IPv6 Balküpü KOVAN

ULAKBIM

17 Ocak 2011

İçindekiler

Bölüm 1

Hakkında

1.1 Belge Hakkında?

Bu belge LaTeXkullanılarak Yavuz Gökırmak, Onur Bektaş ve Serdar Yiğit tarafından Kovan proje sonuç raporu baz alınarak yazılmıştır.

1.2 Kovan nedir?

Kovan bir IPv6 balküpü projesidir. Ulusal IPv6 Altyapısı Tasarımı ve IPv6'ya Geçiş Projesi kapsamında Kovan:

- IPv6 geçiş mekanizmalarının güvenlik açısından incelenmesi
- IPv6 güvenlik açıklarının araştırılması
- IPv6 ağlarında solucan yayılımının incelenmesi
- IPv6 ağlarında mevcut ve yeni güvenlik tehditlerinin belirlenmesi

amacıyla geliştirilmektedir.

Kovan, Kamu Kurum Araştırma Projeleri için Destek Programı (KAMAG 1007) kapsamında Türkiye Bilimsel ve Teknolojik Araştırmalar Kurumu (TÜBİTAK) tarafından desteklenmekte ve TÜBİTAK ULAKBİM tarafından geliştirilmektedir.

1.3 Sistem Gereksinimleri

Kovan'ın sistem gereksinimleri nasıl bir ayarla kullanıldığıyla yakından ilişkilidir. Doğrudan yükleme için tavsiye edilen temel donanım gereksinimleri:

- 2 Gigahertz (GHz) 64-bit (x64) tek çekirdekli işlemci
- 2 Gigabayt (GB) Bellek

Eğer Kovan içinde KVM/QEMU ile işletim sistemi sanallaştırma uygulaması kullanılacaksa önerilen donanım gereksinimleri:

- 2 Gigahertz (GHz) 64-bit (x64) 2 veya 4 çekirdekli işlemci
- 4 Gigabayt (GB) Bellek

Kovan'ın donanım gereksinimleri, kullanılan KVM/QEMU imajı sayısı arttıkça artmaktadır. KVM/QEMU imajı sayısı arttıkça kullanılan bellek miktarı ve işlemci çekirdeği sayısının arttırılması temel bir kural olarak kabul edilebilir. Bir referans noktası olarak test sistemi verilebilir, ULAKBİM Kovan test sistemi aşağıdaki verilen donanımla 12 KVM/QEMU imajını herhangi bir donanım kısıtlaması olmadan çalıştırabilmektedir.

- 2x Intel Xeon CPU E5520 @2.27 Ghz
- 12 Gbyte of RAM
- 2x 160 Gbyte 15.000 RP SAS HDD

1.4 Lisanlama

Kovan GPL lisansı ile lisanslanmıştır. (http://www.gnu.org/licenses/gpl-3.0.html)

Kovan başka bir uygulamaya gömülü ve birlikte kullanılması için ikinci bir ticari lisansa da sahiptir. Daha fazla bilgi için license@ipv6.net.tr adresine yazabilirsiniz.

1.5 Güncel Sürümü Edinme

Kovan'ın son sürümünü http://www.ipv6.net.tr/kovan sitesinden indirebilirsiniz.

Bölüm 2

Sürüm Notları

2.1 Sürümdeki Yenilikler

Bu sürüm, Kovan'ın ilk sürümüdür.

2.2 Önemli Noktalar

- Kovan, FreeBSD 8.1 sürümü üzerinde geliştirilmiştir. Bu yüzden kurulum, çalıştırma ve yapılandırma betikleri 8.1 sürümünde test edilmiştir. Kurulumda FreeBSD'nin port yapısı kullanılması sayesinde Kovan'ın FreeBSD'nin ileri sürümlerde de sorunsuz çalışması beklenmektedir.
- Kovan, yönlendirme protokolü olarak RIP kullanmaktadır. RIP'ın maksimum hop sayısı 15'tir. Bu yüzden 15'ten fazla yönlendiriciden geçmenin gerektiği bir sanal ağ topolojisi yaratılmamalıdır.
- Kovan kurulum betikleri FreeBSD port sistemini FORCE_PKG_REGISTER değeri olmadan kullanmaktadır. Bu nedenle yüklenmeye çalışılan paketler önceden yüklenmişse kurulum başarısız olur. Eğer bu durum ortadan kaldırılmak istenirse FORCE_PKG_REGISTER değerine 1 verilerek sistem çalıştırılabilir (export FORCE_PKG_REGISTER=1).
- ng_bridge nesnesine bağlanabilecek varsayılan cihaz sayısı 32'dir. Bu sayıyı değiştirmek için, "/sys/netgraph/ng_bridge.h" dosyasındaki "NG_BRIDGE_MAX_LINKS" sabiti değiştirilerek çekirdek yeniden derlenmelidir.
- Kovan bazı topolojilerde birden fazla epair nesnesi kullanmakta ve sanal ağı bu şekilde oluşturmaktadır. Böyle bir topolojide çalışıyorken kovan kapatılmak istendiğinde, ardarda çalışan bazı komutlar sistemin panic moduna düşüp kendini yeniden başlatmasına neden olmaktadır. Bu yüzden kovan kapatılırken epair nesnelerinin kaldırılması sırasında 2'şer saniye beklemektedir.

Bölüm 3

Temel Kavramlar

3.1 Sanal Balküpü Altyapısı

Kovan, saldırganları üstüne çekebilmek için gerçek bir ağın davranışını taklit eden sanal balküpü altyapısıdır. Sanal balküpleri, gerçek sistemlere göre daha az kaynak gereksinimiyle kurulabildikleri için tercih edilmektedirler. Gerçek gibi görünen sanal ağ topolojisi kolay uygulanabilir olduğu gibi saldırganlara da gerçekçi görünmektedir. Kovan'ın oluşturduğu gerçekçi sanal ağın, Argos[1] ve Nepenthes[2] gibi balküpleri için bir çalışma ortamı olması öngörülmektedir. Gerçekçi sanal ağın üstüne çeşitli balküpleri veya servisler kurularak çeşitli etkileşim seviyelerini içinde barındıran bir balküpü ağı yaratılabilir.

Kovan mevcut ağ kavramlarını bir arada kullanarak sanal balküpü altyapısı oluşturmaktadır. Genel işleyişe bakıldığında; ilk aşamada sanal cihazlar kullanılarak ağda yer alacak yönlendiriciler, sunucular ve istemciler yaratılmaktadır. Sanal cihazlar FreeBSD Jail[3] sistemi yardımıyla yaratılmaktadır, bölüm 3.1.1'de sanal cihazlar hakkında ayrıntılı bilgi verilmektedir. İkinci aşamada, sanal cihazlar arasındaki sanal kablolar (bağlantılar) yaratılmaktadır. Sanal cihazların bağlanmasında, FreeBSD'nin Netgraph[4] ve Epair[5] sistemleri kullanılmaktadır, 3.1.3 ve 3.1.2 bölümleri altında ayrıntılar anlatılmaktadır. Üçüncü basamağa kadar sanal cihazlar ve aralarındaki bağlantılar oluşturulmaktadır. Üçüncü ve son basamak sanal cihazlar arasındaki iletişimi sağlama amacıyla dinamik yönlendirme protokollerini çalıştırılır. Bu aşamada Quagga[6] yazılımı kullanılmaktadır, bölüm 3.1.4 altında yönlendirme sisteminin ayrıntılarından bahsedilmektedir.

3.1.1 Sanal Cihazlar: FreeBSD Jail

FreeBSD Jail; birbirinden bağımsız, düşük kaynak tüketen sanal sistemler yaratmaya yarayan işletim sistemi seviyesinde bir sanallaştırma sistemidir. Yaratılan sanal sistemlere jail denilmekte ve sistemler gerçek bir sistemin sağlayabildiği birçok özelliği sağlayabilmektedir.

Standart jail kurulumunda, yaratılan jail'lere sadece ana makinenin bulunduğu ağdan (subnet) IP adresi verilebilmektedir. Standart sistemde yaratılan jail'lerin birbirlerine bağlanarak karmaşık ağlar oluşturulması da olanaklı değildir. Kovan, ihtiyacı olan ağ işlemleri için jail sisteminin ileri bir sürümünü kullanmaktadır. Jail sistemine yapılan VIMAGE yaması sayesinde her jail'e bağımsız bir ağ arayüzü atanabilmektedir. Yaratılan ağ arayüzleri uygun cihazlara atanıp, kendi aralarındaki bağlantılar gerçekleştirilerek karmaşık ağ yapıları oluşturulabilmektedir. Jail VIMAGE desteği FreeBSD 8 ile getirilmiştir. 3.1.2 ve 3.1.3 bölümlerinde sanal arayüzlerin nasıl oluşturulduğu ve bağlandığı anlatılmaktadır. Vimage desteğini etkin hale getirilebilmesi için FreeBSD çekirdeği yeniden derlenmelidir (bölüm 5.4.1).

```
1 root# jail -c vnet name=n0 host.hostname=n0 path=/usr/local/kovan/node persist
2 root# ngctl mkpeer eiface ether ether    // ngeth0 yaratıldı
3 root# ifconfig ngeth0 vnet n0
4 root# jexec n0 ifconfig ngeth0 name eth0
5 root# jexec n0 ifconfig eth0 link 00:0c:6e:22:13:0
```

Listing 3.1: Sanal bir ağ arayüzü oluşturup jail'e atamak

3.1'deki komutlarla jail'e vimage desteği verilmesi basit bir örnekle anlatılmaktadır. İlk satırda n0 adlı bir jail yaratılmıştır. İkinci satırda, eiface türünde bir netgraph nesnesi yaratılmıştır. Yaratılan bu nesne bir ağ arayüzü olarak görülebilir (netgraph sistemi hakkında ayrıntılı bilgi 3.1.2 bölümünde verilmektedir). Üçüncü satırda önceden yaratılan nethgraph nesnesi ifconfig komutuyla jail'in içine atılır. 4-5 numaralı satırlarda netgraph nesnesinin adı değiştirilerek bir fiziksel adres atanır. Bu komutlar yardımıyla, sanal bir bilgisayar yaratılıp sanal bir ağ arayüzüne sahip olması sağlanmıştır.

Kovan, jail sistemini kullanarak yönlendirici, sunucu ve istemci gibi ağ bileşenlerini oluşturmaktadır. Her ağ cihazı bir sınıfta yer almakta ve özellikleri bulundukları sınıfa göre belirlenmektedir. Kovan'ın dört cihaz sınıfı bulunmaktadır; yönlendirici, düğüm, karadelik ve monitor. İsminden de anlaşılabileceği gibi yönlendirici sınıfı yönlendirici cihazları için kullanılmaktadır. Yönlendirici sınıfında bulunan cihazlarda, yönlendirme için gereken bütün yazılımlar bulunmaktadır. Monitör cihazında sanal ağı gözlemlemek için gereken NetSNMP, Nagios, NFSEN ve MRTG yazılımları bulunmaktadır. Düğüm cihazı sunucu veya istemci olarak kullanılabilmektedir. Düğüm cihazına istenilen servis yüklenebilmektedir. Kovan'ın sanal HTTP, SMTP, FTP ve DNS servisleri de düğüm cihazında yüklü olarak bulunmaktadır. Karadelik sınıfında, kullanılmayan IPv6 adreslerine gelen trafiğin analizi yapılmaktadır.

3.1.2 Sanal Bağlar: Netgraph

Netgraph sistemi çeşitli ağ işlemleri yapılmasını sağlayan ağ nesneleri sağlamaktadır. Netgraph nesneleri, hook adı verilen özel bağlantı nesneleriyle birbirlerine bağlanarak karmaşık ağlar oluşturulmaktadır. Kovan ' ın, netgraph nesnelerini sanal cihazlara atayıp bu nesneleri birbirine bağlamasıyla ağ yapıları oluşturulmaktadır. Kolay anlaşılması açısından, netgraph nesneleri ağ arayüzü, hook'lar ise ağ kabloları olarak görülebilir.

Kovan'da kullanılan netgraph nesneleri:

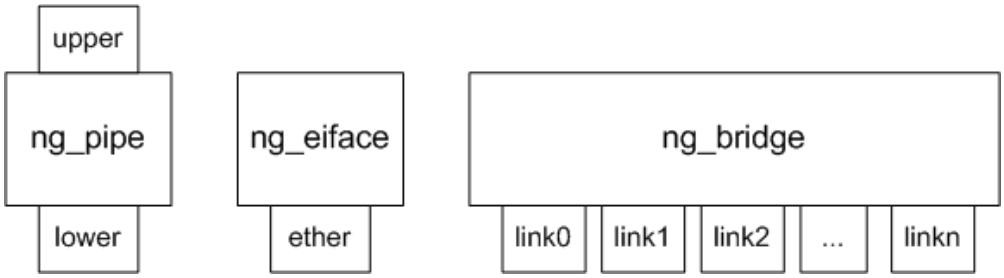
- ng_eiface: Eiface türündeki netgraph nesnesi sanal bir ethernet arayüzü sağlamaktadır. Yaratılan bir eiface nesnesine ifconfig komutu yardımıyla erişilebilmektedir. Kovan ng_eiface nesnelerini iki durumda kullanmaktadır; iki cihazı doğrudan birbirine bağlarken veya bir cihazı bir köprü cihaza bağlarken.

- ng_bridge: Netgraph'ın köprü nesnesi birden fazla nesnenin birbirine bağlanmasında kullanılmaktadır. Köprü üstündeki her bağlantı ethernet paketlerini alıp-gönderme işlemi yapmaktadır. Alınan her pakette, köprü hangi düğümün hangi bağlantıda olduğunu belirler. Bilinen bir hedefe giden paketler sadece ilgili bağlantıya gönderilir, diğer trafikten ayırılır. Köprü nesnesinin bu davranışı hub'dan farklılık gösterir, çünkü hub gelen paketi tüm bağlantılara çoklayarak gönderir. Kovan, köprü nesnesini, birden fazla ağ arayüzünü birbirine bağlayarak yerel ağlar oluşturmak için kullanır.

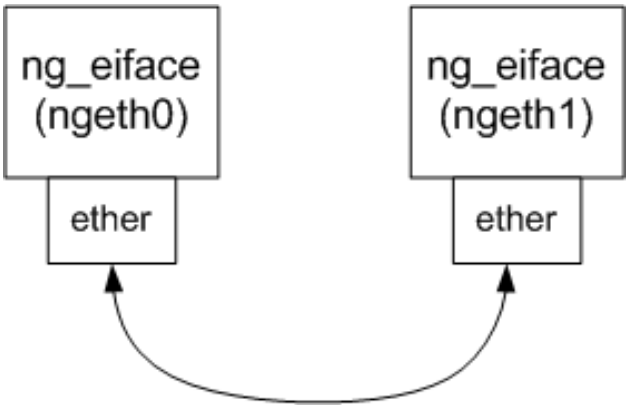
- ng_pipe: Ng_pipe nesnesi bant-genişliği, gecikme ve paket kaybını taklit ederek trafik şekillendirmesi yapmaktadır. Bu nesne, upper ve lower adlı iki hook'a sahiptir. Upper hook'tan gelen trafik lower hook'a; lower hook'tan gelen trafik de upper hook'a yönlendirilmektedir. Delay dışındaki bütün parametreler paketin yönüne göre ayrı ayrı verilebilmektedir. Kovan, pipe nesnesini hem doğrudan bağlantıda hem de düğüm-köprü bağlantılarında kullanmaktadır. Bağlantı parametreleri değiştirilerek sanal ağ daha gerçekçi gösterilebilmektedir.

Şekil 3.1, Kovan'da kullanılan ng_eiface, ng_bridge ve ng_pipe netgraph nesnelerini göstermektedir.

Şekil 3.2 netgraph nesneleri ve hook'ları hakkında basit bir örneği göstermektedir. Bu örnekte, ether adında birer hook'u olan iki ng_eiface nesnesi -ngeth0 ve ngeth1- yaratılmıştır. İki nesneyi birbirine bağlamak için ether hookları bağlanmıştır. Bahsedilen yapı, 3.2'de gösterilen komutlar yardımıyla yapılabilmektedir. Birinci ve ikinci satırlar ngeth0 ve ngeth1 adlı iki ng_eiface nesnesi yaratmaktadır. Nesne isimleri sistem tarafından otomatik olarak atanmaktadır fakat sonradan değiştirilme olanağı da bulunmaktadır. Yaratılan bir ng_eiface nesnesi, ngethX formatında bir ismi otomatik olarak alır, burada "X" sistemdeki kaçıncı ng_eiface nesnesi olduğunu gösterir. Bundan dolayı, ilk satırda yaratılan nesne ngeth0 adını alırken, ikinci



Şekil 3.1: Kovan'da kullanılan Netgraph nesneleri



Şekil 3.2: İki eiface nesnesininin bağlanması

satırdaki nesne ngeth1 adını almaktadır. Üçüncü satır, yaratılan iki nesne arasındaki bağı kurar. "ngctl connect" komutu -sırasıyla- nesne isimleri ve nesnelerin hook'larının isimlerini argüman olarak alır ve nesneleri belirtilen hook'larda birbirlerine bağlar.

```
1 root# ngctl mkpeer eiface ether ether
2 root# ngctl mkpeer eiface ether ether
3 root# ngctl connect ngeth0: ngeth1: ether ether
```

Listing 3.2: İki ng_eiface nesnesini bağlama

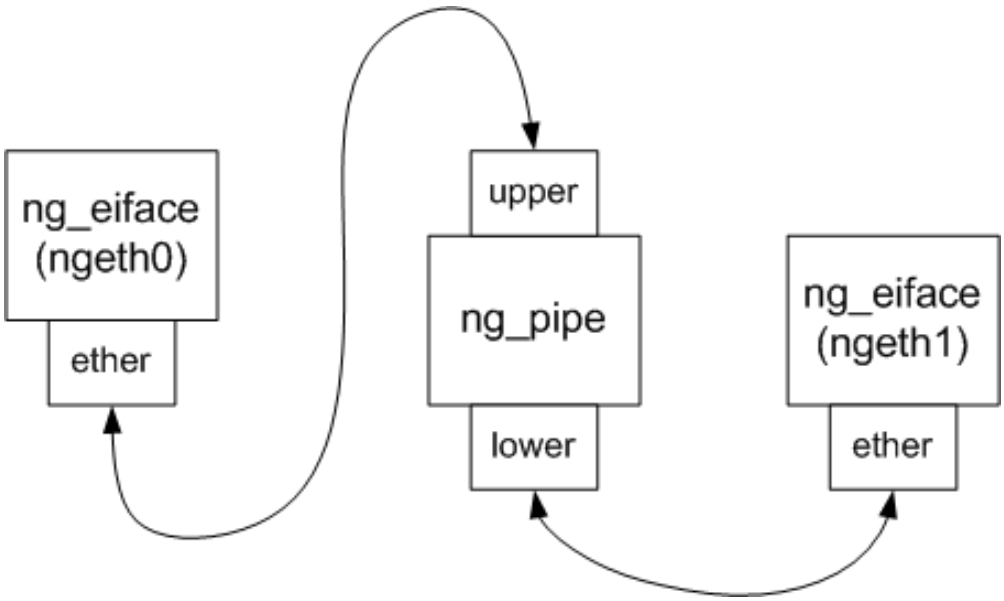
Şekil 3.3'te gösterilen yapıda, iki nesne birbirlerine bağlanırken araya bir ng_pipe nesnesi koyularak daha karmaşık bir yapı elde edilmiştir.

Pipe nesnesi; bant-genişliği(bandwidth), gecikme(delay) ve bit hatası oranı(BER) gibi bağlantı özelliklerinin değiştirilebilmesi için bir mesaj mekanizması sunar. Liste 3.3'te iki nesnenin bir pipe nesnesi aracılığıyla birbirine bağlanmasını sağlayan komutlar yer almaktadır. Üçüncü satır, bir pipe nesnesi yaratarak ngeth0 nesnesini pipe'in upper hook'una bağlar.

```
ngctl mkpeer [hedef] tür hook1 hook2
```

Dikkat edilmesi gereken bir nokta, netgraph nesnelerinin tek başlarına yaratılamadıklarıdır, bir nesneyi yaratmak için bağlanacağı nesneyi de belirtmek gerekir. Bağlanacağı "hedef" belirtilmese dahi, varsayılan socket nesnesine bağlanır. Liste 3.3'te yaratılan nesnelerin varsayılan socket nesnesine bağlanması bir örnektir.

```
1 root# ngctl mkpeer eiface ether ether
2 root# ngctl mkpeer eiface ether ether
```



Şekil 3.3: İki nesnenin pipe ile birbirine bağlanması

```
3  root# ngctl mkpeer ngeth0: pipe ether upper
4  root# ngctl name ngeth0:ether n0-n1
5  root# ngctl connect n0-n1: ngeth1: lower ether
6  root# ngctl msg n0-n1: setcfg { bandwidth=50000 }
7  root# ngctl msg n0-n1: setcfg { delay=10 }
8  root# ngctl msg n0-n1: setcfg { upstream={ BER=2 } }
9  root# ngctl msg n0-n1: setcfg { upstream={ duplicate=1 } }
10 root# ngctl msg n0-n1: setcfg { downstream={ fifo=1 } }
```

Listing 3.3: İki ng_eiface nesnesinin pipe ile bağlanması

Liste 3.3'ün 4. satırı pipe nesnesine bir isim vermektedir. Pipe gibi nesneler yaratılırken varsayılan bir isimleri yoktur, bu yüzden sonradan bu nesnelere bir isim vermek gerekmektedir. Dördüncü satır, ngeth0'ın ether hook'una bağlanmış pipe nesnesine n0-n1 ismini vermektedir.

Liste 3.4'te 4 ng_eiface nesnesi yaratılarak bir köprü nesnesi aracılığıyla birbirlerine bağlanmaktadır. Köprü nesnesi, n tane nesneyi birbirine bağlayabilecek link0, link1, link2 ... linkn adlı n tane hook'a sahiptir. Örneği anlaşılır kılmak amacıyla bağlantılar arasında pipe nesnesi kullanılmamıştır fakat ng_eiface ve ng_bridge nesneleri arasında pipe nesnesi de kullanılabilmektedir.

```
1  root# ngctl mkpeer eiface ether ether
2  root# ngctl mkpeer eiface ether ether
3  root# ngctl mkpeer eiface ether ether
4  root# ngctl mkpeer eiface ether ether
5  root# ngctl mkpeer ngeth0: bridge lower link0
6  root# ngctl name ngeth0:lower test-bridge
7  root# ngctl connect ngeth1: test-bridge: ether link1
8  root# ngctl connect ngeth2: test-bridge: ether link2
9  root# ngctl connect ngeth3: test-bridge: ether link3
```

Listing 3.4: Dört eiface nesnesinin köprüyle bağlanması

Sanal sistemler dışında birbirlerine uygun bir şekilde bağlanan netgraph nesneleri, uygun cihazlara atanarak karmaşık ağ yapıları oluşturulmaktadır. Netgraph nesnelerini sanal cihazlara atamak bağlantılara zarar vermemektedir. Netgraph nesnesi bir vimage jail'ine(bölüm 3.1.1) atandığında, ifconfig komutu yardımıyla diğer ağ arayüzleri gibi görüntülenebilmektedir.

Netgraph sistemi birçok ng_* nesnesi sunmaktadır fakat Kovan bünyesinde sadece bu nesnelerin ufak bir alt kümesi kullanılmaktadır. Kovan'ın ileri sürümlerinde kullanılan nesne türü çeşitlendirilebilir.

3.1.3 Sanal Bağlar: Epair

Epair sistemi netgraph gibi ağ cihazlarının birbirlerine bağlanması için kullanılmaktadır. Epair ve netgraph sistemleri arasında iki önemli farklılık bulunmaktadır.

- Epair sistemi, netgraph gibi karmaşık hook mekanizmaları içermemektedir. Bunun yerine, epair nesneleri ifconfig komutu yardımıyla kolayca yaratılıp dağıtılabilmektedirler.
- Epair sisteminin kullanımı kolay olmasına karşın netgraph sisteminin önemli bir avantajı bulunmaktadır. Netgraph nesneleri arasında, pipe nesnesi kullanılarak ağ parametreleri değiştirilebilir ve daha gerçekçi bir ağ yaratılabilir. Epair böyle bir imkan sunmamaktadır.

Epair, sanal bir kablo ile birbirine bağlanmış bir çift sanal ethernet arayüzü yaratmaktadır. İki cihaz'ın birbirine bağlanabilmesi için her epair ucu bir sanal cihaza atanmalıdır.Liste 3.5'te iki cihaz'ın birbirine doğrudan bağlanabilmesi için gereken epair komutları verilmektedir. Bu komutların çalıştırılmasından önce n0 ve n1 adlı iki jail'in 3.1.1 bölümünde anlatıldığı gibi yaratıldığı varsayılmaktadır.

```
1  root# ifconfig epair create
2  epair0a
3  root# ifconfig epair0a vnet n0
4  root# ifconfig epair0b vnet n1
5  root# jexec n0 ifconfig
6  lo0: flags=8008<LOOPBACK,MULTICAST> metric 0 mtu 16384
7          options=3<RXCSUM,TXCSUM>
8  epair0a: flags=8842<BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
9          ether 02:da:6a:00:04:0a
10 root# jexec n1 ifconfig
11 lo0: flags=8008<LOOPBACK,MULTICAST> metric 0 mtu 16384
12         options=3<RXCSUM,TXCSUM>
13 epair0b: flags=8842<BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
14         ether 02:00:00:00:05:0b
```

Listing 3.5: İki cihaz'ın birbirine epair ile bağlanması

Liste 3.5'ın ilk satırında epair0a ve epair0b adlı bir epair çifti yaratılmaktadır. 2. ve 3. satırlar her bir epair ucunu bir cihaza atar. 4. ve 5. satırda görülebileceği gibi sanal arayüzler, cihazların içinde gerçek arayüzler gibi görünmektedir.

Liste 3.6'te dört sanal cihaz bir köprü cihazı aracılığıyla birbirlerine bağlanmaktadır.

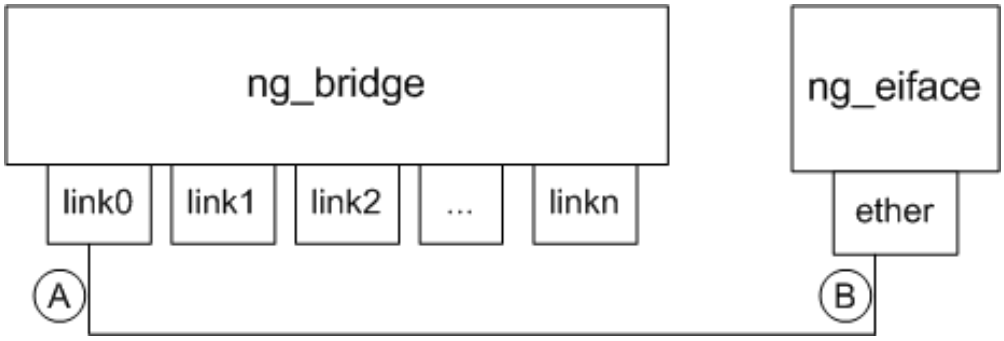
```
1  root# ifconfig epair create
2  epair0a
3  root# ifconfig epair create
4  epair1a
5  root# ifconfig epair create
6  epair2a
7  root# ifconfig epair create
8  epair3a
```

```
 9  root# ifconfig epair0b vnet n0
10  root# ifconfig epair1b vnet n1
11  root# ifconfig epair2b vnet n2
12  root# ifconfig epair3b vnet n3
13  root# ifconfig bridge create
14  bridge0
15  root# ifconfig bridge0 addm epair0a
16  root# ifconfig bridge0 addm epair1a
17  root# ifconfig bridge0 addm epair2a
18  root# ifconfig bridge0 addm epair3a
```

Listing 3.6: Dört cihazın köprü ile bağlanması

1-3-5-7 satırlarında dört tane epairXa-epairXb (X, çiftin yaratılma sırasını gösterir. İlk çift için X 0, ikinci çift için 1...) formatında çiftler oluşturulmaktadır. 9-12 arasındaki satırlarda çiftlerin "b" tarafları ilgili sanal cihaza atanır. 13. satırda bir köprü nesnesi yaratılır. 14-18 arası satırlarda, önceden yaratılan epair çiftlerinin "a" tarafları köprü nesnesine eklenir. Köprü bir anahtarlama cihazı gibi sanal cihazları birbirine bağlar.

Kovan'ın geliştirme aşamalarının başında epair sistemi tüm bağlantılar için kullanılmaktaydı. Geliştirme aşamalarının ileri safhalarında bağlantı parametrelerini değiştirebilme özelliğinden dolayı netgraph sistemi bağlantıların büyük çoğunluğunda kullanılmaktadır. Çoğu bağlantının netgraph sistemiyle yapılmasına karşın, epair bağlarının kullanıldığı iki istisnai durum bulunmaktadır. İlk durum, şekil 3.4'teki gibi bir köprü bağlantısı netgraph ile yapıldığı durumda bağlantının "A" tarafına bir IP verilmek istendiğinde yaşanmaktadır. Netgraph sisteminde bu uca bir IP adresi verilememektedir. Kovan'da bahsedilen durum, bütün sanal ağ'ın AnaSistem'in ağ'ına bağlandığında ortaya çıkmaktadır. Kovan bu durumda epair sistemini kullanmaktadır.



Şekil 3.4: Kovan epair kullanımı istisnası

İstisna yaşanan ikinci durum ise QEMU/KVM imajlarının sanal ağa bağlanmasında yaşanmaktadır. Bu imajların bağlanabilmesi için ifconfig aracılığıyla erişilebilen bir köprü nesnesine ihtiyaç duyulmaktadır. Bu yüzden, QEMU/KVM imajlarının bağlanmasında epair sistemi kullanılmaktadır, ayrıntılar 8.1 bölümünde anlatılmaktadır.

Sanal cihazlar birbirlerine bağlandıktan sonra her cihaza uygun adres verilerek yönlendirmeler yapılmalıdır. Yönlendirme işlemi 3.1.4 bölümünde anlatılmaktadır.

3.1.4 Dinamik Yönlendirme

Kovan, dinamik yönlendirmeyi açık kaynak kodlu Quagga yazılımını kullanarak gerçekleştirmektedir. IPv4 yönlendirmesi için RIPv2, IPv6 içinse RIPng kullanılmaktadır. RIP, link state yönlendirme protokollerine oranla kolay ayarlanabilir olması ve daha az işlemci gücü kullanması dolayısıyla seçilmiştir. Her yönlendirici için RIP, RIPNG ve zebra ayar dosyaları başlangıç aşamasında otomatik olarak oluşturulmaktadır. Ayar

Zebra	2601
Rip	2602
Ripng	2603

Tablo 3.1: Quagga Yönetim Port Numaraları

dosyaları yönlendirici jail'inin altında yaratılmaktadır. Dosyaların konumu ise /usr/local/etc/quagga altında yönlendirici ismiyle olusturulan klasördür. Örneğin, anarouter adlı bir yönlendiricinin ayar dosyaları:

```
/usr/local/etc/quagga/anarouter/ripd.conf
/usr/local/etc/quagga/anarouter/ripng.conf
/usr/local/etc/quagga/anarouter/zebra.conf
```

isimleri ile oluşturulmaktadır. Quagga telnet aracılığıyla ayarlar yapılmasını sağlayan bir yönetim arayüzü sağlar. Tablo 3.1'de yer alan port numaraları kullanılarak telnet yardımıyla yönlendirme ayarları yapılabilmektedir.

3.2 Durum Kaydetme/Yükleme (Stateful)

Kovan farklı ayar dosyalarıyla çalıştırılarak farklı sistemler oluşturulabilir. Her ayar dosyası farklı ağ yapıları ve farklı servisler içerebilir. Kovan çalışması esnasında flow, ağ istatistikleri, bant genişliği kullanımı vb verileri kaydetmektedir. Eğer kullanıcı başka bir ayar dosyasıyla Kovan'ı çalıştırmak isterse ve mevcut topolojinin de silinmesi istenmiyorsa, çalışan sistemin bir kopyası alınabilir. Kovan'ın çalışan sistemiyle ilgili durum bilgileri taşıyan bütün dosyalar .tar dosyası haline getirilerek kovan_path/states/ altına kaydedilir ve böylece mevcut durum yedeklenebilir. Sistemde durum bilgisi taşıyan dosyalar 6 başlıkta toplanabilir:

- Kovan ayar dosyaları
- MRTG ayar dosyaları ve üretilmiş web dosyaları
- NFSEN ayar dosyaları ve veritabanları
- NAGIOS ayar dosyaları ve veritabanları
- MYSQL veritabanları
- İstatistik grafikleri

kovanState betiğinin argümanları ve kullanımları aşağıda gösterilmiştir. Çalışan sistemin durumu kaydedilirken bir durum ismi verilmesi gerekmektedir. Betik verilen bu isme bir zaman bilgisi ekleyerek kaydetmektedir. Daha önceden kaydedilen bir durum tekrar yüklenmek istendiğinde isminin verilmesi yeterli olmaktadır.

```
root# ./kovanState
usage: kovanState --save state_name
       kovanState --load state_name
```

Durum kaydetme işlemi çalıştırıldığında listede yer alan dosyalar bir tar dosyası haline getirilir. Durum yükleme çağrıldığında ise verilen durum ismine karşılık gelen .tar dosyası açılarak içindeki dosyalar gerekli konumlara kopyalanır. Mevcut çalışan durumu kaydetmeden yeni bir durum yüklenirse çalışan sistemin bilgileri kaybedilir. Durum bilgisi taşıyan dosyaların bir listesi aşağıda verilmiştir.

```
# Kovan configuration files
nodeJailRoot/usr/local/kovan/etc
```

```
# MRTG configuration files and produced web files
monitorJailRoot/usr/local/etc/mrtg
monitorJailRoot/usr/local/www/mrtg

# NFSEN configuration files and database files
monitorJailRoot/usr/local/var/nfsen/profiles/live/*
monitorJailRoot/usr/local/var/nfsen/profiles-data/live/*
monitorJailRoot/usr/local/var/nfsen/profiles-stat/live/*.png
monitorJailRoot/usr/local/var/nfsen/profiles-stat/live/*.rrd
monitorJailRoot/usr/local/var/nfsen/run/*
monitorJailRoot/usr/local/etc/nfsen.conf

# NAGIOS configuration files and database files
monitorJailRoot/usr/local/etc/nagios/*.cfg
monitorJailRoot/var/spool/nagios/archives/*
monitorJailRoot/var/spool/nagios/checkresults/*
monitorJailRoot/var/spool/nagios/rw/*
monitorJailRoot/var/spool/nagios/status.dat
monitorJailRoot/var/spool/nagios/nagios.log
monitorJailRoot/var/spool/nagios/objects.cache

# MYSQL database files
monitorJailRoot/var/db/mysql/*

# Statistic graph files
monitorJailRoot/usr/local/www/stats/*.png
```

3.3 İzleme

3.3.1 SNMP

Kovan SNMP izleme için NET-SNMP paketini kullanmaktadır. Örnek ayar dosyası (snmp.conf) Kovan'ın kaynak kodları klasöründen yönlendirici jail'in /etc klasörüne kurulum aşamasında kopyalanmaktadır. Bant genişliği grafiklerinin çizimi için MRTG kullanılmaktadır. İzleme cihazında çalışacak MRTG, kovanMRTG betiği yardımıyla ayarlanmaktadır. Daha ayrıntılı bilgi için lütfen bölüm 4.3'e bakınız.

Kovan'ın ayar dosyaları işlenirken, yönlendirici türündeki her cihazda snmpd çalıştırılarak bütün IPv6 arayüzleriyle ilişkilendirilir. Snmpd hem IPv4 hem de IPv6 dinleyebilmesine karşın, Kovan snmpd'yi sadece IPv6 trafiğini udp portu 161'de dinleyecek şekilde çalıştırır. Kovan her yönlendirici için aşağıdaki komut ve argümanlar ile snmpd çalıştırır.

```
/usr/local/sbin/snmpd -c /etc/snmpd.conf udp6:161
```

3.3.2 NETFLOW

Kovan, softflowd programını netflow üretici, nfsen'i ise netflow toplayıcı olarak kullanır. İki uygulama da install.sh betiği ile kurulur.

Softflowd

Kovan, türü yönlendirici olan bütün cihazlarda softflowd'yi çalıştırır. Netflow izlemeyele ilgili olarak, Kovan ayar dosyasını işlerken şu parametrelere bakar:

```
Eğer cihaz türü yönlendirici ise
    yönlendirici jail id'sini al
yönlendirici üstündeki arayüzlerin sayısını al
Eğer cihaz türü monitör ise
    izleme cihazının IPv6 adresini al
nfsen_start_port değerini al
```

Kovan bu bilgileri bağımsız softflowd uygulamaları çalıştırmak için kullanır. Flow toplatıcı nfsen uygulaması monitor cihazında bulunduğu için, izleme cihazının IPv6 adresi hedef adres olarak kullanılır. Ayar dosyası işlenirken, nfsen_start_port değeri ilk yönlendiricinin ilk arayüzü için atanır sonrasında karşılaşılan her yönlendirici arayüzü için sayı bir arttırılır.

```
/usr/local/sbin/softflowd -v 9 -i interface -m 1000 -n[MonitorIPv6address]:port numarası
```

Netflow izlenmesinin daha iyi anlaşılabilmesi için bir örnek verilebilir. Aşağıdaki gibi bir yapı olması durumunda:

- iki yönlendirici: router1 ve router2.
- Router1 iki arayüze and Router2 ise üç arayüze sahip
- Monitör cihazının IPv6 adresi 2001:a98:13:3000::12
- Nfsen_start_port 9000

Kovan Router1'de şu komutları çalıştırır:

```
/usr/sbin/softflowd -v9 -i eth0 -m 1000 -n[2001:a98:13:3000::12]9000
/usr/sbin/softflowd -v9 -i eth1 -m 1000 -n[2001:a98:13:3000::12]9001
```

Router2'de ise:

```
/usr/sbin/softflowd -v9 -i eth0 -m 1000 -n[2001:a98:13:3000::12]9002
/usr/sbin/softflowd -v9 -i eth1 -m 1000 -n[2001:a98:13:3000::12]9003
/usr/sbin/softflowd -v9 -i eth2 -m 1000 -n[2001:a98:13:3000::12]9004
```

NFSEN

Kovan, nfsen uygulamasını yüklemek, ayarlamak ve çalıştırmak için kovanNfsen'i kullanır. Kovan çalıştırıldıktan sonra, kovanNfsen çalıştırılarak monitör cihazında nfsen.conf ayarlanmalıdır. KovanNfsen, http://monitorIPv6Adresi/nfsen altındaki web sayfalarını oluşturur. Ayrıntılı bilgi için bölüm 4.4'e bakınız.

3.3.3 Servis İzleme

Kovan'daki servisleri izlemek için Nagios uygulaması kullanılmaktadır. Nagios; ağ cihazlarının erişilir olup/olmadığını kontrol ettiği gibi cihazlar üstünde çalışan servislerin de durumlarını otomatik olarak kontrol etmektedir. Nagios her yönlendirici için aşağıdaki başlıkları kontrol etmektedir:

- sistemin çalışma/çalışmama durumu
- snmpd uygulaması
- softflowd uygulaması
- zebra uygulaması
- RIPNG uygulaması

- RIP uygulaması

Nagios her düğüm için aşağıdaki başlıkları kontrol etmektedir:

- sistemin çalışma/çalışmama durumu
- Kovan servisleri (http, smtp, ftp, dns)

Daha fazla bilgi için Bölüm 4.2'ye bakınız

Kaynakça

[1] G. Portokalidis, A. Slowinska, H. Bos, "Argos: an emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation", Proceedings of ACM SIGOPS EUROSYS 2006, 2006.

[2] P. Baecher, M. Koetter, T. Holz, M. Dornseif, "The Nepenthes Platform: An Efficient Approach to Collect Malware", Proceedings of 9th International Symposium (RAID 06), Sayfa : 165 184, 2006.

[3] M. Zec, "Implementing a Clonable Network Stack in the FreeBSD Kernel", Proceedings of USENIX 2003 Annual Technical Conference, Sayfa: 137 - 150, 2003.

[4] http://people.freebsd.org/~julian/netgraph.html

[5] FreeBSD Epair,
http://bsdbased.com/category/virtualization/

[6] http://www.quagga.net/

Bölüm 4

Araçlar

Kovan kurulup başarılı bir şekilde çalıştırıldıktan sonra, sistemin izlenmesi kullanıcılar için önemli bir gereksinim haline gelmektedir. Bölüm 3.3 ' te açıklanan sistem ve ağ izleme araçları, kovan kurulumu sırasında monitor jail ' ine kurulmuştur. Kovan kullanıcısı monitor jail ' ini ve araçları elle yapılandırabilir fakat bu çok zaman alacaktır. Bu yapılandırma sürecini kolaylaştırmak ve hızlandırmak için betikler yazılmış ve "kovan/bin" dizinine koyulmuştur.

Kovan ' ın genel ayarlarını kullanabilmek için, betikler kovan yapılandırma dosyasını parametre olarak almak zorundadırlar. Ayrıca betiklerin işlevsel olarak yaptığı iş aynı olduğundan, bu işler 3 ortak fonksiyonda toplanmış ve "install", "configure", "execute" olarak adlandırılmışlardır.

- install : gerekli yapılandırma dosyalarını kontrol eder, yapılandırma dizinlerini oluşturur ve ilgili dosyaları bu dizinlere kopyalar.Yapılandırma dosyası mevcut olsa dahi yenisini oluşturur ve eskisinin üzerine yazar.
- configure : yapılandırma dosyaları üzerinde düzenli ifadeler (regular expressions) çalıştırarak dosyaları hazırlar.
- execute : uygulamayı çalıştırır, eğer uygulama çalışır haldeyse yeniden başlatır.

"install" parametresini bir aracın sadece ilk yapılandırması sırasında kullanınız. Eğer mevcut çalışan uygulamanın yapılandırma dosyaları üzerinde elle değişiklikler yapar ve ilgili betiği install parametresi ile çalıştırırsanız, betik dosyaların üzerine yazar (overwrite) ve yapılandırmanızı kaybedersiniz.

Hazır betiklerden herhangi birini kullanarak uygulamayı çalışır hale getirdikten sonra, sistemde yaptığınız bir değişiklikle ilgili uygulamayı güncellemek için sadece "configure" parametresini kullanınız. Örneğin ; nagios servisini yapılandırdınız ve çalıştırdınız. Daha sonra sisteme yeni bir bilgisayar eklediniz. Bu servisin nagios ' un yapılandırmasına eklenmesi için betiği şu şekilde kullanınız:

```
root#./kovanNagios -f kovan_ayar_dosyasi --configure
```

Kovan kurulduktan hemen sonra araçları yapılandırmak için, ilgili betikleri sırasıyla install, configure ve execute parametreleriyle çalıştırabilir veya parametreleri tek tek vermek yerine birlikte de kullanabilirsiniz.

```
root#./kovanNagios -f kovan_conf_file --install --configure --execute
     --mail mail@mail.com
```

Parametrelerin ne sırayla verildiği önemli değildir. Yukarıdaki örnekte "mail" parametresinin verildiği gibi her işlevsel parametre için verilmesi gereken diğer parametreler ayrıca belirtilmelidir. Bu parametreleri ve diğer seçenekleri görmek için "help" ' i kullanabilirsiniz :

```
root#./kovanNagios --help
Usage:  ./kovanNagios [-c kovan_conf_file] [--mail mail_address] [--snmp snmp_community]
                      [--install|--configure|--execute]
Options :
        -c kovan_conf_file                : kovan configuration file
        --snmp snmp_community             : snmp community name
        --mail mail_address               : mail address of nagios user
        --install                         : checks for files and copy them for configuring
        --configure                       : executes regex on files and configures them
        --execute                         : runs the binary
```

Bazı yapılandırma dosyaları birden fazla betik tarafından değiştirildiğinden, karışıklığı önlemek ve hangi kısmın hangi betik tarafından değiştirildiğini belirtmek amacıyla yorum standardı belirlenmiştir. Yorum standardı :

```
##kvn\_"Betik Adı"\_s
....
....
....
##kvn\_"Betik Adı"\_f
```

şeklindedir. Bu yorum satırları arasında kalan kısım kovan_BetikAdı betiği tarafından değiştirilmiş veya eklenmiştir anlamına gelmektedir. Örneğin kovanNagios betiği apache ' nin httpd.conf dosyasını değiştirdikten sonra şu satırlar görünecektir :

```
##kvn_nagios_s
    <Directory /usr/local/www/nagios>
    Options None
    AllowOverride None
    Order allow,deny
    Allow from all

    </Directory>

    <Directory /usr/local/www/nagios/cgi-bin>
    AllowOverride None
    Options ExecCGI
    Order allow,deny
    Allow from all

    </Directory>

    ScriptAlias /nagios/cgi-bin/ /usr/local/www/nagios/cgi-bin/
    Alias /nagios /usr/local/www/nagios
##kvn_nagios_f
```

4.1 kovanApachePhp

Monitor jail ' inin çalıştırdığı izleme araçları, web arayüzü üzerinden daha detaylı bilgi sunmaktadır. Bu araçların web arayüzünden verdiği raporları inceleyebilmek için araçlardan önce web sunucusunu yapılandırmak gerekmektedir. Apache (sürüm 2.2.17) web sunucusu, PHP (sürüm 5.3.3) desteği ile kurulum aşa-

masında yüklenmektedir. Apache ve PHP uygulamalarını ve monitordeki diğer dosyaları yapılandırmak için kovanApachePhp betiği yazılmıştır.

Araçlar kısmında da bahsedildiği gibi yapılandırılacak dosyaların hazırlanması "–install" parametresi kullanılarak sağlanır :

```
root#./kovanApachePhp -c kovanDefault.conf --install
/usr/local/kovan/monitor/usr/local/etc/apache22/httpd.conf checked
/usr/local/kovan/monitor/usr/local/etc/php.ini checked
INSTALL OK : 1
```

"install" kısmında betik sadece gerekli dosyaları kontrol etmektedir. Varsayılan Apache kurulumu yapılandırma dosyalarını Kovan ' ın istediği şekilde konumlandırmış olduğundan herhangi bir kopyalama işlemine gerek kalmamaktadır. Apache, Php ve sistem dosyalarının yapılandırılması için :

```
root# ./kovanApachePhp -c kovanDefault.conf --configure
/usr/local/kovan/monitor/usr/local/etc/apache22/httpd.conf checked
/usr/local/kovan/monitor/usr/local/etc/apache22/httpd.conf.temp file created
regex : Listen defined
regex : DirectoryIndex defined
regex : AddType defined
mv /usr/local/kovan/monitor/usr/local/etc/apache22/httpd.conf.temp /usr/local/kovan/moni
tor/usr/local/etc/apache22/httpd.conf
/usr/local/kovan/monitor/etc/hosts.temp file created
regex : 2001:a98:14:5::2    monitor_host defined
mv /usr/local/kovan/monitor/etc/hosts.temp /usr/local/kovan/monitor/etc/hosts
/usr/local/kovan/monitor/usr/local/etc/php.ini.temp file created
regex : date.timezone defined
mv /usr/local/kovan/monitor/usr/local/etc/php.ini.temp /usr/local/kovan/monitor/usr/local
/etc/php.ini
CONFIGURE OK : 1
```

Yapılandırma sırasında kovanApachePhp:

- Apache'nin çalışacağı IPv6 adresini tanımlar.
- Apache ' ye PHP desteğinin verilmesini sağlayan satırları ekler.
- Sistemin IPv6 adresi ve alanadı eşleşmesinin yapıldığı /etc/hosts dosyasını değiştirir.
- PHP için Timezone tanımını yapar.

Yapılandırmayı tamamlandıktan sonra Apache uygulamasını çalıştırmak için:

```
root#./kovanApachePhp -c kovanDefault.conf --execute
apache22 is not running.
Performing sanity check on apache22 configuration:
httpd: Could not reliably determine the server's fully qualified domain name, using
2001:a98:14:5::2 for ServerName
Syntax OK
Starting apache22.
httpd: Could not reliably determine the server's fully qualified domain name, using
2001:a98:14:5::2 for ServerName
EXECUTE OK : 1
```

Bu adımdan sonra tarayıcınızın adres satırına http://[MonitorIPv6Adresi] yazdığınız zaman KOVAN MONITOR ana sayfası görüntülenecektir. Diğer araçları da yapılandırdıktan sonra anasayfadaki linkler düzgün çalışacaktır.

4.2 kovanNagios

Nagios, birçok izleme yeteneği olan açık kaynak kodlu bir uygulamadır. Temel olarak yönlendiricilerin, düğümlerin ve servislerin çalışır durumda olup olmadıklarını izlemekte kullanılır. Nagios'un otomatik olarak ayarlanması için "kovanNagios" betiği geliştirilmiştir. Nagios ' un ping6 ' yı kullanarak ağdaki IPv6 düğümlerin çalışırlığını test edebilmesi için "kovanNagiosCheck" adlı eklenti yazılmıştır. Ayrıca Nagios, yönlendiricilerde bulunan softflowd, zebra gibi uygulamaları da snmp aracılığıyla test etmekte ve tüm bu bilgileri web arayüzünden grafiklerle göstermektedir.

Nagios, ana ayar dosyası nagios.cfg'ye ek olarak resource.cfg, cgi.cfg, commands.cfg, contacts.cfg, templates.cfg , timeperiods.cfg ve localhost.cfg adlı dosyalara sahiptir. Nagios kurulduğunda, ayar dosyaları *.cfg-sample formatında varsayılan içerikle oluşturulur. Bu dosyaların kullanılabilmesi için *.cfg uzantılı olacak şekilde nagios klasörüne kopyalanmaları gerekmektedir.

Monitor jaili kovan dizini altında oluşturulduğundan, nagios vb. uygulamalarda bu dizinin altında bulunacaktır. Bu dizin yolunu elde etmek ve gerekli diğer değerlere ulaşabilmek için, betikler kovan yapılandırma dosyasına ihtiyaç duyarlar. Tüm betikler çalıştırılırken bu dosyayı -c parametresi ile alırlar. Aşağıdaki komutla yapılandırılacak dosyalar uygun dizinlere kopyalanır ve gerekli diğer dosyalar da kontrol edilir :

```
root#./kovanNagios -c kovanDefault.conf --install
/usr/local/libexec/nagios/kovanNagiosCheck exists
cp /usr/local/kovan/monitor/usr/local/etc/nagios/nagios.cfg-sample   /usr/local/kovan/moni
tor/usr/local/etc/nagios/nagios.cfg
cp /usr/local/kovan/monitor/usr/local/etc/nagios/cgi.cfg-sample      /usr/local/kovan/moni
tor/usr/local/etc/nagios/cgi.cfg
cp /usr/local/kovan/monitor/usr/local/etc/nagios/resource.cfg-sample /usr/local/kovan/moni
tor/usr/local/etc/nagios/resource.cfg
cp /usr/local/kovan/monitor/usr/local/etc/nagios/objects/commands.cfg-sample    /usr/local/
kovan/monitor/usr/local/etc/nagios/objects/commands.cfg
cp /usr/local/kovan/monitor/usr/local/etc/nagios/objects/contacts.cfg-sample    /usr/local/
kovan/monitor/usr/local/etc/nagios/objects/contacts.cfg
cp /usr/local/kovan/monitor/usr/local/etc/nagios/objects/timeperiods.cfg-sample /usr/local/
kovan/monitor/usr/local/etc/nagios/objects/timeperiods.cfg
cp /usr/local/kovan/monitor/usr/local/etc/nagios/objects/templates.cfg-sample   /usr/local/
kovan/monitor/usr/local/etc/nagios/objects/templates.cfg
cp /usr/local/kovan/monitor/usr/local/etc/nagios/objects/localhost.cfg-sample   /usr/local/
kovan/monitor/usr/local/etc/nagios/objects/localhost.cfg
INSTALL OK : 1
```

Nagios'un web arayüzünün düzgün çalışabilmesi için httpd.conf dosyası içinde değişiklikler yapmak gerekmektedir. "install" modunda httpd.conf dosyasının ve kovanNagiosCheck betiğinin varlığı kontrol edilir. Herhangi birinin olmaması durumunda "install is not executed" hatası vererek çalışmayı sonlandırır.

Nagios'un ayar dosyalarının yapılandırılması ve httpd.conf'ta gerekli değişikliklerin yapılması için :

```
root# ./kovanNagios -c kovanDefault.conf --mail admin@administrator.com --snmp public --configure
/usr/local/kovan/monitor/usr/local/etc/apache22/httpd.conf.temp file created
regex : <Directory> checked
mv /usr/local/kovan/monitor/usr/local/etc/apache22/httpd.conf.temp /usr/local/kovan/monitor/
usr/local/etc/apache22/httpd.conf
/usr/local/kovan/monitor/usr/local/etc/nagios/objects/templates.cfg..temp file created
regex : check_command -> check-kovan-alive defined
mv /usr/local/kovan/monitor/usr/local/etc/nagios/objects/templates.cfg.temp /usr/local/
kovan/monitor/usr/local/etc/nagios/objects/templates.cfg
/usr/local/kovan/monitor/usr/local/etc/nagios/objects/contacts.cfg.temp file created
regex : email defined
```

```
mv /usr/local/kovan/monitor/usr/local/etc/nagios/objects/contacts.cfg.temp /usr/local/
kovan/monitor/usr/local/etc/nagios/objects/contacts.cfg
/usr/local/kovan/monitor/usr/local/etc/nagios/objects/commands.cfg.temp file created
regex : check_udp defined
regex : kovanNagiosCheck added end of the file
mv /usr/local/kovan/monitor/usr/local/etc/nagios/objects/commands.cfg.temp /usr/local/
kovan/monitor/usr/local/etc/nagios/objects/commands.cfg
/usr/local/kovan/monitor/usr/local/etc/nagios/objects/localhost.cfg.tmp file created
jexec: jail "anarouter" not found
regex : anarouter host definition defined
jexec: jail "ituh" not found
regex : ituh host definition defined
jexec: jail "comuh" not found
regex : comuh host definition defined
jexec: jail "monitor_host" not found
regex : monitor_host host definition defined
jexec: jail "comur1" not found
regex : comur1 host definition defined
jexec: jail "cdep1" not found
regex : cdep1 host definition defined
jexec: jail "cdep2" not found
regex : cdep2 host definition defined
jexec: jail "yavuz" not found
regex : yavuz host definition defined
jexec: jail "murat" not found
regex : murat host definition defined
jexec: jail "onur" not found
regex : onur host definition defined
jexec: jail "emre" not found
regex : emre host definition defined
regex : anarouter -  softflowd service definition defined
regex : ituh -  softflowd service definition defined
regex : comuh -  softflowd service definition defined
regex : comur1 -  softflowd service definition defined
regex : anarouter -  snmpd service definition defined
regex : ituh -  snmpd service definition defined
regex : comuh -  snmpd service definition defined
regex : comur1 -  snmpd service definition defined
regex : yavuz -  kovan_dns service definition defined
regex : onur -  kovan_http service definition defined
mv /usr/local/kovan/monitor/usr/local/etc/nagios/objects/localhost.cfg.tmp /usr/local/
kovan/monitor/usr/local/etc/nagios/objects/localhost.cfg
/usr/local/kovan/monitor/usr/local/etc/nagios/cgi.cfg.temp file created
regex : use_authentication defined
mv /usr/local/kovan/monitor/usr/local/etc/nagios/cgi.cfg.temp /usr/local/kovan/moni
tor/usr/local/etc/nagios/cgi.cfg
CONFIGURE OK : 1
```

kovanNagios "–configure" parametresi ile çalışırken iki ek parametre almaktadır. İlki, "kovanNagiosCheck" eklentisinin ihtiyacı olan snmp community parametresidir ve "–snmp değer" şeklinde verilir. Bu parametre snmpwalk kullanarak cihazları sorgularken gereklidir. İkinci parametre ise, Nagios ' un sistem yöneticisine uyarı epostası atarken kullanacağı eposta adresi bilgisidir.

Yapılandırma kısmında yapılan işlerin büyük bir bölümü düzenli ifadeler aracılığıyla gerçekleştirilir. Örneğin Nagios'un web dizin yolu ve cgi ayarları bu şekilde değiştirilmektedir. Yapılandırma sırasında kovanNagios :

- kovanNagiosCheck, check-kovan-alive, check_udp, komutlarını tanımlar.
- mail adresi gibi admin ile iletişimde kullanılacak olan değişkenleri atar.
- İzlenecek konak ve servis tanımlarını yapar.
- Web arayüzü yetkilendirmesi ile ilgili değişkenleri düzenler.
- Apache ' nin Nagios web dizinini görüntülemesi için gerekli ifadeleri ekler.

kovanNagios yapılandırma sırasında herhangi bir hatayla karşılaşılırsa " CONFIGURE IS NOT EXECUTED" şeklinde çıktı verir ve sonlanır. Yapılandırmayı tamamlandırdıktan sonra Nagios ' u çalıştırmak için :

```
root#./kovanNagios -c kovanDefault.conf --execute
nagios is not running.
Performing sanity check of nagios configuration: OK
Starting nagios.
apache22 is running as pid 88060.
Performing sanity check on apache22 configuration:
httpd: Could not reliably determine the server's fully qualified domain name, using
2001:a98:14:5::2 for ServerName
Syntax OK
Stopping apache22.
Waiting for PIDS: 88060.
Performing sanity check on apache22 configuration:
httpd: Could not reliably determine the server's fully qualified domain name, using
2001:a98:14:5::2 for ServerName
Syntax OK
Starting apache22.
httpd: Could not reliably determine the server's fully qualified domain name, using
2001:a98:14:5::2 for ServerName
EXECUTE OK : 1
```

"EXECUTE OK" çıktısını gördükten sonra, KOVAN MONITOR ana sayfasındaki "Services (Nagios)" linkini takip ederek Nagios web arayüzüne ulaşabilir, cihazların ve servislerin durumunu görebilirsiniz.

4.3 kovanMrtg

MRTG, sanal ağdaki bant-genişliği kullanımının izlenmesi amacıyla kullanılmaktadır. MRTG'nin ayar dosyalarını üreten "configmaker" ve web sayfalarını üreten "indexmaker" betikleri bulunmaktadır. Crontab tarafından her beş dakikada bir çalıştırılan MRTG, web arayüzünden sunduğu grafikleri güncellemektedir.

MRTG dosyalarını hazırlamak için :

```
root# ./kovanMrtg -c ../etc/kovanDefault.conf --install
rm -rf /usr/local/kovan/monitor/usr/local/etc/mrtg
mkdir -p /usr/local/kovan/monitor/usr/local/etc/mrtg
rm -rf /usr/local/kovan/monitor/usr/local/www/mrtg
mkdir -p /usr/local/kovan/monitor/usr/local/www/mrtg
mkdir -p /usr/local/kovan/monitor/usr/local/www/mrtg/bandwidth
mkdir -p /usr/local/kovan/monitor/usr/local/www/mrtg/pps
INSTALL OK : 1
```

Çıktıda da görüldüğü üzere, MRTG ' nin yapılandırma ve web dizinleri oluşturulur. MRTG yapılandırması için :

```
root# ./kovanMrtg -c kovanDefault.conf --configure
jexec MONITOR csh -c "/usr/local/bin/cfgmaker --enable-ipv6  --snmp-options=:::::2
--global \"Options[_]: growright, bits\"   --global WorkDir:/usr/local/www/mrtg/bandwidth
 \"public@[2001:a98:14::6]\"  \"public@[2001:a98:14::2]\" \"public@[2001:a98:14::e]\"
 \"public@[2001:a98::173]\"  \"public@[2001:a98:14::a]\"    | sed 's/1250000/1250000000/g'
   | sed 's/10.0 Mbits/1 Gbps/g'   > /usr/local/etc/mrtg/bandwidth.conf"
jexec MONITOR csh -c "/usr/local/bin/cfgmaker --enable-ipv6  --snmp-options=:::::2
--if-template=/etc/MrtgPpsTemplate.txt  --global WorkDir:/usr/local/www/mrtg/pps
--global \"Options[_]: growright, bits\"      \"public@[2001:a98:14::6]\"  \"public@[
2001:a98:14::2]\"  \"public@[2001:a98:14::e]\"  \"public@[2001:a98::173]\"
\"public@[2001:a98:14::a]\"   > /usr/local/etc/mrtg/pps.conf"
SNMPopen failed: Unable to resolve the UDP/IPv4 address "[2001:a98:14::6]" at /usr/local
/bin/cfgmaker line 147
SNMPWALK Problem for public@[2001:a98:14::a]:::::2 at /usr/local/lib/perl5/site_perl/
5.10.1/MRTG_lib.pm line 1717
jexec MONITOR csh -c "/usr/local/bin/indexmaker --title 'Kovan MRTG Bandwidth Statistics'
/usr/local/etc/mrtg/bandwidth.conf --output /usr/local/www/mrtg/bandwidth/bandwidth.html"
jexec MONITOR csh -c "/usr/local/bin/indexmaker --title 'Kovan MRTG pps Statistics'  /usr
/local/etc/mrtg/pps.conf --output /usr/local/www/mrtg/pps/pps.html"
jexec MONITOR csh -c 'crontab -l > /tmp/mrtg_cron'
/usr/local/kovan/monitor/tmp/mrtg_cron.temp file created
regex : 5,10,15,20,25,30,35,40,45,50,55  * * * *  /usr/local/bin/mrtg /usr/local/etc/
mrtg/bandwidth.conf
5,10,15,20,25,30,35,40,45,50,55  * * * *  /usr/local/bin/mrtg /usr/local/etc/mrtg/pps.conf
defined
mv /usr/local/kovan/monitor/tmp/mrtg_cron.temp /usr/local/kovan/monitor/tmp/mrtg_cron
/usr/local/kovan/monitor//usr/local/etc/apache22/httpd.conf.temp file created
regex : <Directory> checked
mv /usr/local/kovan/monitor//usr/local/etc/apache22/httpd.conf.temp /usr/local/kovan/moni
tor//usr/local/etc/apache22/httpd.conf
CONFIGURE OK : 1
```

"SNMPopen failed: Unable to resolve the UDP/IPv4 address" ve "SNMPWALK Problem for public@[2001:a98:14::a]:::::2" şeklindeki uyarı mesajları göz ardı edilebilir. Yapılandırma sırasında kovanMrtg:

- configmaker betiğini kullanarak kovan yapılandırma dosyasından okunan yönlendirici IPv6 adreslerine göre mrtg.conf dosyasını oluşturur.
- indexmaker betiğini kullanarak mrtg.conf ' tan web dosyaları türetir ve bunları web dizini altına kopyalar.
- Apache ' nin MRTG web dizinini görüntülemesi için gerekli ıfadeleri ekler.
- son olarak crontab ' ı her 5 dakikada bir grafikleri güncelleyecek şekilde ayarlar.

```
5,10,15,20,25,30,35,40,45,50,55  * * * *  /usr/local/bin/mrtg /usr/local/kovan/mrtg/mrtg.conf
```

Yapılandırma tamamlandıktan sonra crontab çalıştırılır, çünkü MRTG cron üzerinden çalıştırılmaktadır.

```
root#./kovanMrtg -c kovanDefault.conf --execute
/usr/bin/crontab -u root /tmp/mrtg_cron
apache22 is running as pid 88274.
Performing sanity check on apache22 configuration:
httpd: Could not reliably determine the server's fully qualified domain name,
```

```
using 2001:a98:14:5::2 for ServerName
Syntax OK
Stopping apache22.
Waiting for PIDS: 88274.
Performing sanity check on apache22 configuration:
httpd: Could not reliably determine the server's fully qualified domain name,
using 2001:a98:14:5::2 for ServerName
Syntax OK
Starting apache22.
httpd: Could not reliably determine the server's fully qualified domain name,
using 2001:a98:14:5::2 for ServerName
EXECUTE OK : 1
```

Çıktıda görüldüğü gibi mrtg cron ifadesinin yer aldığı dosya cron tarafından çalıştırılır ve Apache yeniden başlatılarak MRTG web arayüzü güncellenir. KOVAN MONITOR anasayfasındaki "Bandwidth Graphs" linki takip edilerek grafiklere ulaşılabilir.

4.4 kovanNfsen

Nfsen, Netflow aracılığıyla sanal yönlendiricilerden gönderilen flow verilerini toplar. Bu verilerle ilgili istatistikleri üreterek web arayüzünden grafiklerle sunan bir uygulamadır. Nfsen ' in flow topladığı kaynaklar nfsen.conf dosyasında belirtilir. Aşağıdaki komutla yapılandırılacak dosyalar uygun dizinlere kopyalanır ve gerekli diğer dosyalar da kontrol edilir :

```
root# ./kovanNfsen -f kovanDefault.conf --install
rm -f /usr/local/kovan/monitor/usr/local/etc/nfsen.conf
cp /usr/local/kovan/monitor/usr/local/etc/nfsen-dist.conf  /usr/local/kovan/
monitor/usr/local/etc/nfsen.conf
State reconfig ....
Reconfig: No changes found!
INSTALL OK : 1
```

Dosyalar hazırlandıktan sonra Nfsen dosyalarının ve web arayüzünün yapılandırılması için :

```
root# ./kovanNfsen -c kovanDefault.conf --configure -s 4
/usr/local/etc/nfsen.conf.temp file created
regex : \%sources defined
regex : USER defined
regex : WWWUSER defined
regex : WWWGROUP defined
regex : SUBDIRLAYOUT defined
mv /usr/local/kovan/monitor/usr/local/etc/nfsen.conf.temp /usr/local/kovan/monitor
/usr/local/etc/nfsen.conf
/usr/local/kovan/monitor/usr/local/etc/apache22/httpd.conf.temp file created
regex : <Directory> defined
mv /usr/local/kovan/monitor/usr/local/etc/apache22/httpd.conf.temp /usr/local/kovan/monitor
/usr/local/etc/apache22/httpd.conf
CONFIGURE OK : 1
```

Yapılandırma sırasında kovanNfsen:

- Flow kaynaklarını ve çeşitli değişkenlerini tanımlar,
- Apache ' nin Nfsen web dizinini görüntülemesi için gerekli ifadeleri ekler.

Son olarak Nfsen ' i çalıştırmak için:

```
root#./kovanNfsen -c kovanDefault.conf --execute
New sources to configure : BLACKHOLE_eth1 R2_eth1 FW_eth3 FW_eth2 OMURGA_eth2 OMURGA_eth0
FW_eth0 OMURGA_eth3 R2_eth0 R1_eth1 FW_eth1 BLACKHOLE_eth0 OMURGA_eth1 R1_eth0
Remove configured sources: upstream1 peer1
Continue? [y/n]
Add source(s): BLACKHOLE_eth1 R2_eth1 FW_eth3 FW_eth2 OMURGA_eth2 OMURGA_eth0 FW_eth0
OMURGA_eth3 R2_eth0 R1_eth1 FW_eth1 BLACKHOLE_eth0 OMURGA_eth1 R1_eth0:
Add source 'BLACKHOLE_eth1'
Add source 'R2_eth1'
Add source 'FW_eth3'
Add source 'FW_eth2'
Add source 'OMURGA_eth2'
Add source 'OMURGA_eth0'
Add source 'FW_eth0'
Add source 'OMURGA_eth3'
Add source 'R2_eth0'
Add source 'R1_eth1'
Add source 'FW_eth1'
Add source 'BLACKHOLE_eth0'
Add source 'OMURGA_eth1'
Add source 'R1_eth0'
Delete source(s): upstream1 peer1:
Delete source 'upstream1'
Delete source 'peer1'
No pid file found for nfsend - please restart manually
Starting nfcapd: BLACKHOLE_eth1[32453] R2_eth1[32456] FW_eth3[32459] FW_eth2[32462]
OMURGA_eth2[32465] OMURGA_eth0[32468] FW_eth0[32471] OMURGA_eth3[32474]
R2_eth0[32477] R1_eth1[32480] FW_eth1[32487] BLACKHOLE_eth0[32490]
OMURGA_eth1[32493] R1_eth0[32496].
Starting nfsend.
apache22 is running as pid 32222.
Performing sanity check on apache22 configuration:
httpd: Could not reliably determine the server's fully qualified domain name,
using 2001:a98:14:5::2 for ServerName
Syntax OK
Stopping apache22.
Waiting for PIDS: 32222.
Performing sanity check on apache22 configuration:
httpd: Could not reliably determine the server's fully qualified domain name,
using 2001:a98:14:5::2 for ServerName
Syntax OK
Starting apache22.
httpd: Could not reliably determine the server's fully qualified domain name,
using 2001:a98:14:5::2 for ServerName
EXECUTE OK : 1
```

Nfsen kaynaklarda bir değişiklik olursa öncelikle "nfsen reconfig" komutunun çalışmasını şart koşar. Bu komut ile yeni kaynaklar için onay alınır. Daha sonra uygulamayı yeniden başlatır ve kaynaklar eklenir. Apache ' nin kendi yapılandırmasını güncellemesi için de web sunucusu yeniden başlatılır. Herhangi bir hata mesajı alınmazsa KOVAN MONITOR sayfasındaki "NetFlow (Nfsen)" linki takip edilerek Nfsen web arayüzüne ulaşılabilir.

4.5 kovanState

Lütfen Bölüm 3.2'ye bakınız.

4.6 kovanStats

Tüm servisler ve karadelik, loglarını düzenli olarak monitör cihazına göndermektedirler(Bakınız Bölüm 13). Monitor ' de toplanan bu loglardan istatistik çıkarmak ve grafiklerini çizmek için kovan_graph betiği yazılmıştır. kovan_graph aşağıdaki seçeneklere göre ayrıntılı istatistikler çıkarabilir. Kovan kullanıcısının genellikle saatlik ve günlük istatistiklere ihtiyacı olacağı düşünüldüğünden, bunları gerçekleştiren betikler yazılmış ve kovanStats betiği altında toplanmıştır. KovanStats yapılandırma sırasında :

- kovan_graph ' ın çalışacağı dizinleri ve dosyaları hazırlar.
- Grafik oluşturan betiklerin çalışacağı saat dilimlerine göre cron ' u düzenler.
- Apache ' nin istatistik web dizinini görüntülemesi için gerekli ifadeleri ekler.

kovan_graph, Monitor jaili içerisinde kovan dizininde bulunmaktadır.

```
root# ./kovan_graph
usage: kovan_graph --outputfile NAME --xaxis TITLE --yaxis TITLE --query QUERY
--header HEADER
```

Birinci argüman (–outputfile) üretilecek grafik dosyasının adını belirlemektedir. Grafikte yer alan X/Y eksenlerinin isimleri ise –xaxis ve –yaxis argümanları aracılığıyla verilmektedir. Dördüncü argüman (–query) mysql veritabanında gerekli bilgiyi çekecek sql cümlesini belirlemektedir. Burada kullanılacak sql cümlelerinin iki kolonlu bir cevap dönmesi gerekmektedir; ilk kolondaki değerler x ekseninde kullanılırken ikinci kolon y ekseninde kullanılmaktadır. Son argüman grafiğin başlığını belirlemektedir.

kovanStats betiği aşağıdaki gibi kullanılarak istatistik yapılandırması gerçekleştirilebilir.

```
root# ./kovanStats -c kovanDefault.conf --install
INSTALL OK : 1
root# ./kovanStats -c kovanDefault.conf --configure
jexec MONITOR csh -c 'crontab -l > /tmp/stats_cron'
regex : 10  * * * *  /usr/local/kovan/bin/kovanGraph_hourly.sh
0 3  * * *  /usr/local/kovan/bin/kovanGraph_daily.sh checked
mv /usr/local/kovan/monitor/tmp/stats_cron.temp /usr/local/kovan/monitor/tmp/stats_cron
/usr/local/kovan/monitor//usr/local/etc/apache22/httpd.conf.temp file created
regex : <Directory> checked
mv /usr/local/kovan/monitor//usr/local/etc/apache22/httpd.conf.temp /usr/local/kovan/
monitor//usr/local/etc/apache22/httpd.conf
CONFIGURE OK : 1
root# ./kovanStats -c kovanDefault.conf --execute
/usr/bin/crontab -u root /tmp/stats_cron
apache22 is running as pid 35676.
Performing sanity check on apache22 configuration:
httpd: Could not reliably determine the server's fully qualified domain name,
using 2001:a98:14:5::2 for ServerName
Syntax OK
Stopping apache22.
Waiting for PIDS: 35676.
Performing sanity check on apache22 configuration:
httpd: Could not reliably determine the server's fully qualified domain name,
```

```
using 2001:a98:14:5::2 for ServerName
Syntax OK
Starting apache22.
httpd: Could not reliably determine the server's fully qualified domain name,
using 2001:a98:14:5::2 for ServerName
EXECUTE OK : 1
```

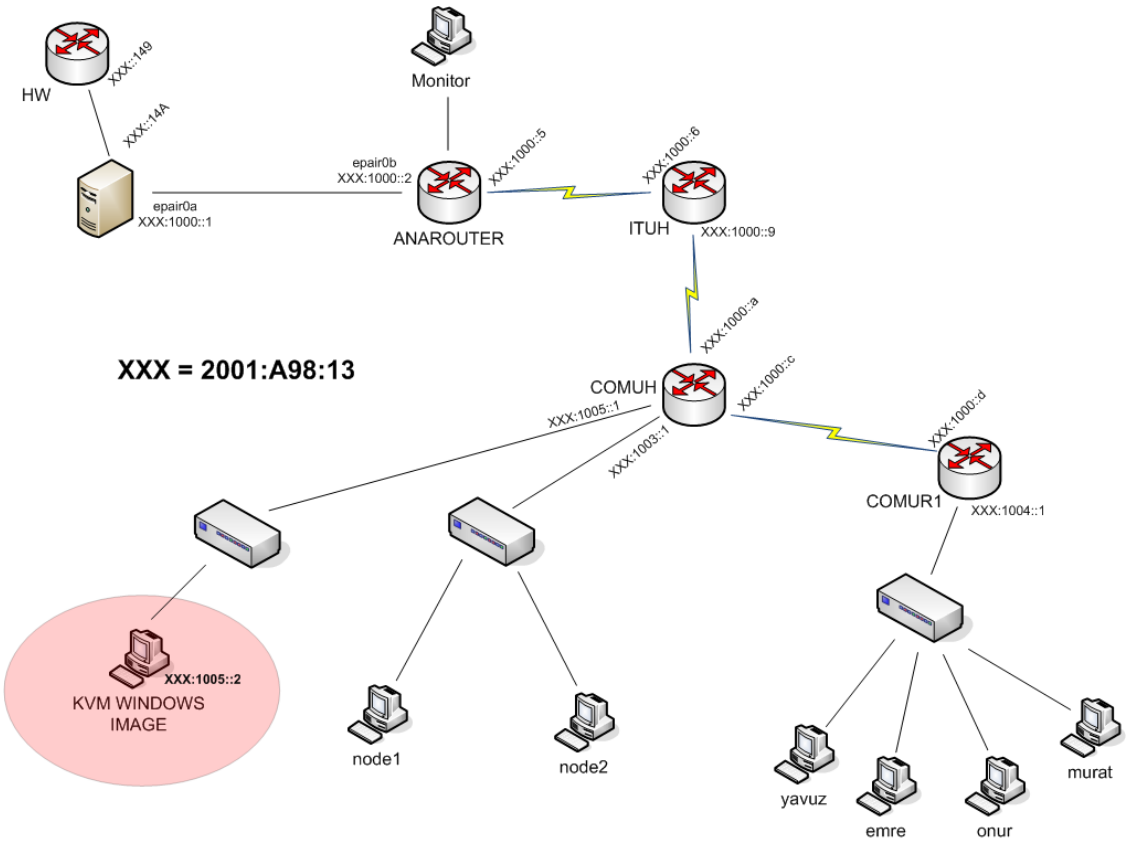
Apache yeniden başlatıldıktan sonra KOVAN MONITOR sayfasından "Honeypot Statistics " linki takip edilerek grafiklere ulaşılabilirsiniz.

4.7 kovanManageLinks

kovanManageLinks cihazlar arası bağlantıların özelliklerini değiştirmek için kullanılan bir araçtır. kovanManageLinks kullanılarak, bütün bağlantıların (fiziksel arayüze kovan'ı bağlayan bağlantı hariç) özellikleri görülebilir, değitirilebilir ve varsayılan ayarlara çekilebilir. Aşağıda betiğin kullanımı ve argümanları görülebilmektedir:

```
root# ./kovanManageLinks
usage:        kovanManageLinks --all [--show]
              kovanManageLinks --show link_name
              kovanManageLinks --reset link_name
              kovanManageLinks --delay=X(s|ms|us)
                   --bandwidth=Y(gbps|mbps|kbps|bps)
                   --upstream_ber=Z --downstream_ber=T link_name
```

"–all" argümanı sistemdeki tüm "değiştirilebilir" bağlantıları listelemektedir. Örnek bir kullanım için, kovanManageLinks betiğini şekil 4.1'te gösterilen ağ üzerinde çalıştırabiliriz.



Şekil 4.1: Örnek Kovan Topolojisi

"–all" kullanılarak tüm bağlantıların listelenmesi liste 4.1'te gösterilmektedir.

```
1  root# ./perl kovanManageLinks --all
2  ituh-comuh
3  yavuz-bridge2
4  onur-bridge2
5  cdep1-bridge1
6  cdep2-bridge1
7  comuh-comur1
8  comur1-bridge2
9  anarouter-monitor_host
10 anarouter-ituh
11 comuh-bridge1
12 murat-bridge2
13 emre-bridge2
```

Listing 4.1: Tüm bağlantıları listeleme

"–show" argümanı bir bağlantının ya da bütün bağlatıların ayrıntılarını listeler. Liste 4.1'te görülen herhangi bir bağlantının ayrıntılarına liste 4.2'deki gibi erişilebilmektedir:

```
1  root# ./kovanManageLinks --show comuh-comur1
2  comuh-comur1:
```

```
3     Bandwidth: default
4     Delay: default
5     UpStream BER: default
6     DownStream BER: default
```

Listing 4.2: Bir bağlantının ayrıntısı

kovanManageLinks tek bir bağlantının ayrıntılarını sergileme yerine tüm bağlantıları ayrıntılı olarak listelemek için de kullanılabilmektedir (liste 4.3):

```
1  root# ./kovanManageLinks --show --all
2  ituh-comuh:
3     Bandwidth: default
4     Delay: default
5     UpStream BER: default
6     DownStream BER: default
7
8  yavuz-bridge2:
9     Bandwidth: default
10    Delay: default
11    UpStream BER: default
12    DownStream BER: default
13
14 onur-bridge2:
15    Bandwidth: default
16    Delay: default
17    UpStream BER: default
18    DownStream BER: default
19
20 cdep1-bridge1:
21    Bandwidth: default
22    Delay: default
23    UpStream BER: default
24    DownStream BER: default
25
26 cdep2-bridge1:
27    Bandwidth: default
28    Delay: default
29    UpStream BER: default
30    DownStream BER: default
31
32 comuh-comur1:
33    Bandwidth: default
34    Delay: default
35    UpStream BER: default
36    DownStream BER: default
37
38 comur1-bridge2:
39    Bandwidth: default
40    Delay: default
41    UpStream BER: default
42    DownStream BER: default
43
```

```
anarouter-monitor_host:
    Bandwidth: default
    Delay: default
    UpStream BER: default
    DownStream BER: default

anarouter-ituh:
    Bandwidth: default
    Delay: default
    UpStream BER: default
    DownStream BER: default

comuh-bridge1:
    Bandwidth: default
    Delay: default
    UpStream BER: default
    DownStream BER: default

murat-bridge2:
    Bandwidth: default
    Delay: default
    UpStream BER: default
    DownStream BER: default

emre-bridge2:
    Bandwidth: default
    Delay: default
    UpStream BER: default
    DownStream BER: default
```

Listing 4.3: Details of one link

kovanManageLinks betiği sadece bağlantıları listelemekte değil onların ayarlarında değişiklik yapılmasında da kullanılabilmektedir. Gecikme(delay), bant genişliği(bandwidth), upstream BER(bit hata oranı) ve downstream BER(bit hata oranı) gibi bağlantı özellikleri kovanManageLinks betiği aracılığı ile değiştirilebilir. Gecikme değeri "–delay=X(s|ms|us)" argümanıyla belirlenir; burada X gecikmenin miktarını, "s|ms|us" de gecikmenin birimini belirtir. Band genişliği "–bandwidth=Y(gbps|mbps|kbps|bps)" argümanıyla belirlenir; burada Y bant genişliğinin miktarını, "gbps|mbps|kbps|bps" de birimini belirtir. BER değerleri sayı olarak verilmektedir, örneğin, eğer BER değeri T ise bit hatası oranı 1/T olarak belirlenir.

Liste 4.4 bağlantı özelliklerinin değiştirilmesiyle ilgili bir örneği göstermektedir. Bu örnekte, anarouter-ituh bağlantısının gecikmesi 3 milisaniye ve bant genişliği de 5 mbps olarak belirlenmektedir.

```
root#./kovanManageLinks --delay=3ms --bandwidth=5mbps anarouter-ituh
anarouter-ituh:
    Bandwidth: 5Mbps
    Delay: 3ms
    UpStream BER: default
    DownStream BER: default
```

Listing 4.4: Tüm bağlantıları listeleme

4.8 kovanMonitorServices

Araçlar kısmında da belirtildiği gibi betiklerin tamamı (kovanManageLinks hariç) ortak parametrelerle çalışmaktadır : "install", "configure" ve "execute". kovanMonitorServices betiği, bu parametrelerin her birinin yaptığı işi tüm betikleri çağırarak yapmakta ve kısa yoldan tüm araçları yapılandırmaktadır.

Yukarıda diğer betiklerin yapılandırılmasında olduğu gibi kovanMonitorServices betiği için de aynı adımları takip ederek yapılandırmayı kısa yoldan bitirebilirsiniz. kovanMonitorServices kullanım seçenekleri aşağıdaki şekildedir.

```
1 root# ./kovanMonitorServices
2 usage:  kovanMonitorServices --conf conffile --snmp snmppasswd --mail adminmail
3                              [--list] [--install|--configure|--execute] [--all]
```

"–list" parametresi –install, –configure, –execute veya –all parametrelerinden biri ile kullanılmaktadır. "–all" parametresi sırasıyla install, configure ve execute parametreleri ile betiği çalıştırır.

Bölüm 5

Kurulum

5.1 Yeni Başlayanlara Öneriler

- Eğer tecrübeli bir FreeBSD kullanıcısı değilseniz önceden hazırlanmış Kovan KVM/QEMU imajlarının kullanılması tavsiye edilmektedir.
- Orta seviyeli FreeBSD kullanıcıları Kovan kurulumu için 5.3 Çabuk Kurulum kısmında belirtilen adımları takip edebilirler. Kovan'ın hızlı bir şekilde kurulabilmesi için önerilen yöntem bu adımların takip edilmesidir.
- Çok iyi bir FreeBSD kullanıcısıysanız ve sunucuma her programı kendim kurarım diyorsanız 5.4 Adım Adım Kurulum kısmı altındaki basamakları takip edebilirsiniz.
- Kurulum betikleri FreeBSD 8.1 ile sorunsuz çalışmaktadır. Kurulum için FreBSD 8.1 sürümünün kullanılması önerilmektedir.

5.2 Zorunlu Ayarlar

- Kovan yeni yüklenmiş bir FreeBSD sistemine kurulmalıdır. Kurulumun, üzerinde başka servisler çalışan bir sistem üstünde kurulması sorunlara yol açabilmektedir.
- Kovan kurumundan önce işletim sisteminin IP ve DNS yapılandırması tamamlanmış olmalıdır. Kovan'ın yalın IPv6 çalıştırılması durumunda /etc/resolv.conf DNS yapılandırma dosyasına IPv6 üzerinden erişilebilen DNS sunucusu eklenmelidir. Benzer şekilde eğer Kovan ikili yığın olarak çalıştırılacak ise DNS sunucularıda her iki protokol için tanımlanmalıdır.
- Kovan'ı kurmadan önce varsayılan parolaları değiştirmek isterseniz Bölüm 9.1 kısmına bakınız.
- Kovan kurulum betikleri bash kabuğunu kullanmaktadır. Bu nedenle yeni kurulan sisteme bash kabuğu yüklenmelidir.
- Kovan kurulum betikleri kurulum esnasında herhangi bir hata olduğunda sonlanacak şekilde yazılmıştır.
- Kovan adım adım kurulum yöntemi ile kurulacak ise FreeBSD port sisteminin çalışabilmesi için IPv4 bağlantısına da ihtiyaç bulunmaktadır.
- Kovan'ın kurulumu sırasında gerekli programlar kurulum betikleri tarafında FreeBSD Port sistemini kullanarak derlenmektedir. Bu derleme işlemi sonucunda yüklenmek istenen port sisteme daha önceden kurulmuş ise kurulum betiği yükleme işlemini durdurur. Bu durumu ortadan kaldırmak için FORCE_PKG_REGISTER 'a 1 değeri verilebilir (export FORCE_PKG_REGISTER=1).

5.3 Çabuk Kurulum

Kovan'ı kurmadan önce varsayılan parolaları değiştirmek isterseniz Bölüm 9.1 kısmına bakınız. Kovan kurulumu iki aşamada yapılmakta ve bu iki aşama için iki ayrı betik kullanılmaktadır. Kurulumun birinci aşaması için **preInstall.sh** betiği kullanılmaktadır. İlk aşamanın başlatılması için sistemin IP ve DNS yapılandırması yapıldıktan sonra root kullanıcısı ile preInstall.sh betiğinin çalıştırılması yeterlidir. preInstall.sh Kovan yüklenmeden önce FreeBSD işletim sistemindeki:

- Port ve kaynak kodlarının güncellenmesi
- Çekirdeğe VIMAGE desteği verilerek derlenmesi
- IPv6 yönlendirme desteğinin açılması
- İşletim sistemi IPC ayarlarını değiştirilmesi
- Ezjail, perl, p5YAMLTiny, p5GetoptLong, automake,snort portlarının

kurulum işlemlerini gerçekleştirerek sistemi yeniden başlatmaktadır. preInstall betiğinin kurulum süresi internet bağlantı hızınız ve sunucunuzun donanım özelliklerine göre 1 saat kadar sürebilir.

Kurulumun ikinci aşamasında install.sh betiği kullanılmaktadır. **install.sh** betiği

- Monitor, router, node, blackhole jaillerinin ve bu jailler altında gerekli programların kurulumu
- Kovan kaynak kodlarının derlenerek kurulum sırasında verilen PREFIX tanımına göre kopyalama

işlemlerini gerçekleştirmektedir. Install.sh betiği 3 adet argüman almaktadır.

1. **PREFIX** Kovan kurulum dizini
2. **IP** Kovan jail kurulumunu gerçekleştirmesi için sunucu ile aynı alt ağ adresine sahip IP adresi
3. **INTERFACE** İnternete bağlanmak için kullanılan fiziksel ağ arabirimi

em0 ağ arabirimine 10.10.20.3 IP adresi atanmış bir sunucuda Kovan'ın /usr/local/kovan altına kurulması için install.sh betiğinin

```
1 install.sh /usr/local/kovan 10.10.20.4 em0
```

argümanları ile çalıştırılması yeterlidir. install.sh betiğinin kurulum süresi internet bağlantı hızınız ve sunucunun donanım özelliklerine göre 1-2 saat kadar sürebilir.

5.4 Adım adım Kurulum

5.4.1 FreeBSD'nin Kovan Kurulumu İçin Hazırlanması

Kovan'ı kurmadan önce varsayılan parolaları değiştirmek isterseniz Bölüm 9.1 kısmına bakınız.

FreeBSD kaynak ağacının güncellenmesi(mecburi değil)

Cvsup uygulaması FreeBSD port ağacını güncellemek için kullanılabilir. Cvsup portunu yüklemek için

```
1 root#pkg_add -r cvsup-without-gui
```

komutunu kullanabilirsiniz. Bu aşamanın ardından aşağıdaki satırları içeren bir dosya (/etc/stable-supfile) yaratılmalıdır. RELENG satırı kurulum yapılan FreeBSD sürümüne göre değiştirilmelidir.

```
*default host=tulumba.ulakbim.gov.tr
*default base=/usr
*default prefix=/usr
*default release=cvs tag=RELENG_8_1
*default delete use-rel-suffix
src-all
```

FreeBSD portlarının güncellenmesi(mecburi değil)

FreeBSD port ağacının güncellenmesi için Portsnap komutu kullanılabilir.

```
root#portsnap -s portsnap2.freebsd.org fetch
root#portsnap extract
```

Diğer bir yöntem cvsup ile portların da güncellenmesidir. Aşağıdaki satırlar stable-supfile'a yazıldığı takdirde hem portlar hem de kaynak kodları güncellenir.

```
*default host=cvsup.freebsd.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=RELENG_8_1
*default delete use-rel-suffix
src-all
ports-all tag=.
```

Bu aşamalardan sonra FreBSD kaynak kodları ve port ağacı cvsup komutuyla aşağıdaki gibi güncellenebilir.

```
root#cvsup  /stable/supfile
```

Cron kullanılarak bu işlemler otomatik hale getirilebilir.

```
30 12 * * * /usr/sbin/portsnap update
```

Çekirdeğin VIMAGE desteği ile derlenmesi

FreeBSD 8.1 ve yukarısı için aşağıdaki satır KERNEL yapılandırma dosyasına (AMD64 sistemler için /usr/local/sys/amd64/conf/GENERIC) eklenir.

```
options VIMAGE
```

FreeBSD 8.0 kullanılıyorsa, SCTP ile ilgili satır silinmelidir.

```
#options SCP
```

Çekirdek derlenir (çekirdek yapılandırma dosyasının /usr/local/sys/amd64/conf/GENERIC olduğu varsayılmıştır)

```
root# cd /usr/src/sys/amd64/conf
root# config GENERIC
root# cd ../compile/GENERIC/
root#make cleandepend && make depend && make && make install
```

Sistem yeniden başlatıldığında değişiklikler devreye girmektedir.

IPv6 desteği ve IPv6 yönlendirmesinin açılması

Aşağıdaki satırlar /etc/rc.conf dosyasına eklenerek FreeBSD'ye IPv6 desteği verilir.

```
ipv6_enable="YES"
```

Aşağıdaki satırlar /etc/sysctl.conf dosyasına eklenir.

```
net.inet.ip.forwarding=1
net.inet6.ip6.forwarding=1
security.jail.allow_raw_sockets=1
```

IPC ayarlarının yapılması

Aşağıdaki satırları /boot/loader.conf dosyasına ekleyiniz.

```
kern.ipc.shmall=32768
kern.ipc.shmmax=134217728
kern.ipc.semmap=256
kern.ipc.semmni=256
kern.ipc.semmns=512
kern.ipc.semmnu=256
```

5.4.2 Kovan İçin Gerekli Portların kurulumu

Kovan, Jail'lerin kurulumu için ezjail uygulamasını kullanmaktadır.

```
root#cd /usr/ports/sysutils/ezjail
root#make -DBATCH install clean
root#ezjail-admin install
```

Aşağıdaki uygulamaların da kurulması gerekmektedir.

```
root# cd /usr/ports/lang/perl5.10 && make -DBATCH install clean
root# cd /usr/ports/textproc/p5-YAML-Tiny/ &&   make -DBATCH    install clean
root# cd /usr/ports/devel/p5-Getopt-Long &&   make -DBATCH  install clean
root# export CONFIGURE_ARGS="--enable-ipv6 --enable-gre --enable-targetbased
--enable-decoder-preprocessor-rules --enable-zlib"
root# cd /usr/ports/devel/automake111 && make -DBATCH install clean
root# cd /usr/ports/security/snort && make -DBATCH -DWITH_SNORTSAM install clean
```

5.4.3 Jail Kurulumu

Dört çeşit Kovan jail türü bulunmaktadır, router, monitor, node ve blackhole. Kovan'ın bütün özelliklerinin kullanılabilmesi için jaillerin tümü kurulmalıdır . Her bir jail tipi içini gerekli uygulamalar FreeBSD port ağacı yardımıyla derlenerek yüklenmektedir. Daha uzun sürmesine karşın kurulacak uygulamalara IPv6 desteğinin verildiğinden emin olunabilmesi için kaynak kodundan derleyerek kurulum yapma opsiyonu seçilmiştir. Jail'lerin kurumu için aşağıdaki adımların takip edilmesi gerekmektedir.

1. ezjail portu kullanılarak kurulum yapılmalıdır (örn ezjail-admin create)
2. Jailler Kovan kurulum dizini altında kurulmalıdır (örn /usr/local/kovan)

3. Jaillerin isimleri ile kurulum dizinlerinin ismi aynı olmalıdır (örn monitor jaili için /usr/local/kovan/-montitor)

4. /usr/ports mount_nullfs komutu ile jail dizini altında /usr/ports dizini mount edilmelidir (örn mount_nullfs -o ro /usr/ports /usr/jail_path/usr/ports)

5. Her bir jail tipi için aşağıdaki komutlar çalıştırılmaldır.

Node

```
root# pkg_add -r bash
root# mkdir -p /usr/local/kovan/etc
root# cd /usr/ports/ports-mgmt/portaudit && make install clean
root# /usr/local/sbin/portaudit -Fda
root# cd /usr/ports/textproc/p5-YAML-Tiny/ ; make -DBATCH -DFORCE_PKG_REGISTER install clean
root# cd /usr/ports/devel/p5-Getopt-Long ; make -DBATCH -DFORCE_PKG_REGISTER install clean
```

Router

```
root# pkg_add -r bash
root# mkdir -p /usr/local/kovan/etc
root# cd /usr/ports/ports-mgmt/portaudit && make install clean
root# /usr/local/sbin/portaudit -Fda
root# cd /usr/ports/net-mgmt/net-snmp ; make -DBATCH -DWITH_IPV6 install clean
root# cd /usr/ports/net/quagga && make  -DBATCH -DWITH_RTADV -DWITH_SNMP install clean
root# cd /usr/ports/textproc/p5-YAML-Tiny/; make -DBATCH    -DFORCE_PKG_REGISTER install clean
root# cd /usr/ports/devel/p5-Getopt-Long; make -DBATCH  -DFORCE_PKG_REGISTER install clean
root# cd  /usr/ports/net-mgmt/softflowd ; make -DBATCH install clean
```

Monitor

```
root# cd /usr/ports/ports-mgmt/portaudit && make install clean
root# /usr/local/sbin/portaudit -Fda
root# pkg_add -r bash
root# mkdir -p /usr/local/kovan/etc
root# cd /usr/ports/www/apache22 && make  -DBATCH install clean
root# cd /usr/ports/lang/php5 && make -DBATCH -DWITH_APACHE -DWITH_IPV6
root# cd /usr/ports/net-mgmt/net-snmp && make -DBATCH -DWITH_IPV6 install
root# cd /usr/ports/net-mgmt/nfsen && make  -DBATCH install clean
root# cd /usr/ports/net-mgmt/mrtg/&& make -DBATCH  -DWITH_IPV6 -DWITH_SNMP install clean
root# cd /usr/ports/net-mgmt/nagios-plugins && make -DBATCH -DWITH_JAIL -DWITH_IPV6
-DWITH_NETSNMP -DWITH_FPING install clean
root# /usr/sbin/pw groupadd nagios
root# /usr/sbin/pw adduser nagios nagios
root# cd /usr/ports/net-mgmt/nagios  && make -DBATCH  install clean
root# cd /usr/ports/textproc/p5-YAML-Tiny/; make -DBATCH install clean
root# cd /usr/ports/devel/p5-Getopt-Long; make -DBATCH  install clean
root# cd /usr/ports/databases/mysql51-server ; make -DBATCH  install clean
root# cd /usr/ports/databases/libdbi-drivers ; make -DBATCH -DWITHOUT_PGSQL -DWITHOUT_SQLITE3
 -DWITH_MYSQL install clean
root# cd /usr/ports/sysutils/syslog-ng3 ; make -DBATCH -DWITH_SQL -DWITH_IPV6 -DWITH_PCRE
 -DWITHOUT_SPOOF install clean
root# /usr/local/etc/rc.d/mysql-server onestart
```

```
root# echo "create database kovan" | mysql -uroot
root# echo "CREATE USER 'logger'@'localhost' IDENTIFIED BY '231905'" | mysql -uroot
root# echo "GRANT ALL ON kovan.* TO 'logger'@'localhost'" | mysql -uroot
root# mysqladmin -u root password 231905
root# /usr/local/etc/rc.d/mysql-server onestop
root# cd /usr/ports/databases/p5-DBI ; make -DBATCH  install clean
root# cd /usr/ports/databases/p5-DBD-mysql ; make -DBATCH  install clean
root# cd /usr/ports/graphics/p5-GD-Graph ; make -DBATCH  install clean
```

Blackhole

```
root# pkg_add -r bash
root# mkdir -p /usr/local/kovan/etc
root# cd /usr/ports/ports-mgmt/portaudit && make install clean
root# /usr/local/sbin/portaudit -Fda
root# cd /usr/ports/textproc/p5-YAML-Tiny/; make -DBATCH -DFORCE_PKG_REGISTER install clean
root# cd /usr/ports/devel/p5-Getopt-Long; make -DBATCH  -DFORCE_PKG_REGISTER install clean
root# cd /usr/ports/net-mgmt/net-snmp && make -DBATCH -DWITH_IPV6 install clean
root# cd /usr/ports/net-mgmt/softflowd ; make -DBATCH install clean

```

5.4.4 Kovan Kaynak Kodlarının Derlenmesi

Kovan kaynak kodlarını derlemek ve kurmak için Kovan kaynak kodlarını bulunduğu dizinde aşağıdaki komutları çalıştırmanız yeterlidir.

```
root#cd kovan_host
root#make
root#make install
root#cd ..
root#cd kovan_jail
root#make
root#make install
```

Kovan Kurulumu bitmiştir, Kovan'ı ilk kez çalıştırmadan önce gerekli yapılandırma için Bölüm 6 Kovan Ayarlama kısmına geçebilirsiniz.

5.5 Önceden Hazırlanmış VM imajlarının kullanılması

Kovan'ın sadece AMD64 için hazırlanmış QEMU/KVM imajı bulunmaktadır. VM sürümüyle Kovan kurulumu yapabilmek için öncelikle http://www.ipv6.net.tr/kovan adresinden KVM/QEMU dosyaları indirilmelidir. İmajların çalışabilmesi için sistemde bir KVM/QEMU uygulaması kurulu olmalıdır. Bu bölümde FreeBSD için QEMU kurulumu anlatılmaktadır, Linux için kullandığınız dağıtımın belgelerinde KVM ile ilgili bilgi bulabilirsiniz.

5.5.1 FreeBSD/QEMU Kurulumu

Lütfen FreeBSD'de güncel QEMU kurulum için "http://wiki.freebsd.org/qemu" adresine bakınız:

Kurulum

Qemu portunu FreeBSD port sistemi yardımıyla kurunuz:

```
root#cd /usr/ports/emulators/qemu
root#make -DBATCH install clean
root#cd /usr/ports/emulators/kqemu-kmod
root#make -DBATCH install clean
```

Gerekli çekirdek modüllerini yükleyiniz:

```
root#kldload kqemu
root#kldload aio
```

/etc/rc.conf dosyasına aşağıdaki satırlar eklenerek kqemu modülünün otomatik yüklenmesi sağlanabilir:

```
kqemu_enable="YES"
```

Ayarlar ve Çalıştırma

Birer bridge ve tap arayüzü yaratılır, tap arayüzü bridge arayüzüne bağlanır (fiziksel arayüzün em0 olduğu kabul edilmiştir):

```
root#ifconfig bridge
root#ifconfig tap
root#ifconfig bridge0 addm em0 addm tap0 up
```

Kovan FreeBSD imajı 2 Gbyte bellek ve 2 çekirdek ile vnc 1'de çalıştırılır

```
qemu-system-x86_64 -m 2048 -smp 2  -net nic -net tap,name=tap0, -hda kovan.amd64.kvm.1.0.img
-boot c -vnc :1
```

Lütfen ayrıntılı bilgi için QEMU belgelerine bakınız.

Bölüm 6

Kovan'ı Ayarlama

Kovan, sanal ağ topolojileri oluşturma ve yalancı servisleri bu topoloji üstünde dağıtmak için bir ayar dosyası kullanmaktadır. Kovan ayar dosyası şunları içerir:

- Jail konumları, fiziksel arayüzler ve Kovan'ın temel konumu gibi sabit değerler.
- Sanal cihazlar ve cihazlarla ilgili varsayılan yönlendirici gibi bilgiler.
- Sanal cihazları birbirlerine bağlayan sanal bağlantılar.
- Sanal cihazlar üstünde çalıştırılacak sanal servislerin tanımları.

Ayar dosyaları, XML ve JSON'la karşılaştırıldığında daha okunabilir bir yapıda olan YAML formatında yazılmıştır. Sonraki bölümlerde Kovan'ın ayar dosyasındaki her bölüm ayrıntılı olarak anlatılmaktadır.

6.1 Sabitler

Kovan'ın çalışabilmesi için ayar dosyasına yazılması gereken iki sabit değer vardır:

- **physical_ether**: AnaSistem üstünde bulunan ve bütün sanal sistemin gerçek ağa çıkmasını sağlayacak fiziksel ağ arayüzünün ismidir. Kovan sanal ağı gerçek ağa çıkmak için fiziksel arayüze ihtiyaç duymaktadır.
- **kovan_dir**: Kovan'ın yükleneceği temel klasör konumunu gösterir. Varsayılan konum değeri "/usr/local/kovan" olarak belirlenmiştir. Kovan'ın yüklenmesi aşamasında (bölüm 5.4.4) farklı bir değer verilmediği takdirde varsayılan değer kullanılmaktadır.

Örnek ayar dosyasından alınan ilgili kısım, liste 6.1'ta görülebilmektedir. Bu örnekte, Kovan ağı AnaSistem'in re0 arayüzüne bağlanmaktadır ve bütün sistem /usr/local/kovan altına yüklenmektedir.

```
1 physical_ether: re0
2 kovan_dir: /usr/local/kovan
```

Listing 6.1: Kovan Ayarları Sabitler bölümü

6.2 Cihazlar

Bütün ağ cihazları bu bölüm altında tanımlanmaktadır. Kovan'da tanımlanabilecek 4 çeşit cihaz bulunmaktadır, cihaz türüne göre cihaz tanımındaki bazı alanlar kullanılmayabilir.

- **name**: Sanal cihazın adını belirleyen değerdir ve AnaSistem'de çalıştırılan **jls** komutunun çıktısında cihazlar bu isimleriyle tanımlanmaktadırlar. Sanal cihazlara isimleri yardımıyla erişilerek, istenen komut **jexec name komut** şeklinde çalıştırılabilir.

- **type**: Cihaz tipi, sanal cihaz yaratılırken kullanılacak jail şablonunun belirlenmesini sağlar. Kullanılabilecek cihaz türleri: yönlendirici, düğüm, monitör ve karadelik olarak belirlenmiştir. Yönlendirici tipi yönlendirici olması düşünülen cihazlarda kullanılırken, düğüm tipi istemci/sunucu cihazlara verilmelidir. Özel bir tür olan monitör, her tasarımda en fazla bir tane tanımlanabilmektedir. Monitör türüne sahip bir cihaz tanımlanması durumunda, sistem yöneticilerinin sistemi izleyebilmesi için çeşitli araçların kurulu olduğu (bölüm 3.3) bir cihaz yaratılmaktadır. Kara delik tipinden bir adet yaratılması önerilmekle birlikte istenirse birden fazla kara delik tipinde sanal cihaz yaratılabilir. Karadelik, kovan_dump uygulaması yardımıyla kendisine gelen trafikten flow oluşturabilmektedir.

- **distribute**: Bu değişken sadece **router** türündeki cihazlar için kullanılmaktadır. Değer olarak 1 verilmesi router RIPng veya RIP yönlendirme protokolü üzerinden varsayılan ağ geçidi "default route" bilgisi dağıtmaktadır.

- **default_route**: Bu değişkende sanal cihazların statik yönlendirici bilgileri yer almaktadır.Varsayılan yönlendirici değerleri yazılırken şu format kullanılmalıdır: "-inet default 10.10.10.1" veya "-inet6 default 2001:a98:13:4000::1".

Örnek ayar dosyasından alınan ilgili kısım, liste 6.2'de görülebilmektedir. Bu örnekte localhost, anarouter ve ituh adlı yönlendiriciler tanımlanmıştır. anarouter cihazı için "distribute" değeri verildiği için varsayılan yönlendirici değerlerini dağıtması istenmiştir. ituh adlı yönlendirici herhangi bir yönlendirme bilgisine sahip değildir ve bu bilgileri dinamik yönlendirme ile alacaktır. monitor_host adında bir monitör cihazı tanımlanarak sistemin izlenebilir olması sağlanmıştır. Tanımlanan iki kullanıcı bilgisayarına IPv4 yönlendirme değerleri de verilerek ikili-yığın çalışabilmeleri sağlanmıştır. Cihazlar bölümünde tanımlanan "localhost" adlı cihaz özel bir tanımdır. AnaSisteme karşılık gelen bu cihaza karşılık herhangi bir sanal cihaz yaratılmamaktadır.

```
1  devices:
2   - name: localhost
3     type: router
4     default_route:
5      - "-inet6 -net 2001:a98:13:4000::/60 2001:a98:13:4000::2"
6      - "-inet -net 10.10.10.0/24 10.10.10.2"
7   - name: anarouter
8     type: router
9     distribute: 1
10    default_route:
11     - "-inet6 default 2001:a98:13:4000::1"
12     - "-inet default 10.10.10.1"
13  - name: ituh
14    type: router
15    distribute: 0
16  - name: monitor_host
17    type: monitor
18    default_route:
19     - "-inet6 default 2001:a98:13:4000::11"
20     - "-inet default 10.10.30.1"
21    nfsen_start_port: 9000
22  - name: user1
23    type: node
24    default_route:
```

```
25      - "-inet default 10.10.60.1"
26    - name: user2
27      type: node
28      default_route:
29      - "-inet default 10.10.60.1"
```

Listing 6.2: Kovan Configuration Constants Section

6.3 Bağlantılar

Bu bölümde sanal cihazlar arasındaki sanal bağlantılar tanımlanmaktadır. Kullanıcılar, tanımlanan her sanal cihazı bağlantılar bölümünde kullanmak zorunda değillerdir fakat bağlantılar bölümünde kullanılan bir cihazın önceden tanımlanması gerekmektedir. Kovan bünyesinde kullanılabilen bridge(köprü) ve direct(doğrudan) türünde iki bağlantı bulunmaktadır. Bridge türü, ikiden fazla cihazın birbirine bağlanmasını sağlarken; direct türü iki cihazın birbirine bağlanmasını sağlamaktadır. Aşağıda listenen değerler bağlantılar bölümünde kullanılmaktadır:

- **type**: Bağlantının türünü belirleyen bu değişken "bridge" ve "direct" değerlerini alabilmektedir. Bağlantı durumunun "direct" olması durumunda "peers"(bağlantıya katılan cihazlar) listesinde 2 tane cihaz olması gerekirken, türün "bridge" olması durumunda "peers" listesinde 2'den fazla cihaz yer alabilir.
- **bandwidth**: (doğrudan bağlantılar için geçerli) Bu değişken doğrudan bir bağlantının ağ genişliğini belirlemektedir. Ağ genişliği belirlenirken kullanılabilecek olası birimler: Gigabits Per Second(Gbps), Megabits Per Second(Mbps), Kilobits Per Second(Kbps) ve Bits Per Second(bps).
- **delay**: (doğrudan bağlantılar için geçerli) Bu değişken doğrudan bir bağlantının gecikme değerini belirlemektedir. Kullanılabilecek gecikme birimleri: Mikrosaniye(us), Milisaniye(ms) ve Saniye(s) olarak belirlenmiştir.
- **downstream_ber**: (doğrudan bağlantılar için geçerli) Bu değişken doğrudan bir bağlantının downstream yönündeki bit hata değerini belirlemektedir. Bu değişkene tam sayı değerler verilebilmektedir. Örneğin, "2" verilmesi durumunda, downstream bit hata oranı 1/2 olarak belirlenmektedir.
- **upstream_ber**: (doğrudan bağlantılar için geçerli) Bu değişken doğrudan bir bağlantının upstream yönündeki bit hata değerini belirlemektedir. Bu değişkene tam sayı değerler verilebilmektedir. Örneğin, "2" verilmesi durumunda, upstream bit hata oranı 1/2 olarak belirlenmektedir.
- **peers**: Bu belirteçten sonra, bağlantıda yer alacak cihazların listelendiği bölüm başlamaktadır.
  - **name**: Cihazlar bölümünde tanımlanan ve bağlantıda yer alacak cihazın adı.
  - **ip_addresses**: (doğrudan bağlantıları için geçerli) Doğrudan bağlantılar iki sanal cihazı birbirine bağlayan sanal kablolar olarak düşünülebilirler. İki cihazı birbirine bağlayan bu kabloların iki ucun bulunmakta ve bu uçlara IPv4/IPv6 adresleri verilebilmektedir. ip_addresses değişkeni, sanal kablonun ilgili ucundaki IP adresini belirlenmesini sağlar. İkili-yığın bir bağlantı istenmediği sürece IPv4 adresi verilmesi gerekmemektedir.
  - **b_ip_address**: (köprü bağlantıları için geçerli) Bir cihaz köprüye bağlandığında, bağlantının köprü ucu (bridge_side: b_side) ve cihaz ucu(node_side: n_side) olmak üzere iki ucu bulunmaktadır. Bu değişken, bağlantının köprü ucunun IP adresini belirlemektedir.
  - **n_ip_address**: (köprü bağlantıları için geçerli) Bir cihaz köprüye bağlandığında, bağlantının köprü ucu (bridge_side: b_side) ve cihaz ucu(node_side: n_side) olmak üzere iki ucu bulunmaktadır. Bu değişken, bağlantının cihaz ucunun IP adresini belirlemektedir.

```
connections:
 - type: bridge
   peers:
    - name: localhost
    - name: anarouter
      b_ip_address:
       - "inet6 2001:a98:13:4000::1 prefixlen 126"
       - "inet 10.10.10.1/24"
      n_ip_address:
       - "inet6 2001:a98:13:4000::2 prefixlen 126"
       - "inet 10.10.10.2/24"

 - type: direct
   bandwidth: 10mbps
   delay: 1ms
   downstream_ber: 2
   peers:
    - name: anarouter
      ip_addresses:
       - "inet6 2001:a98:13:4000::5 prefixlen 126"
       - "inet 10.10.20.1/24"
    - name: ituh
      ip_addresses:
       - "inet6 2001:a98:13:4000::6 prefixlen 126"
       - "inet 10.10.20.2/24"

 - type: bridge
   peers:
    - name: comur1
      ip_addresses:
       - "inet6 2001:a98:13:4004::1 prefixlen 64"
       - "inet 10.10.70.1/24"
      prefix: "2001:a98:13:4004::/64"
    - name: yavuz
      ip_addresses:
       - "inet6 2001:a98:13:4004::200 prefixlen 64"
       - "inet 10.10.70.2/24"
    - name: emre
      ip_addresses:
       - auto6
       - "inet 10.10.70.3/24"
```

Listing 6.3: Kovan Ayarları Bağlantılar bölümü

6.4 Servisler

Bu bölümde sanal ağ oluşturulduktan sonra çalıştırılacak sanal servislerin tanımları yer almaktadır. Servis tanımları basit bir yapıya sahiptir.

- **name**: Servis ismi değişkeni bazı servisler için önem taşımaktadır fakat her servis için farklı bir isim verilmesi önerilmektedir. Yönlendiriciler üstünde softflowd servisi kullanılmak isteniyorsa servis ismi

değişkenine tam olarak "softflowd" yazılmalıdır.

- **node**: Servisin çalıştırılacağı cihaz(ları) belirleyen bir değişkendir. Bu parametre "monitor_host" gibi bir cihaz ismi olabileceği gibi all-nodes ve all-routers gibi bir cihaz grubu ismi de olabilir.

- **command**: Sanal servisin çalıştırılmasında kullanılacak komutu belirler. Bu alanda IP6 veya IP4 gibi özel kelimeler kullanılabilir. IP6 kelimesi kullanılması durumunda, ayar dosyası işlenirken ilgili cihazın IPv6 adresi alınarak IP6 kelimesi yerine kullanılır. Bu sayede cihaz IP'si bilinmeden IP'ye bağlı servisler tanımlanabilmektedir.

Örnek ayar dosyasından alınan ilgili kısım, 6.4'te görülebilmektedir. Bu örnekte, monitor_host ve yavuz adlı cihazlarda syslogd servisinin çalıştırılması için servis tanımları yapılmıştır. Softflowd servisinin ise tüm yönlendiricilerde çalışması sağlanmıştır. Son olarak, kovan_dns servisinin yavuz cihazında, o cihazın IP6 adresi ile çalışması sağlanmıştır.

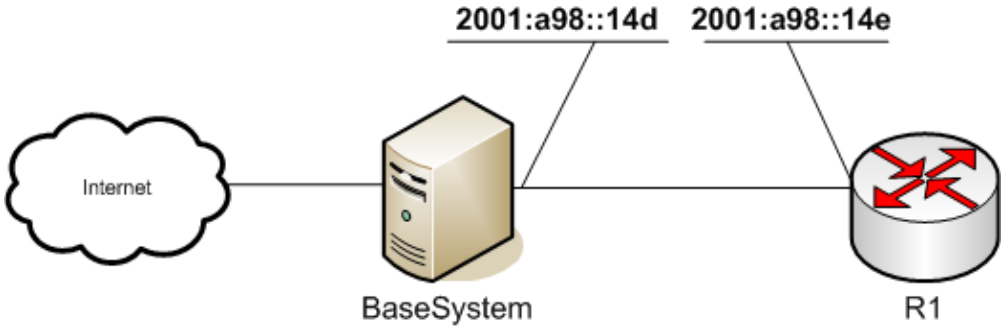
```
1  services:
2   - name: syslogd
3     node: monitor_host
4     command: syslogd -P monitor.pid
5   - name: syslogd
6     node: yavuz
7     command: syslogd -P yavuz.pid
8   - name: softflowd
9     node: all-routers
10    command: /usr/local/sbin/softflowd
11  - name: kovan_dns
12    node: yavuz
13    command: /usr/local/kovan/bin/kovan\_dns -h {IP6} -p 53 -c /usr/local/kovan
          /etc/dns/named.conf  -d /usr/local/kovan/etc/dns.pid
```

Listing 6.4: Kovan Configuration Services section

6.5 Örnek Ağ Topolojileri

Bu bölümde kovan'ın ayarlama sürecinin daha iyi anlaşılabilmesi için üç örnek gösterilmektedir.

Şekil 6.1'te küçük bir ağ topolojisi gösterilmektedir. Bu topolojide sadece bir sanal cihaz üretilerek Ana-Sisteme bağlanmıştır.



Şekil 6.1: Küçük ağ topolojisi

Liste 6.5, şekil 6.1'deki ağın ayarlarını göstermektedir. jail_roots konumlarının verildiği bölümde sadece "router" tipi için jail konumun tanımlanması gerekmektedir çünkü sadece "router" tipinde bir cihaz yaratılmıştır. Diğer tipler node, monitor ve blackhole'un tanımlanmasına gerek yoktur. Cihazlar bölümünde iki cihaz tanımlanmıştır. İlk cihazın ismi özel bir isim olan "localhost" olarak belirtilmiştir. Kovan'da "localhost" ismine sahip cihaz sanal bir cihaz olarak yaratılmamaktadır, bu cihaz AnaSistem cihazını belirtmektedir. localhost cihazının tanımında, sanal ağ için gereken statik yönlendirme bilgisi default_route altında bulunmaktadır. Bağlantılar bölümünde, sadece sanal cihazı AnaSisteme bağlayan bağlantı tanımlanmıştır, bu bağlatıda köprü türü kullanılmaktadır.

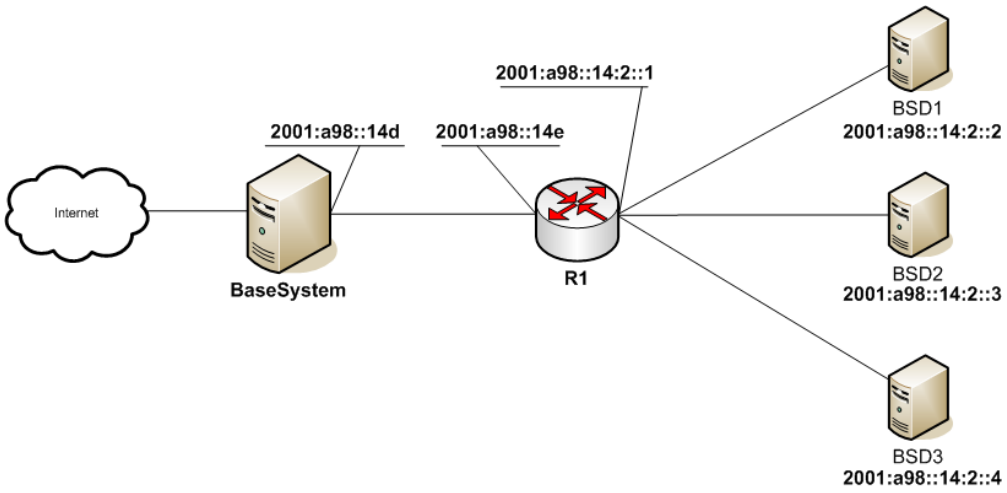
```
---
physical_ether: re0
kovan_dir: /usr/local/kovan

jail_roots:
 router: /usr/local/kovan/router

devices:
 - name: localhost
   type: router
   default_route:
    - "-inet6 -net 2001:a98:14::/48 2001:a98:14::E"

 - name: R1
   type: router
   distribute: 1
   default_route:
    - "-inet6 default 2001:a98:14::D"

connections:

 - type: bridge
   peers:
    - name: localhost
    - name: R1
      b_ip_address:
       - "inet6 2001:a98:14::D prefixlen 126"
      n_ip_address:
       - "inet6 2001:a98:14::E prefixlen 126"
```

Listing 6.5: Küçük Ağ topolojisi ayar dosyası

Şekil 6.2'te orta ölçekli bir sanal ağ örneği gösterilmektedir. Orta ölçekli ağda, küçük ölçekli ağa ek olarak, üç sanal cihaz yaratılarak R1 yönlediricisine bağlanmıştır.

Liste 6.6, şekil 6.2'deki orta ölçekli ağı oluşturan ayarları göstermektedir. Küçük ölçekli ağda bulunan cihazlara ek olarak üç sanal host (BSD1, BSD2, BSD3) "devices" bölümünde tanımlanmıştır. Tanımlanan üç cihaz R1 yönlendiricisine bir köprü cihazı aracılığıyla bağlanmıştır.

```
---
physical_ether: re0
kovan_dir: /usr/local/kovan

jail_roots:
 router: /usr/local/kovan/router
```



Şekil 6.2: Orta ölçekli ağ topolojisi

```
 node: /usr/local/kovan/node
 monitor: /usr/local/kovan/monitor
 blackhole: /usr/local/kovan/blackhole

devices:
 - name: localhost
   type: router
   default_route:
    - "-inet6 -net 2001:a98:14::/48 2001:a98:14::E"

 - name: R1
   type: router
   distribute: 1
   default_route:
    - "-inet6 default 2001:a98:14::D"

 - name: BSD1
   type: node

 - name: BSD2
   type: node

 - name: BSD3
   type: node

connections:

 - type: bridge
   peers:
    - name: localhost
    - name: R1
      b_ip_address:
```

```
      - "inet6 2001:a98:14::D prefixlen 126"
     n_ip_address:
      - "inet6 2001:a98:14::E prefixlen 126"

- type: bridge
  peers:
   - name: R1
     ip_addresses:
      - "inet6 2001:a98:14:2::1 prefixlen 64"
     prefix: "2001:a98:14:2::/64"

   - name: BSD1
     ip_addresses:
      - "inet6 2001:a98:14:2::3 prefixlen 64"
   - name: BSD2
     ip_addresses:
      - "inet6 2001:a98:14:2::4 prefixlen 64"
   - name: BSD3
     ip_addresses:
      - "inet6 2001:a98:14:2::5 prefixlen 64"
```

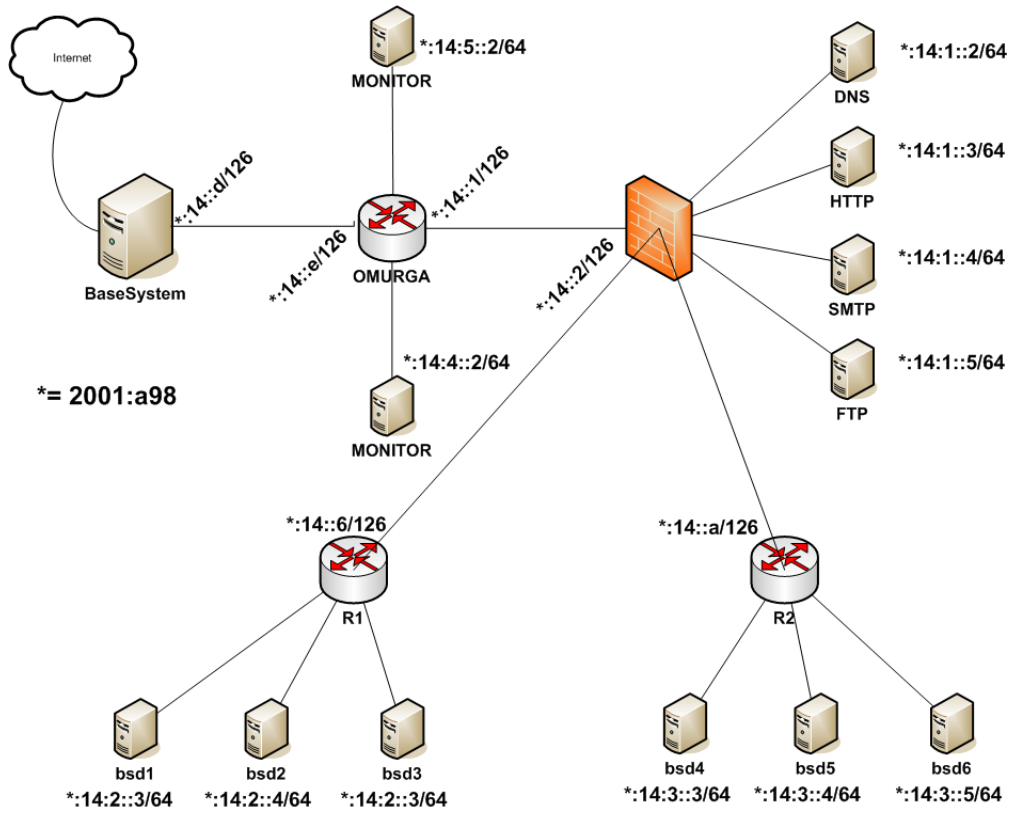
Listing 6.6: Orta ölçekli topoloji ayar dosyası

Küçük ve orta ölçekli topolojilerde cihazlar üstünde herhangi bir servis tanımlanmamıştır. Topolojilerin çalışıp çalışmadığı ping6/traceroute6 gibi araçlarla test edebilebilir.

Son örnek olan büyük ölçekli topolojide (Şekil 6.3), küçük ve orta ölçekli topolojilerin aksine birçok servis tanımlanmıştır.

Büyük ölçekli topolojide üç yönlendirici, bir güvenlik duvarı (bu cihaz da router türünde tanımlanmıştır), dört sunucu ve altı istemci tanımlanmıştır. Küçük ve orta ölçekli topolojilerden farklı olarak büyük ölçekli topolojide router<->router bağlantılarında "direct" türündeki doğrudan bağlantılar da kullanılmaktadır. Ayar dosyasının son bölümünde ise sanal cihazlar üstünde çalıştırılacak servisler tanımlanmaktadır.

```
---
physical_ether: re0
kovan_dir: /usr/local/kovan

jail_roots:
 router: /usr/local/kovan/router
 node: /usr/local/kovan/node
 monitor: /usr/local/kovan/monitor
 blackhole: /usr/local/kovan/blackhole

devices:
 - name: localhost
   type: router
   default_route:
    - "-inet6 -net 2001:a98:14::/48 2001:a98:14::E"

 - name: OMURGA
   type: router
   distribute: 1
   default_route:
    - "-inet6 default 2001:a98:14::D"
```



Şekil 6.3: Büyük ölçekli topoloji

```
- name: FW
  type: router
  distribute: 0

- name: R1
  type: router
  distribute: 0

- name: R2
  type: router
  distribute: 0

- name: MONITOR
  type: monitor
  distribute: 0
  default_route:
   - "-inet6 default 2001:a98:14:5::1"
  nfsen_start_port: 9000

- name: BLACKHOLE
  type: blackhole
```

```
    default_route:
     - "-inet6 -net 2001:a98:14::/48  2001:a98:14:4::1"

 - name: BSD1
   type: node
 - name: BSD2
   type: node
 - name: BSD3
   type: node
 - name: BSD4
   type: node
 - name: BSD5
   type: node
 - name: BSD6
   type: node
 - name: DNS
   type: node
 - name: HTTP
   type: node
 - name: FTP
   type: node
 - name: SMTP
   type: node

connections:

 - type: bridge
   peers:
    - name: localhost
    - name: OMURGA
      b_ip_address:
       - "inet6 2001:a98:14::D prefixlen 126"
      n_ip_address:
       - "inet6 2001:a98:14::E prefixlen 126"

 - type: bridge
   peers:
    - name: localhost
    - name: BLACKHOLE
      n_ip_address:
       - "inet6 2001:a98::173 prefixlen 125"

 - type: direct
   peers:
    - name: OMURGA
      ip_addresses:
       - "inet6 2001:a98:14:5::1 prefixlen 64"
    - name: MONITOR
      ip_addresses:
       - "inet6 2001:a98:14:5::2 prefixlen 64"
```

```
- type: direct
  peers:
   - name: OMURGA
     ip_addresses:
      - "inet6 2001:a98:14:4::1 prefixlen 64"
   - name: BLACKHOLE
     ip_addresses:
      - "inet6 2001:a98:14:4::2 prefixlen 64"

- type: direct
  peers:
   - name: OMURGA
     ip_addresses:
      - "inet6 2001:a98:14::1 prefixlen 126"
   - name: FW
     ip_addresses:
      - "inet6 2001:a98:14::2 prefixlen 126"

- type: direct
  peers:
   - name: FW
     ip_addresses:
      - "inet6 2001:a98:14::5 prefixlen 126"
   - name: R1
     ip_addresses:
      - "inet6 2001:a98:14::6 prefixlen 126"

- type: direct
  peers:
   - name: FW
     ip_addresses:
      - "inet6 2001:a98:14::9 prefixlen 126"
   - name: R2
     ip_addresses:
      - "inet6 2001:a98:14::a prefixlen 126"

- type: bridge
  peers:
   - name: R1
     ip_addresses:
      - "inet6 2001:a98:14:2::1 prefixlen 64"
     prefix: "2001:a98:14:2::/64"

   - name: BSD1
     ip_addresses:
      - "inet6 2001:a98:14:2::3 prefixlen 64"
   - name: BSD2
     ip_addresses:
      - "inet6 2001:a98:14:2::4 prefixlen 64"
   - name: BSD3
     ip_addresses:
```

```
      - "inet6 2001:a98:14:2::5 prefixlen 64"

  - type: bridge
    peers:
     - name: R2
       ip_addresses:
        - "inet6 2001:a98:14:3::1 prefixlen 64"
       prefix: "2001:a98:14:3::/64"

     - name: BSD4
       ip_addresses:
        - "inet6 2001:a98:14:3::3 prefixlen 64"
     - name: BSD5
       ip_addresses:
        - "inet6 2001:a98:14:3::4 prefixlen 64"
     - name: BSD6
       ip_addresses:
        - "inet6 2001:a98:14:3::5 prefixlen 64"

  - type: bridge
    peers:
     - name: FW
       ip_addresses:
        - "inet6 2001:a98:14:1::1 prefixlen 64"
       prefix: "2001:a98:14:1::/64"

     - name: DNS
       ip_addresses:
        - "inet6 2001:a98:14:1::2 prefixlen 64"
     - name: HTTP
       ip_addresses:
        - "inet6 2001:a98:14:1::3 prefixlen 64"
     - name: SMTP
       ip_addresses:
        - "inet6 2001:a98:14:1::4 prefixlen 64"
     - name: FTP
       ip_addresses:
        - "inet6 2001:a98:14:1::5 prefixlen 64"

services:
  - name: softflowd
    node: all-routers
    command: /usr/local/sbin/softflowd
  - name: cron
    node: MONITOR
    command: /etc/rc.d/cron onestart
  - name: snmpd
    node: all-routers
    command: /usr/local/sbin/snmpd -c /etc/snmpd.conf udp6:161
  - name: snmpd
    node: BLACKHOLE
```

```
  command: /usr/local/sbin/snmpd -c /etc/snmpd.conf udp6:161
- name: syslogd
  node: HTTP
  command: syslogd -P syslog.pid
- name: logger
  node: HTTP
  command: tail -f /var/log/messages | grep kovan | logger -h 2001:a98
      :14:5::2 -P 510 &
- name: kovan_dns_6
  node: DNS
  command: /usr/local/kovan/bin/kovan\_dns -h {IP6} -p 53 -c /usr/local/kovan
      /etc/dns/named.conf -d /usr/local/kovan/etc/dns6.pid
- name: kovan_mail_6
  node: SMTP
  command: /usr/local/kovan/bin/kovan\_smtp -h {IP6} -p 25 -c /usr/local/
      kovan/etc/smtp/ipv6go\_mail.conf -d /usr/local/kovan/etc/smtp6.pid
- name: kovan_http_6
  node: HTTP
  command: /usr/local/kovan/bin/kovan\_http -h {IP6} -p 80 -w /usr/local/
      kovan/etc/www/ -d /usr/local/kovan/etc/http6.pid
- name: log_sql
  node: MONITOR
  command: /usr/local/etc/rc.d/mysql-server onestart
- name: syslog
  node: MONITOR
  command: /usr/local/sbin/syslog-ng
- name: flow_logger
  node: BLACKHOLE
  command: /usr/local/kovan/bin/kovan\_dump -i eth0 -f 'ip6␣and␣host␣not␣
      2001:a98::173' | logger -h 2001:a98:14:5::2 -P 510 &
```

Listing 6.7: Büyük ölçekli topoloji ayar dosyası

Bölüm 7

Kovan'ı Çalıştırma

Kovan bölüm 5'te anlatıldığı şekilde yüklendikten sonra, çalıştırılarak sanal balküpü ağı kurulabilir. Bu bölümde Kovan uygulamasının çalıştırılabileceği olası senaryolarda anlatılmaktadır. Bölümdeki senaryolar, Kovan'ın varsayılan konumu olan "/usr/local/kovan"a yüklendiği varsayılmaktadır.

Aşağıda kovan uygulamasının kullanımı gösterilmektedir. "kovan" sanal balküpü ağını oluşturmada ve silmede kullanılan ana uygulamadır.

```
root# ./kovan
Configuration file is not given
usage: kovan --conf conf_file --list
       kovan --conf conf_file --execute
       kovan --conf conf_file --remove --list {destroys system}
       kovan --conf conf_file --remove --execute {destroys system}
```

7.1 Kovan Ayarlarını Kontrol Etme

Bölüm 6'da Kovan ayarları ayrıntılı olarak anlatılmaktadır. Örnek bir kovan ayar dosyası /usr/local/kovan/etc/ altında bulunmaktadır. Yeni bir ayar dosyası ise, okuma izni olan herhangi bir konuma konulabilir.

Kovan, ayar dosyasının kontrolünü dosyayı işlerken yapmaktadır. Fakat, karmaşık ağ yapılarının oluşturulması için gereken ayarların doğru olup olmadığının kararının verilmesi her zaman otomatik olarak olanaklı olamamaktadır, bazı durumlarda insan müdahalesi gerekmektedir. Bu yüzden, Kovan sistemi oluştururken çalıştıracağı bütün komutları kullanıcıya gösterdiği bir listeleme modu içermektedir.

```
1  root# ./kovan -c ../etc/kovanTopology.conf -l
2  kldload ng_ether
3  kldload ng_bridge
4  kldload ng_eiface
5  kldload ng_iface
6  mount_nullfs -o ro /usr/jails/basejail  /usr/local/kovan/router/basejail/
7  ...
8  jail -c vnet name=onur host.hostname=onur path=/usr/local/kovan/node persist
9  ...
10 # It is normal to see "jexec: jail 'xxx' not found" errors in this execution
       mode!
11 jexec: jail "yavuz" not found
12 ...
13 jexec onur /usr/local/kovan/bin/kovan_http -h  -p 80 -w /usr/local/kovan/www/
       -d /usr/local/kovan/etc/http.pid
```

Listing 7.1: Kovan Ayarlarının Listelenmesi

Dikkat edilmesi gereken önemli bir nokta, Kovan'ın listeleme modunda eğer önceki komutun çalışmasıyla elde edilecebilecek bir değer gerekiyorsa bunun gösterilemediğidir. Örneğin, (liste 7.1) servisler bölümünde onur cihazının üstünde bir http servisi tanımlanmıştır. Fakat onur cihazı listeleme modunda henüz yaratılmış olmadığı için HTTP servisinin çalıştırılacağı IP adresine listeleme modunda erişilememektedir. 9. satırdaki uyarı mesajı bu yüzden verilmektedir.

7.2 Kovan'ı Çalıştırma ve Durdurma

Genel uygulamaların aksine Kovan, tek bir çalıştırılabilir dosyadan oluşan tekil bir uygulama değildir. Birçok sistemin bir araya gelerek oluşturduğu karmaşık bir sistemdir. Kovan sanal balküpü ağı için gerekli olan nesneleri yaratarak bunları birbirine bağlar. Liste 7.2'de örnek bir çalışma görebilirsiniz. Bu örnekte Kovan uygulaması, kovanTopology.conf ayar dosyasıyla çalıştırılmıştır.

```
1  root# ./kovan -c ../etc/kovanTopology.conf -e
2  command: kldload ng_ether
3  command finished
4  command: kldload ng_bridge
5  command finished
6  command: kldload ng_eiface
7  command finished
8  command: kldload ng_iface
9  command finished
10 command: mount_nullfs -o ro /usr/jails/basejail  /usr/local/kovan/router/
       basejail/
11 command finished
12 ...
13 command: jail -c vnet name=anarouter host.hostname=anarouter path=/usr/local/
       kovan/router allow.raw_sockets  persist
14 ...
15 command: jexec comuh /usr/local/sbin/snmpd -c /etc/snmpd.conf udp6:161
16 command finished
17 ...
18 command: jexec onur /usr/local/kovan/bin/kovan_http -h 2001:a98:13:4004::201 -
       p 80 -w /usr/local/kovan/www/ -d /usr/local/kovan/etc/http.pid
19 command finished
```

Listing 7.2: Kovan'ı örnek bir ayarla başlatmak

Kovan çalıştırılırken oluşturulan nesneler ve sistem kaynakları kovan'ın durdurulması sırasında silinmektedir. Başlangıç işlemi sırasında, yaratılan tüm nesneler /usr/local/kovan/topology.tmp adlı geçici bir dosyaya yazılmaktadır. Durdurma işlemi sırasında bu nesneler sırayla silinmektedirler (liste 7.3).

```
1  root# ./kovan -c ../etc/kovanTopology.conf -e -r
2  command: route delete -inet6 -net 2001:a98:13:4000::/60 2001:a98:13:4000::2
3  delete net 2001:a98:13:4000::/60: gateway 2001:a98:13:4000::2
4  command finished
5  command: route delete -inet -net 10.10.10.0/24 10.10.10.2
6  delete net 10.10.10.0: gateway 10.10.10.2
7  command finished
8  command: jail -r anarouter
```

```
9  command finished
10 ...
11 ...
```

Listing 7.3: Çalışan Kovan'ı durdurmak

Bir bilgisayarda iki Kovan sistemi çalıştırılması olanaklı değildir. Sistemde çalışan bir sistem olup olmadığı /usr/local/kovan/topology.tmp dosyasının varlığına göre karar verilir. Eğer bu dosya mevcutsa, sistemde bir kovan uygulaması çalışmaktadır. İkinci kovan uygulamasının çalışması durumunda sistem hata vermektedir (liste 7.4).

```
1 root# ./kovan -c ../etc/kovanTopology.conf -e
2 ! /usr/local/kovan/topology.tmp exists
3 Destroy previous configuration before establishing new one
4 usage:  kovan --conf conf_file --list
5         kovan --conf conf_file --execute
6         kovan --conf conf_file --remove --list {destroys system}
7         kovan --conf conf_file --remove --execute {destroys system}
```

Listing 7.4: Kovan'ı Hatalı Başlatmak

Bölüm 8

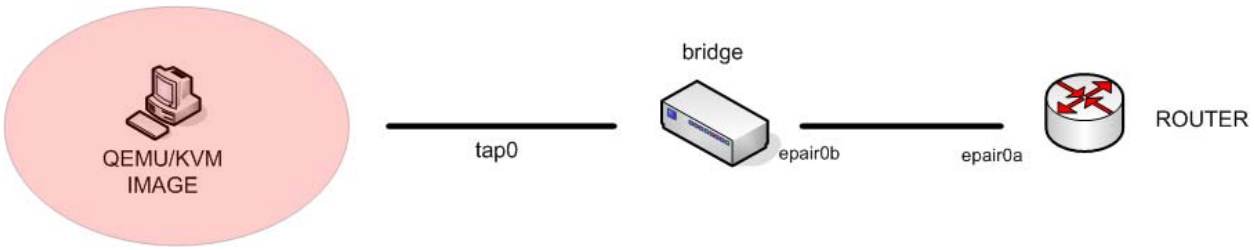
İleri Özellikler

8.1 QEMU/KVM imajları (Windows, Solaris, Centos...) Bağlama

Kovan'a QEMU/KVM imajları bağlanarak, balküpü ağının etkileşim seviyesi arttırılabilir. Bir balküpüne saldırmadan önce saldırganlar, aktif/pasif imza çıkarma yöntemleriyle hedef işletim sistemini ve servislerini tanımlamaya çalışırlar. Saldırının bu aşaması keşif olarak adlandırılır. QEMU/KVM imajlarının kullanılmasıyla, saldırganın gerçek işletim sistemi ve sanalları ayırması zorlaşacaktır Kovan'ı daha gerçekci göstermek için, farklı işletim sistemlerinin QEMU imajları sisteme eklenebilir. Örneğin, Windows işletim sistemine sahip makinelerin çalıştığı bir laboratuvarı simüle etmek amacı ile aşağıdaki yöntemlerden biri kullanılabilir:

1. Kovan'daki cihazları sanal windows servisleriyle çalıştırmak.
2. Gerçek Windows QEMU imajlarını Kovan'ın içinde kullanmak.

İkinci yöntemin kullanılması halinde hedef sistemin gerçek olup olmadığının anlaşılması zorlaşmaktadır. Bir QEMU imajını Kovan ağına bağlayabilmek için bir epair çiftine, bir tap arayüzüne ve bir köprü nesnesine ihtiyaç duyulmaktadır. QEMU imajını Kovan'a bağlamak için, tap ve epair arayüzlerinin köprü nesnesine Şekil 8.1'deki gibi bağlanması gerekmektedir. Köprü iki arayüzü bağlamak için kullanılmaktadır.



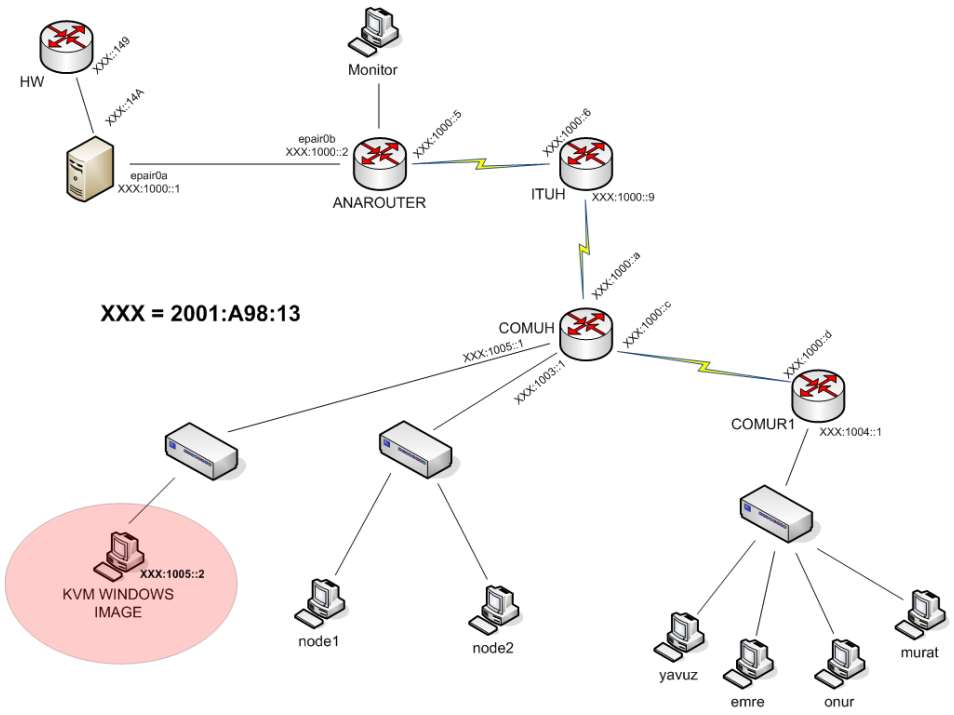
Şekil 8.1: QEMU imajını Kovan yönlendiricisine bağlamak

Bir QEMU imajını Kovan'a bağlamak için izlenmesi gereken 10 basamak bulunmaktadır:

1. FreeBSD Ana Sistemi, QEMU imajlarını çalıştırabilecek şekilde bölüm 5.5.1'de anlatıldığı şekilde ayarlanmalıdır.
2. AnaSistem'e QEMU yüklenmelidir.
3. Sanal bir ağ yapısı tasarlanmalıdır.
4. Sanal ağ'da atanacak IPv4/IPv6 adresleri belirlenmelidir.

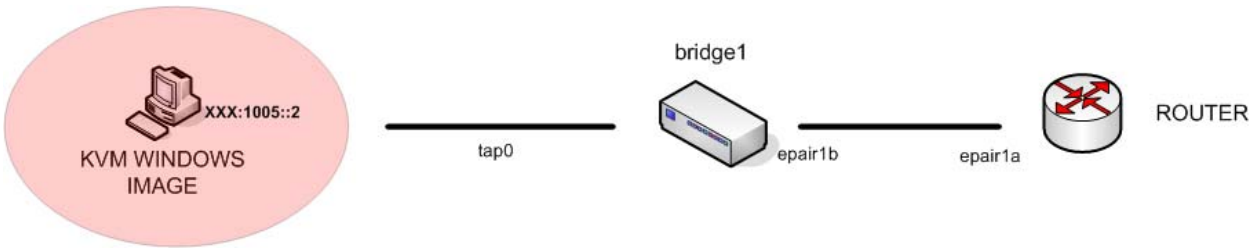
5. QEMU imajının bağlanacağı yönlendirici ayarlanmalıdır.
6. Kovan için bir ayar dosyası hazırlanmalıdır.
7. QEMU imajının ağ topolojisinde nereye bağlanacağına karar verilmelidir.
8. Kovan hazırlanan ayar dosyasıyla çalıştırılmalıdır.
9. Qemu imajı ilgili yönlendiriciye bağlanmalıdır.
10. QEMU imajı çalıştırılmalıdır.

Kavramın daha iyi anlaşılabilmesi için bir örnek verilmiştir. example1.conf adlı ayar dosyası Şekil 8.1'deki topolojiyi oluşturmaktadır.



Şekil 8.2: QEMU VM Örnek topology (example1.conf)

1. Bölüm 5.5.1'i takip ediniz.
2. QEMU belgeleri takip edilerek bir imaj hazırlayınız. Örnek boyunca win7.img adlı bir Microsoft Windows 7 imajı kullanıldığı varsayılmaktadır.
3. 3-7 basamakları için examples klasörü altındaki example1.conf adlı dosyayı kullanınız.
4. Kovan'ı çalıştırınız.
5. QEMU imajını, COMUH adlı yönlendiriciye Şekil 8.3 'teki gibi bağlayınız.
   (a) bridge, tap ve epair arayüzlerini oluştur



Şekil 8.3: COMUH yönlendiricisine QEMU Windows 7 imajı bağlanması

```
root# ifconfig bridge create
brigde1
root# ifconfig tap create
tap0
root# ifconfig epair create
epair1a
```

(b) epair1b adlı ucu köprü nesnesine(bridge1) ekle

```
root# ifconfig bridge1 addm epair1b
```

(c) epair1a adlı ucu Kovan COMUH yönlendiricisine ata

```
root# ifconfig epair1a vnet comuh
```

(d) tap arayüzünü köprü nesnesine ekle

```
root# ifconfig bridge1 addm tap0
```

(e) epair1a arayüzüne IPv4&IPv6 adreslerini ver.

```
root# ifconfig bridge1 up
root# ifconfig epair1b up
root# jexec comuh ifconfig epair1a up
root# jexec comuh ifconfig epair1a 10.10.80.1 netmask 255.255.255.0 up
root# jexec comuh ifconfig epair1a inet6 2001:a98:13:1005::1 prefixlen 64 up
root# ifconfig tap0 up
```

Yönlendirme otomatik olarak ayarlanacaktır.

6. Kovan FreeBSD imajını 1 Gbyte bellek ve 1 çekirdek kullanacak şekilde başlat.

```
qemu-system-x86_64 -m 1024 -smp 1  -net nic -net tap,name=tap0,
    -hda win7.img -boot c -vnc :1
```

7. Ekran çıktısını VNC1'e vermiş olan imaja vncviwer programları kullanarak bağlanabilirsiniz.

Bölüm 9

Güvenlik

9.1 Kovan Parolaları

Kovan Kurulumu sırasında birçok parola ile gelmektedir. Güvenlik açısından bu paroların Kurulumdan Önce değiştirilmesi gerekmektedir.

- **MYSQL Paroları** 231905

  Monitor jaili altında kurulan Mysql şifrelerini değiştimek için Kovan kaynak kodlarının olduğu dizinlerde bulunan 3 dosyada değişiklik yapmak gerekmektedir

  1. install.sh altındaki

     ```
     CREATE USER logger@localhost IDENTIFIED BY 231905" | mysql -uroot
     GRANT ALL ON kovan.* TO logger@localhost" | mysql -uroot
     mysqladmin -u root password 231905
     ```

     satırlarındaki 231905 parolası yerine yeni parola yazılarak.
  2. kovan_host/etc/syslog-ng.conf dosyasında

     ```
     host("localhost") username("logger") password("231905")
     ```

     satırlarındaki 231905 parolası yerine yeni parola yazılarak.
  3. kovan_jail/kovan_graph/kovan_graph dosyasında

     ```
     my \$sql\_password = "231905";
     ```

     satırındaki 231905 parolası yerine yeni parola yazılarak.

- **Quagga Yönlendirici paroları:** 1504 kovan_host/bin/kovanQuaggaConf programında **iki farklı satırda** geçen

  ```
  password 1504
  ```

  1504 parolasını istediğiniz ile değiştirin.

- **Kovan KVM/QEMU imajı root parolası:** kovan

- **SNMP Şifresi:** pide

  Kaynak kodlarında kovan_host/etc/snmpd.conf dosyasındaki

  ```
  rocomminity6 pide
  ```

  satırındaki pide'yi yeni şifre ile değiştirin.

9.2 Kovan'a Uzaktan Erişimin Kısıtlanması

Kovan kurulumunda herhangi bir güvenlik duvarı kuralı aktive edilmemektedir. Kovan üzerinden verilen servislere erişimin kısıtlanması için Kovan'ın kurulduğu sunucuya veya Kovan QEMU/KVM imajı olarak çalışıyor ise ana makine üzerine güvenlik duvarı kuralları yazılabilir. Aşağıda 10.20.30.40 ve 2001:a98:dede:efe:fafa:1 adresine sahip Linux Ana makineye erişim kısıtlaması için kullanılabilecek örnek netfilter kuralları verilmiştir. Betikte ana makine üzerindei SSH ve VNC portlarına 192.168.3.0 ve 2001:a98:dede:baba:: ağından bağlantıya izin verilmiş ve diğer ağlardan erişim kısıtlanmıştır. Benzer şekilde, KOVAN_NET ve KOVAN_NET6 ağlarındaki QUAGGA portlarına da erişim kısıtlanmıştır.

```
#!/bin/bash

IPTABLES="/sbin/iptables"
IP6TABLES="/sbin/ip6tables"

#### Bridge Interface
BRIDGE=br0
### KVM Linux Server Interfaces
MNG_IP="10.20.30.40"
MNG_IP6="2001:a98:dede:efe:fafa:1"

### System administrators net
MNG_NET="192.168.3.0/24"
MNG_NET6="2001:a98:dede:baba::/64"
KOVAN_NET="193.168.0.0"
KOVAN_NET6="2001:a98::"
### VNC Port Range
VNC_RANGE="5900:5920"
QUAGGA_RANGE="2601:2603"

#
# Flush (-F) all specific rules
#
$IPTABLES -F INPUT
$IPTABLES -F FORWARD
$IPTABLES -F OUTPUT
$IPTABLES -F -t nat

#### Flush Rules For IPv6
$IP6TABLES -F INPUT
$IP6TABLES -F FORWARD
$IP6TABLES -F OUTPUT

### Accept Everyting
$IPTABLES -P INPUT ACCEPT
$IPTABLES -P FORWARD ACCEPT
$IPTABLES -P OUTPUT ACCEPT

### Accept Everyting For IPv6

$IP6TABLES -P INPUT ACCEPT
$IP6TABLES -P FORWARD ACCEPT
```

```
$IP6TABLES -P OUTPUT ACCEPT


### Allow management IP address SSH and VNC connection
$IPTABLES -A INPUT -p tcp -i $BRIDGE -s $MNG_NET -d $MNG_IP -m multiport --
    dports 22,$VNC_RANGE -m state --state NEW -j ACCEPT
$IP6TABLES -A INPUT -p tcp -i $BRIDGE -s $MNG_NET6 -d $MNG_IP6 -m multiport
    --dports 22,$VNC_RANGE -m state --state NEW -j ACCEPT
$IPTABLES -A INPUT -p udp -i $BRIDGE -s $MNG_NET -d $MNG_IP -m multiport --
    dports 22,$VNC_RANGE -m state --state NEW -j ACCEPT
$IP6TABLES -A INPUT -p udp -i $BRIDGE -s $MNG_NET6 -d $MNG_IP6 -m multiport
    --dports 22,$VNC_RANGE -m state --state NEW -j ACCEPT

### Allow management IP address QUAGGA connection
$IPTABLES -A INPUT -p tcp -i $BRIDGE -s $MNG_NET -d $KOVAN_NET -m multiport
    --dports $QUAGGA_RANGE -m state --state NEW -j ACCEPT
$IP6TABLES -A INPUT -p tcp -i $BRIDGE -s $MNG_NET6 -d $KOVAN_NET6 -m
    multiport --dports $QUAGGA_RANGE -m state --state NEW -j ACCEPT

### Deny all other connection to SSH and VNC
$IPTABLES -A INPUT -p tcp -i $BRIDGE -d $MNG_IP -m multiport --dports 22,
    $VNC_RANGE -m state --state NEW -j DROP
$IP6TABLES -A INPUT -p tcp -i $BRIDGE -d $MNG_IP6 -m multiport --dports 22,
    $VNC_RANGE -m state --state NEW -j DROP
$IPTABLES -A INPUT -p udp -i $BRIDGE -d $MNG_IP -m multiport --dports 22,
    $VNC_RANGE -m state --state NEW -j DROP
$IP6TABLES -A INPUT -p udp -i $BRIDGE -d $MNG_IP6 -m multiport --dports 22,
    $VNC_RANGE -m state --state NEW -j DROP

### Deny connection to KOVAN quagga Ports
$IPTABLES -A INPUT -p tcp -i $BRIDGE -d $KOVAN_NET -m multiport --dports
    $QUAGGA_RANGE -m state --state NEW -j DROP
$IP6TABLES -A INPUT -p tcp -i $BRIDGE -d $KOVAN_NET6 -m multiport --dports 22,
    $QUAGGA_RANGE -m state --state NEW -j DROP
```

Listing 9.1: AteşDuvarı ayarları

Bölüm 10

Başka Yazılımlarla Kullanım

Kovan'da servis olarak Kovan ile birlikte gelen DNS, FTP, WEB ve SMTP sunucuları kullanılabileceği gibi başka programlar da kullanılabilir.Programların kullanımı için tek yapılması gereken programın router, monitor, blackhole, node jaillerinden birine kurulduktan sonra kovan ana yapılandırma dosyasında bir servis olarak başlatılması için gerekli ayarların yapılmasıdır. Servislerin nasıl tanımlanacağı ile ilgili bilgi Bölüm 6.4'te verilmiştir. Node jail'i altında apache sunucu kurulumu aşağıda verilmiştir.

10.1 Apache

Node tipi jail altında apache kurulumu için pkg_add komutu kullanılabilir.

```
pkg_add -r apache22
```

Apache kurulduktan ve gerekli yapılandırma tamamlandıktan sonra node1 ismindeki sanal cihazda apache başlatılması için kovan ana ayar dosyasında services tanımı altına

```
 - name : gercek_apache
   node : node1
   command: /usr/local/etc/rc.d/apache22 onestart
```

satırlarının yazılması yeterlidir.

Bölüm 11

Gelecekteki Çalışmalar

Kovan'ın bundan sonraki geliştirme aşamalarında Kovan saldırılara karşı daha akıllı hale getirilebilir. Kovan'da kullanılan karadeliğe yapılan saldırılardaki hedef adreslerinin belirli düzen içinde olduğu görülmüştür. Kovan bu düzenleri tespit ederek sırada taranacak IPv6 adresinde otomatik olarak sanal cihaz oluşturacak şekilde geliştirilebilir. Benzer şekilde normalde servis çalışmayan bir IPv6 adresine saldırı olması durumunda bu adreste bir servis başlatılabilir.

Kovan durum denetimli yapısından dolayı iki farklı yapılandırma arasında geçiş yapabilir. Bu özellik saldırıların sayısı ve kullandığı bant genişliği arttıkça saldırı tipinin yakalamaya yönelik olarak farklı yapılandırmaya yönelik sistemler tasarlanmak için kullanılabilir. Örneğin tek bir IPv6 adresine çok sayıda saldırı olduğu zaman saldırı karşısında sistem kaynaklarını verimli kullanmak için az sayıda cihaz içeren bir yapılandırmaya geçilirken, port taraması tarzında saldırıların olduğu durumlarda cihaz sayısını çok fazla olduğu bir yapılandırmaya geçecek akıllı sistemler tasarlanabilir.

Bölüm 12

Servisler

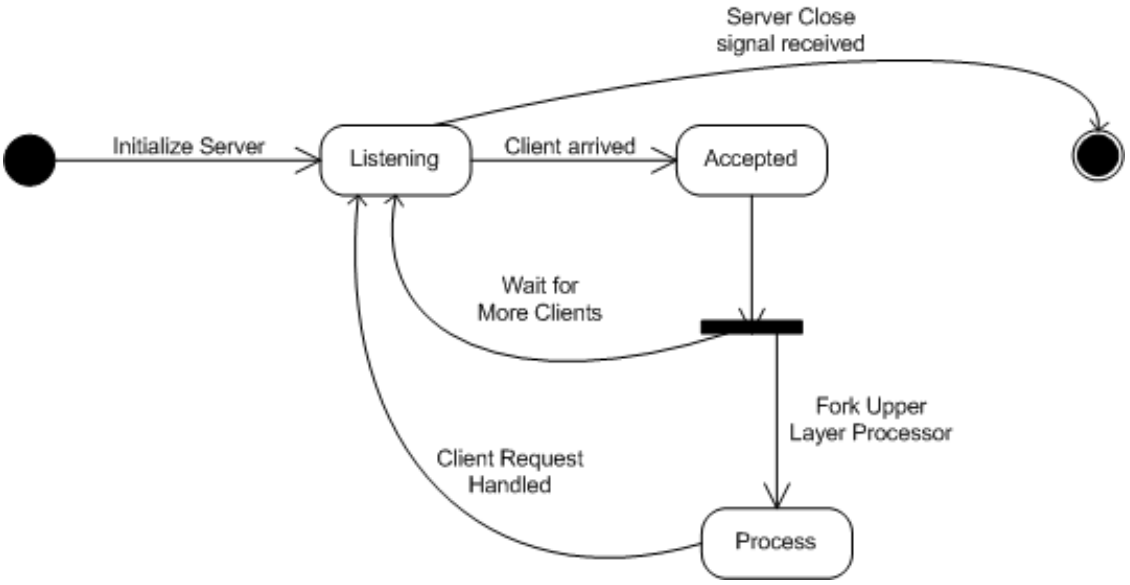
Kovan uygulama seviyesi servisleri kullanarak saldırganların ilgisini çekmeye çalışmaktadır. Gerçek bir servis ya da yalancı bir servis kullanmak sistem yöneticisinin tercihlerine bağlıdır. Apache, bind ve postfix gibi gerçek servisler kullanıldığında etkileşim seviyesi yüksek olmaktadır. Gerçek bir servisin kullanılmasındaki avantaj, saldırganın balküpünü tespit etmesinin zorlaşmasıdır. Öte yandan, etkileşim seviyesi yüksek bir servis kullanılması durumunda saldırganın sistemi ele geçirme olasılığı da artmaktadır. Etkileşim seviyesi düşük bir sistem kullanılması durumunda da saldırganın balküpünü tespit etme olasılığı artarken ele geçirme olasılığı düşmektedir.

Kovan, yaygın servislerin az kaynak tüketen ve her özelliği sağlamayan versiyonlarını içermektedir. HTTP, DNS, SMTP ve FTP servisleri; generic_server adı verilen uygulama seviyesi servis geliştirme alt yapısı üzerine inşa edilmiştir. Bütün servisler ve generic_server altyapısı C dili ile geliştirilerek, sistem kaynaklarının düşük seviyelerde tüketilmesi sağlanmıştır. Üretilen servisler birer örnek teşkil etmektedir, generic_server alt yapısı kullanılarak başka TCP/UDP servisleri de geliştirilebilir.

12.1 Genel Sunucu Altyapısı

Ağ geliştiricileri, ağ temelli uygulamalar geliştirirken paketlerin yazılıp/okunması, doğruluğunun kontrolü, hedefe erişip/erişmediği gibi birçok sorunla ilgilenmek zorundadırlar. Bu genel sorunlarla her geliştirici karşılaşacağı için işletim sistemi, alt seviyedeki bu gibi işleri kendisi hallederek üst seviyelere kolay bir arayüz sunmaktadır. Fakat; HTTP, DNS, SMTP veya FTP gib ağ servisleri geliştirilirken, işletim sisteminin sağladığı socket arayüzü ile sağlanan olanaklara ek olarak geliştiricilerin uğraşması gereken başka sorunlar da ortaya çıkmaktadır. Çok kanlallı mimari(multi-threaded), paylaşılmış bellek (shared-memory) ve kaynak tüketimi gibi önemli tasarım tercihleri geliştiricilere bırakılmıştır. Kovan bünyesinde geliştirilen genel sunucu alt yapısı(kovan_gs) sayesinde önemli tasarım sorunları halledilerek, servis geliştiricisine basit bir arayüz sağlanmaktadır.

Kovan_gs; bağlantı sağlama, kaynak tüketimi, uygulamanın kanallara bölünmesi ve paylaşılan hafıza gibi işlemleri kendi bünyesinde hallederek servis geliştiricisinin bu ayrıntılarla uğraşmadan uygulamalar geliştirmesini sağlamaktadır. Servis geliştiricisinin; soket açıp dinleme, bağlantıyı alma, gerekiyorsa yeni prosesler yaratarak işi dağıtma gibi rutin işlemlerle uğraşması gerekmemektedir. Kovan_gs üst seviye servisler, sadece ihtiyaç duyulan bilgileri göndermektedirler; servis geliştiricisi, gelen ağ paketini, kimden geldiğini, kaynak-hedef portlar gibi bilgileri kovan_gs'den alabilmektedir. Bu bilgiler ışığında eylemler gerçekleştirerek gerekirse cevapları da kovan_gs aracılığı ile göndermektedirler.

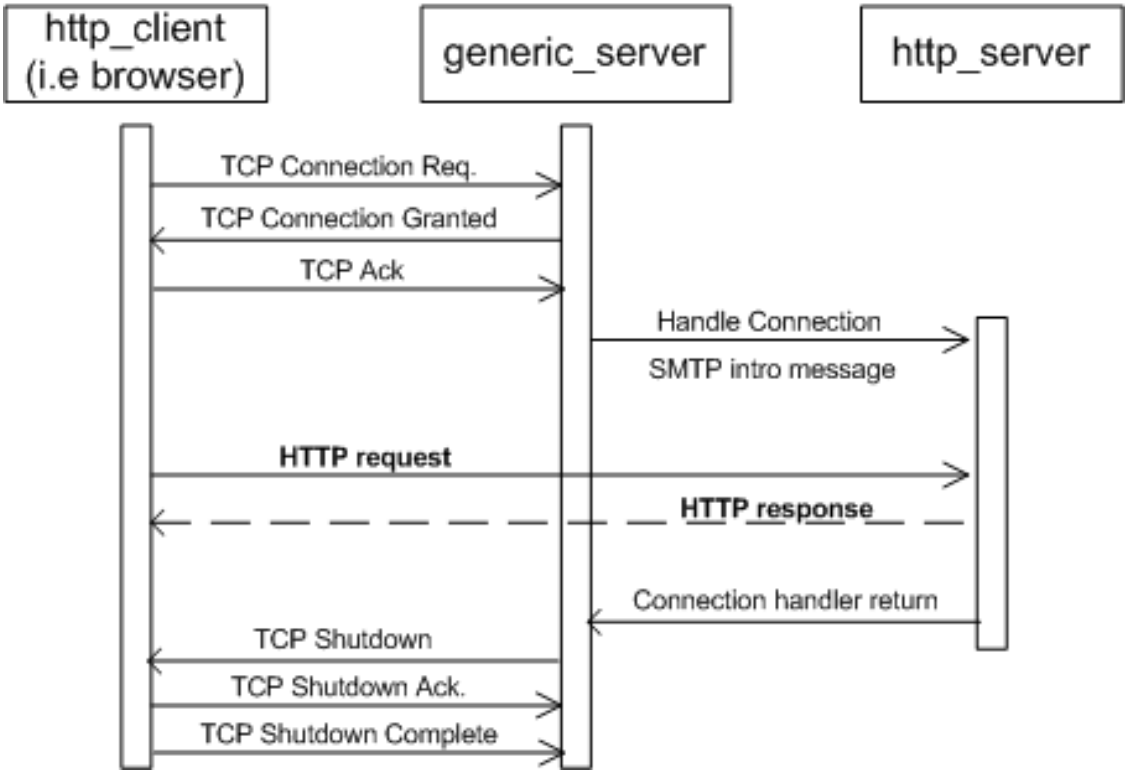


Şekil 12.1: Generic Server's execution states

Şekil 12.1, kovan_gs'nin temel çalışma yapısını göstermektedir. Kovan_gs istemciden gelen istek doğrultusunda bağlantı oluşturur ve bu bağlantıya hizmet etmesi içi bir process yaratır. Oluşturulan proces'e servis methodu verilerek gelen isteğe servise özel cevap verilmesi sağlanır.

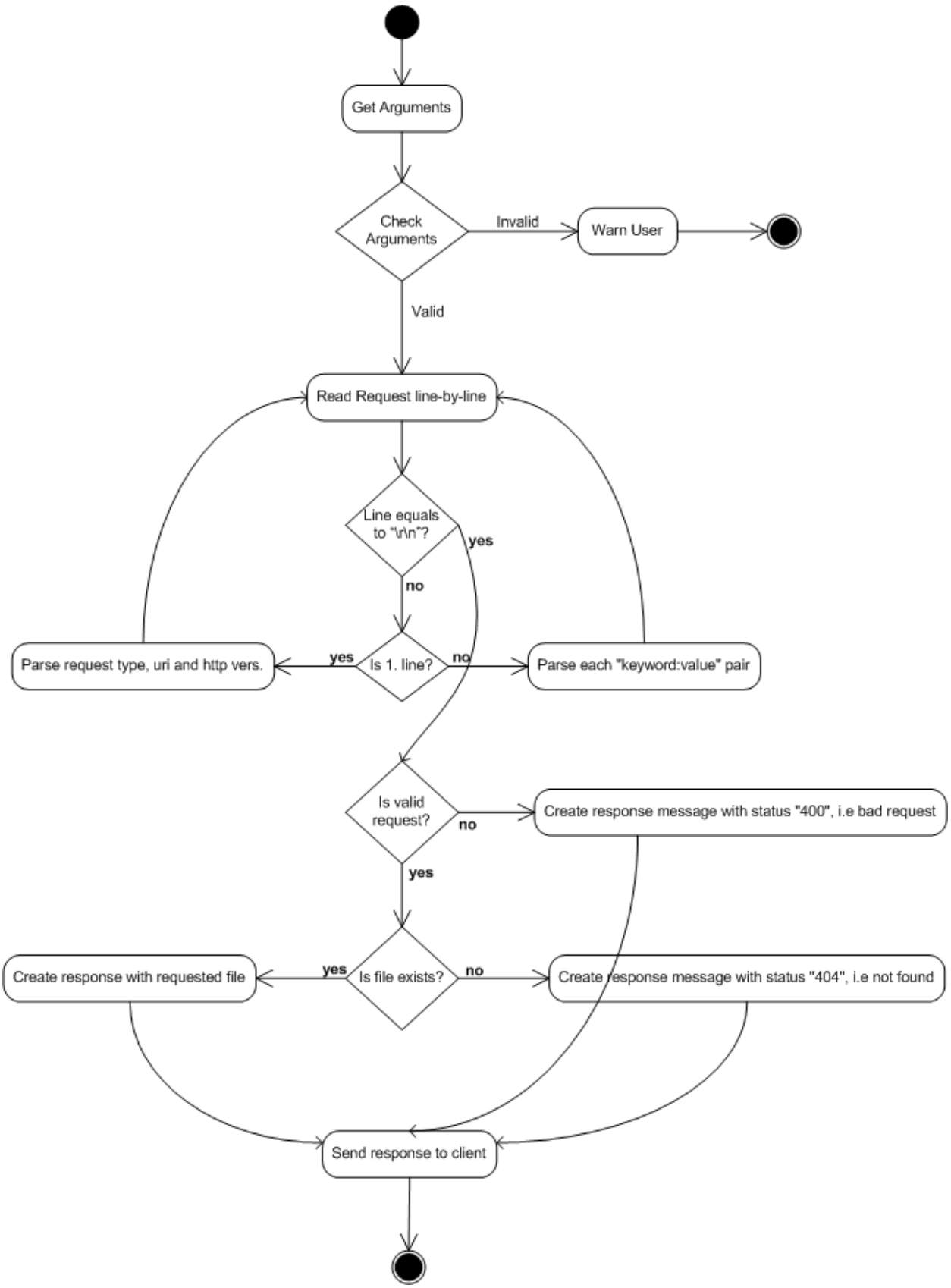
12.2 HTTP

Kovan HTTP servisi kovan_gs altyapısı üzerine inşa edilmiştir. Şekil 12.2'de görüldüğü gibi kovan_gs altyapısı servis öncesi ve sonrası işlemleri yapmaktadır. Yapılması gereken rutin işlemler tamamlandıktan sonra, gerekli olan bilgiler üst seviye olan HTTP prosesine gönderilmektedir. HTTP servisi gelen TCP paketini yorumlayarak cevabını yine kovan_gs yardımıyla istemciye göndermektedir.



Şekil 12.2: HTTP servisi sequence diagram

Şekil 12.3, HTTP servisinin akışını göstermektedir. HTTP paketinin satır-satır okuyan servis ilk aşamada; istek tipi, URL ve http versiyonu bilgilerini elde eder. Sonraki satırlarda keyword:value çiftlerini okuyarak özel bir veri yapısında saklar. HTTP paketi okunduktan sonra servis alınan bilgilerin doğruluğunu kontrol eder. Eğer istek sorunsuz ise servis isteği yerine getirmeye çalışır(istenen dosyayı gönderir), eğer istek hatalı ise istemciye uygun hata mesajı gönderilir.



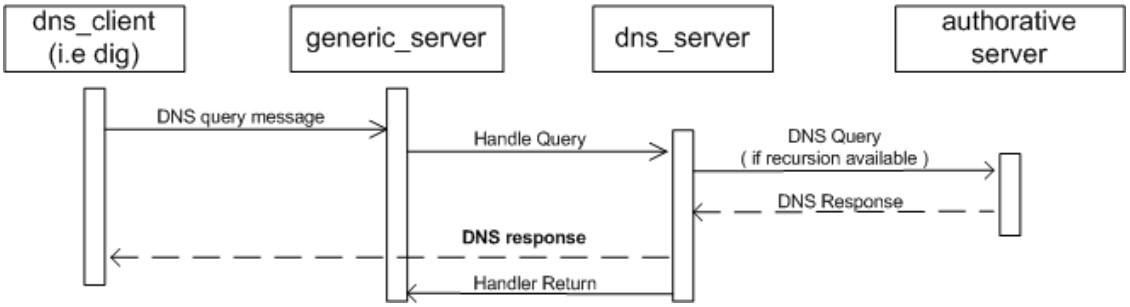
Şekil 12.3: HTTP servisi akış diyagramı

kovan_http üç zorunlu bir tane de tercihe bağlı argüman almaktadır. Eğer servisin bir arkaplan uygulaması olarak çalışması isteniyorsa, "-d pidfile" argümanıyla proses numarasının yazılacağı bir dosya ismi verilmelidir. host ve port argümanları servisin hizmet vereceği IP adresi ve port numarasını belirtmektedir. Son argüman webroot değeri ise web sayfası dosyalarının tutulacağı dosya konumunu belirtmektedir.

```
root# ./kovan_http
usage: kovan_http [-d pidfile] -h host -p port -w webroot
```

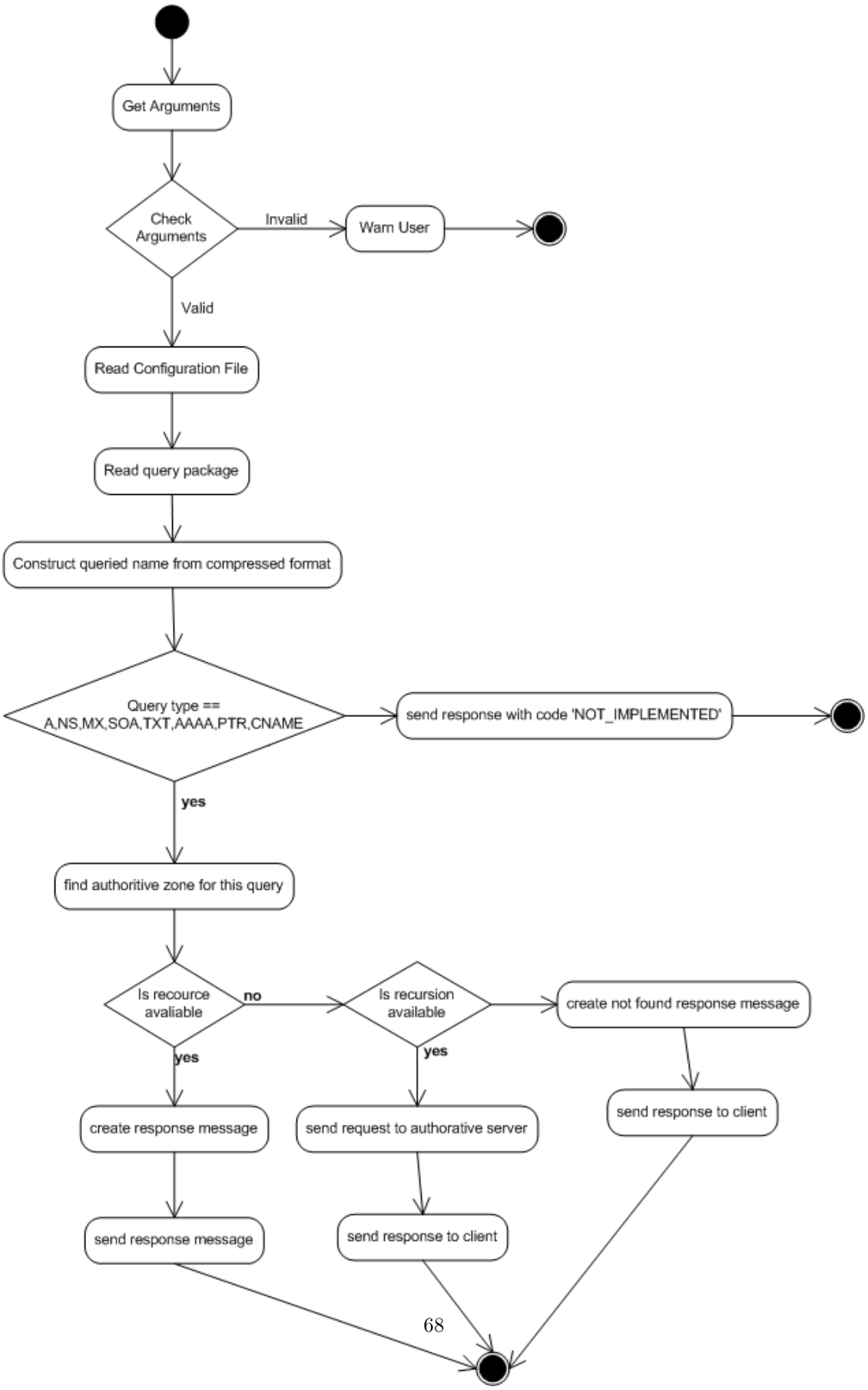
12.3 DNS

Kovan DNS, kovan_gs altyapisi üzerine inşa edilmiştir. Şekil 12.4'te görüldüğü gibi servis öncesi ve sonrası rutin işler kovan_gs tarafından yapılmaktadır. UDP protokolü üstünde çalışan servis, kovan_gs'den gelen veri ışığında istemciye gereken cevabı hazırlayarak gönderir.



Şekil 12.4: DNS sequence diagram

Şekil 12.5, DNS uygulamasının akışını göstermektedir. Servis, ilk basamakta dns ayar dosyalarını okuyarak sonradan kullanacağı tabloları oluşturur. Dosyalar okunurken bir hata oluşması durumunda program çalışmayı durdurur. Her DNS talep mesajında kovan_gs tarafından yaratılan prosesin, dns tablolarını kullanarak istemciye uygun cevabı vermesi sağlanır. Yapılan istek desteklenen bir dns isteği değil ise, cevap verme süreci başlamadan bitirilir.



Şekil 12.5: DNS akış diyagramı

kovan_dns üç zorunlu ve iki tane de tercihe bağlı argümanla aşağıdaki gibi çalıştırılmaktadır. Eğer servisin bir arkaplan uygulaması olarak çalışması isteniyorsa, "-d pidfile" argümanıyla proses numarasının yazılacağı bir dosya ismi verilmelidir. host ve port argümanları servisin hizmet vereceği IP adresi ve port numarasını belirtmektedir. "-r" argümanı DNS'in recursive sorgulara açık olup olmadığını belirtir, "-r 1" verilmesi durumunda kovan_dns recursive sorgulara da cevap verir.

```
root# ./kovan_dns
usage: kovan_dns [-d pidfile] -h host -p port -c config [-r recurse]
```

DNS ayar dosyasının konumu, "-c config" argümanıyla belirtilmektedir (bu konum dosyanın tam konumu -fullpath- olmalıdır). Ayar dosyası sorumlu olunan her alan(zone) için satır içermektedir. Her satır "alan ismi-alan dosyası(zonefile)":

```
root# cat named.conf
"ipv6go.ipv6.net.tr." "ipv6go.ipv6.net.tr.conf"
```

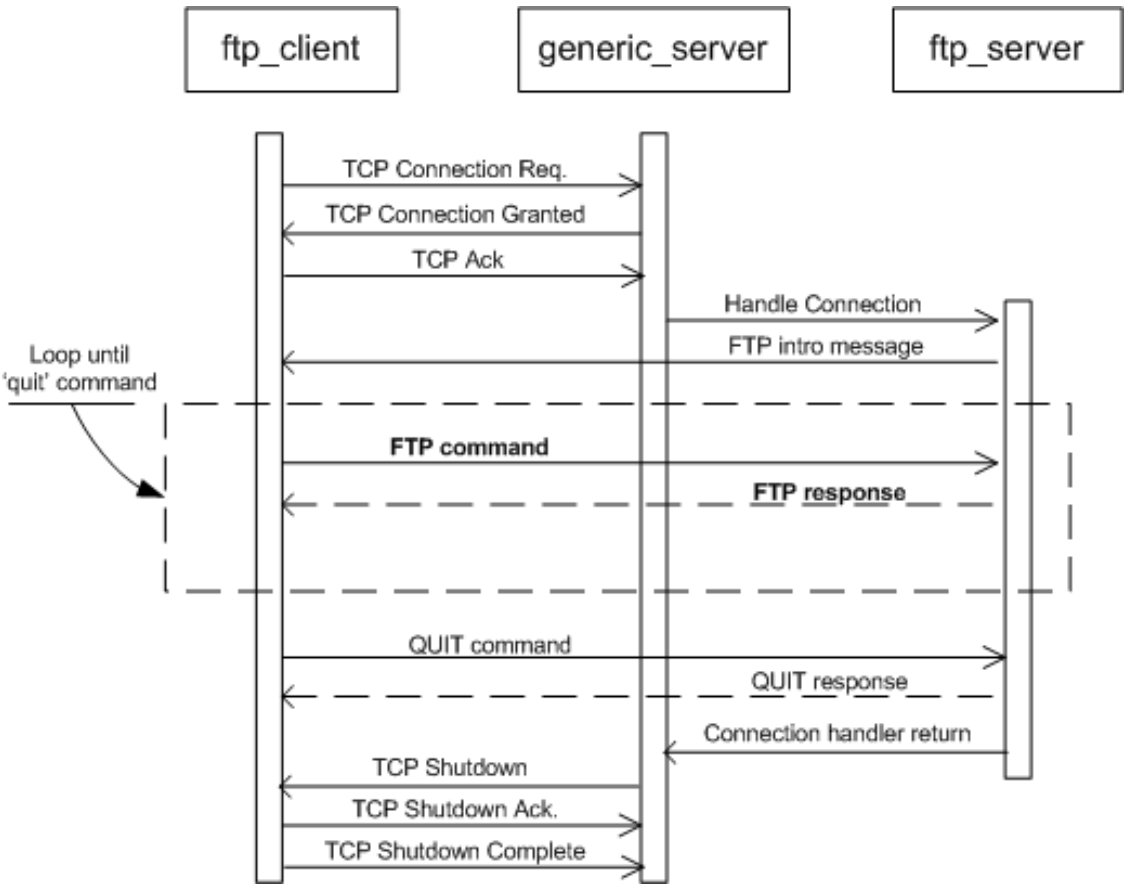
Alan dosyaları gerçek bir DNS alan dosyasının yapısını kullanmaktadırlar (wildcard özelliği gibi bazı ileri özellikler desteklenmemektedir). Örnek bir alan dosyası liste 12.1'te gösterilmektedir:

```
root# cat ipv6go.ipv6.net.tr.conf
$ORIGIN ipv6go.ipv6.net.tr.
$TTL 1h                 ; comments...
ipv6go.ipv6.net.tr. IN  SOA ns1.ipv6go.ipv6.net.tr. hostmaster.ipv6go.ipv6.net
    .tr. 2009082702 28800 7200 604800 600
ipv6go.ipv6.net.tr. NS     ns1.ipv6go.ipv6.net.tr.
ipv6go.ipv6.net.tr. MX     10 mail.ipv6go.ipv6.net.tr.
ipv6go.ipv6.net.tr. AAAA   2001:a98:14:1::3
www    AAAA   2001:a98:14:1::3
mail   AAAA   2001:a98:14:1::4
ns1    AAAA   2001:a98:14:1::2
ftp    AAAA   2001:a98:14:1::5

bsd1.computerscience     AAAA   2001:a98:14:2::3
bsd2.computerscience     AAAA   2001:a98:14:2::4
bsd3.computerscience     AAAA   2001:a98:14:2::5

bsd4.management          AAAA   2001:a98:14:3::3
bsd5.management          AAAA   2001:a98:14:3::4
bsd6.management          AAAA   2001:a98:14:3::5
```

Listing 12.1: Alan dosyası içeriği

12.4 FTP

FTP servisi bir mesaj sunucusu olarak tasarlanmıştır. İstemciye uygun cevaplar dönülerek istenen işin yapıldığı izlenimi verilmektedir. Şekil 12.6 FTP servisinin genel çalışmasını göstermektedir.



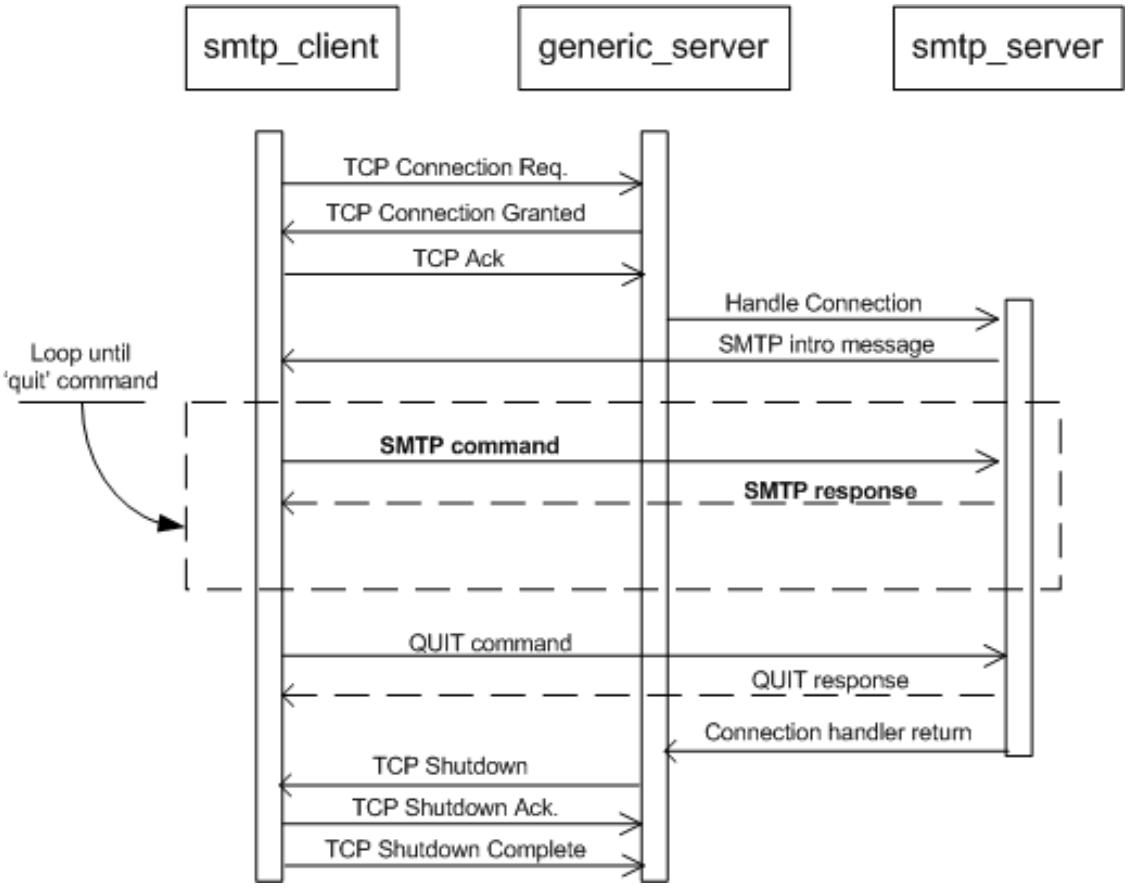
Şekil 12.6: FTP service sequence diagram

kovan_ftp iki zorunlu ve bir tane de tercihe bağlı argümanla aşağıdaki gibi çalıştırılmaktadır. Eğer servisin bir arkaplan uygulaması olarak çalışması isteniyorsa, "-d pidfile" argümanıyla proses numarasının yazılacağı bir dosya ismi verilmelidir. host ve port argümanları servisin hizmet vereceği IP adresi ve port numarasını belirtmektedir.

```
root# ./kovan_ftp
usage: kovan_ftp [-d pidfile] -h host -p port
```

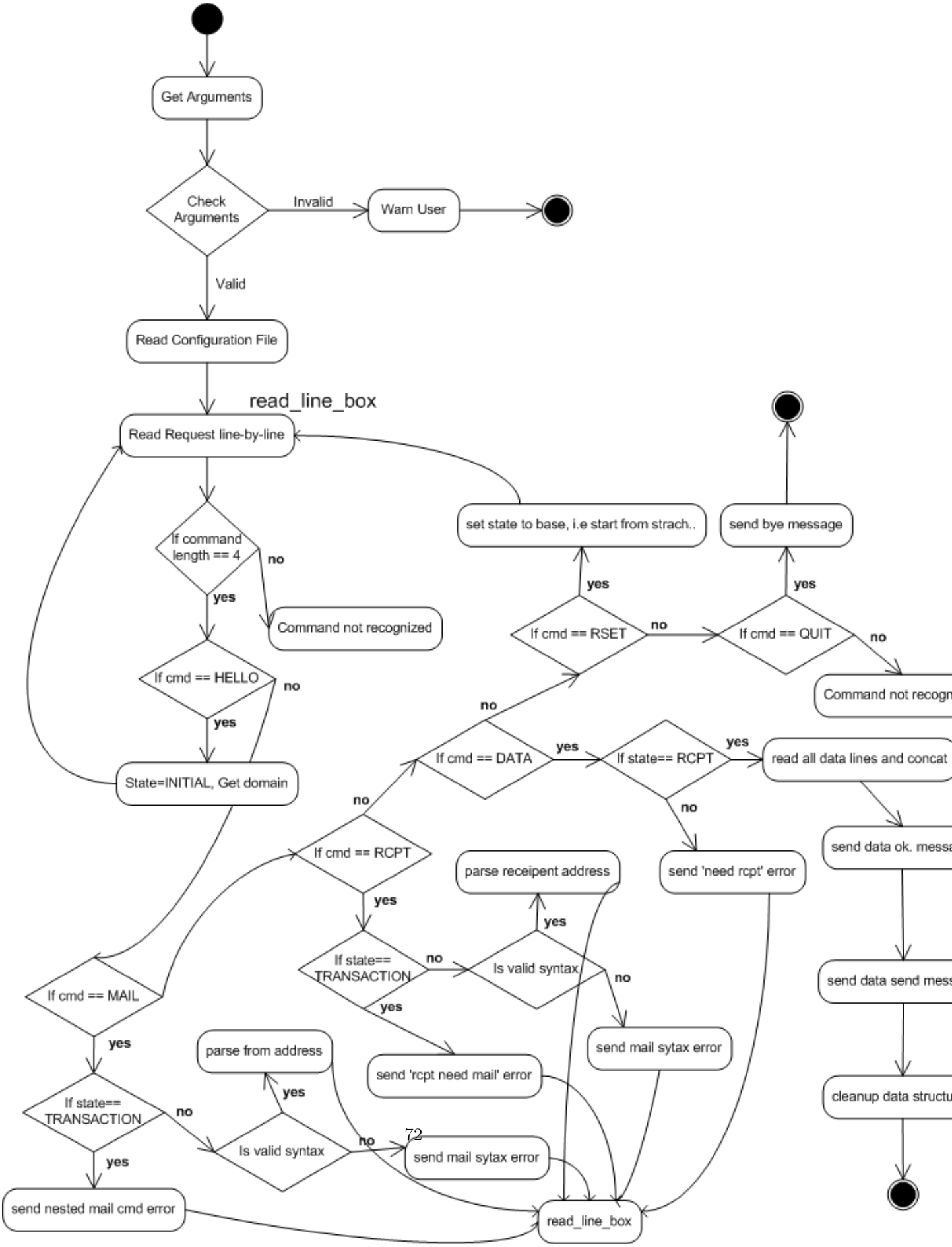
12.5 SMTP

Yapılması gereken rutin işlemler kovan_gs tarafından tamamlandıktan sonra, gerekli olan bilgiler üst seviye olan SMTP prosesine gönderilmektedir. SMTP servisi gelen TCP paketini yorumlayarak cevabını yine kovan_gs yardımıyla istemciye göndermektedir.



Şekil 12.7: SMTP service sequence diagram

SMTP servisi diğer servislerle karşılaştırıldığında daha karmaşık bir akış grafiğine(şekil 12.8) sahiptir. Bunun nedeni smtp'nin etkileşimli ve durumlu(stateful) yapısıdır. Etkileşim, program akışı sırasında durumlar arasında yaşanan geçişlerin artmasına sebep olmaktadır. Ek olarak, durumlu yapı, durumdan duruma geçişlerin sürekli kontrol edilmesini gerektirmektedir.



Şekil 12.8: SMTP servis akış diyagramı

kovan_smtp üç zorunlu ve bir tane de tercihe bağlı argümanla aşağıdaki gibi çalıştırılmaktadır. Eğer servisin bir arkaplan uygulaması olarak çalışması isteniyorsa, "-d pidfile" argümanıyla proses numarasının yazılacağı bir dosya ismi verilmelidir. host ve port argümanları servisin hizmet vereceği IP adresi ve port numarasını belirtmektedir.

```
root# ./kovan_smtp
usage: kovan_smtp [-d pidfile] -h host -p port -c config
```

SMTP ayar dosyasının konumu, "-c config" argümanıyla belirtilmektedir. Ayar dosyası her cevap mesajı için bir satır içermektedir:

```
HELO_250 "250 mail.ulakbim.gov.tr"
MAIL_250 "250 ok"
HELO_FIRST_503 "503 5.5.1 Error: send HELO/EHLO first"
MAIL_SYNTAX_ERR "501 5.5.4 Syntax: MAIL FROM:<address>"
MAIL_NESTED_ERR "503 5.5.1 Error: nested MAIL command"
QUIT_BYE "221 2.0.0 Bye"
RCPT_NEED_MAIL "503 5.5.1 Error: need MAIL command"
RCPT_SYNTAX_ERR "501 5.5.4 Syntax: RCPT TO:<address>"
RCPT_OK "250 2.1.5 Ok"
DATA_NEED_RCPT "503 5.5.1 Error: need RCPT command"
DATA_START "354 End data with <CR><LF>.<CR><LF>"
DATA_OK "250 2.0.0 Ok: queued as 296F511E823"
CMD_NOT_RECOG "502 5.5.2 Error: command not recognized"
```

Listing 12.2: SMTP ayar dosyası

Bölüm 13

Loglama

Kovan, servislerden ve karadelikten log toplamaktadır. Kovan servisleri hata mesajlarını ve erişim mesajlarını /var/log/messages dosyasına yazacak şekilde tasarlanmıştır. /var/log/messages altına yazılan loglar, bir betik tarafından monitör cihazına sürekli olarak gönderilmektedir. Kovan servisleri aşağıdaki genel log yapısını kullanmaktadırlar:

```
Nov 22 15:17:30 HTTP kovan_http: 2001:a98:14:5::2 1755
    2001:a98:14:1::3 80 tcp "GET / HTTP/1.0 404 0" 1290439050
```

Genel log yapısı 10 alan içermektedir, bunlar: log tarihi, kaynak cihaz, kaynak servis, kaynak IP, kaynak PORT, hedef IP, hedef PORT, kullanılan protokol, mesaj, utc zamanı. Kovan'da, tüm servis cihazları aynı jail dosya sistemini kullandıkları için, loglama yapılırken aynı log soketi ve aynı /var/log/messages dosyasını kullanmaktadırlar. Bu yüzden, servisler için sadece bir tek syslog programı çalıştırılmaktadır. Sonuç olarak, genel log formatının ikinci alanı her servis logunda aynı değeri almaktadır, syslog'un çalıştığı sanal cihazın adı bu alanda yer almaktadır.

Kovan servislerinden gönderilen bilgilere ek olarak karadelik'ten de loglar monitör cihazına gönderilmektedir. Karadelik logları aşağıdaki yapıdadır:

```
Nov 22 15:17:30 BLACKHOLE blackhole: 2001:a98:12::151 -1
    2001:a98::abab -1 icmp6 blackhole_flow 1290440635
```

Monitör cihaz tarafında, syslog-ng3 uygulaması kullanılarak loglar toplanır ve veritabanına yazılır. syslog-ng3 uygulaması aşağıda verilen ayar dosyasıyla çalıştırılmaktadır. Ayar dosyasında logların yazılacağı hedef olarak bir mysql veritabanı tanımlanmıştır. syslog-ng3 logları 510. udp portundan alarak, işler ve sonrasında mysql veritabanına ekler. Veritabanı tablosu syslog-ng3 tarafından otomatik olarak yaratılmaktadır.

```
1  options { long_hostnames(off); flush_lines(0); };
2
3  source net {
4      udp6(port(510));
5  };
6
7  parser p_kovan {
8      csv-parser(columns( "KOVAN.SRC_IP", "KOVAN.SRC_PORT",
9      "KOVAN.DEST_IP", "KOVAN.DEST_PORT", "KOVAN.PROTO",
10     "KOVAN.MSG", "KOVAN.TIME")
11     flags(escape-double-char,strip-whitespace)
12     delimiters(" ")
13     quote-pairs('""[]')
14     );
```

```
15 };
16
17 destination d_mysql {
18     sql(type(mysql)
19         host("localhost") username("logger") password("231905")
20         database("kovan")
21         table("logs")
22         columns( "logtime INT UNSIGNED","service varchar(32)",
23         "srcip varchar(64)", "srcport int", "destip varchar(64)",
24         "destport int", "proto ENUM('tcp', 'udp', 'icmp6', 'icmp')",
25         "msg varchar(256)" )
26         values( "${KOVAN.TIME}", "$PROGRAM", "${KOVAN.SRC_IP}",
27         "${KOVAN.SRC_PORT}", "${KOVAN.DEST_IP}", "${KOVAN.DEST_PORT}",
28         "${KOVAN.PROTO}", "${KOVAN.MSG}" )
29         indexes("service")
30     );
31 };
32
33 log { source(net); parser(p_kovan); destination(d_mysql); };
34
```