Title: Automating URL and IP Blocking on Cisco Meraki MX using Python

---

📊 Objective

This document explains how to automate the blocking of **URLs** and **IP addresses** on Cisco Meraki MX devices using a Python script and the official Meraki Dashboard API.

🧱 Background

In network operations, blocking unwanted or malicious URLs/IPs is a frequent task. Doing it manually through the Meraki Dashboard GUI is time-consuming. Automating this via Python provides a scalable, error-free, and faster method to apply content filtering rules across networks.

🗜️ Tools Used

- **Python 3.11 or later**
- **Meraki Python SDK** (`meraki`)
- **.env file** for storing sensitive info (API keys, network IDs)
- **VS Code** or any IDE
- **Command Line** (Windows CMD, PowerShell, or Terminal)

🔒 Prerequisites

1. **Meraki API Key** from your Meraki Dashboard: https://dashboard.meraki.com/account
2. **Organization ID** and a list of **Network IDs** for MX devices
3. Basic Python installed and configured

⚖️ Environment Setup

1. Create and activate virtual environment (optional but recommended)

```
python -m venv .venv
.venv\Scripts\activate
```

2. Install dependencies

```
pip install meraki python-dotenv
```

3. Project structure

```
meraki_blocker/
├── automation.py
├── urls_to_block.txt
├── ips_to_block.txt
└── .env
```

4. `.env` file format (with multiple networks)

```
MERAKI_API_KEY=your_api_key_here
MERAKI_ORG_ID=your_org_id_here

HYDERABAD=N_123456789
MUMBAI=N_987654321
CHENNAI=N_111222333
DELHI=N_444555666
```

🔧What Happens in Background

1. Authentication and Session Initialization

- The script loads the API key from the `.env` file.
- A secure session is created using `meraki.DashboardAPI`, establishing authenticated communication with the Meraki Dashboard.

2. Network Selection

- The script prompts the user to enter a network name (e.g., HYDERABAD), or type `ALL` to apply the blocking rules to every network listed in `.env`.

3. Data Loading

- URLs are loaded from `urls_to_block.txt`
- IPs are loaded from `ips_to_block.txt`

4. Fetching Current Rules from Meraki

- For each target network, the script fetches:

- Current content filtering rules
- Existing Layer 3 firewall rules

5. Merging and Updating

- The new IPs/URLs are merged with the existing rules (without duplicates).
- Each IP gets its own deny rule; URLs are added to the block list.
- The updated lists are pushed to the respective network using:
- `updateNetworkApplianceContentFiltering`
- `updateNetworkApplianceFirewallL3FirewallRules`

6. Output and Logs

- A clear message is printed for each network updated, indicating success or any issues.

🧮 Sample Execution (Windows)

```
python automation.py
```

Expected Prompts and Output:

```
Enter network name (or type 'ALL' to apply to all): all

🔧 Applying rules to HYDERABAD (N_123456789)
🔗 Done for HYDERABAD

🔧 Applying rules to MUMBAI (N_987654321)
🔗 Done for MUMBAI

...and so on...

🎉 All updates completed.
```

📊 Best Practices

- Keep your `.env` file clean and organized with consistent naming
- Avoid duplication of rules in your input files
- Validate all network names before execution

Troubleshooting

Issue	Possible Cause	Solution
`404 Not Found`	Wrong network ID or not an MX network	Use helper script or double-check `.env`
`APIKeyError`	Key missing or invalid	Ensure API key is loaded properly from `.env`
`FileNotFoundError`	Input file not found	Make sure both `urls_to_block.txt` and `ips_to_block.txt` exist

You're All Set!

You can now automate bulk URL and IP blocking either to a single selected network or across all networks listed in your .env file. This setup can be enhanced into a scalable and secure enterprise-grade workflow.
End of Document

How to Use This Automation

When we run the script file, it prompts the user to enter a network name. The user can enter a single network name or type 'ALL' to apply the blocking rules to all networks. The script then displays the output messages based on the user input