ПРИМЕНА СКРИПТЕ ЗА АУТОМАТИЗАЦИЈУ ПРОЦЕСА КОНФИГУРИСАЊА РАЧУНАРСКЕ МРЕЖЕ

Никола Курбалија[1] Горан Зајић[2] Милан Павловић[3] Марија Зајегановић[4] Слободан Чабаркапа[5]

Резиме: Убрзани развој рачунарских мрежа допринео је побољшању квалитета услуга и живота. Све већи број уређаја укључен је у Интернет ствари (*IoT*), што омогућава да се многи процеси убрзају и да се споје ствари које до сада нису могле да се споје. У позадини свега налази се инфраструктура која то омогућава. Да би се данашњи стандарди, којих је сваког дана све више, испоштовали, инфраструктура на којој почивају данашњи сервиси и услуге је често велика и комплексна. Велики број уређаја је међусобно повезан и синхронизовано ради у циљу непрестаног пружања најквалитетније могуће услуге. Конфигурација и одржавање оваквих система представља прави изазов и захтева да се ти процеси што је могуће више аутоматизују. Један од начина да се процеси конфигурације и одржавања мреже аутоматизују је помоћу програмских језика, односно, софтверски дефинисаним умрежавањем (*SDN*). У овом раду ће бити представљен процес конфигурације рачунарске мреже помоћу скрипте написане у програмском језику *Python*.

Кључне речи: *IoT (Internet of Things), SDN (Software defined networking), Python*, конфигурација, скрипта

USING A SCRIPT FOR AUTOMATING THE PROCESS OF CONFIGURING AND MAINTAINING A COMPUTER NETWORK

Abstract: Accelerated development of computer networks has contributed in an increasing quality of service and life overall. More and more devices are becoming a part of Internet of Things (IoT), which enables for many processes to be accelerated and for many things to be connected, which was hard to imagine in the past. Beneath lies the infrastructure that makes all of that possible. To meet evergrowing modern day standards, the underlying infrastructure is often big and complex. Huge amount of devices is interconnected and is working in sync to provide best possible quality of service. Configuring and maintaining such systems can be a real challenge, and often demands for those processes to be automated as much as possible. One of the ways to automate processes of configuring and maintaining computer networks is through the use of programming languages, which is also known as Software Defined Networking (SDN). In this paper, a process of configuring a computer network using a script written in Pyhton programming language, will be presented.

Key words: *IoT (Internet of Things), SDN (Software defined networking), Python*, configuration, script

1. УВОД

Један од начина да се процес конфигурације и одржавања мрежних уређаја аутоматизује је употреба програмских језика, односно софтверски дефинисано умрежавање. Аутоматизацијом датих процеса постиже се велика уштеда у количини времена коју је неопходно уложити у њих. Поред уштеде у количини времена које је потребно уложити, могуће је извршити задатке које не би било практично извршити мануелно. Када је реч о алатима који се користе за аутоматизацију, *Python* програмски

[1] Спец.струк.инж, Академија техничко-уметничких струковних студија Београд одсек Висока школа за информационе и комуникационе технологије, Здравка Челара 16, Београд, e-mail: nikola.kurbalija@ict.edu.rs
[2] Др, Академија техничко-уметничких струковних студија Београд одсек Висока школа за информационе и комуникационе технологије, Здравка Челара 16, Београд, e-mail: goran.zajic@ict.edu.rs
[3] Др, Академија техничко-уметничких струковних студија Београд одсек Висока школа за информационе и комуникационе технологије, Здравка Челара 16, Београд, e-mail: milan.pavlovic@ict.edu.rs
[4] Мр, Академија техничко-уметничких струковних студија Београд одсек Висока школа за информационе и комуникационе технологије, Здравка Челара 16, Београд, e-mail: marija.zajeganovic@ict.edu.rs
[5] Дипл.инж., Академија техничко-уметничких струковних студија Београд одсек Висока школа за информационе и комуникационе технологије, Здравка Челара 16, Београд, e-mail: slobodan.cabarkapa@ict.edu.rs

језик се издваја. То је последица велике заједнице која сваког дана доприноси развоју тог програмског језика. Велика заједница значи велики број доступних библиотека, које су битне за развој и симплификацију кода. Велика заједница, такође, значи и велики број ресурса доступних за учење, као и велики број људи доступних као помоћ у решавању проблема. Још једна добра карактеристика овог програмског језика је то што је програмски језик високог нивоа. Обично не постоји потреба за писањем додатног кода, који ће спојити друга два дела кода. Није потребно мануелно чистити искоришћену меморију и радити алокацију и делокацију исте. Све претходно наведене карактеристике и предности *Python* програмског језика чине да буде један од примарних избора при избору алата за аутоматизацију рада рачунарских мрежа [2].

У овом раду, биће представљена употреба програмског језика *Python* и јавно доступних библиотека при аутоматизацији процеса конфигурације рачунарских мрежа. Као развојно окружење у коме је написан код, коришћен је *PyCharm*, а уређаји, чија конфигурација се аутоматизује користе *Cisco IOS* оперативне системе.

2. PYTHON ПРОГРАМСКИ ЈЕЗИК

2.1. Настанак прогамског језика Python

Програмске језике обично развијају велике компаније са доста запослених који се деле у тимове. Због тога обично није могуће повезати име неког појединца са целим пројектом. Једна занимљива чињеница везана за *Python* јесте то што представља (већински) дело једне особе. Творац програмског језика *Python* је Гвидо Ван Росум, рођен 1956. године у Холандији. Наравано, Гвидо Ван Росум није развио и унапредио све компоненте програмског језика сам. Брзина којом се овај програмски језик шири пре свега је резултат непрекидног рада великог броја програмера, тестера, корисника и ентузијаста. Гвидо Ван Росум је 1999. године дефинисао 4 циља за *Python :*

- Да буде лак и интуитиван језик, моћан попут оних које праве главни конкуренти
- Да код буде отворен и доступан свима, како би могли да допринесу даљем развоју
- Да тај код буде разумљив, попут енглеског језика
- Да буде прикладан за свакодневне задатке

Након више од двадесет година, јасно је да су ови циљеви постигнути [3].

2.2. Одлике Python-а

Python је програмски језик високог нивоа, опште намене. Најчешће користи интерпретатор. Синтакса је таква да омогућава писање веома прегледних програма и врло лако се учи [1].

3. СОФТВЕРСКИ ДЕФИНИСАНО УМРЕЖАВАЊЕ

3.1. Почетне конфигурације

Део процеса конфигурације мрежних уређаја, који није аутоматизован, уско је повезан за безбедност истих. Да би са централизованог места конфигурисали и одржавали уређаје, потребно је успоставити безбедну везу са њима. Неопходно је припремити мрежне уређаје са основном конфигурацијом, која подразумева *IP* адресу на виртуелном интерфејсу, као и подешавање *SSH* протокола на виртуелним линијама рутера.

На слици 1 је приказан пример почетне конфигурације на једном од мрежних уређаја у топологији. У датој конфигурацији приказује се начин на који се кофигуришу *IP* адреса и *SSH* протокол, који су кључни за успостављање конекције са удаљене локације путем *Python* скрипте.

```
username nikola privilege 15 password 0 cisco
line vty 0 4
 login local
 transport input ssh
 exit
hostname Core_SW1
ip domain-name master2045.ict
crypto key generate rsa modulus 1024
interface vlan 1
 no shutdown
 ip address 192.168.121.3 255.255.255.0
 exit
```

Слика 1 – Пример почетне конфигурације

3.2. Учитавање модула

Након што је извршена почетна конфигурација на мрежним уређајима, потребно је припремити и *Python* скрипту за приступ мрежним уређајима и отварање конекције. Претходно је у раду напоменуто колико је важно то што је заједница људи, која користи програмски језик Python велика. То се пре свега огледа у великом броју доступних библиотека. Велики број доступних библиотека значи и могућност да се код и функције које се налазе у њима искористе. Уместо писања кода који би омогућио остварење везе са оперативним системом рутера, могуће је искористити код који је већ неко написао и учинио јавно доступним.

Први део скрипте садржи линију кода којом се самом оперативном систему назначава да је у питању скрипта која је написана у *Python* програмском језику. Након тога, увозе се модули, релевантни за реализацију пројекта.

```
#!/usr/bin/env python

#Linija koda kojom se uvoze moduli neophodni za izradu projekta
from netmiko import ConnectHandler
```

Слика 2 – Први део скрипте

Библиотека коришћена за израду овог рада је *netmiko*. На слици 2 је приказана линија кода, којом се из поменуте библиотеке увозе модули потребни за отварање конекције према мрежним уређајима. Модул коришћен у изради овог рада је *ConnectHandler* из библиотеке *netmiko*.

3.3. Дефинисање мрежних уређаја

Након учитавања библиотеке и модула, неопходно је дефинисати тип мрежних уређаја који се налазе у топологији, као и *IP* адресе преко којих им је могуће приступити. Ово се постиже употребом речника у *Python* програмском језику. Кључне вредности, које су битне за остваривање конекције са мрежним уређајима су тип оперативног система који ти уређаји користе, њихова *IP* адреса, корисничко име и

лозинка, који су подешени и који се користе при безбедном повезивању путем *SSH* конекције. Корисничко име, лозинка, као и *IP* адреса подешени су у првом делу припреме, као склоп почетне конфигурације мрежних уређаја, која се врши мануелно.

```
8   CoreSW1 = {
9       'device_type': 'cisco_ios',
10      'ip': '192.168.121.3',
11      'username': 'nikola',
12      'password': 'cisco',
13  }
14  CoreSW2 = {
15      'device_type': 'cisco_ios',
16      'ip': '192.168.121.4',
17      'username': 'nikola',
18      'password': 'cisco',
19  }
```

Слика 3 – Дефинисање мрежних уређаја

На слици 3 су приказане линије кода којима се дефинишу два уређаја из топологије. На тај начин дефинисани су и остали уређаји, који се у топологији налазе. Назив речника асоцира на име самог мрежног уређаја. Као додатан начин да се сама рачунарска мрежа заштити, могуће је вредности које одговарају корисничком имену и шифри учитати из засебног текстуалног фајла.

4. КОНФИГУРИСАЊЕ УРЕЂАЈА

4.1. Конфигурисање приступног слоја

Након што су сви уређаји у топологији дефинисани, могуће је отворити конекције ка истим и проследити им конфигурационе фајлове. Први на реду су уређаји који се налазе у приступном слоју.

```
38  #Definisanje access sviceva
39  access_devices = [AccessSW1, AccessSW2]
40  i = 1
41  for device in access_devices:
42      #Priprema konfiguracionih fajlova
43      with open('ios_access'+str(i)+'_config') as cfg:
44          access_cfg = cfg.read().splitlines()
45          print(access_cfg)
46      #Otvaranje konekcije prema access svicevima
47      net_connect = ConnectHandler(**device)
48      output = net_connect.send_config_set(access_cfg)
49      print(output)
50      i = i + 1
```

Слика 4 – Отварање конекције и слање конфигурационих фајлова уређајима у приступном слоју

Помоћу *for* петље, приказане на слици 4, аутоматски се пролази кроз све приступне свичеве, који се налазе у листи званој *access_devices*. Променљива *i* представља бројач, који се инкрементује сваким проласком кроз петљу. Та променљива је касније искоришћена за учитавање конфигурационих фајлова. Конфигурациони фајлови

приступних свичева разликују се у редном броју. Први пут, када се изврши *for* петља, биће учитан конфигурациони фајл за први приступни свич. Следећи пролазак кроз петљу, бројач ће бити једнак броју два, што значи да ће бити учитан конфигурациони фајл за други приступни свич. Због тога је битно у листу редом додати мрежне уређаје, од оног са најмањим редним бројем, ка највећем. Такође, неопходно је водити рачуна и исправно именовати саме конфигурационе фајлове.

Када је конфигурациони фајл учитан, потребно је команде које се у њему налазе негде сачувати. Због тога је креирана локална променљива *access_cfg*. Све што се налази у конфигурационом фајлу, чува се на кратко у тој локално променљивој. Због прегледности, функцијом *print* се исписује садржај променљиве. Након тога, отвара се конекција ка мрежном уређају, помоћу функције *ConnectHandler*. Пошто је функција позвана у оквиру петље, конекцију није неопходно затварати. Када програм прође кроз тело петље, конекција се аутоматски затвара. Ово није случај када се та функција користи ван петље. Методом *send_config_set* уређају ка којем је отворена конекција шаље се претходно сачувани сет команди *access_cfg.* Да би се водило рачуна о евентуалним грешкама и да би све било прегледно, функцијом *print* се исписује све што се на рутеру догоди, након што му се проследи сет команди. На слици 5 је приказан део конфигурационог фајла, касније сачуваног у сету команди.

```
1   vtp mode transparent
2   spanning-tree mode rapid-pvst
3   errdisable recovery cause all
4   ip name-server 8.8.8.8
5   vlan 10
6    name ServerRoom
7   vlan 20
8    name Office1
9   vlan 30
10   name Office2
11  vlan 40
12   name Office3
```

Слика 5 – Приказ дела конфигурације једног од уређаја

4.2. Конфигурисање језгра мреже

На слици 6 је приказан део скрипте за конфигурацију уређаја у језгру мреже.

```
52  #Pripremanje konfiguracionog fajla za core sviceve
53  with open('ios_core_config') as cfg:
54      core_cfg = cfg.read().splitlines()
55      print(core_cfg)
56  #Definisanje core sviceva
57  core_devices = [CoreSW2, CoreSW1]
58  #Otvaranje konekcija prema core svicevima
59  for device in core_devices:
60      net_connect = ConnectHandler(**device)
61      output = net_connect.send_config_set(core_cfg)
62      print(output)
```

Слика 6 – Приказ дела скрипте у коме се врши конфигурација уређаја у језгру мреже

На исти начин на који је извршено конфигурисање приступног слоја, врши се и конфигурисање језгра мреже. Уређаји који припадају језгру мреже групишу се у посебну листу. Садржај конфигурационог фајла чува се у променљивој, а касније се прослеђује самом мрежном уређају.

С обзиром да уређаји у језгру користе исту конфигурацију у овом примеру, није неопходно користити бројач у *for* петљи. Након отварања конекције, методом *send_config_set*, као и код приступних уређаја, прослеђује се сет команди који се чувају у променљивој.

4.3. Конфигурисање рутера у језгру мреже

Рутер који се налази у језгру мреже, логички је раздвојен од осталих уређаја који се ту налазе. То је последњи уређај у топологији који је неопходно конфигурисати. За разлику од приступног слоја и осталих уређаја у језгру мреже, сада нема групе уређаја коју је неопходно конфигурисати, те се губи потреба за *for* петљом. Довољно је конфигурациони фајл рутера учитати, сачувати у једну променљиву, а затим отворити конекцију ка рутеру. Овог пута неопходоно је, након што се рутеру проследи сет команди, затворити конекцију која је отворена. У претходним случајевима када се из петље *for* изађе, конекција се аутоматски затварала.

```
64  #Pripremanje konfiguracije rutera
65  with open('ios_router_config') as cfg:
66      router_cfg = cfg.read().splitlines()
67      print(router_cfg)
68  #Otvaranje konekcije prema ruteru
69  net_connect = ConnectHandler(**CoreRouter)
70  #Prosledjivanje konfiguracije i prikazivanje rezultata
71  output = net_connect.send_config_set(router_cfg)
72  print(output)
73  net_connect.disconnect()
```

Слика 7 – Приказ дела скрипте у коме се врши конфигурација рутера

На слици 7 је приказано затварање претходно отворене конекције према рутеру које се постиже коришћењем методе *disconnect.*

Последњи корак је покретање саме скрипте. Неопходно је обезбедити да рачунар, са кога се скрипта покреће, буде повезан са уређајима у мрежи и да им може приступити преко *IP* адреса које су наведене у речницима.

5. ЗАКЉУЧАК

Развој рачунарских мрежа уско је повезан са аутоматизацијом процеса. Најлакши начин да се процеси аутоматизују је помоћу програмских језика. *Python* програмски језик се издваја, због своје природе и великог броја људи који га користи баш у ту сврху. Већ спремне библиотеке омогућавају чак и мање искусним програмерима, поготово онима који долазе из света мрежног инжењерства, да се на врло једноставан начин укључе у процес и почну да аутоматизују. Многе индустријске обуке већ подразумевају рад са разним алатима за аутоматизацију, између осталог и са *Python* програмским језиком.

Иако постоји још доста простора за напредак, процес преласка на софтверски дефинисано умрежавање је увелико започео. Питање је времена када ће и процеси које још увек радимо мануелно бити аутоматизовани. *Python* ће по свему судећи бити једна од главних алатки за постизање тог циља. Аутоматизација процеса, како у свету рачунарских мрежа, тако и у другим сферама *IT* сектора и живота, требало би да допринесе побољшању квалитета живота и услуга.

6. ЛИТЕРАТУРА

[1] Chou, Eric – Mastering Python networking. Your one-stop solution to using Python for network automation, DevOps, and test-driven development (2018, Packt Publishing)

[2] Zajeganović M., Zajić G., Kurbalija N., Pavlović M., Čabarkapa S. (2019) *Simulating computer networks with Mininet*. 5th International conference on Knowledge management and informatics. Kopaonik. Serbia. ISBN 978-86-6211-115-9

[3] Cisco Networking Academy (2018): *PCAP: Programming Essentials in Python*. San Jose: Cisco