DOI: 10.33947/2316-7394-v7n1-3521

DESENVOLVIMENTO DE FERRAMENTA DE GERENCIAMENTO DE FUNCIONALIDADES DE SWITCHES, ROTEADORES E CME DA CISCO PARA OTIMIZAÇÃO DE PROCESSO DE CONFIGURAÇÃO PARA GARANTIA DE QUALIDADE COM USO DA LINGUAGEM PYTHON.

DEVELOPMENT OF A CISCO SWITCHES, ROUTERS AND CME FUNCTIONALITY MANAGEMENT TOOL TO OPTIMIZE THE CONFIGURATION PROCESS FOR QUALITY ASSURANCE USING THE PYTHON LANGUAGE.

Gabriel de Sousa Campos[1], Filipe Carvalho da Paixão[2]

RESUMO

O presente artigo demonstra como desenvolver uma ferramenta capaz de otimizar os processos de configuração e garantir qualidade nos comandos efetuados em equipamentos de rede Cisco, limitando-se a *switches, routers e callm*anager. Para tanto, é utilizado um módulo específico em Python, denominado Paramiko. Trata-se de um módulo capaz de habilitar conexões remotas em equipamentos de rede através do protocolo SSH. Os testes são efetuados em um ambiente Linux, respeitando as devidas hierarquias de acessos de usuários, como *root* ou configurador global da Cisco. Deu-se ênfase na importância do uso do protocolo SSH ao invés do Telnet, para que qualquer acesso remoto seja criptografado e por tanto seguro. O Artigo trata-se de uma demonstração simples de otimização, porém sem pretensão de respostas conclusivas em ambiente Cisco.

PALAVRAS-CHAVE: SSH. Cisco. Python. Switch. Router. Callmanager. CME. Otimização-Cisco. PABX Cisco

ABSTRACT

The present article demonstrates how to develop a tool capable of optimizing the configuration processes and guaranteeing quality in the commands made in Cisco network equipment, being limited to switches, routers and callmanager. For this, a specific module in Python, called Paramiko, is used. It is a module capable of enabling remote connections on network equipment through the SSH protocol. The tests are performed in a Linux environment, respecting the appropriate hierarchies of user access, such as root or Cisco global configurator. Emphasis was placed on the importance of using the SSH protocol instead of Telnet, so that any remote access is encrypted and therefore secure. The article is a simple demonstration of optimization, but without pretension of conclusive answers in the Cisco environment.

KEYWORDS: *SSH. Cisco. Python. Switch. Router. Callmanager. CME. Optimization-Cisco. Cisco PABX*

[1] Aluno do Curso de Graduação em Ciência da Computação da Universidade UNIVERITAS/UNG.
[2] Aluno do Curso de Graduação em Ciência da Computação da Universidade UNIVERITAS/UNG.

INTRODUÇÃO

A Cisco é uma empresa líder no fornecimento de soluções de comunicação e de equipamentos de rede, voltados para todo tipo de empresa. A Cisco trabalha com a venda e prestação de serviços para diversos equipamentos e funcionalidades como, por exemplo: *routers*, *switches*, aplicações de vídeo conferência, web conferência entre outros serviços.

Segundo um estudo realizado pela International Data Corporation (IDC*)*, a Cisco terminou o primeiro trimestre de 2018 com 7,6% de aumento e uma participação no mercado de 53,4% (Mehra, Jirovsky, & Shirer, 2018). Devido ao aumento de mercado que a Cisco obteve no primeiro trimestre de 2018, tem-se o seguinte cenário: Uma grande quantidade de empresas e poucos profissionais qualificados no mercado para gerenciar toda a estrutura de rede de maneira ágil, preservando sempre a possibilidade de expansão na rede, boas práticas de segurança, tempo necessário para configuração e a precaução com erros humanos que podem causar falhas de rede ou em alguns casos perda de configuração total do equipamento, obrigando o administrador de rede a efetuar toda a configuração do equipamento novamente.

Devido à carência de profissionais qualificados e alternativas para o controle das configurações via *command-line interface* (CLI*)* que se baseia em controlar um equipamento através de linhas de comando. Muitas empresas acabam acionando consultorias especializadas para que toda a configuração e gerenciamento dos equipamentos sejam efetuados. Essas configurações incluem: configuração de *Virtual Lan* (VLAN), gerenciamento de portas de rede, roteamento, configuração de ramais, *virtual private network* (VPN), balanceamento de carga, tolerância a falhas entre outras funcionalidades. Também é necessária a manutenção preventiva nos equipamentos, como, por exemplo, atualização de *firmware* e correções de segurança que são efetuadas na atualização do sistema *Internetwork Operating System* (IOS). Durante o processo de configuração inúmeros erros de configuração podem ocorrer, o mais comum é o comando *Register config 0x4* estar habilitado, uma vez que o equipamento é reiniciado todas as configurações efetuadas são apagadas, obrigando o administrador a configurar o equipamento novamente. Para que uma configuração efetuada através de *CLI* seja correta é necessário que o administrador tenha pleno conhecimento de boas práticas de segurança voltado para equipamentos da rede Cisco. Tais conhecimentos são adquiridos através de certificações específicas que a Cisco oferece.

Pensando no caso de pequenas e médias empresas a Cisco desenvolveu um protocolo de gerenciamento de equipamento chamado *Cisco Smart Install*, muitas empresas optaram pelo uso desse protocolo devido a proposta de facilitar a gestão e implantação dos equipamentos de rede que suportam o protocolo, porém países como China, Estados Unidos, Rússia e Irã sofreram ataques cibernéticos onde os *hackers* exploraram uma vulnerabilidade no protocolo *Cisco Smart Install* que permitiu reescrever a imagem do (IOS*)* da Cisco.

Segundo uma matéria publicada no site da Kaspersky e Cisco Talos (Kasperksy, 2018), mais de 168.000 dispositivos, sendo em sua maioria *switches* e *routers*, contêm essa vulnerabilidade. O protocolo *Cisco Smart Install* foi desenvolvido com o objetivo de servir como instrumento para os administradores de rede que trabalham com Cisco, de maneira que a gestão dos equipamentos se torne mais eficiente e simples.

A proposta do artigo é demonstrar como desenvolver uma ferramenta baseada em *Python* (open-source) capaz de gerenciar, padronizar e simplificar a configuração de *switches*, *routers* e *callmanagers* (CME) da Cisco*,* a fim de otimizar os processos de configuração que envolvam toda parte de comando via *CLI*. Desta forma o responsável pela rede estará isento de conhecimentos técnicos que só são adquiridos através de certificações fornecida pela Cisco ou empresas parceiras. De modo que o administrador poderá gerenciar os equipamentos de rede, e as empresas poderão contar com uma alternativa para gestão da estrutura sem que haja uma demanda de tempo muito alta, erros humanos relacionados a digitação, e necessidade de contatar uma empresa terceira ou especializada para efetuar a configuração dos equipamentos. Com base no cenário explorado o objetivo deste artigo é contribuir com a comunidade de redes Cisco, garantindo assim boas práticas de segurança e otimizando a qualidade e tempo do

processo de configuração, de modo que diminua significativamente a probabilidade de erro humano nas configurações via *CLI*.

MATERIAIS E MÉTODOS

Através de um estudo profundo que teve como embasamento teórico os processos de configuração necessário para se manter qualidade e segurança nas configurações efetuadas via o protocolo de comunicação *secure shell* (SSH), a finalidade deste artigo é demonstrar como desenvolver uma ferramenta que otimize os processos de configuração sem que haja a necessidade recorrente de interação humana nos comandos efetuados via linha de comando, a fim de diminuir erros de configuração e otimizar o tempo e qualidade que leva para configurar um equipamento.

O Cisco IOS é o sistema operacional padrão da empresa Cisco que é utilizado na maioria dos equipamentos como *switch* e *routers.* O Cisco IOS tem como funcionalidade permitir a comunicação na rede, roteamento, criptografia, autenticação, firewall, aplicação de políticas entre outras funcionalidades que o Cisco IOS é capaz de gerenciar.

Para análise da ferramenta desenvolvida pela Cisco com o objetivo de ser uma alternativa para pequenas e médias empresas, o *Cisco Smart Install*, foi utilizado a ferramenta de pesquisa *Shodan.* O *Shodan* é um mecanismo de busca para determinados dispositivos conectados à internet (Matherly, 2017). É possível encontrar dispositivos usando um determinado protocolo como, por exemplo, o protocolo *Cisco Smart Install*. O Shodan pode ser utilizado quase da mesma forma que o Google, como um buscador na internet. Um exemplo de busca que pode ser utilizado é a quantidade e localização dos equipamentos conectados à internet, sendo possível filtrar por país ou cidade.

Bot é um programa de computador desenvolvido para automatizar processos repetitivos para o usuário e, em alguns casos, encontrar informações de maneira automatizada (Mimi, 2016). A palavra *Bot* vem de "*Robot*", que em português significa Robô, um mecanismo programável para realizar ações de maneira remota e repetitiva. Foi identificado um *Bot* no *Shodan* que tem como finalidade exercer a função de buscador de dispositivos conectados à internet. O objetivo do *Bot* é usar o *Shodan* para encontrar os dispositivos que estão conectados e ainda permanecem com o protocolo *Cisco Smart Install* em estado ativo.

O *Cisco Smart Install* é um recurso de configuração plug-and-play - que nada mais é que o conceito de instalar e ligar - que tem como finalidade o gerenciamento de imagem para *switches e routers*. De modo que é possível enviar um equipamento para um local e instala-lo na rede e liga-lo sem a necessidade de uma programação no dispositivo (Cisco, 2018). De acordo com a Imagem 1, é apresentado um exemplo de como verificar se um determinado equipamento da Cisco tem suporte ao protocolo *Cisco Smart Install* basta digitar o comando ***show vstack config*** e o sistema *IOS* irá retornar o status do protocolo.

```
switch1      # show vstack config
      Função: Client
       .
       .
       .

    switch2 # show vstack config
      Função: Client (SmartInstall habilitado)
       .
       .
       .

    switch3 # show vstack config
      Capacidade: Client
     Oper Mode: Enabled
      Função: Client
       .
       .
       .
```

Figura 1 – Linha de comando "***show vstack config"*** que demonstra o Protocolo Habilitado.

Hacker é um indivíduo com conhecimento superior em uma determinada área (Leitão Marques Filho, 2010). Existem diversos tipos de *Hacker* sendo um deles o "*Black Hat",* o *Hacker* que se enquadra no termo "*Black Hat*" não respeita nenhuma ética e usa todo conhecimento adquirido para fins criminosos. Podemos citar como exemplo um ataque que ocorreu em novembro de 2017. *Hackers* se aproveitaram de uma falha no protocolo *Cisco Smart Install* que lhes permitiu invadir *switches* e *routers* com esse

protocolo em estado ativo, assim os malfeitores reescreveram a imagem do sistema *IOS* que os permitiu roubar informações e alterar arquivos de configuração, aproveitando para deixar uma mensagem nos equipamentos invadidos.

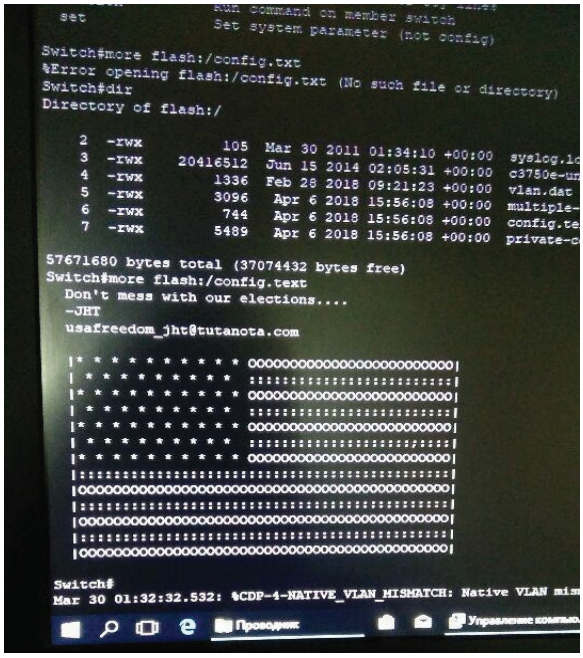

Figura 2 - Mensagem deixada pelos malfeitores "**Não mexa com nossas eleições**".

Local Area Network (LAN) é um agrupamento de computadores interligados entre si, em um mesmo local físico onde é possível a troca de informações. Dentro do contexto de uma rede local temos também o conceito de *Virtual Lan* (VLAN), é ela que divide uma rede *LAN* em grupos e permite que computadores estejam ligados fisicamente a *switches* que possam estar separados e se comunicar via rede virtual. (Mendes, 2015)

NAT (*Network Address Translation*) não é considerado um protocolo e nem se refere a um padrão, ele é apenas uma série de tarefas que um roteador ou outro equipamento equivalente deve converter endereços *IP* entre redes diferentes. Também é responsável por analisar todos os pacotes de dados que passam por ele, além de realizar essa troca de maneira correta, ou seja, ele troca um *IP* de origem pelo do roteador destinado, além de cadastrar em tabelas e relacionar a porta com o *IP* de origem com o intuito de devolver ao emissor o pedido feito. (Mendes, 2015)

Para estudo e testes de configurações básicas nos equipamentos é necessário utilizar o *Cisco Packet Tracer.* O *Cisco Packet Tracer* é um simulador de redes desenvolvido pela Cisco, sendo capaz simular redes a cabo, redes sem fio ou por fibra óptica. É possível colocar vários do mesmo equipamento interligado e realizar suas configurações de conexão com a rede, é possível também simular grandes redes tornando-a uma ótima ferramenta para quem está estudando na área de redes ou necessita de um ambiente simulado para testes. (Fernandes de Almeida, 2016)

A linguagem Python é uma linguagem de alto-nível e de fácil aprendizado, contendo diversas bibliotecas bem estruturadas. A linguagem Python possui uma sintaxe clara e concisa, que favorece a leitura do código. (Borges, 2010). Uma linguagem de alto nível como Python, tem como características: facilidade na leitura do código, facilidade na compreensão do código e facilidade na manutenção do código, e ainda se trata de uma linguagem multiplataforma. Isso porque Python trabalha juntamente com um interpretador que foi desenvolvido através das linguagens C e C++. Visto que C++ é a linguagem base para todo o contexto computacional existente, podemos encontrar diversos compiladores para praticamente todas as plataformas existentes. A linguagem Python tem uma comunidade de desenvolvedores bem ativa, devido a essa quantidade de desenvolvedores ativos é possível explorar diversos módulos que já foram desenvolvidos. É possível citar alguns exemplos de módulos desenvolvidos em Python: Ler páginas da internet, exibir gráficos ou criar planilhas. Para que o uso da linguagem Python seja possível no presente artigo é necessário efetuar a instalação do Python em suas diferentes versões para Windows ou Linux através do site oficial da organização. (Rossum, 2005)

Para desenvolvimento da ferramenta é necessário uma *Integrated Development Environment* (IDE) que suporte à linguagem Python. No presente artigo é utilizado a IDE PyCharm. Da mesma desenvolvedora e IDE (ambiente de desenvolvimento integrado) para linguagem PHP e HTML da JetBrains, o PyCharm é uma ótima opção para se programar em Python, o ambiente possui diversas ferramentas e recursos para sua utilização que facilitam o trabalho do programador na tarefa de desenvolvimento,

disponibilizando bibliotecas e temas caso seja solicitado pelo programador (Reis, 2016). Essa é uma ferramenta de desenvolvimento de software gratuita disponível tanto para Windows e Linux. Antes de fazer a utilização do software é preciso fazer o download do Python (https://www.python.org/downloads), em seguida já se pode fazer o download e instalação do PyCharm (https://www.jetbrains.com/pycharm/download/#section=windows).

O Protocolo de comunicação necessário para acessar os equipamentos remotamente é o *Secure Shell* (SSH), o protocolo SSH é um protocolo de comunicação com arquitetura cliente e servidor. O protocolo SSH possui um processo de criptografia transparente, sendo assim os usuários podem trabalhar normalmente sem saber que a comunicação na rede está criptografada (Barret & Silverman, 2001). O protocolo SSH possui compatibilidade com todas as plataformas, como por exemplo: Windows, Linux, Mac, entre outros. Tem como características autenticação para determinar a identidade do usuário de forma confiável, criptografia para que apenas o destinatário receba as informações de maneira inteligível, e integridade para que não haja a alteração nos dados transmitidos. A porta padrão para comunicação via SSH é a porta 22, podendo ser alterada com base nos requisitos de segurança pré-estabelecidos de cada empresa. (Pinto, 2013) Por padrão estabelecido pela Cisco o método de criptografia utilizada nas conexões SSH é o método *Rivest Shamir Adleman* (RSA). O método de criptografia *RSA* é uma criptografia do tipo assimétrica pois são usadas duas chaves no processo de criptografia. Tratando-se da primeira chave também conhecida como chave pública, é a chave utilizada para criptografar os dados que são enviados na rede. A segunda chave também conhecida como chave privada é a chave responsável por descriptografar os dados que foram enviados. É possível utilizar softwares gratuitos para acesso remoto de equipamentos de rede que suportam o protocolo ou até mesmo bibliotecas específicas para desenvolvedores como por exemplo o Paramiko.

Para que a conexão via SSH seja possível através da linguagem Python, é necessário utilizar um módulo específico chamado Paramiko. Paramiko é uma implementação do protocolo SSH que pode oferecer funcionalidades tanto como cliente quanto servidor (Paramiko, 2018). O Paramiko é um módulo da linguagem Python que fornece diversas funcionalidades voltadas para conexão SSH, sendo possível criar conexões e gerenciar equipamentos através da linguagem Python. Tratando-se de um módulo Python é necessário baixar o Paramiko no repositório Github e efetuar a instalação através do gerenciador de pacotes *PIP*. Após a instalação ter sido efetuada, para implementar o módulo Paramiko podemos criar uma classe em Python denominada “SSH_Connection” e após a definição do método construtor efetuarmos a instância da classe SSHClient como a seguir na imagem 3.

```
1  #!/usr/bin/python
2
3  from paramiko import SSHClient
4  import paramiko
5
6  class SSH_Connection:
7      def __init__(self):
8          self.ssh = SSHClient()
9          self.ssh.load_system_host_keys()
10         self.ssh.set_missing_host_key_policy
11         (paramiko.AutoAddPolicy())
12         self.ssh.connect(hostname='IP_HOST'
13         ,username='Usuário_Host'
14         ,password='Senha_do_usuário_host')
```

Figura 3 - Implementando a classe Paramiko e efetuando conexão com um equipamento através de um endereço *IP*.

PuTTY é um programa de acesso remoto desenvolvido para diversas plataformas com a finalidade de se comunicar com outros aparelhos através de diversos protocolos, sendo um deles o SSH que será utilizado para efetuar conexões remotas em equipamentos de rede. (Almeida Santos, 2009) O PuTTY pode ser baixado através do link: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html, sendo necessário escolher a plataforma em que ele será executado. Além disso ele pode guardar recursos e configurações para posteriormente serem executadas, além de recursos de chave para o SSH em dois tempos de entrada e saída.

Para ativar o protocolo SSH nos equipamentos cisco switch, router ou callmanager é necessário acessar o equipamento via serial console através do software gratuito Putty. Para que a primeira conexão ao equipamento seja possível via serial console, é necessário um cabo rollover com uma ponta padrão serial e o outro padrão RJ45. Também é necessário um cabo padrão console USB Serial RS232 em que uma ponta é serial e a outra USB, assim a ponta RJ45 do cabo rollover é plugada no equipamento que esta sendo configurado, na sequência a ponta serial do cabo rollover é plugada na ponta serial do cabo RS232 que por sua vez a ponta USB é plugada no computador do administrador. Após todos os cabos estarem devidamente conectados é necessário verificar a qual porta COM o dispositivo está plugado.

Para exemplo neste artigo, recomenda-se a utilização do software Putty que suporta a comunicação via Serial, SSH e Telnet, trata-se de um software gratuito com suporte a diversos sistemas operacionais. Após a conexão com o equipamento ter sido efetuada via *serial* a primeira configuração efetuada é referente ao acesso e segurança do equipamento. Através do comando ***enable*** que permite acesso ao modo configurador e posteriormente o comando ***configure terminal*** para efetivamente entrar no modo configurador, por padrão o equipamento em modo de fábrica não pede senha de autenticação, sendo necessário configurar manualmente para que o modo configurador seja acessível somente pelo administrador de rede. Através do comando ***enable password <senha definida pelo administrador>*** lançado no modo configurador, é possível configurar uma senha que será solicitada a cada vez que o administrador entrar em modo configurador. Após a senha ter sido configurada, ainda em modo administrador é necessário definir um *IP* para o equipamento, através do comando ***interface vlan 1*** é possível entrar na interface padrão do equipamento e definir um *IP* de teste através do comando ***ip add 192.168.0.1 255.255.255.0*** com a sua respectiva máscara de rede, após a configuração de *IP* do equipamento é necessário habilitar a interface de rede através do comando ***no shutdown*** e em seguida os comandos ***exit*** para sair do modo configurador e ***copy running-config startup-config*** para salvar as configurações efetuadas.

Para configurar o acesso via SSH é necessário configurar um nome de domínio ainda em acesso serial, através do comando ***ip domain-name <nome do domínio>*** no modo configurador, trata-se de uma restrição obrigatória da Cisco. Após configurar o nome de domínio é necessário configurar as duas chaves do método *RSA* através do comando ***crypto key generate rsa*****,** em seguida deve-se estipular a quantidade de bits para o módulo sendo que o mais utilizado é o valor de 512, não sendo recomendado um valor abaixo por questões de boas práticas de segurança. Após configurar as duas chaves no valor de 512 bits foi usado o comando ***line vty 0 4*** que é utilizado para definir e controlar as conexões remotas. Dentro das configurações do ***line vty 0 4***, é estipulado que o tipo de conexão será via protocolo SSH através do comando ***transport input ssh.*** Também é especificado que o sistema da Cisco irá solicitar uma autenticação para cada conexão remota efetuada, através do comando ***login local.***

Para o desenvolvimento da ferramenta é necessário primeiramente importar a biblioteca do Paramiko através do comando: ***import paramiko.*** Em seguida deve-se instanciar a classe ***SSHClient*** que foi importada através da primeira linha do script que está sendo desenvolvido, o ***import paramiko.*** Através do comando: ***self.ssh.set_missing_host_key_policy(paramiko.autoAddPolicy())*** é definido o que fazer quando a chave de um servidor não é encontrado no arquivo ***"~/.ssh/known_hosts"*** que é gerenciado pelo SSH. . O comando ***self.ssh.set_missing_host_key_policy()*** faz com que o ***Paramiko*** aceite o servidor que está sendo acessado pela primeira vez e em seguida cadastre no arquivo ***"~/.ssh/known_hosts".*** Após as configurações acima terem sido definidas corretamente, deve-se especificar o endereço *IP* do dispositivo em conjunto com usuário e senha que foi definido nas configurações de SSH do dispositivo. Para isso deve-se complementar o script com o seguinte comando: ***self.ssh.connect(hostname='ip',username='username',password='username_password')*** em que é necessário especificar os seguintes parâmetros: ***IP*** do equipamento que será acessado remotamente, ***username*** que no caso é o usuário SSH configurado previamente no equipamento e o ***username_password*** que se refe-

re a senha que foi configurado junto com o usuário. Após especificar endereço ip, usuário e senha, deve-se criar uma variável denominado session para abrir efetivamente a conexão através da linha de comando: ***session = ssh.get_transport().open_session()***.

Um *switch* ou *comutador* é um dispositivo com capacidade de dividir a rede em diferentes domínios e que permite interligar diversos dispositivos a rede através de portas físicas para que funcionem em um único segmento de rede (Pior, 2008). O número de portas de um *switch* pode variar de acordo com a regra de negócio do ambiente corporativo. É possível encontrar diversas marcas e modelos no mercado, podemos citar algumas marcas que hoje se destacam como por exemplo: Cisco, Hewlett Packard, Dell entre outras marcas. Geralmente um *switch* é escolhido levando em consideração a quantidade de portas que serão necessárias para conectar uma determinada quantidade de dispositivos. Existem *switches* de 24 portas, 48 portas e até 50 portas, que é o caso do *switch 3com 4500.* Tratando-se de um switch com portas que suportam 100 megabit por segundo, toda a configuração de portas é tratada com o prefixo “Fa”. Em alguns switches é necessário usar o prefixo “Gi” pois suportam 1000 megabit por segundo. Essa particularidade vai depender do equipamento que está sendo configurado. Na imagem a seguir podemos ver um exemplo de um método denominado “ligar_porta_switch”, passando como parâmetro “server_address” responsável por guardar o endereço IP do servidor, “server_username” responsável por guardar o usuário SSH do equipamento, “server_password” responsável por guardar a senha previamente configurada para o usuário SSH do equipamento e “interface” que está responsável por guardar o número da porta que será ligada no switch.

```
import paramiko

def configure_fastethernet_ports(
        server_address,
        server_username,
        server_password,
        interface):
```

Figura 4 – Definindo um método denominado ***“configure_fastethernet_ports”*** que passa como parâmetro o endereço do servidor, usuário SSH previamente configurado, senha do usuário SSH previamente configurado e “interface” que define a porta que será ligada no switch.

```
session.exec_command("enable")
stdin = session.makefile('wb', -1)
stdout = session.makefile('rb', -1)
stdin.write(server_password + '\n')
session.exec_command("configure terminal")
session.exec_command("interface FastEthernet 0/" + interface)
session.exec_command("no shutdown")
session.exec_command("end")
session.exec_command("write")
stdin.flush()
print(stdout.read().decode("utf-8"))
```

Figura 5 – Criando um canal de comunicação para enviar os comandos via SSH para o switch e ligando uma porta do switch no qual está sendo passado como parâmetro na chamada do método ***“configure_fastethernet_ports”***.

Callmanager é um componente para processamento e gestão de chamadas telefônicas (Cisco, 2013). O callmanager é o PABX IP da Cisco, possuindo diversas funcionalidades restritas a cada modelo de equipamento. Através de um *callmanager* é possível efetuar uma conferência com pelo menos 3 ligações simultâneas, transferência de ligações e gerenciar ramais. Um modelo que é possível ser usado como exemplo é o Cisco 2821 voltado para empresas de pequeno porte.

```
def configure_dhcp_local(
        server_address,
        server_username,
        server_password,
        ip_address,
        subnet,
        default_router
        ):
```

Figura 6 – Criando um método denominado ***“configure_dhcp_local”***, passando como parâmetro o endereço IP e máscara de rede que está sendo configurado.

```
session.exec_command("enable")
stdin = session.makefile('wb', -1)
stdout = session.makefile('rb', -1)
stdin.write(server_password + '\n')
session.exec_command("configure terminal")
session.exec_command("ip dhcp pool Voice")
session.exec_command("network " + ip_address + " " + subnet)
session.exec_command("default-router " + default_router)
session.exec_command("end")
session.exec_command("write")
```

Figura 7 – Criando um servidor DHCP local.

Um *router* é um dispositivo que tem o objetivo de interligar duas redes diferentes, sendo que a função primordial de um *router* é o reenvio de pacotes entre as duas diferentes interfaces de rede (Pior, 2008). Um *router* pode suportar diversos protocolos como por exemplo: *DHCP*, *NAT*, *VPN* entre outros. Alguns *routers* também podem ter a funcionalidade de firewall e controle de conteúdo. Para exemplo de otimização é possível criar um método denominado ***"configure_vlan"*** que passa como parâmetro o número identificador da VLAN, como mostra na imagem a seguir.

```
def configure_vlan(
        server_address,
        server_username,
        server_password,
        vlan_id):
```

Figura 8 – Criando um método chamado ***"configure_vlan",*** passando como parâmetro o número identificador na variável "vlan_id" que o administrador deseja configurar na VLAN.

```
session.exec_command("enable")
stdin = session.makefile('wb', -1)
stdout = session.makefile('rb', -1)
stdin.write(server_password + '\n')
session.exec_command("configure terminal")
session.exec_command("vlan " + vlan_id)
session.exec_command("exit")
session.exec_command("write")
stdin.flush()
print(stdout.read().decode("utf-8"))
```

Figura 9 – Criando um canal de comunicação para enviar os comandos via SSH e criando uma VLAN com base no parâmetro que foi passado na chamada do método ***"configure_vlan"***.

DISCUSSÃO

Primeiramente para a decisão do desenvolvimento da ferramenta foi necessário estudar a situação atual do procedimento de configuração nos equipamentos. Para estudo do atual procedimento que é realizado pelos administradores que se baseia em configurar os equipamentos via Cisco Smart Install, foi identificado através do buscador Shodan que diversos equipamentos estão com o protocolo Cisco Smart Install ativo, tornado os equipamentos vulneráveis uma vez que no ano de 2017 houve um ataque que foi baseado no protocolo Cisco Smart Install. Os hackers desenvolveram um bot que utilizou o buscador Shodan para capturar todos os equipamentos da Cisco que estavam utilizando o protocolo.

Para que as configurações efetuadas sigam boas práticas de segurança, é necessário que o administrador de redes possua uma certificação que é retirada após a avaliação dos conhecimentos adquiridos pelo curso efetuado. A certificação necessária para administrar *switches* e *routers* é a *CCNA Routing & Switching*. A certificação *CCNA Routing & Switching* engloba configurações básicas de *switch* e *routers* como configurações relacionadas a log de dados, portas, *VLAN* entre outras funcionalidades. O custo médio do exame final para a emissão do certificado tem um custo total de US$ 300,00.

Com a intenção de otimizar linhas de comando fazendo uso da linguagem Python que de acordo com (Vicente Rocha, 201?) uma automação de linhas de comando pode passar de 769 para 27 linhas de comando para apenas 27 linhas em um processo de configuração de 12 switches tendo um aproveitamento de 96%, e o processo de configurar 12 routers pode passar de 176 linhas para 19 linhas com apenas 89% de aproveitamento. A linguagem Python é uma linguagem de alto-nível e de fácil aprendizado, contendo diversas bibliotecas bem estruturadas, uma dessas bibliotecas é o Paramiko que é utilizado para que se tenha uma conexão via protocolo SSH, sem a utilização desse protocolo não seria possível ativar a conexão, além de ser capaz de criar conexões e gerenciar outros dispositivos.

Uma conexão SSH tem a mesma função que o protocolo Telnet, acessar equipamentos de rede remotamente em uma rede interna ou externa. O pro-

tocolo SSH é um protocolo de comunicação cliente servidor que possui um processo de criptografia forte e transparente, diferente do protocolo Telnet no quesito de segurança pois o protocolo Telnet não possui criptografia e trabalha com base na ideia de texto interativo. Baseado no funcionamento do protocolo Telnet é possível capturar todos os comandos que estão sendo realizados em uma conexão remota através de um software capaz de capturar protocolos, o que torna o protocolo Telnet vulnerável a ataques.

Devido ao alto custo de um equipamento Cisco, foi necessário avaliar possíveis sistemas parecidos com a hierarquia de acesso IOS para criação de um ambiente de testes parecido, ou seja, um sistema em que houvesse diversos níveis de acesso como usuário root do Linux ou modo configurador da Cisco, de modo que fosse possível simular um sistema IOS afim de obter um resultado próximo do esperado.

Devido a grande comunidade de desenvolvedores Python, foi analisado que, demonstrar como desenvolver uma ferramenta para otimizar os processos de configuração de equipamentos Cisco, atrairia um público cada vez maior para o desenvolvimento voltado para redes, trazendo novos módulos e ampliando as possibilidades de configurações que podem ser efetuadas via Paramiko.

O desenvolvimento da ferramenta discutida no presente artigo limita-se às microempresas que compõem de 9 até 19 empregados, empresas de pequeno porte que compõe de 10 até 49 empregados e empresas de médio porte que compõe de 50 a 99 empregados, para empresas de comércio e serviços, segundo anuário publicado pela empresa Sebrae. (Sebrae, 2013)

RESULTADO ESPERADO

Espera-se que seja possível obter uma otimização nos processos de configuração via linha de comando, fornecendo a comunidade de rede cisco uma alternativa viável e prática para gerenciar *switches*, *routers* e *callmanager*. Através do módulo Paramiko espera-se que seja possível diminuir a quantidade de comandos efetuados para configurar funcionalidades simples dos equipamentos, desenvolvendo métodos ou até mesmo classes em Python que possivelmente possam ser implementados em sistemas mais robustos com uma interface gráfica mais amigável. Com a utilização do protocolo SSH através do módulo Paramiko, espera-se que a segurança envolvida no processo de configuração seja mais efetiva do que as configurações efetuadas via protocolo Telnet. Com a demonstração do desenvolvimento da ferramenta idealizada como Open-Source, espera-se que a comunidade de desenvolvimento Python seja capaz de desenvolver sistemas mais simples a fim de diminuir a demanda de profissionais qualificados na área de redes Cisco, criando uma cultura maior no que se refere à segurança fornecida pelo protocolo SSH.

CONCLUSÃO

A demonstração do presente artigo possibilitou uma análise no processo de configuração de switches, routers e callmanager em que foi possível demonstrar uma alternativa simples e gratuita para a comunidade de redes Cisco. Com o aumento que a Cisco obteve no primeiro trimestre de 2018, a demanda de profissionais qualificados para efetuar as configurações nos equipamentos envolvidos no presente artigo aumentou, devido a essa situação, através do estudo e demonstração de como otimizar os processos de configuração, espera-se que seja possível fornecer a comunidade de desenvolvedores Python uma alternativa para desenvolver interfaces amigáveis que se utilizam da ferramenta que foi demonstrada no artigo afim de suprir a necessidade de profissionais qualificados, facilitando o processo de configuração e fornecendo mais oportunidade de emprego para administradores de rede sem experiência com equipamentos Cisco.

Com o ambiente de teste Linux, foi possível demonstrar como desenvolver uma ferramenta que envia um pacote de comandos via SSH respeitando a hierarquia de acessos, porém como os testes foram realizados em um ambiente Linux devido a falta de acesso a um equipamento Cisco, o resultado em ambiente Cisco tornou-se inconclusivo. Espera-se que em um ambiente Cisco a conexão e envio dos comandos seja possível.

REFERÊNCIAS

ALMEIDA SANTOS, V. P. **Telecom**. 2009. Disponível em: Telecom.uff: http://www.telecom.uff.br/pet/petws/downloads/dicas/DicasPetTeleSsh.pdf. Acesso em: 27 nov. 2018.

BARRET, D.; SILVERMAN, R. **SSH, the secure shell:** the definitive guide. 2001. Disponível em: http://dbmanagement.info/Books/Others/SSH_The_Secure_Shell_The_Definitive_Guide.pdf. Acesso em: 6 de set. 2018.

BORGES, L. E. **Python para desenvolvedores**. Rio de Janeiro, Brasil, 2010. Acesso em: 7 de set. 2018. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/3252265/mod_resource/content/1/b_Borges_Python_para_desenvolvedores_2ed.pdf>. Acesso em: 7 de set. 2018.

CISCO. **Cisco unified communications manager administration guide**, Release 10.0(1), 2013. Disponível em: <https://www.cisco.com/c/en/us/td/docs/voice_ip_comm/cucm/admin/10_0_1/ccmcfg/CUCM_BK_C95ABA82_00_admin-guide-100.html>. Acesso em: 12 de mai. 2018.

CISCO. **Smart install configuration guide**. 2018. Disponível em: Cisco.com: <https://www.cisco.com/c/en/us/td/docs/switches/lan/smart_install/configuration/guide/smart_install/concepts.html>. Acesso em: 6 set. 2018.

DIAS, K. L.; SILVA, D. **CISCO packet tracer**. Disponível em: <https://docente.ifrn.edu.br/jeffersonduarte/disciplinas/redes-de-computadores-e-aplicacoes/aulas/tutorial-sobre-o-cisco-packet-tracer>. Acesso em: 23 out. 2018.

FERNANDES DE ALMEIDA, A. **Academia**. 2016. Disponível em: <https://www.academia.edu/29628834/Impla%C3%A7%C3%A3o_de_redes_Cisco_Packet_Tracer>. Acesso em: 26 out. 2018.

KASPERKSY. **Kasperksy lab**. 2018. Disponível em: <https://www.kaspersky.com/blog/cisco-apocalypse/21966/>. Acesso em: 6 de abr. 2018.

LEITÃO MARQUES FILHO, G. **Hackers e Crackers na internet:** as duas faces da moeda. *Revista eletrônica Tematica*, 2010. Disponível em: <http://www.insite.pro.br/2010/janeiro/hackers_crackers_internet.pdf>. Acesso em: 28 de out. 2018.

MATHERLY, J. **Complete guide to shodan**. Em J. Matherly, Complete Guide to Shodan (p. 1), 2017. Disponível em: https://the-eye.eu/public/Books/qt.vidyagam.es/library/zz_unsorted/shodan.pdf. Acesso em: 29 de out. 2018.

MEHRA, R.; JIROVSKY, P.; SHIRER, M. **IDC**. 2018. Disponível em: <https://www.idc.com/getdoc.jsp?containerId=prUS43900818>. Acesso em: 7 de set. 2018.

MENDES, D. R. **Redes de computadores:** teoria e prática, 2015.

MIMI. **What is a bot?** 2016. Disponível em: <https://cdn2.hubspot.net/hubfs/53/assets/hubspot.com/research/reports/What_is_a_bot_HubSpot_Research.pdf?t=1492209311951>. Acesso em: 6 set. 2018.

PARAMIKO. **Paramiko.org.** 2018. Disponível em: <https://media.readthedocs.org/pdf/paramiko-www/latest/paramiko-www.pdf>. Acesso em: 19 set. 2018.

PINTO, R. D. **Renesp**. 2013. Disponível em: <http://renesp.com.br/files/materiais/2013/ssh.pdf>. Acesso em: 27 nov. 2018.

PIOR, R. **dd.fc.up.pt.** 2008. Disponível em: <http://www.dcc.fc.up.pt/~rprior/1314/LR/Equipamento.pdf>. Acesso em: 25 de nov. 2018.

ROSSUM, G. V. **Universiadade Federal do Rio de Janeiro.** 2005. Disponível em: <https://www.dcc.ufrj.br/~fabiom/python/tutorialpython.pdf>. Acesso em: 23 de nov. 2018.

SEBRAE. **Serviço brasileiro de apoio às micro e pequenas empresas**. 2013. Disponível em: <http://www.sebrae.com.br/Sebrae/Portal%20Sebrae/Anexos/Anuario%20do%20Trabalho%20Na%20Micro%20e%20Pequena%20Empresa_2013.pdf>. Acesso em: 9 de set. 2018.

VICENTE ROCHA, U. **Editorarealize**. Disponível em: https://www.editorarealize.com.br/revistas/joinbr/trabalhos/TRABALHO_EV081_MD4_SA39_ID2412_15092017194938.pdf. Acesso em: 15 out. 2018.