FERRAMENTAS DE AUTOMAÇÃO DE REDES PARA ISPS EXPLORANDO PARAMIKO, NETMIKO E NAPALM

nic.br aiGon

OLÁ, EU SOU

MATHEUS LEAL

Engenheiro de Telecomunicações pelo INATEL e Especialista em Network Security.

Produtor de Conteúdo no Youtube, fomentando o conhecimento em Tecnologia.

nic.br aiGon

SOBRE O CURSO

agenda

PARAMIKO

NETMIKO

NAPALM

- Conceitos
- Pré-requisitos
- Estrutura básica
- Laboratório

nic.br aiGon

PARAMIKO

PYTHON LIBRARIES

conceito

Em programação, uma biblioteca é uma coleção de códigos pré-compilados que podem ser utilizados posteriormente em um programa para algumas operações específicas e bem definidas. Além de códigos pré-compilados, uma biblioteca pode conter documentação, dados de configuração, modelos de mensagens, classes e valores, entre outros.

PARAMIKO

conceito

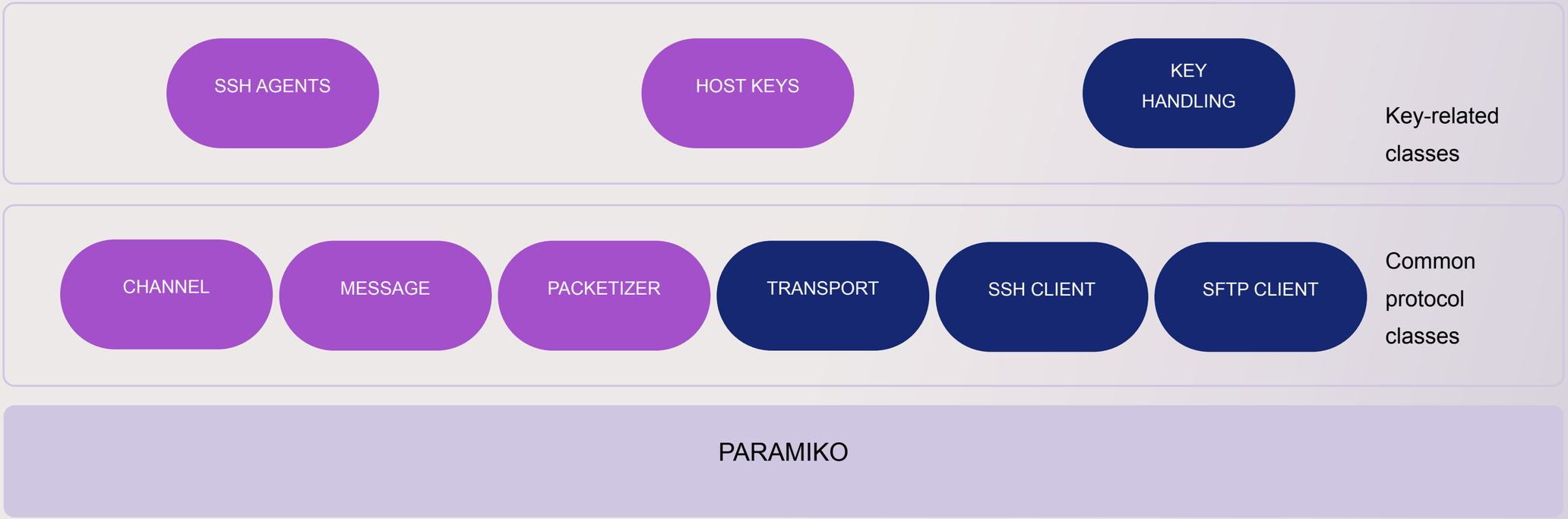
PARAMIKO é uma implementação em Python do protocolo SSHv2, atuando como cliente e servidor. Serve de base para bibliotecas SSH de alto nível e ideal para executar comandos remotos ou transferir arquivos.

Inspirado no OpenSSH, Paramiko é versátil e útil em cenários complexos.

Referência: https://github.com/paramiko/paramiko/blob/main/README.rst

PARAMIKO

arquitetura



Referência: SSH Principles and Practices - Pag 20 - HCIP-Datacom-Network Automation Developer V1.0 Training material

nic.br aiGon

PARAMIKO

pré-requisitos

PHYTON

- Necessário para usar o PARAMIKO
- Deve ser versão 3.6 ou superior

PIP

- Gerenciador de pacotes do PHYTON
- Essencial para instalar e gerenciar bibliotecas

Referência: https://github.com/paramiko/paramiko

nic.br aiGon

PARAMIKO

estrutura básica

```python
from napalm import get_network_driver
from rich import print as rprint

# Define o driver para o dispositivo IOS
huawei_device = get_network_driver("huawei_vrp")

# Define as credenciais e o endereço IP do dispositivo em um dicionário
device_credentials = {
    'hostname': '10.0.255.1',
    'username': 'netautomation',
    'password': 'Admin@123',
    'optional_args': {},  # Argumentos opcionais, se necessário
}

# Cria a instância do dispositivo com as credenciais
my_device = huawei_device(**device_credentials)

# Abre a conexão com o dispositivo
my_device.open()

# Coleta as informações do dispositivo
results = my_device.get_facts()

# Imprime as informações
rprint(results)

# Fecha a conexão com o dispositivo
my_device.close()
```

Referência: https://docs.paramiko.org/en/latest/api/client.html

nic.br aiGon

HANDS 0N

NETMIKO

NETMIKO

conceito

NETMIKO é uma biblioteca versátil projetada para simplificar conexões CLI com dispositivos de rede, desempenhando um papel crucial na automação de rede. Ao coletar dados de comandos de exibição e efetuar alterações de configuração, Netmiko se destaca. Além disso, a biblioteca se empenha em reduzir a complexidade, tornando a automação mais acessível e eficiente.

Referência: https://github.com/ktbyers/netmiko/blob/develop/README.md

nic.br aiGon

NETMIKO

plataformas suportadas

O NETMIKO oferece suporte a uma variedade de plataformas, incluindo marcas de destaque no universo dos Internet Service Providers (ISP), tais como Cisco, Huawei, Juniper e MikroTik.

Referência: https://github.com/ktbyers/netmiko/blob/develop/PLATFORMS.md

nic.br aiGon

NETMIKO

pré-requisitos

PHYTON

- Necessário para usar o NETMIKO
- Deve ser versão 3.8 ou superior

PIP

- Gerenciador de pacotes do PHYTON
- Essencial para instalar e gerenciar bibliotecas

Referência: https://github.com/ktbyers/netmiko/blob/develop/README.md

nic.br aiGon

NETMIKO

estrutura básica

```python
# Importa a classe ConnectHandler da biblioteca netmiko
from netmiko import ConnectHandler

# Cria um dicionário com as configurações necessárias para a conexão
cisco_device = {
    'device_type': 'cisco_xe',  # Define o tipo do dispositivo, neste caso um Cisco IOS XE
    'host':   '10.0.254.1',     # Endereço IP do dispositivo ao qual vamos nos conectar
    'username': 'netautomation',# Nome de usuário para a autenticação
    'password': 'Admin@123',    # Senha para a autenticação
    'port' : 22,                # Porta para a conexão SSH, 22 é o padrão
}

# Utiliza o dicionário de configuração para criar uma conexão SSH com o dispositivo
net_connect = ConnectHandler(**cisco_device)

# Envia um comando para o dispositivo e armazena a saída em uma variável
output = net_connect.send_command('show ip int brief')

# Exibe a saída do comando
print(output)
```

Referências:

https://github.com/ktbyers/netmiko/blob/develop/README.md#getting-started-1

https://ktbyers.github.io/netmiko/docs/netmiko/index.html#netmiko.ConnectHandler

https://ktbyers.github.io/netmiko/docs/netmiko/index.html#netmiko.BaseConnection

nic.br aiGon

HANDS 0N

NAPALM

NAPALM

conceito

Network Automation and Programmability Abstraction Layer with Multivendor support

É uma biblioteca Python que implementa um conjunto de funções para interagir com diferentes sistemas operacionais de dispositivos de rede usando uma API unificada. Suporta vários métodos para conectar-se aos dispositivos, manipular configurações ou recuperar dados.

Referência: https://github.com/napalm-automation/napalm

nic.br aiGon

NAPALM

benefícios

Abstração Multivendor: NAPALM fornece uma interface única para interagir com diferentes dispositivos de rede, independente do fabricante;

Facilidade de Uso: NAPALM simplifica a automação de tarefas de rede ao fornecer métodos de alto nível para tarefas comuns, como a coleta de dados de configuração e status dos dispositivos;

Integração com Ferramentas de Automação: NAPALM pode ser facilmente integrado com ferramentas populares de automação e orquestração, como Ansible, Salt, e StackStorm, permitindo a criação de fluxos de trabalho automatizados complexos e robustos.

NAPALM

pré-requisitos

PHYTON

- Necessário para usar o NAPALM
- Deve ser versão 3.7 ou superior

PIP

- Gerenciador de pacotes do PHYTON
- Essencial para instalar e gerenciar bibliotecas

Referência: https://github.com/napalm-automation/napalm?tab=readme-ov-file

nic.br aiGon

NAPALM

plataformas suportadas

_	EOS	Junos	IOS-XR (NET-CONF)	IOS-XR (XML-Agent)	NX-OS	NX-OS SSH	IOS
Driver Name	eos	junos	iosxr_netconf	iosxr	nxos	nxos_ssh	ios
Structured data	Yes	Yes	Yes	No	Yes	No	No
Minimum version	4.15.0F	12.1	7.0	5.1.0	6.1 [1]	6.3.2	12.4(20)T
Backend library	pyeapi	junos-eznc	ncclient	pyIOSXR	pynxos	netmiko	netmiko
Caveats	EOS		IOS-XR (NET-CONF)		NXOS	NXOS	IOS

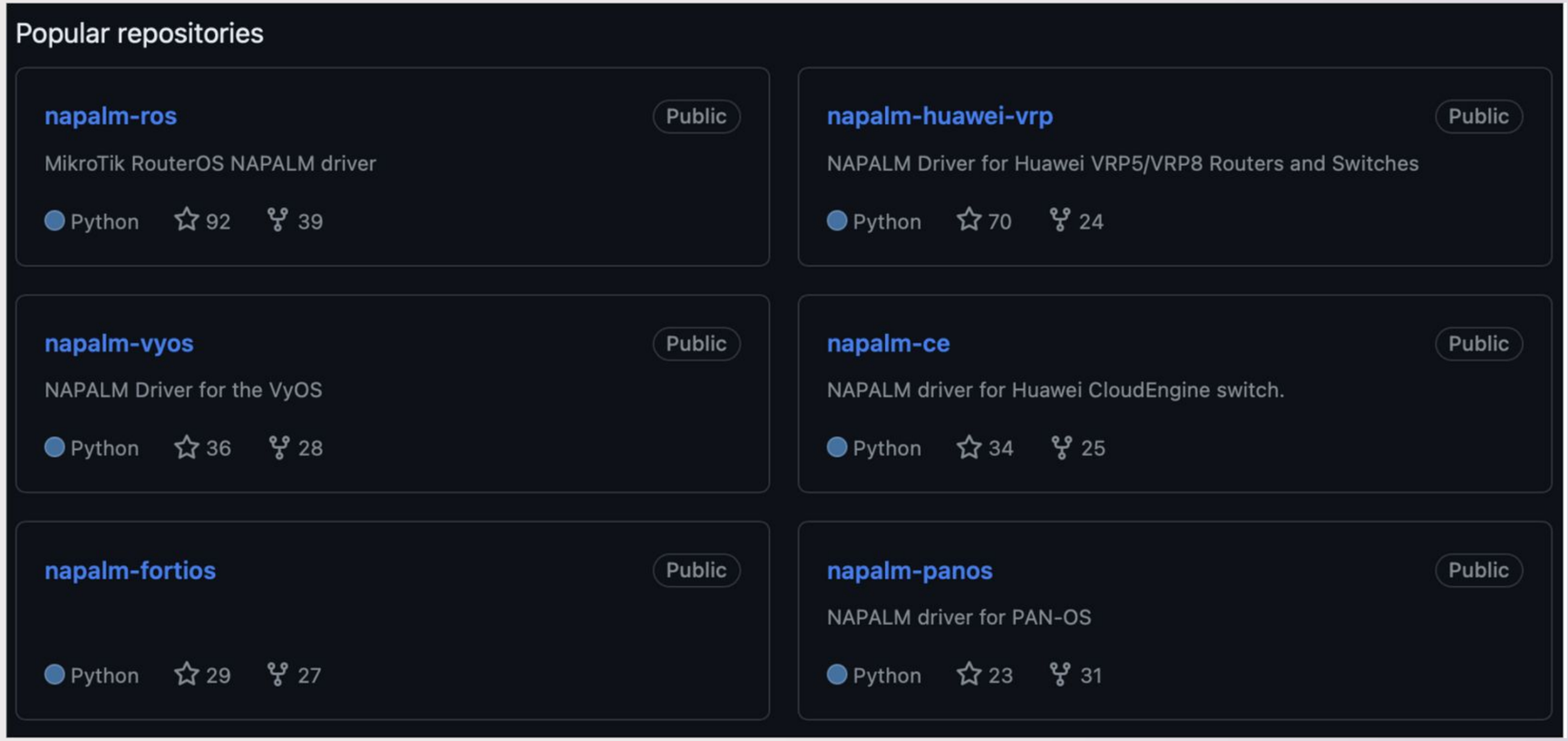
Referência: https://napalm.readthedocs.io/en/latest/support/index.html#general-support-matrix

nic.br aiGon

NAPALM

plataformas suportadas

community



Referência: https://github.com/napalm-automation-community

nic.br aiGon

NAPALM

getters suportados

	EOS	IOS	IOSXR	IOSXR_NETCONF	JUNOS	NXOS	NXOS_SSH
get_arp_table	✅	✅	❌	❌	❌	❌	✅
get_bgp_config	✅	✅	✅	✅	✅	❌	❌
get_bgp_neighbors	✅	✅	✅	✅	✅	✅	✅
get_bgp_neighbors_detail	✅	✅	✅	✅	✅	❌	❌
get_config	✅	✅	✅	❌	✅	✅	✅
get_environment	✅	✅	✅	✅	✅	✅	✅
get_facts	✅	✅	✅	✅	✅	✅	✅
get_firewall_policies	❌	❌	❌	❌	❌	❌	❌
get_interfaces	✅	✅	✅	✅	✅	✅	✅
get_interfaces_counters	✅	✅	✅	✅	✅	❌	✅
get_interfaces_ip	✅	✅	✅	✅	✅	✅	✅
get_ipv6_neighbors_table	❌	✅	❌	❌	✅	❌	❌
get_lldp_neighbors	✅	✅	✅	✅	✅	✅	✅
get_lldp_neighbors_detail	✅	✅	✅	✅	✅	✅	✅
get_mac_address_table	✅	✅	✅	✅	✅	✅	✅
get_network_instances	✅	✅	❌	❌	✅	✅	✅
get_ntp_peers	❌	✅	✅	✅	✅	✅	✅
get_ntp_servers	✅	✅	✅	✅	✅	✅	✅
get_ntp_stats	✅	✅	✅	✅	✅	✅	❌

Referência: https://napalm.readthedocs.io/en/latest/support/index.html#getters-support-matrix

nic.br aiGon

NAPALM

getters suportados

community

Get info

API	Description
get_facts()	Return general device information
get_config()	Read config
get_arp_table()	Get device ARP table
get_mac_address_table()	Get mac table of connected devices
get_interfaces()	Get interface information
get_interfaces_ip()	Get interface IP information
get_interfaces_counters()	Get interface counters
get_lldp_neighbors()	Fetch LLDP neighbor information

Config

API	Description
cli()	Send any cli commands
load_merge_candidate()	Load config
compare_config()	A string showing the difference between the running configuration and the candidate configuration
discard_config()	Discards the configuration loaded into the candidate
commit_config()	Commits the changes requested by the method load_replace_candidate or load_merge_candidate

Referência: https://github.com/napalm-automation-community/napalm-huawei-vrp

NAPALM

estrutura básica

```python
from napalm import get_network_driver
from rich import print as rprint

# Define o driver para o dispositivo IOS
huawei_device = get_network_driver("huawei_vrp")

# Define as credenciais e o endereço IP do dispositivo em um dicionário
device_credentials = {
    'hostname': '10.0.255.1',
    'username': 'netautomation',
    'password': 'Admin@123',
    'optional_args': {},  # Argumentos opcionais, se necessário
}

# Cria a instância do dispositivo com as credenciais
my_device = huawei_device(**device_credentials)

# Abre a conexão com o dispositivo
my_device.open()

# Coleta as informações do dispositivo
results = my_device.get_facts()

# Imprime as informações
rprint(results)

# Fecha a conexão com o dispositivo
my_device.close()
```

Referências: https://napalm.readthedocs.io/en/latest/index.html

nic.br aiGon

HANDS 0N