**OpenFlow protokolü kullanılarak SDN anahtarlarında IP adresleri engelleme kurallarının tanımlanması**

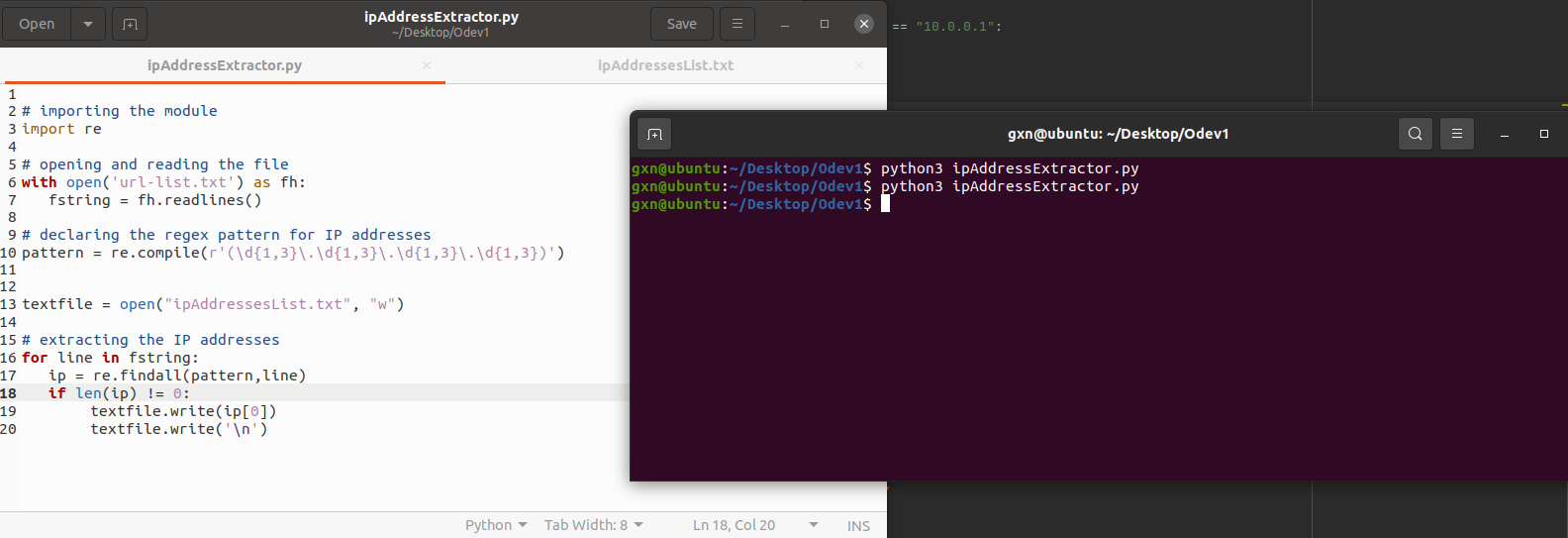
OpenFlow protokolü kullanarak anahtar ve sunucular arasındaki iletişimde kuralları belirlemek için sanal bir anahtar yapısı kullanılır. Bunlar genel SDN anahtarlarıdır. SDN anahtarları, anahtarların beyinleri gibi çalışır.

Bu ödev içerisinde bu topolojideki düşünme birinin neyi yapacağını belirlememiz gerekmektedir. Eğer kuralları belirleyen birim bizim SDN anahtarı ise, kuralımız ne ise orada programlamamız gerekecektir. Kuralımız bu durumda, IP adresleri bloklamaktır. Bu bloklanacak IP adresleri de bize usom içerisindeki listeden gelecektir. İlk olarak wget komutu ile bu veriyi yerel dosya sistemimize indirelim.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Dosyayı çektikten sonra dosya içerisinde gözlemleme yapalım. Bu dosya içerisinde nslookup gerektiren ve gerektirmeyen içerikler var. O yüzden içerideki ipleri tespit edip bir dosyaya yazalım bu dosyadan yola çıkarak.



Bir python modülü yazılır. Bu modülde ip’nin regex patterni yöntemiyle liste içerisindeki bu örüntüye uyan içerikler farklı bir dosyaya yazılır. Böylelikle usom içerisindeki ip’ler direk bir dosyaya ayrıştırılmış olacaktır. Kodu çalıştırdıktan sonra 5000 civari bir ip çıkarıldığı görülür.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Url listesindeki hazır ip’ler bir dosyaya alındı, fakat listede ip olmayan içerikler var. DNS sorgulaması yapılıp, bu linklerin hangi IP’lere karşılık gelindiğinin bulunması gerekmektedir. Böylelikle topladığımız tüm liste, tek bir format üzerinden oluşacaktır. Kullandığımız SDN Anahtar yapısı da zaten IP’ler üzerinden kontroller yapacaktır. Domain isimleri üzerinden kontrol yapmamalıdır.

Bu yüzden aynı şekilde farklı bir python modülü daha yazılır. Bu modülde farklı olarak düşünüp, url-list dosyasındaki ip olmayanlar üzerinden bir liste yapıp onlar üzerinde işlemler yapmalıyız. Yine regex yöntemiyle ama bu sefer regexte bulunanlar dışındakilere bakınır. Böylelikle domain olanlar ortaya çıkar. Bu bilgiler ışığında, her domain için, bir sunucu sorgusu atmamız gerekecektir.

Normal şartlarda bu sorgu **addr=socket.gethostbyname(site)** üzerinden her domain için atılabilir. Fakat çalışmalarda gözlemlenen duruma göre bunu yapmamız günleri alabilir. Bu yüzden farklı bir yöntem ya da kurgu ile bu sorgunun yapılması gerekmektedir. Bu yüzden **thread** kullanımına göre sorgular kurgulandı. 50 thread üzerinden daha hızlı bir şekilde ilerlediği keşfedildi. Fakat thread ile çalışırken de dikkat edilmesi gereken noktalar olmuştur. Sorgu atıldıktan sonra dosyaya yazıldığı için, dosyaya yazılma işlemlerinin tamamen tamamlandığına emin olunması gerekmektedir, ama ilk başta yazılan verilerin kaybolduğu gözlemlenmiştir. Bu yüzden dosya yazmadan önce bir **lock** oluşturulup, dosya yazma işlemi tamamlanınca **lock release** edilmiştir. Kod ile ilgili detaylar **Github** linki üzerinden incelenebilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ekran görüntüsünde olduğu gibi, erişebildiği sunucuları dosyaya yazıp, erişemediklerini atlamaktadır. Bu işlem thread’lede olsa bir süre almaktadır. Ekran görüntüsü tüm verilerden oluşmamaktadır.,

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Uzun süren kara listedeki IP’leri toplama işleminden sonra bunları bir dosyaya atıp, SDN anahtarın bu dosyayı takip etmesini sağlamamız gerekmektedir. Hazırdaki SDNSwitch.py dosyasını kopyalayıp üzerinde eklemeler yapmaya başlanır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Yukarıdaki kodda göründüğü gibi SDN paket almadan önce bazı tanımlama işlemleri eklenir. Bu bizim kuralımıza göre tüm kara listedeki ipleri bir liste objesine önceden çekmemizle başlar. Global bir değişken olarak tutulur. “listFlowAdded” diye ekstra bir değişken daha tutulur, bu değişkenin amacı flow’ların eklenip eklenmediğini kontrol eden durumu tutmasıdır.

SDN’nin kurallarına devam etmeden önce, bizim rahat test edebileceğimiz bir test topolojisine ihtiyacımız olacaktır. O yüzden python ile custom topoloji modülü oluşturulur.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada 8 sunucu, 8 anahtar kurulu bir topoloji planlanmaktadır. Ekstradan bu durumda anahtarlardan birine id tanımlanıp, o id üzerinden SDN anahtara farklı kontroller eklenebilir. Yukarıdaki resimde s3’e özel id kontrolü yaptırılıp, bu anahtara özel kural tanımlanabilir.



Eğer testlerimizde sunucuya kara listedeki bir IP tanımlayıp test etmek istiyorsak bu kod satırı ile sağlanabilir. Sunucunun adresi, bu IP ile tanımlanacaktır.

SDN anahtarına devam edecek olursak, SDN anahtarında global bir kara liste var. Gelen paketlerin kontrol edilip, bu liste içerisinde olup olmadığı kontrol edilmelidir.

**ipv4\_pkt = pkt.get\_protocol(ipv4.ipv4)**  bu komut ile gelen paketin ipv4 üzerinden gelip gelmediği anlaşılır. Bunun içerisinde de **ipv4\_pkt.src** veya **ipv4\_pkt.dst** ile de gelen paketin geldiği yer veya gideceği yerin tespiti yapılabilir. Bu aşamada bunların kullanılıp, kara listeden mi olup olmadığının kontrol edilmesi gerekecektir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ekran görüntüsünde olan kod görüntüleri bu kuralların hepsinin eklenmiş halidir. Aşama aşama üzerinden geçelim.

İlk olarak if ile paket ipv4 ‘müdür o kontrol edilir. Eğer ipv4 ise paket’in src veya dst bilgilerinin kara listede olup olmadığı sorgulanır. Bu python içerisindeki count fonksiyonu ile bakılıp 0’dan büyük olması listede olduğu anlamına gelmektedir. Bu kontrollerden sonra da kara listenin tamamı henüz flow’lara eklendi mi diye kontrol edilmesi için listFlowAdded değişkeni kontrol edilir. Eklenmediyse, sonraki adım listede olup olmadığı ve eğer eklenmek isterse diye, anahtarın bu kurala ekleneceği anahtarla aynı id’ye sahip olup olmadığı kontrol edilir. Bu kontrolleri’de geçerse tüm kara liste flow olarak eklenir.

Sadece kara listeden bir yerden gelmesine rağmen neden hepsini tek seferde ekledik? Çünkü birer birer eklendiğinde, eklenmeye devam edilmediği görülmüştür. O yüzden ekleme yapıldığında hepsinin tek seferde yapılması gerekmektedir. Bu durum az sonra ekran görüntüsü gösterilecek komutlar üzerinden doğrulanmıştır.

Kara listedeki tüm ip’lerin paketlerde src veya dst olduğu flow’lar eklenir ve kurallar eklenmiş olmaktadır.

Test adımlarına geçelim:

İlk olarak ryu-manager SDNSwichWithUsomList.py komutu ile remote bir SDN anahtar yazılımı başlatılır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Baştaki loglara bakıldığında, iplerin ilk başta dosyadan çekildiği görülür.

Sıradaki adımımız, bu SDN anahtarın bağlanacağı bir topoloji oluşturmaktır. Oluşturduğum custom Topoloji dosyasını kullanarak alttaki komut ile başlatılır.

**sudo mn --controller remote --custom BlackListTopology.py --topo=project –nat**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

SDN anahtar topolojiye bağlandı ve paketleri yayınlamaya başladı. Henüz aranan ipv4 paketi gelmediği için bir kural görünmesi beklenmemektedir. Bunu switch s3 üzerindeki flow’lara bakarak doğrulayabiliriz.

**sudo watch -d -n 1 ovs-ofctl dump-flows s3 -O OpenFlow14**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

65535 direk başlangıçta oluşturduğu tek flow olduğu için, henüz herhangi bir flow olmadığı görülmektedir. Şimdi kara listeyi buraya ekleyebilmek için o IP’lerden birine istek atmamız gerekecektir. Bunu da s3’te görmek için s3’e bağlı bir sunucu üzerinden bu pingleme işleminin yapılması lazımdır.

**xterm h1** yapılarak h1 sunucusunun sanal makinesi açılır. Listedeki herhangi bir adrese ping atılması için komut başlatılır.

**ping 46.119.35.231** komutu çalıştırılır ve dönüş alınamaz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Aynı şekilde SDN anahtarındaki loğlara bakılır ve kural eklenirken eklediğimiz loğların görüldüğü anlaşılır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şimdi eklenen kuralı görmek için s3’ün dump flow’una bakalım.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Listedeki her eleman için kural eklendiği görülür. Paketlerin aksiyonunda da drop olmasından dolayı, gelen paketlerin dönüş almayacağıyla ilgili olduğu da anlaşılabilir.

Anahtara göre özel kontrol yaptığımızdan emin olmak için **xterm h3** yapıp aynı ping üzerinden s4 anahtarı üzerindeki kurallara bakarız.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Yaptığımız id kontrolü sayesinde topolojide sadece s3 anahtarında kuralların eklendiği görülür. Tüm anahtarlarda bu kuralları görmek istiyorsak, DPID kontrolünün kaldırılması gerekmektedir.

Peki kara listedeki IP’lerden biri anahtarların birinin IP’si olsaydı?

Bu durumu test etmek için de dokümanın başındaki gibi topoloji dosyasında h2’ye bir tane listedeki IP’lerden birini veririz. H1 h2’ye s3 anahtarı ile bağlı olduğu için h1 h2’ye ping attığı an flow’lar eklenmesi beklenmektedir.



Öncesi:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Sonrası:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Kodların linklerine <https://github.com/goksunonal/SDNSmartSwitch> adresi üzerinden erişilebilir. Text dosyaları oluşturulduğu için eklemedim.

**Okuduğunuz için Teşekkürler,**

**Göksun ÖNAL**

**SBG 550 Ödev 1**

**2111111034**