/azure/sentinel/tutorial-enrich-ip-information>. Later in this lab, in step 4, we create a playbook from scratch.    6. Successfully run a playbook. This playbook automatically checks IP addresses in your incidents against a threat intelligence source. It records each result in its relevant incident. You can use any playbook that is relevant to your incident.    7. Click on **Try the new experience** switch to return to the old incident hunting interface. Click on the **Comments** tab. You can comment on an incident so it can assist SOC analysts in investigating it.    8. On the left-hand side below, click on the **Investigate** button.    9. It shows the investigation graph to investigate how this event occurred and take further actions in the future to mitigate this type of incident in Microsoft Sentinel. Hover over the **RDP-Login-Failures** to dig deeper into this incident. |
| **Step 3: Investigate**  1. In the SOC incident workflow investigation step, we used the **Identity & Access workbook** to investigate incidents on the **Incident-Hunt-VM1** virtual machine. In this workbook, you should observe the top 10 activities that happened in the VM. Then, you can examine **User activities** and how many users try to log into a VM. Also, review **Machine activities** and how processes are running in a VM.    2. Scroll down to observe the **Full details** of a VM. You can deeply investigate the incidents in a VM.    3. Scroll down to analyze the **Processes graph** and **Process details**.    4. Scroll down. You can scrutinize the attacks deeply.    5. Click on the **Logs** under **Overview (Preview)**. Enter the following **SecurityAlert** KQL query. Click on the **Run** button to check how many alerts have been generated.    6. Copy and paste the below-proved KQL in the code editor. Then click on the **Run** button. This KQL query retrieves security alerts from the past 7 days, displaying their generation time, name, description, associated MITRE ATT&CK tactics, and techniques, sorted by the most recent alerts.   |  | | --- | | SecurityAlert  | where TimeGenerated >= ago(7d)  | project TimeGenerated, AlertName, Description, Tactics, Techniques  | order by TimeGenerated desc |     7. Select any result that appears after running a query. Then click on the **Add bookmark**.    8. Enter the RDP Brute Force Attack name of the bookmark. Click on the **MITRE ATT&CK** dropdown to select tactics, techniques, and sub-techniques of the MITRE ATT&CK framework. In the search bar, type **Credential Stuffing**. Click on the **Credential Access > T1110 – Brute Force**. Select the **T1110.004 – Credential Stuffing**. Again, in the search bar, type **Remote Desktop**. Click on the **Lateral Movement > T1020 – Remote Services**. Select the **T1021.001 – Remote Desktop Protocol**. Selected **6** MITRE ATT&CK tactics. Follow the same procedure as we did in step 1. Click on the **Create** button.    9. Click on the **Hunting** under **Threat Management**. Click on the **Bookmark** tab. Select the **RDP Brute Force Attack** bookmark. Click on the **Incident actions** drop-down. Then click on the **Add to existing incident**. Select any incident from **Adding bookmark(s) to an existing incident**. Also,note down the **Incident ID**. Click on the **Add** button.    10. Click on the **Incident** under **Threat Management**. Select the incident with an incident ID that you added a bookmark to; for instance, in this lab, it is **177**. On the left-hand side in **Evidence,** you should see that **1** bookmark is associated with this incident. You can also verify by clicking on the **Bookmark** tab. |
| **Step 4: Respond**  1. On the **Hunting** dashboard, click on the **Livestream** tab. You can respond to a threat in real-time using a livestream. In the livestream, the same query we used in log analytics will run continuously in the background. If any threat occurs, it will automatically show a result.    2. The second approach to respond to a threat is via a playbook. In this step, we create a playbook that states that if any RDP brute force attack occurs on a VM, it will shut it down. This is an automation process. Go to the Logic Apps. Click on the **+ Add** button.    3. Select the **Multi-tenant**. Then click on the **Select** button.    4. Select the subscription **vanguardinvest**. Select the resource group **SOC-Workflow**. Enter the name of a logic app: **vm-shutdown-logic-app**. Select the region **UK South**. Click on **Review + create**.    5. Click on the **Create**.    6. Click on the **Identity** under **Settings**.    7. Click on the **Status** switch to **On**. Then click on the **Save** button.    8. Click on the **Yes** button.    9. Click on the **Azure role assignments**.    10. Click on the **+ Add role assignment (Preview)**. In **Scope,** select **Subscription**. In **Subscription,** select **vanguardinvest**. In **Role,** select **Virtual Machine Contributor**. Click on the **Save** button.    11. Click on the **Logic app designer** under **Development Tools**.    12. Click on **Add a trigger**.    13. In **Add a trigger** search bar, type **Sentinel**. Select the **Microsoft Sentinel incident**.    14. Added **Microsoft Sentinel Incident** trigger. Click on the **Change connection**.    15. Click on the **Add new** button.    16. Select **Managed identity** in **Authentication**. Enter the connection name **SentinelConnect**. Select **System-assigned managed identity** in **Managed Identity**. Click on the **Create new** button.    17. Click on the **+ Add an action**.    18. In the search bar, type **vm** under **Add an action**. Then Click on **See more**.    19. Select the **Power off virtual machine**.    20. Select **Managed identity** in **Authentication**. Enter the name of the connection **SentinelConnect**. Select **System-assigned managed identity** in **Managed Identity**. Click on the **Create new** button.    21. Select **vanguardinvest** in the **Subscription Id**. In the resource group, select **SOC-Workflow**. In **Virtual Machine,** select **Incident-Hunt-VM1**. Then click on the collapse **icon >>**.    22. Click on the **Save** button.    23. Go to the **Microsoft Sentinel** dashboard. Click on the **Automation** under **Configuration**. Click on the **Active playbooks** tab to see the **vm-shut-down-logic-app** logic app.    24. Click on the **Analytics** under **Configuration**. Before adding a playbook to an analytics rule, disable the rule first to disable it. Right-click on a rule, then click on **Disabled**. To edit the settings of a rule, Right-click on the **RDP-Login-Failures** analytics rule. Click on the **Edit**.    25. Click on the **Automated response**. Then click on the **+ Add new**.    26. Enter the name of the automation rule **shutdown-vm**. In **Trigger,** select **When incident is created**. In Actions, select **Run Playbook**.    27. Click on the **Manage playbook permissions**.    28. Select the **SOC-Workflow** resource group. Then click on the **Apply** button.    29. Select the **vm-shutdown-logic-app**. Then click on the **Apply** button.    30. Added automation rule in an **RDP-Login-Failures** analytics rule. Click on the **Next : Review + create >** button.    31. After validations pass, click on the **Save** button.    32. Select the **RDP-Login-Failures** analytics rule. Then, Right-click on the **RDP-Login-Failures** analytics rule. Click on the **Enabled**.    33. Go to the **Virtual Machines**. Notice that the **Incident-Hunt-VM1** is **Running**.    34. Again, RDP the **Incident-Hunt-VM1** as we did in step 1. Create an RDP brute attack. Enter a fake **user name** and **password**. Then click on the **OK** button.    35. Go to the **Virtual Machines**. Notice this time that the **Incident-Hunt-VM1** is **Stopped**.    36. Go to the Logic App. Click on the **Run history** under **Development Tools**. Hence, the playbook runs successfully. Click on the **Succeeded**.    37. The green tick shows that the logic app triggers successfully and shutdown a VM. |
| **Step 5: Handle False Positive**  1. There are many methods to handle false positives in Microsoft Sentinel. For this, you can create a separate analytics rule. The following is the explanation of the KQL query.  **Exclude Specific Destination IP**: The where Computer != “1.2.3.4” condition excludes any failed login attempts where “1.2.3.4” is the target (destination) IP.  **Count and Detect**: The query counts failed attempts by Computer, Account, and IpAddress and triggers only if there are 3 or more failed login attempts for other IPs.  Copy and paste the KQL query below in the **Rule query** code editor.   |  | | --- | | SecurityEvent  | where EventID == 4625 and LogonType == 3  | where Computer != "1.2.3.4" // Exclude attempts to this IP address  | summarize FailedAttempts = count() by Computer, Account, IpAddress  | where FailedAttempts >= 3 // Detects failed attempts on all other IPs |     Use the two links below to handle false positives in Microsoft Sentinel according to your specific scenarios.  <https://learn.microsoft.com/en-us/azure/sentinel/false-positives>  <https://techcommunity.microsoft.com/t5/microsoft-sentinel-blog/handling-false-positives-in-azure-sentinel/ba-p/2158352> |