**SOC Project Report on SOAR Implementation Using Tines and LimaCharlie**

**Introduction:**

In today’s rapidly evolving cybersecurity landscape, **Security Orchestration, Automation, and Response (SOAR)** solutions are becoming essential components of Security Operations Centers (SOCs). The primary goal of SOAR is to automate repetitive tasks, enabling SOC teams to focus on more complex incidents, streamline workflows, and enhance incident response through structured playbooks.

This report details the process of integrating **Tines** as the SOAR platform with **LimaCharlie** as an Endpoint Detection and Response (EDR) system. The project demonstrates how SOAR can be effectively used in a SOC environment to automate responses to detections, such as password recovery tool activity, and escalate actions based on user input. This project also emphasizes the importance of understanding the functionality of tools, irrespective of the specific vendor, by focusing on core SOAR concepts rather than proprietary features.

**Project Objectives:**

1. **Automate Detection and Response**: Automate the detection of password recovery tools on Windows machines using LimaCharlie and respond to incidents using Tines.
2. **Design and Implement Playbooks**: Create a custom playbook (referred to as “story” in Tines) that integrates alerts with communication tools such as Slack and email for incident notification and escalation.
3. **User Interaction in Incident Response**: Provide a mechanism for user input to determine further actions, such as isolating a machine upon detecting unauthorized activity.
4. **Develop Hands-On SOAR and EDR Experience**: Gain practical experience with SOAR and EDR platforms by setting up detection rules, generating events, and automating responses.

**Project Workflow Overview:**

1. **Preparation**:
   * Install **Tines** as the SOAR platform and **LimaCharlie** as the EDR solution.
   * Deploy a Windows Virtual Machine (VM) to act as the test environment.
   * Ensure the VM has an active internet connection to communicate with the SOAR and EDR platforms.
2. **Detection Setup**:
   * **Custom Detection Rules in LimaCharlie**: Implement a detection rule to identify the usage of password recovery tools on the Windows VM.
   * **Data Collection**: Start collecting event data from LimaCharlie, which will act as the EDR solution for the project.
3. **Playbook Creation (SOAR Story in Tines)**:
   * Design the **playbook** (story) in Tines, which will automate the response to detections.
   * The playbook will:
     + Send an alert via **email** and **Slack** when a detection is made.
     + Query the user to ask whether the detected machine should be isolated.
     + **Isolate the machine** using LimaCharlie’s capabilities if the user confirms the action.
4. **Slack and Email Integration**:
   * Integrate **Slack** and **email** with the playbook to enable real-time incident notifications and escalations.
   * Test the integration to ensure that notifications are triggered upon detection.
5. **User Interaction**:
   * The playbook will prompt the user with a **confirmation** to isolate the machine if suspicious activity is detected. Based on the user’s response, LimaCharlie will execute an automated isolation process.

**Challenges Encountered:**

1. **Playbook Design Iterations**: The creation of a functional playbook (story) required several iterations. As is common in SOC environments, some planned functions didn’t work as expected, and adjustments were made during the development process.
2. **Integration and Event Handling**: Ensuring smooth integration between LimaCharlie and Tines involved troubleshooting compatibility issues, particularly in passing alerts and data between the platforms. Documenting these integration steps was key to resolving problems efficiently.
3. **Handling User Inputs**: Allowing user interaction in automated processes posed some challenges. The logic required to handle user responses within the playbook was adjusted to ensure accuracy, as incorrectly configured logic could lead to improper incident response actions.

**Results:**

By the end of the project, we have successfully:

* Created a **detection rule** in LimaCharlie to monitor and alert on the use of password recovery tools.
* Integrated **Slack** and **email** notifications into the SOAR playbook, enabling real-time communication of incidents to the SOC team.
* Developed a **fully functional playbook** that provided users with options for manual intervention in incident handling, such as isolating compromised machines.

The project showcased the value of SOAR platforms like Tines in automating SOC workflows and allowed for hands-on experience in managing and responding to security threats in real-time.

**Lessons Learned:**

1. **Tool Agnosticism**: While we used **Tines** as the SOAR platform and **LimaCharlie** as the EDR tool, the principles of SOAR implementation remain largely vendor-agnostic. The key takeaway is that SOC processes should be designed with flexibility in mind, as the tools may vary, but the goals of automation and structured responses stay the same.
2. **Importance of Iteration in Playbook Development**: During the creation of playbooks, it’s crucial to anticipate changes and continuously refine the workflows. Playbooks evolve based on operational requirements, tool limitations, and new security threats.
3. **Practical Experience Matters**: This project reinforced that hands-on practice with SOAR tools is essential for building confidence and technical proficiency in SOC operations. The integration of LimaCharlie and Tines provided valuable insight into orchestrating responses in real-time.

**Conclusion:**

This project successfully demonstrated the use of **SOAR platforms** like Tines to streamline SOC workflows by automating repetitive tasks, providing real-time incident notifications, and enabling user interaction during incident response. The integration of EDR with SOAR platforms is a critical strategy for modern SOC teams to ensure that security incidents are handled efficiently and effectively. This hands-on project provided a solid foundation for future SOC automation projects and real-world application of SOAR tools.

Part 1

A screenshot of a computer screen

Description automatically generated

**LimaCharlie EDR and Tines Integration – Part 2**

**Objective:**

In this phase, my goal is to:

* Install and set up **LimaCharlie EDR** on my **Windows Server 2022 virtual machine (VM)** running in VirtualBox.
* Ensure the VM is enrolled in **LimaCharlie** and is generating **event data**.
* Prepare the environment for creating **custom detection and response rules**.

**Step-by-Step Setup Process:**

1. **Environment Setup (Windows Server 2022 in VirtualBox):**

* I will be using my existing VM running Windows Server 2022 on VirtualBox. It’s important to ensure that the VM has internet connectivity since LimaCharlie needs to communicate with its cloud dashboard to report events.

1. **LimaCharlie Setup:**
   * Go to the **LimaCharlie website** and sign up if you haven’t done so already.
   * Once logged in, follow the LimaCharlie onboarding wizard, which includes creating an organization.
   * Give your organization a meaningful name, such as **"chineme-SOAR-EDR"**.
   * Then select the appropriate **data residency region**, like "USA" or your preferred location.

A screenshot of a computer

Description automatically generated

1. **Creating Installation Keys in LimaCharlie:**
   * Navigate to the **Installation Keys** section on the LimaCharlie dashboard. These keys are used to enroll sensors (my VM) into the LimaCharlie environment.
   * Click on **Create Installation Key** and provide a description such as **"Chineme-SOAR-EDR"**.
   * Save the key for later use during the agent installation process.

A screenshot of a computer

Description automatically generated

1. **Installing the LimaCharlie Agent on Windows Server 2022:**
   * Now that I have my installation key, it’s time to install the **LimaCharlie Agent** on my Windows Server VM.
   * Download the **LimaCharlie Windows Agent** from the platform.
     + In LimaCharlie, navigate to the **Downloads** section and select the Windows agent for **64-bit systems**.

A screenshot of a computer

Description automatically generated

* + Once downloaded, I’ll open **PowerShell as Administrator** to install the agent:
    - Navigate to the directory where the file was downloaded using cd Downloads.
    - Run the installation using the command:

A screen shot of a computer

Description automatically generated

.\limaCharlieInstaller.exe -i <Your\_Installation\_Key>

* + - I’ll ensure the agent installs successfully, and I should see a confirmation message once the installation is complete.

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

1. **Verifying Enrollment in LimaCharlie:**
   * After installation, head back to the **LimaCharlie dashboard** and navigate to **Sensors**.

A screenshot of a computer

Description automatically generated

* + You should now see **Windows Server 2022 VM** listed under the **Sensors List**. The dashboard will provide details such as:
    - Hostname
    - Internal and external IP addresses
    - Enrollment date
    - Platform type
  + I can also see basic analytics like **event collection** and **file system** data.

*Screenshot: Include the LimaCharlie dashboard showing the enrolled sensor (my VM).*

A screenshot of a computer

Description automatically generated

1. **Initial Test Commands:**
   * With my sensor now visible in the LimaCharlie dashboard, it’s time to run some basic commands to verify the setup.
     + I’ll use the **Netstat** command from within the LimaCharlie console to view active network connections on my Windows Server VM:
       - In LimaCharlie, navigate to **Console** under the enrolled sensor and run:
       - I’ll verify that the output shows network connections along with their process IDs (PIDs).

A screenshot of a computer program

Description automatically generated

1. **Preparing for Custom Detection and Response:**
   * Now that my VM is generating telemetry and I’ve verified the communication with LimaCharlie, I’m ready for the next phase of the project: **creating detection and response rules**.
   * The focus will be on generating specific events related to **password recovery tools** and crafting custom responses to mitigate the threat.

**Part 3: Generating Telemetry and Creating Detection & Response Rules in LimaCharlie**

In this part, our main objective is to:

1. Generate telemetry from the **LaZagnea password recovery tool**.
2. Create a **detection and response rule** in LimaCharlie to identify this activity.
3. Set the groundwork for forwarding the detection to **Tines** for automated actions.

**Step 1: Download and Run LaZagne**

The first step is to generate activity by downloading and running **LaZagne**, a popular password recovery tool. LimaCharlie will monitor this execution, allowing us to create a detection rule based on telemetry.

1. **Download LaZagne:**
   * Visit **LaZagne 's GitHub** repository.

Lazagne: <https://github.com/AlessandroZ/LaZagne>

* + Navigate to the **Releases** section and download the **LaZagne.exe** file.

A screenshot of a computer

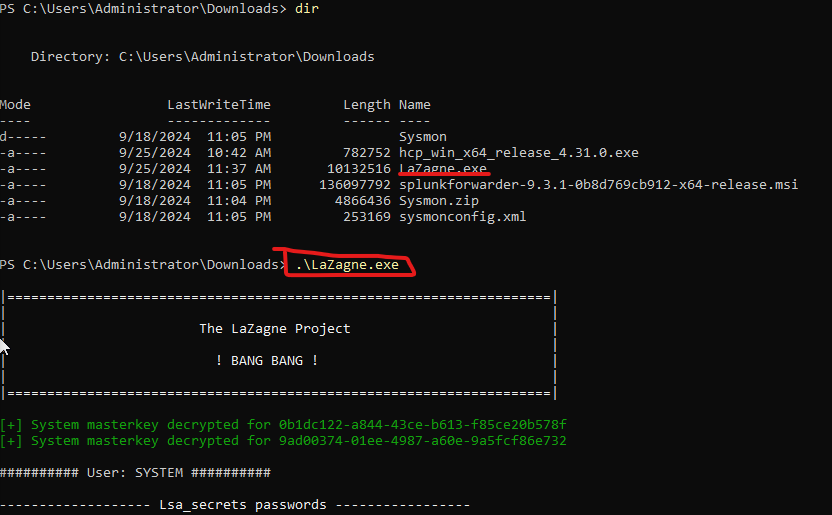
Description automatically generated

1. **Disable Windows Security:**
   * Open **Windows Security** and go to **Virus & Threat Protection**.
   * Temporarily **disable real-time protection** to prevent Windows Defender from blocking LaZagne.

***Note****: Disabling real-time protection is necessary for this demo but should be re-enabled after completing the project.*

1. **Running LaZagne:**
   * Open **PowerShell** in the directory where LaZagne is downloaded.
   * Run the tool by typing:

.\LaZagne.exe



1. At this point, LimaCharlie should begin capturing telemetry related to the execution of LaZagne.

**Step 2: Viewing Telemetry in LimaCharlie**

After running LaZagne, we can head over to **LimaCharlie** to view the generated events.

1. **Navigate to the Sensor:**
   * In LimaCharlie, go to the **Sensors List** and select your **Windows Server VM**.
2. **View the Event Timeline:**
   * Navigate to the **Timeline** tab to see recent events. Since we just executed LaZagne, the telemetry should be visible here.
   * Filter the events using the term **LaZagne** to quickly find the specific process creation event.
3. **Telemetry Details:**
   * Click on the event for more details. Here, you'll see useful information such as the **file path**, **command line arguments**, **process ID**, and more.
   * This information will form the foundation for our custom detection rule.

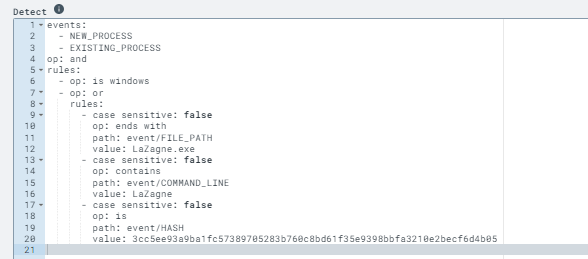
A screenshot of a computer

Description automatically generated

**Step 3: Creating the Detection Rule in LimaCharlie**

Now that we have telemetry from LaZagne, it’s time to create a **detection and response (DNR) rule** in LimaCharlie. This rule will detect the execution of LaZagne in the future and trigger a response.

1. **Navigate to Detection & Response (DNR) Rules:**
   * Go to the **Automation** section in LimaCharlie.
   * Click on **New Rule** to create a new detection rule.
2. **Base the Rule on Existing Templates:**
   * Instead of starting from scratch, use an existing rule as a foundation.
   * Search for **Credential Access** related rules, as LaZagne is a password recovery tool. Select a rule related to **process creation** and view its content.
3. **Customizing the Rule for LaZagne:**
   * Edit the rule to fit our use case. The key event type we're interested in is **new process creation**.
   * We will match the **file path** and **command line arguments** from the telemetry we viewed earlier.
   * For example, the rule can trigger if the file path ends with **LaZagne.exe** or if the **command line** contains specific arguments (like **all**).
4. **Refining the Criteria:**
   * The rule should check:
     + **Process Creation Event**: The event type should be either **new process** or **existing process**.
     + **File Path**: The file path should end with LaZagne.exe (case-insensitive).
     + **Command Line**: The command line should contain **all** or similar relevant arguments from the telemetry.
     + **File Hash**: Optionally, you can match the hash of the LaZagne executable to ensure accuracy.

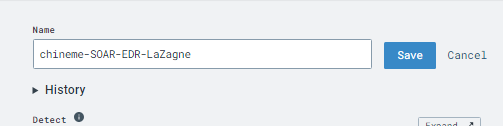


A screenshot of a computer program

Description automatically generated

Here is the GitHub link to the rule: [GitHub - MyDFIR/SOAR-EDR-Project](https://github.com/MyDFIR/SOAR-EDR-Project)

1. **Saving the Rule:**
   * Once customized, name the rule something relevant like **"LaZagne Detection Rule"** and save it.



**Step 4: Testing the Rule**

After saving the rule, it’s important to test whether it works as intended.

1. **Testing with Past Events:**
   * Go back to the previously captured event for LaZagne.
   * Use LimaCharlie’s **Test Event** functionality by pasting the raw event data into the test field. Run the test and ensure that the rule is correctly detecting the event.

A screenshot of a computer

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

1. **Running LaZagne Again:**
   * Clear the previous telemetry for a clean slate.
   * Run **LaZagne.exe** again in your Windows Server VM.
   * Check **LimaCharlie’s detection tab** to verify that the rule triggered and generated a detection.

A screenshot of a computer

Description automatically generated

**Conclusion:**

In this part of the project, we successfully:

* Generated telemetry from the **LaZagne password recovery tool**.
* Used that telemetry to create a **custom detection and response rule** in LimaCharlie.
* Tested the rule to ensure that it correctly identifies the execution of LaZagne.

**Part 4: Integrating Slack, Tines, and LimaCharlie for Automation**

In this phase, we will:

1. Set up **Slack** to receive alerts.
2. Configure **Tines** to connect with LimaCharlie.
3. Test the connection between LimaCharlie and Tines to ensure that **SOAR** (Security Orchestration, Automation, and Response) processes are receiving the detections generated by LimaCharlie.

**Step 1: Setting up Slack**

The first tool we’ll configure is **Slack**, where we’ll be sending our alerts from LimaCharlie via Tines.

1. **Create a Slack Workspace**:
   * Go to [**Slack.com**](https://slack.com) and click on the "Get Started Free" button to create an account.
   * Use a **valid email account**, as Slack will send a confirmation code to complete the signup process.
   * Name your workspace something meaningful like **"chineme-SOAR-EDR"**.
2. **Create an Alerts Channel**:
   * In your new workspace, create a dedicated channel for alerts:
     + Click on **Add Channels** and select **Create a New Channel**.
     + Name the channel **alerts** and set it to **public**.
     + Skip inviting people for now.
   * This channel will receive detection alerts from LimaCharlie via Tines.

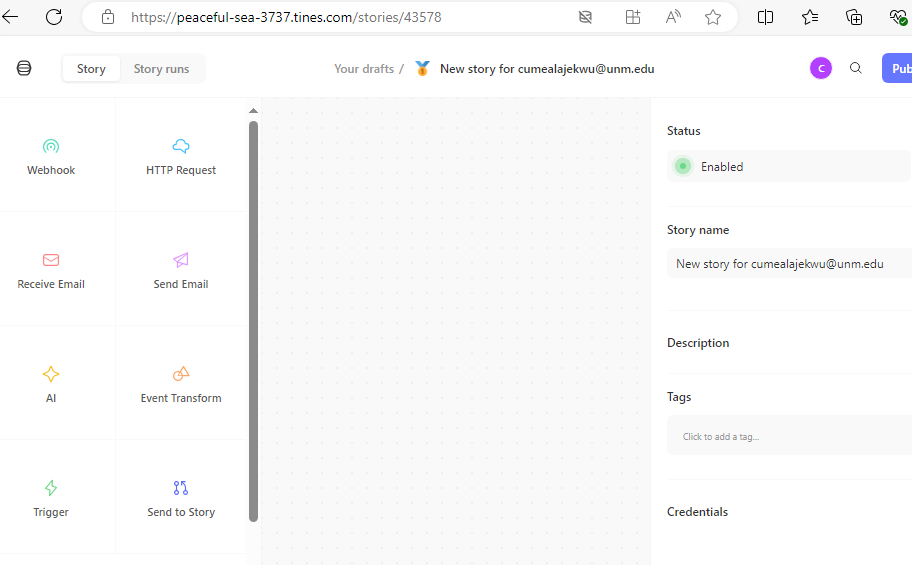
A screenshot of a computer

Description automatically generated

**Step 2: Setting up Tines**

Next, we’ll configure **Tines**, which will handle automation based on the detections reported by LimaCharlie.

1. **Create a Tines Account**:
   * Head over to [**Tines.com**](https://tines.com) and sign up using your email. You can also sign in with **Google** or **Microsoft** accounts.
   * Once logged in, exit the introduction tour by clicking on the "X" and "End Tour."

**

1. **Exploring Tines' Actions**:
   * On the left-hand side, you’ll see different actions you can use to build your playbook (known as **stories** in Tines).
   * For example, you can drag an **HTTP Request** or **Web Hook** action onto the storyboard. We’ll use a **Web Hook** to receive detections from LimaCharlie.
   * Templates are also available if you’d prefer starting with a pre-built example.

*A screenshot of a computer

Description automatically generated*

**Step 3: Connecting LimaCharlie and Tines**

Now that we have both **Slack** and **Tines** set up, the next step is to establish the connection between **LimaCharlie** and **Tines**, ensuring that detection data flows between them.

1. **Create a Web Hook in Tines**:
   * In Tines, drag a **Web Hook** onto the storyboard and name it **Retrieve Detections**.
   * For the **Web Hook URL**, copy the provided link.

A screenshot of a computer

Description automatically generated

1. **Configuring Outputs in LimaCharlie**:
   * Head back to **LimaCharlie** and navigate to your organization’s **Outputs**.
   * Click on **Add Output** and select **Detections** (since we want to capture all detections from our rule).
   * In the output configuration:
     + Select **Tines** as the destination.
     + Paste the **Web Hook URL** from Tines into the **Destination Host** field.
     + Name the output something like **chineme-SOAR-EDR**

A screenshot of a computer

Description automatically generated

1. **Generating a Detection Event**:
   * To test if the connection is working, return to your **Windows Server VM** and run **Lasagna.exe** again:

.\Lasagna.exe

* + This will trigger a detection in LimaCharlie, which should be forwarded to **Tines** via the Web Hook.

A screenshot of a computer

Description automatically generated

**Step 4: Verifying the Integration**

Now that the connection is configured, let’s confirm that **Tines** is receiving detections from LimaCharlie and forwarding them to **Slack**.

1. **Check the Detection in Tines**:
   * Go back to **Tines** and check the **Retrieve Detections** Web Hook.
   * You should see the detection event from **LimaCharlie**. This will contain information like the **command line**, **file path**, **hash**, and **username** associated with the Lasagna execution.

A screenshot of a computer

Description automatically generated

**Conclusion**

At this point, we’ve successfully:

* Set up **Slack** to receive alerts.
* Configured **Tines** to accept detection events from **LimaCharlie** via a Web Hook.
* Tested the connection by generating a detection from the **Lasagna** password recovery tool and forwarding it to Slack.

**Part 5: Building the Final Playbook with LimaCharlie, Tines, and Slack Integration**

Today’s objective is to build our **Playbook** (or story) that will:

1. **Send a message to Slack** with the details of the detection.
2. **Send an email** to notify the team.
3. **Prompt the user** to decide whether to isolate the affected machine.
4. **Isolate the machine** using LimaCharlie if the user chooses to do so.
5. **Send confirmation** of the machine’s isolation status to Slack.

This is the culmination of all the steps we’ve been working on, and by the end of this section, you’ll have a fully automated detection and response Playbook in **Tines** connected to **LimaCharlie**.

**Step 1: Reviewing the Workflow**

Before jumping into the configuration, let's review the **workflow** we created in Part 1.

The workflow for this automation is as follows:

1. **Detection event**: LimaCharlie detects a malicious tool (like **Lasagna**) and sends the detection to **Tines**.
2. **Slack message**: Tines sends a notification to a Slack channel with the detection details.
3. **Email notification**: An email is sent to notify the relevant team about the detection.
4. **User prompt**: Tines prompts the user to decide whether the machine should be isolated.
5. **Isolation**: If the user selects "Yes," LimaCharlie isolates the machine.
6. **Slack confirmation**: Tines sends a confirmation message to Slack that the machine has been isolated.

**Step 2: Connecting Slack to Tines**

1. **Create Slack Workspace**:
   * Ensure you have your **Slack workspace** ready (as set up in Part 4).
   * You will also need a dedicated **alerts channel** where the messages from Tines will be sent.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

I already downloaded the tines app that’s why I cant see it in my search

A screenshot of a computer

Description automatically generated

A screenshot of a white background

Description automatically generated

A purple box with white text

Description automatically generated

1. **Establish the Link Between Tines and Slack**:
   * In **Tines**, drag the **Slack** action from the template library into your Playbook.

A screenshot of a computer

Description automatically generated

* + Use the Slack API to send a message to your **alerts** channel. The message should contain the details of the detection (such as the detection name, time, affected computer, and IP address).
  + Use the Slack **Channel ID** to specify where the message should be sent.

A screenshot of a computer

Description automatically generated

A screenshot of a chat

Description automatically generated

A screenshot of a black screen

Description automatically generated

A computer screen shot of a program code

Description automatically generated

Copy and paste on the notepad!

A screenshot of a computer program

Description automatically generated

Paste in slack(tines)

A screenshot of a computer

Description automatically generated

Run test. Here is the result of the test

A screenshot of a computer

Description automatically generated

**Step 3: Configuring Email Notifications in Tines**

1. **Set up the Email Notification**:
   * Add an **email notification** action in Tines to send the detection details via email.
   * Use your organization's email system or a disposable email provider (like **Squarex**) to test.
   * Ensure the email includes relevant information such as the **detection name**, **computer name**, **IP address**, and a **link to the detection** in LimaCharlie.

*A screen shot of a computer

Description automatically generated*

A screenshot of a computer

Description automatically generated

Use the same step as step 3. Just copy and paste the message you want to receive in your email.

A screenshot of a computer

Description automatically generated

**Step 4: Adding a User Prompt for Isolation Decision**

1. **Create a User Prompt Page**:
   * Now, create a **User Prompt Page** in Tines that asks the user whether they want to isolate the machine.
   * The page should display the detection details for the user to review before making the decision.
2. **Yes/No Decision**:
   * Include **Yes** and **No** buttons on the page.
   * If the user selects **Yes**, the Playbook should continue to isolate the machine using **LimaCharlie**. If the user selects **No**, a message should be sent to Slack indicating that the machine was not isolated.

A screenshot of a computer

Description automatically generated

*A screenshot of a computer

Description automatically generated*

A screenshot of a computer

Description automatically generated

Configuring the workflow so that when you say no, it shows you a message and then sends you a slack message

This can be done with this configuration

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step 5: Isolating the Machine in LimaCharlie**

1. **Automate Machine Isolation**:
   * If the user chooses to isolate the machine, configure Tines to send an **isolation request** to LimaCharlie.

If you are in doubt what to put in the use the search bar ad type in “isolate”

A screenshot of a computer

Description automatically generated

* + Use the **Sensor ID** (which you’ll get from the detection event) to identify the machine to be isolated.

A screenshot of a computer

Description automatically generated

* + Use the **LimaCharlie API** to issue the isolation command.

Under access management in limacharlie you’d click on REST API and copy Org JWT because that is what limacharlie documents suggested to be safer.

A screenshot of a computer code

Description automatically generated

We would create a credential to add to tines so the limaCharlie can link and isolate the machine. In the Value box, that’s where to post the API

A screenshot of a computer

Description automatically generated

1. **Verify Isolation**:
   * After issuing the isolation command, verify that the machine has been successfully isolated by checking the **LimaCharlie dashboard**.

Before we Isolated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

N/B: To start another test click on re-emit in the events tag of user prompt. The ame thing goes for webhook as well. If you want to run another test.

A screenshot of a computer program

Description automatically generated

**Step 6: Sending Confirmation to Slack**

1. **Confirmation Message**:
   * Once the machine has been isolated, send a confirmation message to the Slack channel. This message should include the **isolation status** and the **computer name**.
2. **Customize the Message**:
   * Ensure the message is clear and formatted, for example:

The computer [Computer Name] has been isolated.

Isolation Status: Success

A screenshot of a computer

Description automatically generated

**Step 7: Testing the Entire Playbook**

1. **Run the Playbook**:
   * Generate a **Lasagna.exe** detection on your Windows Server VM.
   * The detection should trigger the entire Playbook, sending a Slack message, an email, and a user prompt to isolate the machine.
2. **Check for Errors**:
   * Test all branches of the Playbook:
     + If the user chooses **No**, ensure the correct message is sent to Slack.
     + If the user chooses **Yes**, verify the machine is isolated and a confirmation message is sent.