Automated GitHub Scanner for Exposed AWS IAM Keys

Overview

This n8n workflow automatically scans GitHub for exposed AWS IAM access keys associated with your AWS account, helping security teams quickly identify and respond to potential security breaches. When compromised keys are found, the workflow generates detailed security reports and sends Slack notifications with actionable remediation steps.

🔑 Key Features

- **Automated AWS IAM Key Scanning**: Regularly checks for exposed AWS access keys on GitHub
- **Real-time Security Alerts**: Sends immediate Slack notifications when compromised keys are detected
- **Comprehensive Security Reports**: Generates detailed reports with exposure information and risk assessment
- **Actionable Remediation Steps**: Provides clear instructions for securing compromised credentials
- **Continuous Monitoring**: Maintains ongoing surveillance of your AWS environment

📋 Workflow Steps

1. **List AWS Users**: Retrieves all users from your AWS account
2. **Split Users for Processing**: Processes each user individually
3. **Get User Access Keys**: Retrieves access keys for each user
4. **Filter Active Keys Only**: Focuses only on currently active access keys
5. **Search GitHub for Exposed Keys**: Scans GitHub repositories for exposed access keys
6. **Aggregate Search Results**: Consolidates and deduplicates search findings
7. **Check For Compromised Keys**: Determines if any keys have been exposed
8. **Generate Security Report**: Creates detailed security reports for compromised keys
9. **Extract AWS Usernames**: Extracts usernames from AWS response for notification
10. **Format Slack Alert**: Prepares comprehensive Slack notifications
11. **Send Slack Notification**: Delivers alerts with actionable information
12. **Continue Scanning**: Maintains continuous monitoring cycle

🛠️ Setup Requirements

Prerequisites

- Active n8n instance
- AWS account with IAM permissions
- GitHub account/token for searching repositories
- Slack workspace for notifications

Required Credentials

1. **AWS Credentials**:
    - IAM user with permissions to list users and access keys
    - Access Key ID and Secret Access Key
2. **GitHub Credentials**:
    - Personal Access Token with search permissions
3. **Slack Credentials**:
    - Webhook URL for your notification channel

⚙️ Configuration

1. **AWS Configuration**:
    - Configure the "List AWS Users" node with your AWS credentials
    - Ensure proper IAM permissions for listing users and access keys
2. **GitHub Configuration**:
    - Set up the "Search GitHub for Exposed Keys" node with your GitHub token
    - Adjust search parameters if needed
3. **Slack Configuration**:
    - Configure the Slack node with your webhook URL
    - Customize notification format if desired

🚀 Usage

Running the Workflow

1. **Manual Execution**: Click "Execute Workflow" to run an immediate scan
2. **Scheduled Execution**: Set up a schedule to run periodic scans (recommended daily or weekly)

Repository Compatibility

This workflow is compatible with both public and private GitHub repositories to which you have access. It will scan all repositories you have permission to view based on your GitHub credentials.

Handling Alerts

When a compromised key is detected:

1. Review the Slack notification for details about the exposure
2. Follow the recommended remediation steps:
    - Deactivate the compromised key immediately
    - Create a new key if needed
    - Investigate the exposure source
    - Update any services using the compromised key

⚠️ Disclaimer

This workflow template is provided for reference purposes only to demonstrate how to automate AWS IAM key exposure scanning. Please note:

- The scanning process may produce false positives as it only matches potential AWS access key patterns
- Always verify any reported exposures manually before taking action
- Disabling or deleting access keys without proper verification could have significant negative impacts on your environment
- Understand which systems and applications rely on identified access keys before deactivating them
- This template should be customized to fit your specific environment and security policies

IMPORTANT: Use this workflow with caution and only after thoroughly understanding your AWS environment. The authors of this template are not responsible for any disruptions or damages resulting from its use.

🔒 Security Considerations

- This workflow requires access to sensitive AWS credentials
- Store all credentials securely within n8n
- Review and rotate access keys regularly

📝 Customization Options

- Adjust GitHub search parameters for more targeted scanning
- Customize Slack notification format and content
- Modify security report generation for your specific needs
- Integrate with additional notification channels (email, MS Teams, etc.)

Optional: Enabling Interactive Slack Buttons

The Slack Block Kit notification format supports interactive buttons that can be implemented if you want to perform actions directly from Slack:

1. **Disable Key**: This button can be configured to automatically disable the compromised AWS IAM access key

2. **View Details**: This button can be set up to show additional information about the exposure
3. **Acknowledge**: This button can be used to mark the alert as acknowledged

To make these buttons functional:

1. **Set up a Slack Socket Mode App**:
    - Create a Slack app in the Slack API Console
    - Enable Socket Mode and Interactive Components
    - Subscribe to the `block_actions` event to capture button clicks
2. **Create an n8n Webhook Endpoint**:
    - Add a new webhook node to receive Slack button click events
    - Create separate workflows for each button action
3. **Implement AWS Key Disabling**:
    - For the "Disable Key" button, create a workflow that uses the n8n HTTP Request node to call the AWS IAM `UpdateAccessKey` API
    - Example HTTP request that can be implemented in n8n:
    - `Method: POST`
    - `URL: https://iam.amazonaws.com/`
    - `Query Parameters:`
    - `  Action: UpdateAccessKey`
    - `  AccessKeyId: AKIAIOSFODNN7EXAMPLE`
    - `  Status: Inactive`
    - `  UserName: {{$json.username}}`
    - `  Version: 2010-05-08`
4. **Update the Slack Message Format**:
    - Modify the `Format Slack Alert` node to include your webhook URL in the button action values
    - Add callback_id and action_id values to identify which button was clicked

This implementation allows for immediate response to security incidents directly from the Slack interface, reducing response time and improving security posture.