line(thin)-notch

RING Fence**:**

**ACI Datacenter Segmentation**

Date: 02/03/2025

Author: William Kimandu

Campus DC

Version: 0.1

**Document History**

**Revisions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision date** | **Revised by** | **Summary of Changes** | **Version** |
| 02/03/2025 | Infra ART DC | Initial draft | 0.1 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Review**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Department** | **Date of Issue** | **Version** |
| William Kimandu | Campus DC | Click or tap to enter a date. |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Contacts**

|  |  |  |
| --- | --- | --- |
| **Name /Team** | **Department /Role** | **Email** |
| IT OPS | TCS |  |
| IT Campus DC ART | ASML/NTT/TCS |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Contents

[1.1 Context: Background 4](#_Toc191465065)

[1.2 Approach 5](#_Toc191465066)

[1.2.1 Phase 1: Segmentation of the ACI Fabric Workloads 5](#_Toc191465067)

[1.2.2 Phase 2: Service Graphs and Security 5](#_Toc191465068)

[1.3 Secure Infrastructure 5](#_Toc191465069)

[1.4 Ansible 5](#_Toc191465070)

[1.5 Ansible: ACI environments ASML Gitlab 6](#_Toc191465071)

[1.6 GitLab 7](#_Toc191465072)

[2. Subnet and Policy Configuration 7](#_Toc191465073)

[2.1 Subnets 7](#_Toc191465074)

[2.2 Contracts 8](#_Toc191465075)

[2.3 Tags 8](#_Toc191465076)

[2.4 Selectors 8](#_Toc191465077)

[2.5 Filters 8](#_Toc191465078)

[2.6 Filter Entries 8](#_Toc191465079)

[2.7 Filter Entry Actions 8](#_Toc191465080)

[3. GitLab CI 8](#_Toc191465081)

[3.1 GitHub Actions - Pipeline 8](#_Toc191465082)

[4. YAML Configuration for NDI PCV ESG Rules 8](#_Toc191465083)

Create Git Environment for DCAuto

**ASML Gitlab Environment**

An automated solution for the deployment of ESG in the ACI fabric using Ansible and GITLAB

ASML Gitlab Environment

Author: William Kimandu

Team: Campus DC

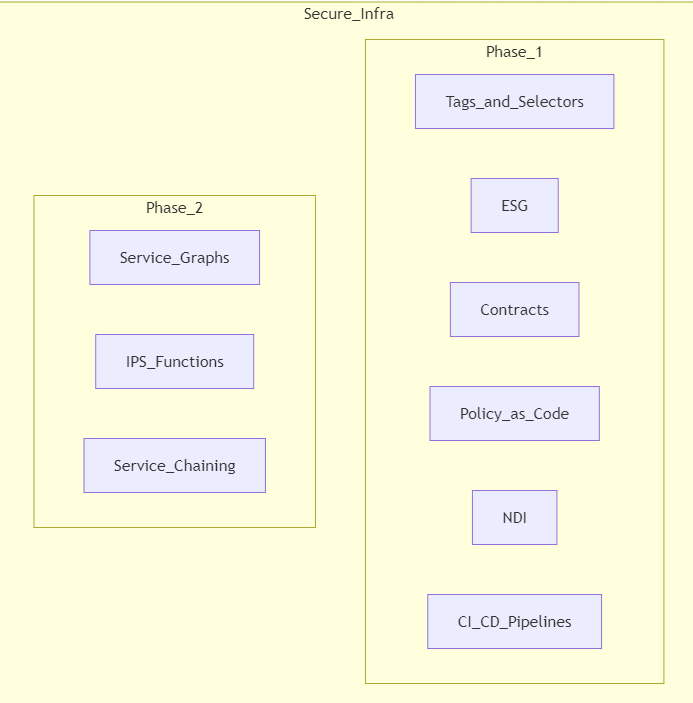
Contact Information:

Email:  
Phone:

Version: 0.4

Date: 2025-01-19

Testing the pipeline in a dummy ACI tenant98



Context: Background

Segmentation of the ACI fabric and deploying ESG in the ACI fabric (RING Fence candidates workloads).

This is automated via IAC—CI/CD pipelines—GITLAB. When IAC is achieved, add the Policy as Code framework to the solution. ESG—endpoint security groups is a granular security policy that can be applied to a group of endpoints.

We need to have a policy on group membership, and this solution is flexible in scope and scale. Secure endpoint groups can leverage the ACI contracts and filters to determine the traffic's forwarding behavior. This capability is available in the ACI fabric and can be automated using Ansible and GITLAB.

At a later stage, we may need to perform deep packet inspection, which can be achieved using the FTD and FMC. This then leverages service graph and service chaining, redirecting traffic to the IPS functions (FTD) for inspection and then back to the ACI fabric.

Approach

To achieve the key milestones of segmenting the ACI fabric and deploying ESG, we will follow a phased approach:

Phase 1: Segmentation of the ACI Fabric Workloads

* **Tags and Selectors**: Define tags and selectors for endpoint (EP) membership.
* **ESG (Endpoint Security Groups)**: Apply granular security policies to groups of endpoints.
* **Contracts**: Establish contracts between EPGs (Endpoint Groups) and ESGs.
* **Policy as Code**: Implement policies using YAML for rules and validation and Jinja Templates for dynamic configurations. Utilize the Robot Framework for rules and validation.
* **Leverage NDI**: Nexus dashboard insights for Pre-change validation and Post-change validation.
* **CI/CD Pipelines**: Use GitLab CI/CD pipelines to automate the workflow.

Phase 2: Service Graphs and Security

* **Service Graphs**: Define service graphs for traffic redirection.
* **IPS Functions**: Implement IPS functions using FTD (Firepower Threat Defense) and FMC (Firepower Management Center).
* **Service Chaining**: Redirect traffic to IPS functions for inspection and then back to the ACI fabric.
* **Policy as Code**: Implement anomaly detection and compliance rules using NDI assurance and PCV ESG rules.
* **CI/CD Pipelines**: Continue using GitLab CI/CD pipelines to automate the workflow.

To achieve the key milestones of segmenting the ACI fabric and deploying ESG, we will follow a phased approach:

Secure Infrastructure

Provide a secure infra as code solution for deploying ESG in the ACI fabric using Ansible and GITLAB. We are tasked with automating the deployment of ESG in the ACI fabric. The deployment will be done using Ansible, and the code will be stored in a GITLAB environment. The following steps will guide you on how to create a GITLAB environment in the ACI fabric.

Ansible

Collections:

* ND: ansible-galaxy collection install cisco.nd
  + Clone repos: https://github.com/CiscoDevNet/ansible-nd.git
  + README: https://github.com/CiscoDevNet/ansible-nd?tab=readme-ov-file
* ACI: ansible-galaxy collection install cisco.aci
  + Clone repos: <https://github.com/CiscoDevNet/ansible-aci.git>
  + README:https://github.com/CiscoDevNet/ansible-aci/blob/master/README.md

Roles:

* Pipelines: GitLab CI

YAML:

* Rules and Validation: Robot Framework
* Jinja Templates: Dynamic configurations

---