Wojskowa Akademia Techniczna

Wydział Elektroniki

**Bezpieczeństwo informacyjne w sieciach programowalnych**

**(Data-Centric Security in Software Defined Networks)**

Zapoznanie oraz praca ze środowiskiem Mininet

Opracował:

dr inż. Konrad Wrona

mgr inż. Sebastian Szwaczyk

Warszawa 2017

**1. Wstęp**

Laboratorium ma na celu zapozananie studenta ze środowiskiem do emulacji sieci SDN o nazwie Mininet oraz protokołem OpenFlow.

**2. Wprowadzenie do środowiska Mininet**

Mininet (<http://mininet.org/>) jest narzędziem pozwalającym na stworzenie realistycznej sieci wirtualnej SDN na poziomie jądra systemu. W tym celu wykorzystywane są wydzielone przestrzenie nazw oraz wirtualne interfejsy.

W laboratorium mininet został pobrany do katalogu /root/mininet i zainstalowany za pomocą skryptu install.sh z opcją -n (instalacja mininet) oraz opcją -f (instalacja kontrolera openflow).

Aby uruchomić emulator mininet wystarczy wydać polecenie:

*$ sudo mn*

Aby wyświetlić pomoc pokazującą możliwe paramtery polecenia *mn* należy wydać polecenie:

*$ sudo mn -h*

Po uruchomieniu środowiska Mininet w konsoli powinny pojawić się informacje o zadaniach wykonanych przez Mininet oraz na końcu powinno zostać uruchomione CLI (Command Line Interface) Mininet, dzięki któremu możliwa jest interakcja ze środowiskiem Mininet.

*\*\*\* Creating network*

*\*\*\* Adding controller*

*\*\*\* Adding hosts:*

*h1 h2*

*\*\*\* Adding switches:*

*s1*

*\*\*\* Adding links:*

*(h1, s1) (h2, s1)*

*\*\*\* Configuring hosts*

*h1 h2*

*\*\*\* Starting controller*

*c0*

*\*\*\* Starting 1 switches*

*s1 ...*

*\*\*\* Starting CLI:*

*mininet>*

W przypadku uruchomienia środowiska Mininet bez żadnych parametrów następuje przyjęcie wielu założeń domyślnych, takich jak:

* użycie topologii minimalnej
* wykorzystanie OpenVSwitch jako implementacji przełącznika SDN
* uruchomienie standardowego kontrolera OpenFlow na porcie 6653
* połączenie wszystkich przełączników do uruchomionego wcześniej kontrolera

Używając komendy *help* można wyświetlić polecenia dostępne w CLI:

*mininet> help*

*Documented commands (type help <topic>):*

*========================================*

*EOF gterm iperfudp nodes pingpair py switch*

*dpctl help link noecho pingpairfull quit time*

*dump intfs links pingall ports sh x*

*exit iperf net pingallfull px source xterm*

*You may also send a command to a node using:*

*<node> command {args}*

*For example:*

*mininet> h1 ifconfig*

*The interpreter automatically substitutes IP addresses*

*for node names when a node is the first arg, so commands*

*like*

*mininet> h2 ping h3*

*should work.*

*Some character-oriented interactive commands require*

*noecho:*

*mininet> noecho h2 vi foo.py*

*However, starting up an xterm/gterm is generally better:*

*mininet> xterm h2*

Emulator można wyłączyć za pomocą polecenia *exit*:

*mininet> exit*

*\*\*\* Stopping 1 controllers*

*c0*

*\*\*\* Stopping 2 links*

*..*

*\*\*\* Stopping 1 switches*

*s1*

*\*\*\* Stopping 2 hosts*

*h1 h2*

*\*\*\* Done*

W przypadku problemów pomocne może być wywołanie komendy:

*$ sudo mn -c*

która usuwa wszystkie pozostałości w przypadku gdy wystąpiły wcześniej błędy w działaniu Mininet.

**ZADANIE 1**

Na podstawie dokumentacji lub analizując logi emulatora narysować topologię domyślnie tworzoną przez Mininet.

**ZADANIE 2**

Należy zaobserwować proces zestawiania połączenia pomiędzy kontrolerem a przełącznikami. W tym celu:

1. Jeśli jest uruchomiony to zatrzymaj emulator
2. Uruchom program wireshark nasłuchując na interfejsie loopback (wyfiltruj ruch tak, aby obserwować tylko ruch pomiędzy kontrolerem a przełącznikami)
3. Uruchom emulator
4. **Dokładnie** przeanalizuj proces zestawiania połączenia

**3. Praca ze środowiskiem Mininet**

Z poziomu CLI Mininet możemy wydawać polecenia zarówno dla samego środowiska Mininet jak i przełączników oraz hostów przez niego stworzonych. Do często używanych poleceń należą:

* *nodes –* wyświetla listę wezłów w sieci
* *net* – pokazuje połączenia pomiędzy węzłami
* *dump* – wyświetla informacje o poszczególnych węzłach
* *help –* wyświetla pełna listę dostępnych poleceń

Można również wykonywać polecenia bezpośrednio na węzłach. Należy najpierw podać nazwę węzła a następnie polecenie jakie ma być na nim wykonane np.:

*mininet> h1 ifconfig*

**ZADANIE 3**

Wyświetlić z poziomu Mininet CLI konfigurację interfejsów sieciowych dla h1, h2 oraz z poziomu systemu operacyjnego konfigurację jego interfejsów. Przeanalizować wyniki i odpowiedzieć na pytanie dlaczego z poziomu systemu operacyjnego nie widać interfejsów hostów h1 i h2?

**4. Testowanie połączenia**

W tym podpunkcie przetestujemy możliwość połączenia pomiędzy hostami h1 i h2. W tym należy

1. Uruchomić sniffer wireshark z nasłuchem na interfejsie loopback
2. Z poziomu CLI Mininet wydać polecenie:  
    *mininet> h1 ping h2*

Jednym z pomocnych poleceń może być:

*mininet> pingall*

pozwalające na przetestowanie połązceń pomiędzy wszystkimi hostami.

**ZADANIE 4**

Przeanalizować komunikację pomiędzy kontrolerem i przełącznikiem podczas realizacji połączenia pomiędzy h1 a h2. Dlaczego pierwszy pakiet ma zdecydowanie większe opóźnienie od pozostałych? Czy w przypadku powtórzenia polecenia dalej występuje taka sytuacja?

**5. Inne topologie**

Oprócz jednej domyślnej topologii, Mininet pozwala na użycie kilku predefiniowanych wzorów. Aby zdefiniować, który ze wzorów chcemy wykorzystać należy użyć parameteru –*topo* podczas uruchamiania emulatora.

**ZADANIE 5**

Uruchomic emulator używając poniższych poleceń:

* $ sudo mn --topo single,3
* $ sudo mn --topo linear,4
* $ sudo mn -- topo tree,depth=2
* $ sudo mn --topo tree, depth=2,fanout=4

Naszkicować jak wyglądają poszczególne topologie.

**6. Wydajność połączeń**

Mininet wykorzystując narzędzie iperf w prosty sposób pozwala na przetestowanie wydajności połączeń:

*mininet> iperf*

**ZADANIE 6**

Przetestować wydajnośc połączenia wykorzystując domyślną topologię.

Używając narzędzia tc, Mininet pozwala na określenie parametrów (opóźnienie, przepustowość) tworzonych łaczy.

**ZADANIE 7**

Uruchomić emulator za pomocą polecenia:

*$ sudo mn --link tc,bw=5,delay=20ms*

Następnie przetestować wydajność łączy i porównać z wynikami otrzymanymi w zadaniu 5. Na koniec wykonać ping pomiędzy h1 i h2. Skąd wynika opóźnienie około 80ms?

**6. Przełącznik OpenVSwitch**

Do tej pory wspomniano tylko o tym, że Mininet wykorzystuje implementację OpenVSwitch jako przełącznik SDN. W tym podpunkcie zapoznamy się z podstawami współpracy z tym przełącznikiem.

Do interakcji z przełącznikiem OpenVSwitch służą polecenia z rodziny *ovs* np.:

* *ovs-vsctl –* polecenia do zarządzania przełącznikami
* *ovs-ofctl –* polecenie pozwalające na wysłanie polecenia do przełącznika za pomocą protokołu OpenFlow

Jak można było zauważyć po wykonaniu zadania 3, w konfiguracji interejsów sieciowych systemu operacyjnego pojawiły się interfejsy zaczynające się od *s1.* Są to interfejsy przełącznika SDN tworzonego podczas startu emulatora Mininet. Aby zobaczyć aktualną konfigurację przełączników należy przejśc na konto root (polecenie: *sudo su)* orazwydać polecenie:

*# ovs-vsctl show*

**ZADANIE 8**

Przeanalizować wyniki otrzymane z polecenia *ovs-vsctl show.* Co oznacza paramter fail\_mode?

Za pomocą polecenia *ovs-ofctl* jesteśmy w stanie np. wyświetlić zainstalowane przepływy:

*# ovs-ofctl dump-flows s1*

**ZADANIE 9**

Wykonać polecenie wylistowania zainstalowanych przepływów przed uruchominiem polecenia ping między hostami h1 a h2 oraz po jego uruchomieniu. Przeanalizować zainstalowane przepływy. Wykorzystać polecenie *ovs-ofctl,* aby usunąć zainstalowane przepływy.

**7. Definiowanie własnych topologii**

Oprócz topologii dostarcznych ze środowiskiem Mininet użytkownik może zdefiniować własną topologię. W tym celu zostało udostępnione API dostępne dla języka Python.

Pierwszym krokiem jest stworzenie skryptu opisującego żądaną topologię. Pusty plik tworzymy poleceniem:

*$touch topo-2sw-2host.py*

Następnie plik otwieramy w edytorze (np. gedit):

*$gedit topo-2sw-2host.py*

Skrpyt uzupełniamy przykładową topologią:

*from mininet.topo import Topo*

*class MyTopo( Topo ):*

*"Simple topology example."*

*def \_\_init\_\_( self ):*

*"Create custom topo."*

*# Initialize topology*

*Topo.\_\_init\_\_( self )*

*# Add hosts and switches*

*leftHost = self.addHost( 'h1' )*

*rightHost = self.addHost( 'h2' )*

*leftSwitch = self.addSwitch( 's3' )*

*rightSwitch = self.addSwitch( 's4' )*

*# Add links*

*self.addLink( leftHost, leftSwitch )*

*self.addLink( leftSwitch, rightSwitch )*

*self.addLink( rightSwitch, rightHost )*

*topos = { 'mytopo': ( lambda: MyTopo() ) }*

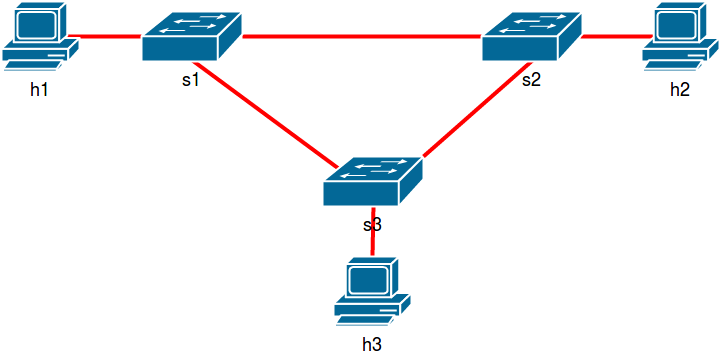
Stworzoną topologię uruchamiamy poleceniem:

*$sudo mn --custom topo-2sw-2host.py --topo mytopo*

Więcej przykłądowych topologii można znaleźć w katalogu */root/mininet/examples.*

**ZADANIE 10**

Na podstawie powyższego przykładu swtworzyć plik *example-topo.py* i zapisać topologię z rysunku nr 1.

  
Rys. 1 Topologia do zaimplementowania

Dodatkowe parametry konfiguracji topologii:

* wszystkie łącza o przepustowości 5 Mb/s
* wszystkie łącza o opóźnieniu 20 ms
* Adresy ip hostów
  + h1 – 192.168.1.1
  + h2 – 192.168.1.2
  + h3 – 192.168.1.3
  + h4 – 192.168.1.4

Rozdział o edytorze wymaga działania x-term!!!, które w tym momencie nie działają przy dockerze!!!

**8. Edytor graficznych**

Do środowiska Mininet dołączany jest również edytor graficzny. **Wspiera on jednak tylko niektóre z dostępnych opcji, w związku z czym nadaje się do wstępnego narysowania topologii, jednak w przypadku bardziej skomplikowanych rozwiązań i tak wymagana jest ręczna edycja skryptu.**

Edytor graficzny można uruchomić za pomocą polecenia:

*sudo python /root/mininet/examples/miniedit.py*

*Uzupełnić dalej jak naprawie x-term*

**\*\*\*Dla ambitnych**

Pod tym linkiem ([http://costiser.ro/2016/06/26/my-sdn-testbed/#.WXrR7Sf-thE](http://costiser.ro/2016/06/26/my-sdn-testbed/" \l ".WXrR7Sf-thE)) można znaleźć instrukcje w jaki sposób samemu stworzyć sieć SDN na swoim komputerze. Tutorial ten opisuje kroki jakie tak naprawdę za nas wykonuje Mininet.