

T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

IPSEC/VPN GEÇİTLERİ İÇİN SCRIPT MIB
TABANLI DAĞITIK YÖNETİM SİSTEMİ

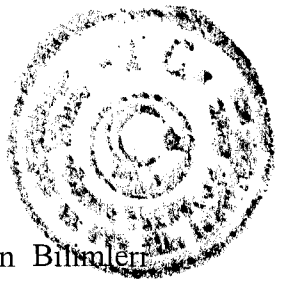
134348

Mücahit MUTLUGÜN
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

TEZ DANIŞMANI
Yrd.Doç.Dr. İbrahim SOĞUKPINAR

GEBZE
2003

134348
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ



Mücahit MUTLUGÜN'ün tez çalışması, G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 16-06-2003 tarih ve .2003-18 sayılı kararıyla oluşturulan jüri tarafından Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS Tezi olarak kabul edilmiştir.

JÜRİ

ÜYE : Yrd. Doç. Dr. İbrahim SOĞUKPINAR
(Tez Danışmanı)

ÜYE : Yrd. Doç. Dr. F. Mehmet GÖKTÜRK

ÜYE : Yrd. Doç. Dr. İlyas KANDEMİR

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
16.07.2003 tarih ve 2003/22 sayılı kararı.



ÖZET

IP ağlarının gelişmesi ile ağların ve ağlardaki elemanların yönetim ihtiyacı paralel olarak artmıştır. IP ağlarının güvenliğini sağlamada kullanılan güvenlik cihazlarından IPSec/VPN geçitlerinin de etkili bir şekilde yönetilmesi gerekmektedir. Günümüzde bu iş için kullanılan çözümler yetersiz kalmaktadır.

IPSec geçitlerinin yönetim ihtiyaçları göz önüne alınarak geliştirilen Script MIB tabanlı dağıtık yönetim sistemi pek çok yönden diğer çözümlerin eksiklerini kapatmakta ve bu yönden iyi bir alternatif olmaktadır.

IPSec geçitleri için Script MIB tabanlı dağıtık yönetim sisteminin önce teorik altyapısı verilmiş, bunun ardından sistemin mimarisi ve elemanları ayrıntılı şekilde açıklanmıştır. Daha sonra da delegasyon için kullanılan script tiplerinden bahsedilmiştir. Bu scriptler için örnek uygulamalar yapılmış ve ardından da bu scriptler üzerinden sistemin verimliliği hesaplanmıştır.

Çalışmanın son kısmında ise sistemin avantajları ortaya konmuş, diğer çözümlerle bazı kıyaslamalar yapılmış ve sistemin eksik bazı noktalarının giderilmesi adına öneriler sunulmuştur.

Script MIB-based distributed management system, which is developed by considering management needs of IPSec/VPN gateways, solves insufficiencies of other solutions' and becomes a good alternative.

At the last part of the work, the advantages of the system and some comparisons with other works are presented. Furthermore, for the sake of overcoming some insufficient points of the presented system, some recommendations are given.

TEŞEKKÜR

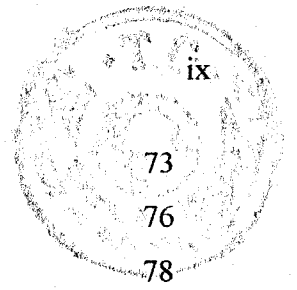
Tez çalışmam boyunca yönlendirmeleri ile büyük pay sahibi olan tez danışmanım Sayın Yrd. Doç. Dr. İbrahim Soğukpınar'a teşekkürlerimi sunarım. Tez çalışmamı destekleyen TÜBİTAK-UEKAE Müdürü Sayın Önder Yetiş'e ve proje müdürüm Sayın Oktay Adalier'e teşekkürlerimi sunarım. Ayrıca bana daima maddi ve manevi yardımlarını eksik etmeyen eşime de teşekkürü bir borç bilirim.



İÇİNDEKİLER DİZİNİ

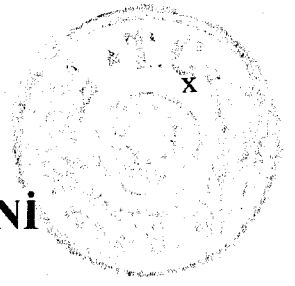
	<u>Sayfa</u>
ÖZET	iv
SUMMARY	v
TEŞEKKÜR	vi
İÇİNDEKİLER DİZİNİ	vii
SİMGELER VE KISALTMALAR DİZİNİ	x
ŞEKİLLER DİZİNİ	xii
ÇİZELGELER DİZİNİ	xiii
1. GİRİŞ	1
1.1. Problemin Genel Tanımı	1
1.2. Ipsec Geçitleri ve Yönetim İhtiyacı	3
1.2.1. IPSec Protokolü ve IPSec/VPN Geçitleri	3
1.2.2. IPSec Geçitlerinin Yönetimi Gereksinimi	7
1.2.3. IPSec Geçitlerinin Güvenlik Ve Konfigürasyon Yönetimi Gereksinimleri	8
1.2.4. Ipsec Geçitlerinin Olay Yönetimi Gereksinimleri	15
1.2.5. IPSec Geçitlerinin Gözlenme Gereksinimleri	16
2. AĞ YÖNETİMİ MİMARİLERİ VE PROBLEM İLE İLGİLİ ÇALIŞMALAR	18
2.1. Ağ Yönetimi Mimarileri	18
2.1.1. Merkezi Yönetim	19
2.1.1.1. SNMP Tabanlı Yönetim	20
2.1.1.2. HTTP (Hyper-Text Transfer Protocol) Tabanlı Yönetim	20
2.1.2. Zayıf Dağıtık Hiyerarşik Ağ Yönetimi	21
2.1.2.1. Telekomünikasyon Alanında	22
2.1.2.2. IP Alanında	22
2.1.3. Güçlü Dağıtık Hiyerarşik Ağ Yönetimi	23
2.1.3.1. Gezgin Kod	24
2.1.3.2. Dağıtık Nesneler	25

2.1.4. Güçlü Dağıtık Kooperatif Ağ Yönetimi	26
2.1.5. Ağ Yönetimi Mimarilerinin Ipsec Geçitleri Yönetimi İçin İncelenmesi	27
2.2. Konuyla İlgili Çalışmalar	31
2.2.1. Merkezi Yönetim	31
2.2.2. SPP (Security Policy Protocol)	33
2.2.3. Politika Tabanlı Ağ Yönetimi	35
3. IPSEC GEÇİTLERİ İÇİN SCRIPT MIB TABANLI DAĞITIK YÖNETİM SİSTEMİ	38
3.1. DISMAN Teknolojileri – Script MIB	38
3.1.1. Script MIB	38
3.1.1.1. Script MIB'in yapısı	38
3.1.1.2. Script MIB'in Haberleşme ve Kullanım Modeli	41
3.1.1.3. Script MIB'in Güvenliği	42
3.2. Yönetim Sistemi	42
3.2.1. Yöneticilerin Yöneticisi	44
3.2.2. Alan Yöneticileri	46
3.2.3. IPSec Yönetim Etmenleri	47
3.2.4. Yönetim Script'leri	49
3.2.4.1. Kullanılan Script Ortamı	49
3.2.4.2. Script Tipleri	50
4. ÖRNEK UYGULAMA YORUMLAR VE KARŞILAŞTIRMA	52
4.1. Örnek uygulamalar	52
4.1.1. IPSec Konfigürasyon Test MIB'i	53
4.1.2. Konfigürasyon Scriptleri	55
4.1.3. Olay Scriptleri	59
4.1.4. Gözlem Scriptleri	64
4.2. Sistemin Avantajları ve Analizler	67
4.2.1. Dağıtık Yönetim	67
4.2.2. Standartlar Kullanılması	67
4.2.3. Kod Dağıtımı	68
4.2.4. Ölçeklenebilirlik	68
4.2.4.1. Konfigürasyon Scriptlerinin Ölçeklenebilirliği	69



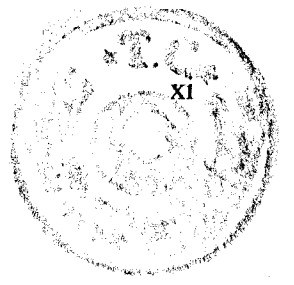
4.2.4.2. Olay Scriptlerinin Ölçeklenebilirliği	73
4.2.4.3. Gözlem Scriptlerinin Ölçeklenebilirliği	76
4.2.5. Hata Kotarma	78
4.2.6. Tekrar Kullanılabilirlik	78
4.2.7. AY'lerin Esnek Yapısı	79
4.2.8. MIB Bağımlılığı Olmaması	79
5. SONUÇLAR VE ÖNERİLER	81
KAYNAKLAR	84
ÖZGEÇMİŞ	88
EKLER	
Ek-1: Test için Geliştirilen IPSec Configuration MIB	
Ek-2: Geliştirilen yönetim scriptlerinin kaynak kodları	





SİMGELER VE KISALTMALAR DİZİNİ

IP	Internet Protocol
IPSec	IP Security Protocol
SNMP	Simple Network Management Protocol
IETF	Internet Engineering Task Force
VPN	Virtual Private Networks
DISMAN	Distributed Management
MIB	Management Information Base
AY	Alan Yöneticisi
YY	Yöneticiler Yöneticisi
AH	Authentication Header
ESP	Encapsulation Security Payload
IKE	The Internet Key Exchange Protocol
SPI	Security Parameters Index
KH	Kripto Haritası
GB	Güvenlik Birliği
ISO	International Organization for Standardization
TMN	Telecommunications Management Network
HTTP	Hyper-text Transfer Protocol
XML	Extensible Markup Language
RMON	Remote Monitoring
CORBA	Common Object Request Broker Architecture
RMI	Remote Method Invocation
OMG	Object Management Group
SPP	Security Policy Protocol
SPS	Security Policy System
PDP	Policy Decision Point
PEP	Policy Enforcement Point
LDAP	Lightweight Directory Access Protocol
COPS-PR	Common Open Policy Service for Policy Provisioning
FTP	File Transfer Protocol
OID	Object Identifier



IYE	IPSec Yönetim Etmeni
UML	Unified Modelling Language
API	Application Programming Interface

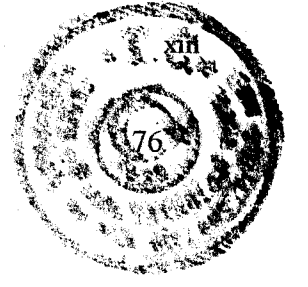




ŞEKİLLER DİZİNİ

<u>Sekil</u>	<u>Sayfa</u>
1.1. Tünel Modu	4
1.2. IPSec AH protokolü paket yapısı	5
1.3. IPSec ESP protokolü paket yapısı	6
1.4. IPSec geçitleri arasında örgü yapısı	10
1.5. IPSec geçitlerinin şube çalışma alanlarına göre gruplanması	12
1.6. IPSec geçitlerinin şube bölgesel özelliklerine göre gruplanması	13
2.1. Delegasyon ve yardımlaşma	19
2.2. Merkezi Yönetim	20
2.3. Zayıf dağıtık hiyerarşik yönetim	22
2.4. Güçlü dağıtık hiyerarşik yönetim	23
2.5. Güçlü dağıtık kooperatif yönetim	27
2.6. IPSec geçitlerinin merkezi yönetimi	31
2.7. SPS politika sorgusu	33
2.8. IETF politika tabanlı ağ yönetimi mimarisi	35
3.1. Script MIB'in yapısı	40
3.2. Sistemin genel mimarisi	44
3.3. YY'nin altsistemleri	45
3.4. AY ve IYE'nin detaylı yapısı	48
4.1. Geliştirilen scriptlerin paket yapıları	52
4.2. IPSec Configuration Test MIB'in yapısı	54
4.3. IPSec Configuration Test MIB'deki güvenlik parametre tabloları	55
4.4. Konfigürasyon script'lerinin UML sınıf diyagramı	58
4.5. Event paketinin UML sınıf diyagramı	60
4.6. Filtre sınıflarının UML sınıf diyagramı	62
4.7. Action sınıfından türeyen gözlem ve olay scriptleri	64
4.8. Gözlem scriptlerinin UML sınıf diyagramı	66
4.9. Manuel tünel konfigürasyon maliyet grafiği	70
4.10. Tünel konfigürasyonu sayısı-verimlilik grafiği	71
4.11. Zamana programlı konfigürasyon verimlilik grafiği	72
0.12. Zamana programlanmış konfigürasyon maliyet grafiği	73

0.43. Olay delegasyonu verimliliği-zaman grafiği



ÇİZELGELER DİZİNİ

Cizelge

Sayfa

2.1. Ağ yönetimi mimarilerinin karşılaştırılması

30

0.2. Çözümlerin kıyaslama tablosu

80



1 GİRİŞ

1.1 Problemin Genel Tanımı

Bilgisayar ağıları son on yılda çok hızlı şekilde gelişti. Kullanıcı sayısı da paralel olarak günden güne artmaktadır. Kamu ve özel sektör kuruluşları da her geçen gün bilgisayarları ve beraberinde bilgisayar ağlarını, özellikle Internet'i daha fazla kullanmaktadırlar. Artık hemen her şirketin, kamu kuruluşunun ve hatta şahısların Web sayfaları mevcuttur. Bunların sonucu olarak bilgiye erişim, paylaşım ve depolama hayatımızda pek çok kolaylıklar sağlamıştır.

Bilgisayar sistemleri kolaylıklarla beraber bazı problemleri de getirmiştir. Bu problemlerin en başında da güvenlik gelmektedir. Çünkü çağımızda bilgi en önemli meta olmuştur ve bilgisayar sistemleri de bilginin depolandığı ve iletildiği ortamların en başında gelmektedir. Özellikle elektronik ticaretin gelişmesi de güvenlik gereksinimlerini had safhaya çıkarmıştır. Bilgisayar ağlarında güvenliğin sağlanabilmesi konusu ciddi bir araştırma konusu olmuş ve bu konuda şimdiye kadar hatırı sayılır emek harcanmıştır. Bu araştırmalar çerçevesinde günümüzde bilgisayar ağlarının güvenliğinin sağlanmasında güvenlik duvarları, nüfuz tespit sistemleri, erişim kontrol sistemleri ve kriptu sistemleri gibi bazı güvenlik sistemleri geliştirilmiştir. Bu sistemler kendilerinden bekleneni çoğu zaman yerine getirirler dahi bunların bulunması güvenlik için yeterli olmamaktadır. Bu sistemlerin istenen güvenlik fonksiyonlarını sağlayabilmeleri için titizlikle konfigüre edilmeleri, alarmlarının değerlendirilmesi ve gözlenmeleri gerekmektedir. Daha genel şekilde ifade edilirse bunların ciddi şekilde yönetilmesi gerekmektedir. Ayrıca bu sistemlerin sayısı arttıkça da yönetim ihtiyacı ve paralel olarak da zorluğu artmaktadır. Bu güvenlik sistemlerini yönetmek için bazı yönetim sistemleri geliştirilmiş olmasına rağmen pratikte bunlar operatörler tarafından manuel olarak yönetilmektedir. Fakat şu açık bir gerçektir ki bu şekilde bir yönetim hataya açık ve tesir gücü az bir yöntemdir. Bu sistemler kullanıma geçirildiğinde gerekli konfigürasyonları yapılsa dahi değişen şartlar konfigürasyon değişiklikleri gerektirmektedir ki bu da hataya daha açık bir iştir. Güvenlik sistemlerinin yönetimi, pek çok şeyi otomatik hale getiren gelişmiş yönetim sistemleri aracılığıyla yapılmalıdır.

Bu tez çalışmasında araştırma konusu olan IPSec (Internet Protocol Security) protokolünü gerçekleyen IPSec/VPN (Virtual Private Networks-Sanal Özel Ağlar) geçitleri de yukarıda bahsedilen güvenlik sistemlerinden biridir. Bu geçitler üzerilerinden geçen trafiği şifreleyerek pek çok seviyede güvenlik sağlanmaktadır. Diğer güvenlik sistemlerinde bulunan yönetim ihtiyacı belki daha fazla şekilde bu cihazlarda bulunmaktadır. Bu cihazların da benzer şekilde ayrıntılı cihaz ve güvenlik konfigürasyonuna ihtiyacı vardır. Bu cihazlar, üzerlerinden geçen trafiğe yapılacak muameleye bulundurdıkları güvenlik politikalarına bakılarak karar vermektedirler. Bu yüzden bu politikaların belirlenmesi ve doğru bir şekilde ayarlanması kritik bir mevzudur. Halihazırdaki uygulamaya bakılırsa, IPSec politikaları genelde her bir cihazda ayrı ayrı ve manuel olarak ayarlanmaktadır ki daha önce bahsedildiği gibi bu yöntem hataya açık ve etkisi ve başarısı düşük bir yöntemdir. Büyük bir organizasyonun pek çok IPSec güvenlik geçidine sahip ve kompleks bir ağında her bir nokta için güvenlik politikalarının belirlenip ayarlanması çok fazla iş yükü getirmektedir. Bazen yapılan küçük hatalar çok büyük güvenlik açıklarına sebep olabilmektedir. Ayrıca iş, bu sistemlerin konfigürasyonunun yapılması ile de bitmemektedir. Bu cihazların ciddi şekilde gözlenmesi, hata ve alarm yönetimlerinin de yapılması gerekmektedir ki sistemde politikaların belirlenmesinde veya konfigürasyonda yapılan hatalar belirlenebilsin ve güvenlik cihazlarının zaman içinde ortaya çıkan problemleri ve durumundaki değişimler değerlendirilebilsin.

Bu noktada bu cihazların yönetimi için kullanılabilecek yöntemler düşünüldüğünde geleneksel ağ yönetim mimarilerinin yeterlilik gösteremediği görülmektedir. Birkaç IPSec geçidinin yönetimi için eski yöntemler yeterlilik gösterebilse bile yüzlerce noktası olan kamuya ait veya büyük şirketlere ait VPN'lerde bu mimariler başarılı olamamaktadır. Alternatifler düşünüldüğünde pek çok dağıtık ağ yönetimi mimarisi ve bu mimarilerin altyapısında kullanılan pek çok protokol bulunmaktadır. Bu noktada böyle bir yönetim ihtiyacını neyin karşılayacağına çok iyi karar verilmesi gerekmektedir. Bazı çözümler vardır ki çok güçlü olmasına rağmen oldukça komplekstir ve pratik hayatta gerçekleştirilmesi ve hatta kullanılması dahi nerede imkansızdır. Bu noktada endüstrinin trendleri gerekli bazı ipuçlarını vermektedir.

Bu tezin hedefi IPSec/VPN güvenlik geçitleri için uygulanabilir ve bu cihazların bütün yönetim ihtiyaçlarını yeterli bir şekilde karşılayabilen bir yönetim

sistemi önermektedir. Burada önerilen sistem bir esnek delegasyon mekanizmalarını içeren bir dağıtık yönetim çözümüdür. Bu çözüm temelde SNMP (Simple Network Management) tabanlı bir çözümdür, fakat burada SNMP geleneksel şekilde kullanımdan farklı olarak güçlü-dağıtık hiyerarşik bir yönetim mimarisinde delegasyon mekanizmalarını da destekler şekilde kullanılmaktadır. Burada SNMP delegasyonu IETF (Internet Engineering Task Force) DISMAN (Distributed Management) Çalışma grubu tarafından standartlaştırılan Script MIB'i (Management Information Base) kullanarak yapmaktadır. Burada önerilen yönetim sisteminde IPsec geçit cihazları kullanım alanlarına göre gruplanmaktadır ve her bir alan da bir AY (alan yöneticisi) tarafından kontrol edilmektedir. Her bir AY Script MIB ve gerekli Script çalışma ortamını bulundurmaktadır. Bütün AY'ler bir YY (Yöneticiler Yöneticisi) tarafından kontrol edilmektedir. YY kendi yönetim sorumluluklarını yönetim scrippleri halinde AY'lere delege etmektedir. YY'nin yönetim sorumlulukları iç ana başlık altında toplanmaktadır ki bunlar konfigürasyon yönetimi, olay yönetimi ve gözlemdir.

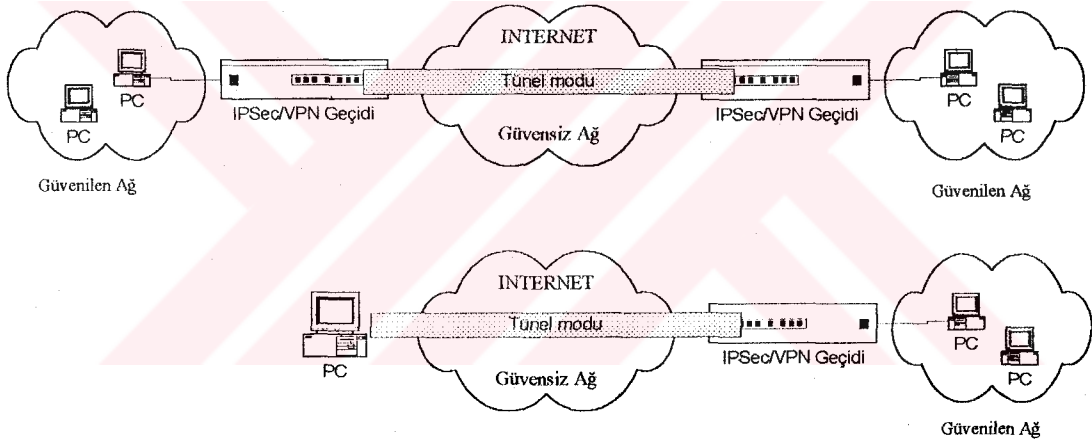
Tezin geri kalanı şu şekilde yapılandırılmıştır: Bu bölümün devamında IPsec protokolü, IPsec/VPN güvenlik geçitleri ve bu cihazların yönetim gereksinimleri açıklanmaktadır. Bölüm 2'de dağıtık ağ yönetimi mimarileri ve bu mimarileri baz alan teknolojiler anlatılmaktadır ve bu mimarilerin IPsec VPN güvenlik geçitlerinin yönetimi konusuna uygulanabilirliği tartışılmaktadır. Ayrıca IPsec/VPN geçitlerinin yönetimi konusunda şimdiye kadar yapılan çalışmalar üzerinde de yine bu bölümde durulmaktadır. Bölüm 3'de Script MIB ve kullanım alanlarından bahsedilmektedir ardından da Script MIB tabanlı dağıtık IPsec/VPN geçitleri yönetim sistemi ayrıntılı bir şekilde ele alınmaktadır. Bölüm 4'de de bu sistemle ilgili uygulamalar, sistemin avantajları, uygulamaların analizleri ve diğer çözümler ile karşılaştırmalar yer almaktadır. Sonuçlar ve öneriler kısmı da Bölüm 5'de yer almaktadır.

1.2 Ipsec Geçitleri ve Yönetim İhtiyacı

1.2.1 IPsec Protokolü ve IPsec/VPN Geçitleri

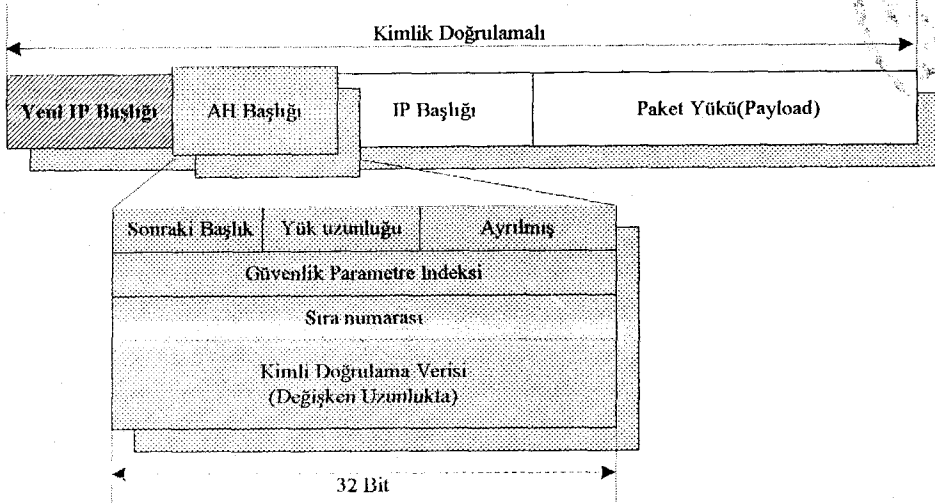
Internet protokolü IP ilk tasarlandığında o günlerde bilgisayar ağları çok sınırlı olduğundan güvenlik hesaba katılmamıştı. Büyük ihtimalle bu protokolün

tasarımcıları bu protokolün bu kadar başarılı olacağını ve bugünlerin dünya çapındaki bilgisayar ağlarının büyüklüğüne ulaşacağını hayal dahi etmemişlerdi. Fakat, IP'nin başarısını müteakip, gelişen ağlarda IP'de güvenlik meselesinin hesaba katılmayışı önemli bir problem oldu. IP'deki güvenlik problemlerini çözmek için, IETF 1998 yılında IPSec ismi altında Internet protokolü için geliştirilen güvenlik mimarisini sundu [1]. IPSec, IP ağlarında gizlilik ve kimlik doğrulama sağlayan bir protokol setinden oluşmaktadır. AH (Authentication Header - Kimlik doğrulama Başlığı) şifreleme olmaksızın kimlik doğrulama imkanı sunarken, ESP (Encapsulation Security Payload - Kaplama Güvenlik Eki) şifreleme ile gizlilik ve kimlik doğrulamayı beraber sunar. Kriptografi olduğunda anahtar yönetimi de kaçınılmaz olmaktadır. IKE (The Internet Key Exchange Protocol – Internet Anahtar Değiş-tokuş Protokolü) AH ve ESP için gerekli olan anahtar değiş-tokuş mekanizmasını gerçekleştirerek bu eksiği kapatmaktadır.



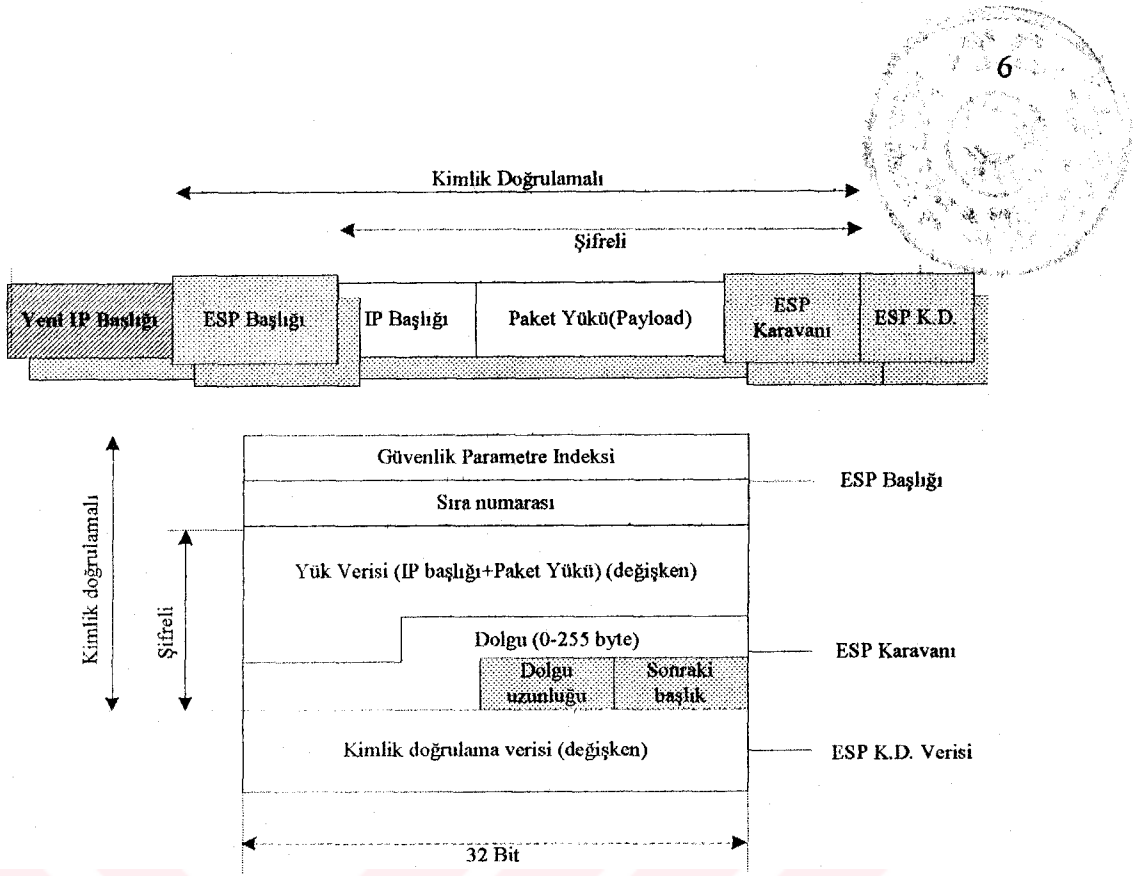
Şekil 1.1. Tünel Modu

IPSec protokolü transport ve tünel olmak üzere iki türdür. Transport mod OSI (Open System Interconnection) dördüncü seviye üstünü korurken, tünel mod üçüncü seviye ve üstünü korumaktadır. Pek çok IPSec geçidi iki modu da destekleyebilmektedir. Burada sadece tünel modu ile ilgili problemlerin üzerinde durulacaktır. Yani burada yönetim hedefi olarak alınan IPSec cihazları sadece tünel mod destekleyenlerdir. Transport mod ile ilgili problemler burada ele alınmayacaktır.



Şekil 1.2. IPsec AH protokolü paket yapısı

IPsec geçitleri üzerlerinden geçen trafiğe seçici güvenlik uygulaması yapmaktadır. Sadece seçilen paketlere geçiş izni verilmekte, geçiş izni olmayan paketler atılmakta ve seçilen paketler de spesifik güvenlik fonksiyonları (AH veya ESP) ile korunmaktadır. IP paketlerinin seçilmesi ve güvenlik kararı IPsec politikalarında tanımlanmaktadır. Bu politikalar Güvenlik Politika Veritabanında (GPV) depolanmaktadır. Bir IPsec politikası iki kısımdan oluşmaktadır: Süzgeç ve karar. Eğer süzme gerçekleşirse karar uygulanmaktadır. Bir politikadaki süzgeç bir IP paketinin başlığındaki değerlere göre süzme yapmaktadır. Mesela, eğer kaynak IP Adres = X, varış IP Adres = Y olan bir süzgeç bir politikada tanımlanmışsa, o zaman kaynak IP'si X ve varış IP'si Y olan bütün IP paketleri seçilecek ve o paketlere o politikadaki karar uygulanacaktır. Burada süzgeç kaynak ve varış IP adreslerinden oluşabildiği gibi IP paketinde bulunan üst katman protokolü, veya kaynak ve varış portu da süzgeçte etkin olabilir. Karar da söylendiği gibi durdur, geçir veya IPsec uygula şeklinde olabilir.



Şekil 1.3. IPsec ESP protokolü paket yapısı

Bu noktada eğer karar IPsec uygula ise bunun anlamı belirli bazı güvenlik parametreleri kullanılarak şifreleme veya kimlik doğrulama ile ilgili IPsec protokolünün seçilen pakete uygulanmasıdır. Politika'nın IPsec uygula kararı şifreleme veya kimlik doğrulama gibi güvenlik protokolleri, algoritmalar, güvenlik uygulamasının başlayacağı nokta ile yine güvenlik uygulamasının biteceği nokta, anahtarlar gibi bazı parametrelere sahiptir. Bu IPsec uygulama parametreleri Güvenlik Birliği (GB) adı verilen kayıtlarda tutulur ve bunlar politika kararları ile ilintilendirilirler. Güvenlik Birliğinin asıl anlamı iki uç noktanın algoritma, anahtarlar ve diğer bazı parametrelerde güvenlik fonksiyonunu uygulamak için anlaşmasıdır. Bir IPsec geçidi bir GB'yı kullanarak şifreleme ve kimlik doğrulama yaparken diğeri şifrelemeyi çözme ve kimliği tasdik etme işini yapar. Üçüncü birisi iki nokta arasında giderken korunan paketi göremez ve değiştiremez.

Daha ayrıntılı şekilde anlatmak gerekirse IPsec geçitlerinde güvenlik fonksiyonları şu şekilde çalışır: Daha önce de bahsedildiği gibi IPsec geçidi üzerinden geçen trafiğe uygulayacağı kuralı güvenlik politikalarına bakarak karar verir. Eğer karar durdur veya geçir ise paketler atılır veya müdahale edilmeden geçirilir. Eğer paketler çıkan yönde ve karar da IPsec uygula ise cihaz önce

halihazırda uygun Güvenlik Birlikleri kurulu olup olmadığına bakar. Eğer kurulu değilse IKE protokolü uzak uçtaki IPSec geçidi ile bir GB üzerinde anlaşmak için konuşur. GB'ler kurulduktan sonra IPSec geçidi GB'de belirlenen algoritma ve anahtarlar ile güvenlik fonksiyonlarını trafik üzerine uygular. Bu noktada tünel modu aktif olduğundan IPSec geçidi yeni oluşturulan IP başlığında kaynak IP adresi olarak kendi arayüzünün IP adresini, varış IP adresi olarak da GB kurulan uzaktaki cihazın arayüzünün IP adresini verir. Diğer yandan uzakuç olarak belirlenen IPSec geçidi da aynı algoritma ve anahtarlarını güvenlik fonksiyonlarında ters yönde uygulayarak paketleri orijinal haline çevirir. Tünel tekniği bütün paketi korur ve gerçek kaynak ve hedefi saldırı analizlerine karşı saklar.

AH ve ESP protokollerinin parametreleri uzantı başlıklarındaki SPI (Security Parameters Index-Güvenlik Parametre İndeksi), Sıra Numarası (tekrarlama ataklarına karşı) gibi alanlarda taşınır. Kimlik doğrulama veya şifreleme için pek çok değişik algoritma vardır. İstenen algoritmalar IPSec politikaları tarafından kullanılan GB'lerde belirlenir.

1.2.2 IPSec Geçitlerinin Yönetimi Gereksinimi

Ağ yönetimi konusunda ISO (International Standards Organization) beş adet OSI ağ yönetimi uygulama alanı tanımlamıştır. Bunlar sırasıyla hata, konfigürasyon, hesap, performans ve güvenlik yönetimidir. (İngilizce bunlar kısaca FCAPS olarak anılır ki fault, configuration, account, performance ve security kelimelerinin baş harfleri alınarak oluşturulmuş bir kısaltmadır) Bunlara kısaca değinmek gerekirse:

Hata yönetimi : Ağda herhangi bir hata oluştuğu zaman bu problemi çözmek ağ yöneticisinin sorumluluğundadır. Bu sorumluluk hatanın tespit ve izole edilmesini kapsamaktadır. Pekçok zaman ağda bulunan cihazlar hatanın düzeltilmesini otomatik olarak halledebilirler. Diğer durumlarda Ağ Yönetim Sistemi (AYS) hatayı tespit eder ve uygun alarmlar ile problemi gerekli yerlere duyurur.

Konfigürasyon yönetimi : Konfigürasyon yönetiminin amacı ağın ve sistemin konfigürasyonunu gözlemek ve gerekli düzenlemeleri ve değişiklikleri yapmaktır. Ayrıca ağdaki elemanların çalışma ve konfigürasyon durumunun güncel olmasını sağlamaktır.

Hesap yönetimi : Hesap yönetiminin amacı ağın kullanım parametrelerini

ölçerek ağın bireysel ve grup kullanımını uygun şekilde kontrol etmektir. Bu tür kontroller ağın problemlerini azaltır ve bütün kullanıcıların ağ erişimlerindeki adaleti maksimize eder.

Performans yönetimi : Performans yönetimi ağın genel performansını kontrol etmek ve performansın kabul edilebilir seviyede olmasını sürdürmek amacını güder. Bunun için AYS verileri toplar ve bu verileri işleyerek bunları ağın performansını ayarlama da kullanır.

Güvenlik yönetimi : Güvenlik yönetimi güvenlik konusunun geniş bir alanını kapsar. Bu alana ağın fiziksel güvenliği de kullanıcıların ağa olan erişimlerinin kontrol edilmesi de dahildir. Güvenlik yönetiminin amacı genel olarak ağın sabote edilmesini önlemek için ağ kaynaklarına erişimi kontrol etmek ve hassas bilgilerin yetkisiz kişiler tarafından erişilmesini önlemektir. Mesela güvenlik yönetimi yapan bir AYS bir ağ kaynağına giriş yapan kullanıcıların jurnallerini (log) gözler. Güvenlik yönetiminin diğer bir yönü de güvenlik duvarları ve kriptografi gibi güvenlik mekanizmalarının yönetimidir. [2],[3]

IPSec/VPN geçitlerinin yönetim ihtiyacından bahsetmek gerekirse, yukarıda açıklanan ağ yönetimi fonksiyonel uygulamalarının dördü bu cihazlarda da bulunmaktadır. IPSec/VPN cihazlarının önem sırasına koyulursa güvenlik yönetimi, konfigürasyon yönetimi, hata yönetimi ve performans yönetimi noktalarında ciddi gereksinimleri vardır. Bu cihaz bir güvenlik mekanizması olan VPN teknolojisini uygulaması hasebiyle güvenlik yönetimi ve paralel olarak da konfigürasyon yönetimi en önce ele alınması gereken yönetim uygulamasıdır. Diğerleri de herbir ağ cihazında ele alınması gereken yönetim uygulamaları olması cihetiyle önem arz etmektedir. Hesap yönetimi ise bazı VPN cihazlarında önem arzetmekle beraber burada ele alınmayacaktır.

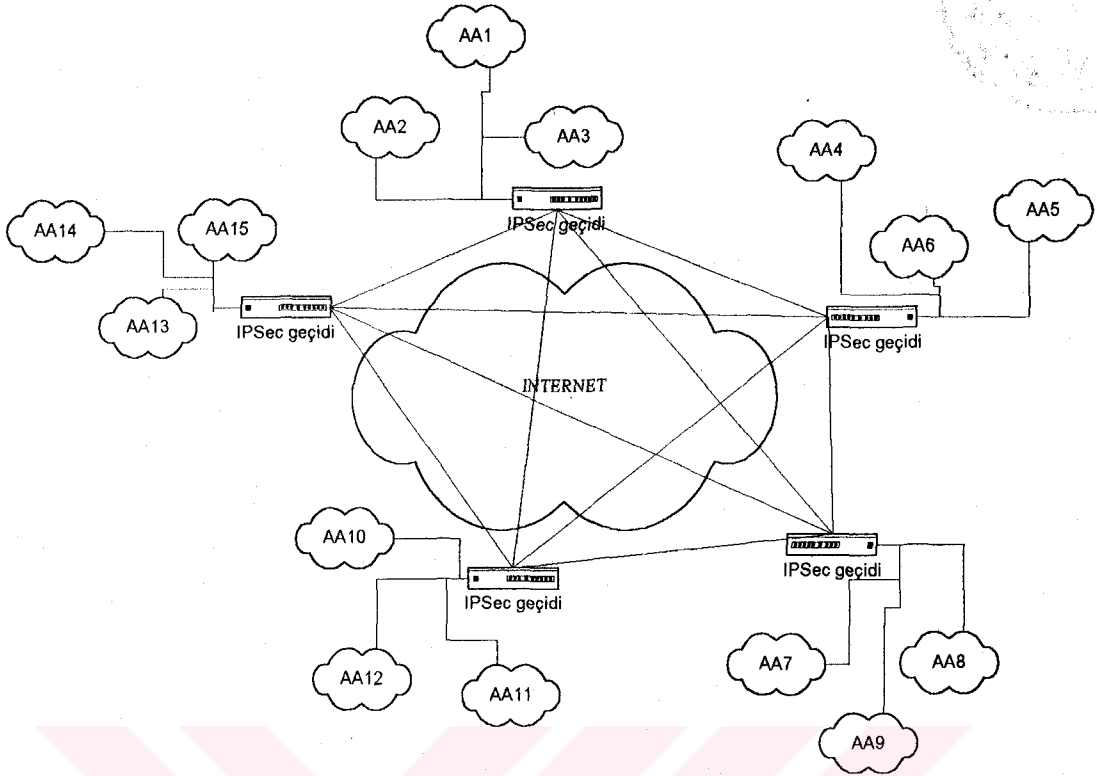
1.2.3 IPSec Geçitlerinin Güvenlik Ve Konfigürasyon Yönetimi Gereksinimleri

İki IPSec geçidi aralarında bir tünel kurarak VPN gerçekleştirebilmeleri için pek çok parametrenin ayarlanması gerekmektedir. Öncelikle kurulan tünelin manuel güvenlik birliği mi yoksa otomatik güvenlik birliği mi kullanacağına karar verilmelidir. Manuel güvenlik birlikleri önceden paylaşılmış anahtarları kullanırken,



otomatik güvenlik birlikleri IKE protokolünü kullanarak oturum başına otomatik anahtar üreterek bunları kullanır. Her manuel GB kullanıldığında iki cihazın GB veritabanlarına birbirlerini uzakuç olarak tanımlayan, aynı SPI numaraları, aynı algoritmalar ve aynı anahtarları kullanan manuel GB tanımı eklenmelidir. Otomatik GB kullanıldığında ise iki cihazın GB veritabanlarına aynı algoritmayı kullanan otomatik GB tanımı eklenmeli bununla beraber yine iki cihazın IKE veritabanlarına aynı uzakuç ve IKE parametrelerini kullanan IKE tanımı eklenmelidir. Daha sonra bu GB ve IKE tanımlarını gruplayan KH (Kripto Haritası) yine her iki cihazın Kripto Haritası veritabanına eklenmelidir. Son olarak da yine her iki cihazda politika tablolarını ayarlamaktır. Yine her iki cihazda giren ve çıkan politika veritabanlarına istenen paket süzgeçleri kullanılarak politika eklenmeli, bu politikalara “Karar” olarak “IPSec uygula” verilmeli ve eklenen Kripto Haritası referans olarak verilmelidir. Yapılan bu konfigürasyonlar her iki cihazda da uygulandıktan sonra bu iki cihaz arasında bir tünel oluşmuş olacaktır. Bu cihazlar arasında politikada belirtilen süzgeçlere uyan IP paketleri ilgili algoritmalar ile şifrelenecek veya kimlik doğrulaması yapılacak ve iki cihaz arasında güvenli olarak gidip gelecektir.

Bu noktada bu cihazların konfigürasyonu ile ilgili şöyle bir problem vardır. İki cihazın arasında oluşan bir haberleşmede IPSec ve IKE protokolleri her iki noktanın da haberleşme ve parametreleri hakkında önceden bilgisi olduğunu kabul ederek çalışır. Yani her iki nokta da bu haberleşme ile ilgili bütün gerekli konfigürasyona sahip olmalıdır. Yukarıda bahsedilen herhangi bir tünel konfigürasyonunda hata yapılırsa haberleşme gerçekleşmeyecektir. Yapılan hatalı konfigürasyon haberleşmenin değişik şekillerde yarım kalmasına sebep olacaktır. Mesela ya cihazlardan biri trafiği kriptografik işleme tabi tutacak, diğeri ise buna uyan bir konfigürasyona sahip olmadığından kendisine gelen paketi atacaktır veya paketler kriptografik işleme tabi tutulmadan atılacaktır. Ayrıca politikalarda yapılan değişik hatalar tünele girmesi gereken trafiğin açık şekilde güvensiz ağa salınmasına sebep olabilmektedir.



Şekil 1.4. IPsec geçitleri arasında örgü yapısı

IPsec geçitlerinde tünellerin kurulmasının zorluğu ile ilgili küçük bir örnek verilmesi yerinde olacaktır [40]. Şekil 1.4'teki örnekte 5 adet IPsec/VPN geçidi arkalarındaki üçer adet altağlar arasında tünel kurma görevi üstlenmiştir. Burada her bir altağın diğeri ile ayrı bir tünelden konuşması istenmektedir. Toplam oluşturulması gereken tünel sayısı düşünülecek olursa bu aşağıdaki formül ile hesaplanabilir:

$$\text{Tünel Sayısı} = N*(N-1)/2 - N \quad (1.1.)$$

Bu formül şu şekilde ortaya çıkmaktadır. Her bir altağdan diğerlerine tünel oluşması durumunda bir altağ için toplam $N-1$ adet tünel gerekmektedir. N altağ ile bu rakam çarpıldığında toplam tünel sayısı elde edilmesi gerekir. Yalnız burada tekrar edenleri çıkarmak için ikiye bölünmesi gerekmektedir. Bu durumda $N*(N-1)/2$ tünel çıkıyor. Tabii bir altağ aynı cihazın arkasında oldukları komşularına tünel kurmayacağı için altağ sayısının da bu rakamdan çıkarılması gerekiyor ki bu durumda da verilen formül ortaya çıkıyor. Örnekte bu formüle göre gereken toplam tünel sayısı: $15*14/2 - 18 = 87$ olmaktadır. Bu durumda her bir cihazda ayarlanması gereken tünel sayısı düşünülürse bu da her bir altağ için $4*3 = 12$ (Uzak cihaz sayısı*altağ sayısı) ve üç altağ olduğundan $3*12 = 36$ adettir. Yani manuel GB

kullanılma durumunda herbir cihazın veritabanına 36 adet GB tanımı, 36 adet KH, 36 adet Giren Politikaya kural ve 36 adet de Çıkan Politikaya kural eklenmelidir. Bu da toplam 144 adet konfigürasyon kaydı anlamına gelmektedir. Eğer otomatik GB kullanılacaksa bu durumda 36 adet IKE tanımı (Bu kullanıma göre değişebilir, her uzakuç için bir IKE tanımlanabilir de bu durumda rakam 4'e düşmektedir) ve kullanılmak istenen algoritma ve IPSec protokolü çeşitliliğine göre de en az 1 olmak üzere pek çok otomatik GB tanımlanabilir. Hemen her durumda da otomatik GB kullanıldığında 144 rakamına yakın bir sayıda konfigürasyon kaydı güvenlik veritabanlarına eklenmek durumundadır.

Yukarıda verilen örneğe göre manuel GB kullanarak tünel oluşturulma durumunda, bir IPSec geçidinin güvenlik veritabanlarına girilmesi gereken konfigürasyon kaydı sayısı formülize edilirse: Toplam IPSec geçidi sayısı GS olsun, herbir IPSec geçidinin arkasında eşit sayıda altağ olduğunu varsayalım ve bunların sayısı da AS olsun. Bu durumda bahsedilen şekilde her altağ arasında ayrı bir tünel oluşturulması durumunda toplam konfigürasyon kaydı sayısı TKKS:

$$TKKS = GS * AS * AS \text{ (GB için)} + GS * AS * AS \text{ (KH için)} + GS * AS * AS \text{ (Giren Politika için)} + GS * AS * AS \text{ (Çıkan Politika için)} \quad (1.2.)$$

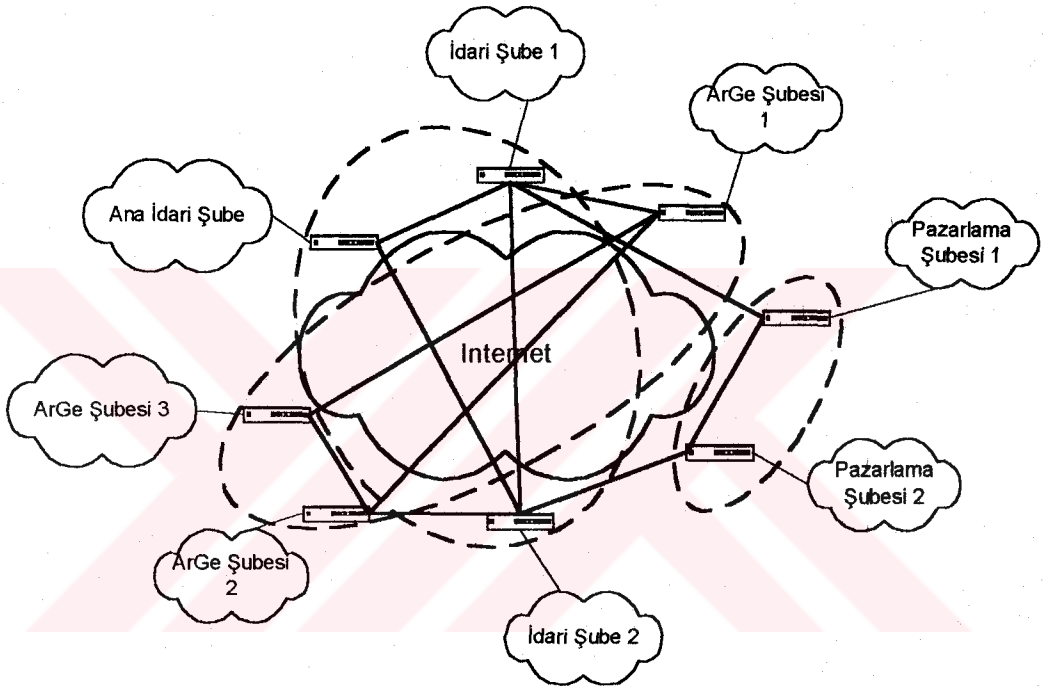
$$TKKS = 4 * GS * AS^2 \quad (1.3.)$$

Bu durumda bu örnek genişletilirse 50 şubeli ve her şubesinde 6 altağ olan bir şirket ağında $TKKS = 4 * 50 * 6^2 = 7200$ olacaktır ki başedilemez bir rakamdır. Burada verilen örnek bu işin uç noktasıdır. En uç nokta ise tünelleri altağlar arasında kurmak değil ağdaki herbir host için kurmaktır.

Pratik hayattaki uygulama her zaman verilen örnekteki kadar kompleks olmamaktadır. Örneğin son verilen örnekteki gibi büyük bir şirket ağında herkes herkesle tünel kurmak zorunda değildir hatta kurmamalıdır da. Çünkü genelde bu tür şirketlerde uygulama herkesin haberleşmesi gerektiği şubelerle bağlantısının olması şeklindedir. Mesela böyle bir şirketin ana merkezinin büyük şubelerdeki yönetim biriminin altağıyla haberleşmesi yeterli olmaktadır. O şubedeki diğer birimlerin altağlarının ana merkez şubesiyle VPN kurmasına gerek yoktur. Ayrıca mesela genelde pek çok şubenin diğerleri ile hiçbir VPN kurmasına ihtiyaç olmamaktadır. Bazen belirli bir bölgede bulunan şubeleri birbirleri ile konuşmaları yeterli olmakta bunlardan sadece biri (örneğin bu bölgenin merkez şubesi) başka bir bölgede bulunan

bir üst merkezle haberleşme ihtiyacı duymaktadır.

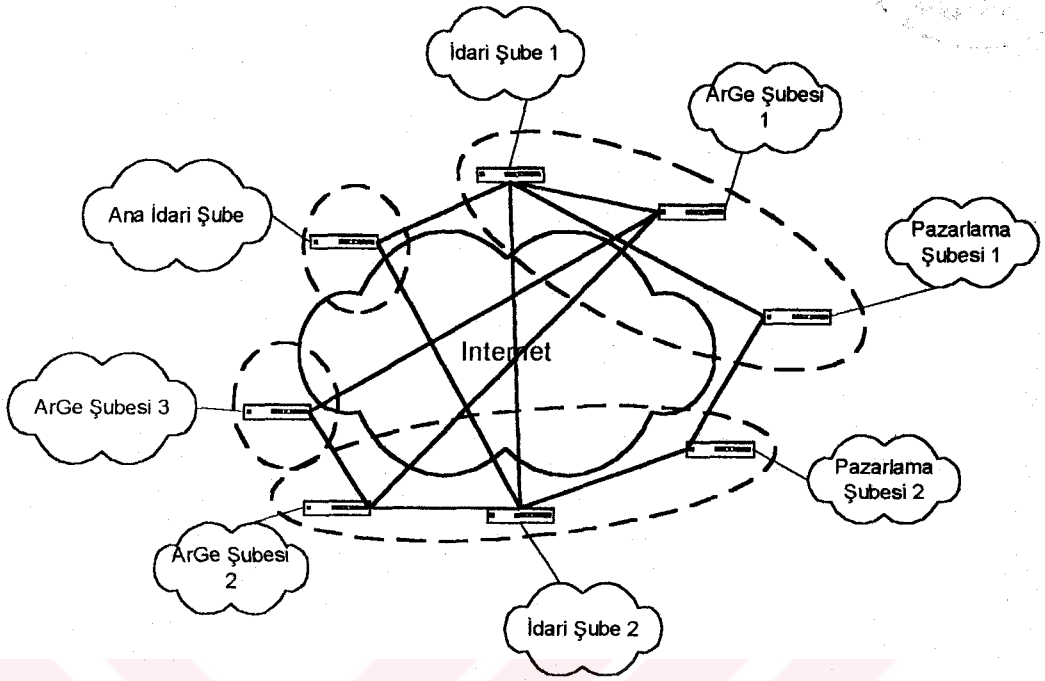
Bahsedilen şekilde bir yapı bölgeler içinde olabileceği gibi birimler arasında da olabilir. Yine farklı bir örnek vermek gerekirse dünyanın pekçok yerinde şubeleri olan bir yazılım şirketinin araştırma-geliştirme üzerine çalışan şubelerinin birbirleri ile haberleşmesi gerekirken bunların sadece bazılarının diğer şubelerle VPN kurması yeterli olmaktadır. Ayrıca aynı konu üzerine çalışan şubeler arasında kurulması gereken VPN'lerin daha ayrıntılı tünellere ihtiyacı olabilirken, şubeler arası VPN'lerde bu ihtiyaç azalmaktadır.



Şekil 1.5. IPSec geçitlerinin şube çalışma alanlarına göre gruplanması

Şimdi yukarıda verilen yazılım şirketi örneği biraz daha incelenecektir. Şekil 1.5. ve Şekil 1.6'da bu örnek, şube çalışma alanlarına göre ve bölgesel özelliklerine göre ayrı ayrı gruplanmıştır. Bu örnekte üç ayrı bölgesel şube ve bir ana idari şube bulunmaktadır. 1 ve 2 nolu bölge idari, pazarlama ve Arge şubelerinden oluşmaktayken 3 nolu bölgede sadece Arge şubesi bulunmaktadır. Ana idari şube de ayrı bir bölgede bulunmaktadır. Burada 1. ve 2. bölgelerdeki Arge şubesi ve pazarlama şubesi idari şubeyle konuşmaktadır ve birbirleriyle konuşmamaktadırlar. Ayrıca her bir şube kendi bölümünden (idari, pazarlama ve Arge) olan bütün şubelerle konuşmaktadır. Şekil 1.5.'te şubeleri Internet'e bağlayan IPSec geçitleri bölümlere göre gruplandırılmıştır. Şekil 1.6'da ise bu IPSec geçitleri bölgelerine göre

gruplandırılmıştır.



Şekil 1.6. IPsec geçitlerinin şube bölgesel özelliklerine göre gruplanması

Bu noktada verilen örnekte her iki gruplandırmanın ne kadar etkili olduğunun incelenmesi gerekmektedir. Burada önce kriter belirlenmesi gerekmektedir. Burada genel kriter grup içinde kalan bağlantı sayısı ile gruplar arası (Şekilde grup çizgilerini kesen ve grup içinde bir geçitte sonlanan çizgi sayısı) çizgilerin sayısıdır.

Önce Şekil 1.5.'teki gruplandırma incelenecektir. Burada üç adet grup bulunmaktadır. İdare bölümünün grubu içinde 3 adet grup içi bağlantı ve 4 adet de bu gruptan başka gruptaki cihazlara olan bağlantı bulunmaktadır. Arge bölümünün grubu içinde 3 adet grup içi bağlantı ve 2 adet de bu gruptan başka gruptaki cihazlara olan bağlantı içermektedir. Pazarlama bölümünün grubu içinde de 1 adet grup içi bağlantı ve 2 adet de bu gruptan başka gruptaki cihazlara olan bağlantı bulunmaktadır.

Şekil 1.6'daki gruplandırma incelenirse de şunlar görülmektedir. Burada dört adet grup bulunmaktadır. 1. bölgedeki grubun içinde 2 adet grup içi bağlantı ve 5 adet de bu gruptan başka gruptaki cihazlara olan bağlantı bulunmaktadır. 2. bölgedeki grubun içinde de aynı şekilde 2 adet grup içi bağlantı ve 5 adet de bu gruptan başka gruptaki cihazlara olan bağlantı içermektedir. 3. bölgedeki grubun içinde tek geçit olduğundan 0 adet grup içi bağlantı ve 2 adet de bu gruptan başka

gruptaki cihazlara olan bağlantı bulunmaktadır. Ana idari şubenin bulunduğu grup da aynı şekilde 0 adet grup içi bağlantı ve 2 adet de bu gruptan başka gruptaki cihazlara olan bağlantı bulundurmaktadır.

Burdaki problemde gruplandırmanın en önemli esprisi grup içi bağlantıların en fazla olması ve gruplar arası bağlantının da mümkün olduğunca az olmasıdır. Bu formülize edilirse gruplandırmanın efektifliği GE:

GİBS : Grup içi bağlantı sayısı

GABS : Gruplar arası bağlantı sayısı

$$GE = GİBS / GABS \quad (1.4.)$$

Verilen örnekte de bölümlere göre gruplandırmada toplam grup içi bağlantı sayısı 7 ve toplam gruplar arası bağlantı sayısı da 8'dir. Bölgelere göre gruplandırmada ise toplam grup içi bağlantı sayısı 4 ve toplam gruplar arası bağlantı sayısı da 14'dir. (1.4)'teki formül burada uygulandığında bölümlere göre gruplandırmanın GE'si $7/8 = 0,875$ iken bölgelere göre gruplandırmanın GE'si de $4/14 = 0,285$ çıkmaktadır. Bu sonuca göre örnekte bölümlere göre gruplandırma bölgelere göre gruplandırmanın neredeyse üç katı GE'ye sahip olmakla daha kullanışlı olmaktadır.

Verilen örnekteki gruplandırmada bölümlere göre sınıflandırmanın daha iyi olması bunun her zaman böyle olduğunu göstermemektedir. Burada aralarında VPN kurulacak ağların arasında nasıl bir haberleşme politikası olduğuna bağlıdır ve tamamen kurumun ağ ve iş yapısına bağlıdır. Burada gösterilmek istenen şudur ki Şekil 1.4.'teki örgü yapısı, yani herkesin bir diğeri ile konuşması pratikte genelde olmayan birşeydir ve zaten kullanışlı da değildir. Bu yüzden pratik hayattaki uygulamalar göz önüne alındığında IPSec geçitleri bir gruplamaya tabi tutulabilir. Bu gruplama bölüme göre, bölgeye göre veya bir başka kritere göre yapılabilir. Fakat her nasıl yapılırsa yapılsın efektif bir gruplama yapılmalıdır ve gruplamanın yönetim açısından efektif olup olmadığının hesaplanmasının teorisi de yukarıda gösterilmiştir. Tezin konusu olan IPSec geçitlerinin yönetimi konusundaki çözümümüz de burada anlatılan gruplama teorisine dayanmaktadır.

Bu noktada herbir cihazın güvenlik ve konfigürasyon yönetimini düşünüldüğünde bunun herbir cihazda manuel olarak yapılamayacağı aşikardır.

Oysaki günümüzde pratik hayata bakılırsa pekçok ağ yöneticisi ve operatörün bu işleri cihazların seri arayüzünden lokal bağlantı ile veya uzaktan SSH (Secure Shell) bağlantısı kullanarak komut satırı arayüzüyle (CLI-Command Line Interface) yapıldığı görülmektedir. Oysa ki bu hataya çok açık ve etkisi düşük bir yönetimdir. Ayrıca bu çözüm dinamik yönetim gereksinimi olan yerlerde çok yetersizdir. Bazı IPsec geçit cihazlarının üretici tarafından sağlanan merkezi yönetim araçları olsa da, bunlar da genelde dinamik güvenlik ve konfigürasyon yönetiminde yetersiz olmaktadır.

1.2.4 IPsec Geçitlerinin Olay Yönetimi Gereksinimleri

Ağ yönetiminin en önemli fonksiyonlarından birisi hata yönetimidir. Hata yönetimi ise genelde yönetimde alttan tetiklenen bir süreçtir. Başta SNMP çatısı (framework) da dahil olmak üzere ağ yönetiminde kullanılan hemen bütün çatılar kendine has bir bildiri (notification) veya başka bir tabirle olay (event) mekanizmasına sahiptir. Bu noktada yönetilen ağ elemanlarından gelen olayların yönetim tarafında toplanması ve değerlendirilip operatöre iletilmesi gereklidir.

IPsec geçit cihazları öncelikle bir ağ elemanı ve de cihaz olması nedeniyle pekçok olay üretebilmektedirler. Klasik anlamda bütün ağ cihazlarında üretilen ve SNMP standardı olan coldstart, warmstart ve authenticationFailure gibi olayların yönetim sisteminde ele dikkate alınması gereken olayların başında gelmektedir. Ayrıca mesela bir IPsec cihazında bir arayüzün arızalanması veya işlemcinin fanının durması gibi donanım olayları sistem yöneticisi için çok şey ifade etmektedir. Ayrıca cihaza lokal bağlantı ve yapılabildiği konfigürasyon değişiklikleri yapılabileceğinden bu tür durumlarda yine ilgili bildiriler ile yönetim sisteminin haberdar edilmesi gerekmektedir.

Yukarıda bahsedilen durumların dışında üreticinin gerçekleştirmesine bağlı olan ve çok değişik sahalarda, özellikle IPsec protokolünün pekçok kritik noktasında cihaz üzerinde olaylar tanımlanabilir. Bu noktada IETF'de draft olarak geliştirilen IPsec ile ilgili SNMP MIB'leri gerekli pekçok olayı tanımlamıştır. Bunlardan IETF'de draft seviyesinde olan ipsec-flow-monitoring-mib'de [5] 15 adet bildiri, yine draft seviyesinde olan ipsec-ike-monitor-mib'de [7] 2 adet bildiri, yine draft seviyesinde olan ipsec-monitor-mib'de [4] 8 adet bildiri ve yine draft seviyesinde

olan ipsp-ipsec-conf-mib'de [6] 2 adet bildiri tanımlanmıştır. Bunlardan örnek vermek gerekirse; ikeProtocolFailure IPSec faz-1 ike tüneli için işlem protokolle ilgili bir hata ile karşılaşırsa protokol yığını tarafından oluşturulup yönetim sistemine yollanan bir olaydır. Ayrıca bunların dışında gerçekleştirmeye özel olarak pekçok olay tanımlanabilir mesela güvenlik politikasındaki her bir kurala bir olay kurulabilir ve o kural çalıştığında ilgili bildiri yönetim sistemine gönderilir.

IPSec geçitleri güvenlik cihazları olması nedeniyle bu cihazlardan gelen hemen her türlü olay ehemmiyetlidir ve yönetim sistemi tarafından mutlaka dikkate alınması gerekir. Bu noktada yönetim sisteminin ağdaki bütün cihazlardan gelen bütün olaylar ile tek tek ilgilenmesi gerekmektedir. Ayrıca bu olaylar belirlenen bazı kurallara göre birbirleriyle ilişkilendirilebilmeli, süzulebilmeli ve bazı durumlarda gereken aksiyon yapılmalıdır. Burada önemli nokta şudur ki önceden çözümü belirlenebilen bazı durumlar için mümkün olabildiğince operatörün müdahalesine gerek kalmadan sistem problemi çözebilmelidir.

Yukarıda bahsedilen olay yönetimi gereksinimleri de cihazda bariz şekilde manuel yapılması çok zor bir aktivitedir. Bu noktada yüzlerle ifade edilebilecek IPSec geçidinden gelecek olayların yönetilip gerekli müdahalelerin otomatik olarak yapılması IPSec geçitlerini yöneten bir yönetim sisteminin olması gereken en önemli özelliklerinden birisidir.

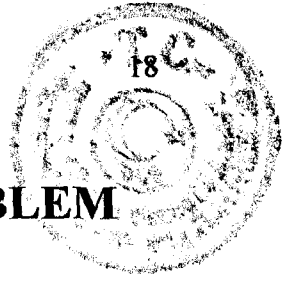
1.2.5 IPSec Geçitlerinin Gözlenme Gereksinimleri

Ağ yönetiminde ağın ve ağdaki elemanların gözlenmesi de çok büyük önem arz etmektedir. Özellikle IPSec geçitleri gibi güvenlik cihazlarının durumu operatör tarafından dikkatle gözlenmelidir. Örneğin cihazlara ulaşılamaması durumunda bu problemin acilen operatöre bildirilmesi gerekmektedir. Cihaza ulaşılamaması çok değişik nedenlerden olabilir. Bu nedenlerin en önemlisi cihazın arızalanması olabilir ki bunun haricinde hatalı konfigürasyon gibi durumlar da olabilir.

IPSec geçitlerinde gözlenmesi gereken diğer hususlar da hemen her protokol seviyesindeki sayaçlar ile ilgilidir. Bu sayaçlar cihazın durumu hakkında pekçok bilgi içermektedirler. IP, TCP, UDP, Arayüz ve SNMP ile ilgili pekçok sayacı içeren standart MIB II ve bunu sonradan genişleten Arayüz MIB'i gibi bazı standart

MIB'ler hemen her ağ cihazında bulunur ve gözlenip analiz edildiğinde pekçok bilgiyi ortaya çıkarırlar. Ayrıca IPSec protokolü ile ilgili IPSec ve IKE gözlem MIB'leri [4] standart hale gelmemiş olmasına rağmen pekçok ticari üründe gerçekleştirilmektedir. Bu MIB'lerde bulunan sayaçlar gözlendiğinde geçidin yazılım problemleri, konfigürasyon problemleri, cihaza yapılan saldırıları ve diğer cihazlarla uyumluluk konularında pekçok bilgi ortaya çıkarabilmektedir.





2 AĞ YÖNETİMİ MİMARİLERİ VE PROBLEM İLE İLGİLİ ÇALIŞMALAR

2.1 Ağ Yönetimi Mimarileri

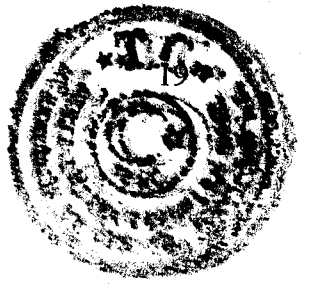
Telekomünikasyon ve veri ağlarının ilk zamanlarında ağların yönetiminin gerekliliği ortaya çıkınca IETF ve ISO yönetim için birer çatı (framework) önerdiler. IETF tarafından önerilen SNMP çatısıyken ISO tarafından önerilen ise OSI yönetim modeliydi [8][9]. SNMPv1 uzun zaman veri ağlarının yönetiminde geniş çapta kullanılırken OSI yönetim modeli aynı başarıyı veri ağlarında gösteremedi. Fakat OSI yönetim modeli telekomünikasyon dünyasında özellikle TMN (Telecommunications Management Network) olarak adapte edildikten sonra her yerde yaygınlaştı.

Yönetim fonksiyonlarının dağıtımı düşünüldüğünde, SNMPv1 tamamen merkezi yönetim mimarisine dayanmaktaydı. Fakat veri ağlarının gelişmesiyle SNMP mimarisi ciddi ölçeklenebilirlik problemleri ile karşı karşıya kaldı. Bununla birlikte, büyük ve gelişmiş ağlar yönetim açısından daha fazla otomasyona ihtiyaç duymaya başladı. Bu sıralarda ağ yönetimi alanında dağıtık hesaplama yöntemlerinin kullanımının gerekliliği daha aşikar şekilde ortaya çıktı. Dağıtıklık seviyesi düşünüldüğünde dağıtık yönetim dört kategoriye bölünebilir: Merkezi yönetim, zayıf dağıtık hiyerarşik yönetim, güçlü dağıtık hiyerarşik yönetim ve güçlü dağıtık kooperatif yönetim. Bu sınıflandırma ilk olarak Flatin [10] tarafından ileri sürüldü ve akademik dünyada dağıtık yönetim paradigmalarının anlaşılması noktasında çok faydalı olarak kabul edildi.

Bu sınıflandırma şu şekilde formülize edilmektedir: M ağdaki yöneticilerin toplam sayısını (en üst seviye ve orta seviye yöneticilerin hepsi) belirtsin ve N de ağda bulunan bütün düğümlerin sayısını (yöneticiler ve yönetilenler birlikte) belirtsin. Bu durumda bu dört sınıfın ayrıtılması şu kriter aracılığıyla yapılabilir:

$M = 1 \Rightarrow$ merkezi yönetim;

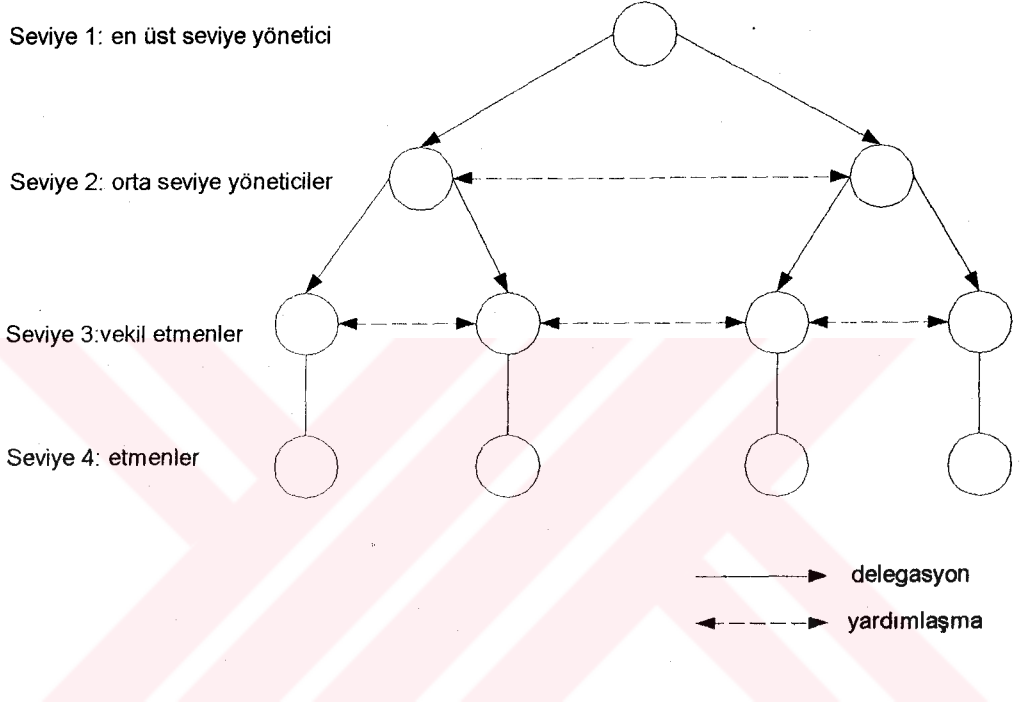
$1 < M \ll N \Rightarrow$ zayıf dağıtık hiyerarşik yönetim;



$1 \ll M < N \Rightarrow$ güçlü dağıtık hiyerarşik yönetim;

$M \approx N \Rightarrow$ kooperatif yönetim. [11]

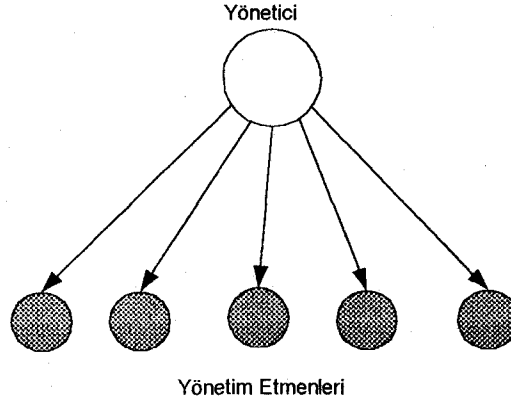
Şekil 2.1 yukarıda bahsedilen yönetim modellerini kabaca göstermektedir. Bu şekilde dikey çizgiler delegasyonu gösterirken yatay çizgiler de yardımlaşmayı göstermektedir. Bu dört paradigma devam eden bölümlerde ayrıntılı bir biçimde açıklandığında bu şekil çok daha manalı olacaktır.



Şekil 2.1. Delegasyon ve yardımlaşma

2.1.1 Merkezi Yönetim

Merkezi yönetimde, bir yönetici yönetim etmenleri ile haberleşerek bütün yönetim işlerini yürütür. Yönetim etmenleri çok fazla özelliği olmayan veri toplayıcıları rolündedirler. SNMP (Simple Network Management Protocol) tabanlı ve HTTP (Hyper-text Transfer Protocol) tabanlı yönetim bu türün tipik iki örneğidir.



Şekil 2.2. Merkezi Yönetim

2.1.1.1 SNMP Tabanlı Yönetim

IP ağlarının neredeyse başlangıcından bu yana, ağlar ve sistemler SNMP ile yönetildiler. Bu yönetim mimarisi yönetim alanında oldukça başarılı oldu. Fakat SNMP mimarisinin en büyük problemi ölçeklenebilirlikti. Doksanlı yıllarda, ağların büyüklüğü ve ağdaki cihazların karmaşıklığı çok ciddi şekilde arttı. Buna paralel olarak SNMP MIB'lerinin büyüklüğü de çok ciddi arttı. SNMP ilk olarak tasarlandığında yönetimin dağıtılması IETF için bir öncelik olmadığından, SNMP merkezi yönetim mimarisini kullandı ve SNMP bu mimari ile yönetim ihtiyaçlarını karşılama noktasında zamanla yetersiz hale geldi.

SNMP'nin ikinci versiyonunda yöneticiler arası haberleşme tanımlanmasına rağmen dağıtık ağ yönetiminde yetersizliği dolayısıyla başarılı olamamıştır. SNMP'nin ikinci versiyonunda tanımlanan dağıtık ağ yönetimi özellikleri zayıf-dağıtık hiyerarşik yönetim kısmında daha sonra bahsedilecektir. SNMP'nin en son versiyonu olan versiyon 3 ise tamamen versiyon ikinin mimarisinin üzerine güvenlik getirmeye odaklanmıştır ve tanımlanan herhangi bir dağıtık yönetim fonksiyonu yoktur.

2.1.1.2 HTTP (Hyper-Text Transfer Protocol) Tabanlı Yönetim

90'lı yılların ortalarında ucuz ve kullanımı kolay olan web teknolojilerinin hızlı gelişiminden dolayı bazı çevreler web teknolojilerinin varolan mimariler için en iyi alternatif olduğunu iddia ettiler. HTTP tabanlı yönetimde bütün haberleşme HTTP

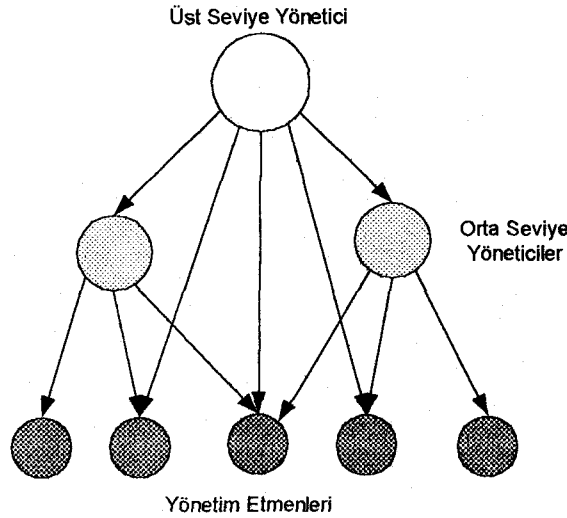
üzerinden gerçekleşmektedir. Bu durum yönetim etmenlerinin web server bulundurmalarını ve yönetimin de bir web gezgini bulundurmasını gerektirmektedir. Yönetici etmenden web sayfasını alır ve bunu gösterir. Etmenler ise iki türlü web sayfası bulundurur: Statik ve dinamik. Statik sayfalar zaman içinde değişmez ve etmen tarafından kaydedilirler. Dinamik sayfalar ise yönetim tarafından gelen isteğe göre dinamik olarak oluşturulurlar ve etmenin belirli bir andaki durumunu yansıtırlar. Statik sayfalar için tipik bir örnek konfigürasyon yönetimi içerikli web sayfası ve dinamik sayfalar için tipik bir örnek de performans yönetimi içerikli web sayfasıdır.

Java teknolojisi bu teknolojiye bir alternatif olarak düşünülebilir. Bu durumda, yönetim gezgininde bir applet çalışır ve HTTP etmen ile yönetim arasındaki haberleşmede kullanılır. XML (Extensible Markup Language) veya String'lerin içine kodlanan yönetim verisi etmen tarafından yukarıya itilebilir veya yönetim tarafından yukarıya çekilebilir [37].

Yönetimin dağıtılması noktasından karşılaştırıldığında bu yönetimin klasik SNMP yönetimi mimarisinden çok farkı yoktur. Genel manada protokol ve tarz farkı vardır.

2.1.2 Zayıf Dağıtık Hiyerarşik Ağ Yönetimi

Zayıf dağıtık yönetim paradigmlarında uygulana muhtelif yerlere yayılmıştır. Telekomünikasyon alanında bu paradigma uzun yıllar TMN tarafından kullanılırken IP alanında sadece RMON (Remote Monitoring) ile küçük bir teşebbüs başarılı oldu. SNMP'nin bazı versiyonlarında bu paradigma denendi fakat istenen başarıya ulaşamadı.



Şekil 2.3. Zayıf dağıtık hiyerarşik yönetim

2.1.2.1 Telekomünikasyon Alanında

TMN çatısında, yönetim hiyerarşik yönetici ağacı içerisinde dağıtılmıştır. Herbir orta seviye yönetici bir yönetim alanından sorumludur ve üst seviye yönetici (yöneticiler yöneticisi) bu yöneticilerin üstündedir ve onları kontrol eder. Eğer bir orta seviye yönetici ile üst seviye yönetici arasındaki haberleşme koparsa, orta seviye yönetici kendi alanındaki gerekli düzeltici işleri yapma yetisine sahiptir.

2.1.2.2 IP Alanında

SNMP dünyasında bazı zayıf dağıtık yönetim çözümleri teklif edildi. Bunlardan birisi SNMPv2 çatısıyla gelen inform mesajıydı [13]. Bu yöneticiler arası haberleşmede kullanılması amaçlanan bir protokol ayrıntısıydı. Fakat bu mimari dağıtık yönetim işlerini yerine getirme konusunda başarılı olamadı ve kullanım noktasında çalışamaz olduğu ispatlandı.

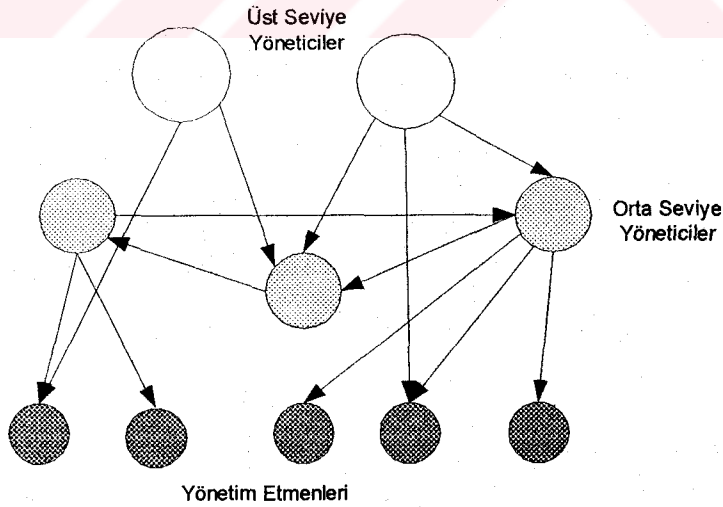
IETF tarafından yapılan zayıf dağıtık yönetim paradigmasının başarılı bir tanımlaması etmenleri daha akıllı hale getiren RMON idi. RMON MIB ilk olarak 1991 yılında ortaya çıktı ve 1995 yılında yenilendi. RMON etmenleri bazı yönetim fonksiyonlarını başarı ile yerine getirdiler. RMON yeteneği olan cihazlarda kullanım istatistiklerini toplayarak yönetim merkezlerine verilerin taşınması ve işlenmesi ağır yükü ortadan kalktı. Bu bazı basit ağ yönetim işlerinin etmenlere delege edilmesiydi. Fakat bu işler statiktir ve zaman içinde değiştirilemez. Bazı basit mekanizmalar kullanılarak ölçüm sayaçları ve bildiriler (trap) RMON MIB ile bağlanır ve yönetim

ile etmen arasında bağlantı kopsa dahi istatistiklerin toplanması işi devam eder. Günümüzde RMON'un IP dünyasında kullanımı yaygın şekilde devam etmektedir.

2.1.3 Güçlü Dağıtık Hiyerarşik Ağ Yönetimi

Zayıf dağıtık yönetim paradigmaları merkezi paradigmaların pek çok problemini, özellikle de ölçeklenebilirlik problemini çözdüler. Fakat zayıf dağıtık yönetim için hala bazı sınırlamalar devam etmektedir. Genel olarak bunlar zindelik (robustness), esneklik ve uygulanabilirliktir. Zindeliği ele alırsak yönetici ile etmen arasında bağlantı koptuğu zaman etmen gerekli düzeltici eylemleri yürütememektedir. Bununla birlikte, zayıf dağıtık yönetim paradigmalarında delege edilen yönetim fonksiyonları statik olduğundan dinamik olarak yeni vazifelerin atanması mümkün olamamaktadır ki bu da bir esneklik problemi olduğunu göstermektedir. Uygulanabilirlik yönüyle de yönetim istasyonlarına odaklanmak daha güçlü yönetim istasyonlarını gerektirmektedir ve ayrıca çoğu zaman hata noktasının bir tane olması uygulanabilir değildir [10].

Bu problemler ile başedebilmek için yöneticiler ile etmenler arasındaki güçlü dağıtım ilk olarak Goldszmidt tarafından Delegasyon ile Yönetim (Management by Delegation –MbD) mimarisinde ortaya atıldı [14].



Şekil 2.4. Güçlü dağıtık hiyerarşik yönetim

Bu çalışmada Goldszmidt, ağlar büyüdükçe aptal veri toplayıcı etmenler ve merkezi yönetim mimarisi yönetim vazifelerini başaramadığı için yönetimin dağıtılmasının gerekliliğinin arttığını gösterdi. Burada fikir basit fakat zekiceydi.

Bütün etmenler yönetim uygulamasında aktif ve ilgili olmalıydı. Elastik sunucuları içeren etmenler yönetim tarafından gönderilip delege edilen programı alabilir ve yürütebilirdi. Delegasyon ile Yönetim fikrini takiben pekçok araştırma yürütüldü ve pekçok teknoloji de gelişti.

Güçlü hiyerarşik yönetim paradigması iki tipte ele alınabilir: Gezgin kod ve dağıtık nesneler.

2.1.3.1 Gezgin Kod

Gezgin kod paradigmalarının hepsi basit bir fikri paylaşmaktadır. Yönetici gerektiği zaman dinamik olarak programları bunları yürütebilecek etmenlere transfer eder. Bu programlar iki sınıfa bölünebilir. İlki zayıf gezginlik olarak tanımlanabilir. Bu türde programlar sadece yürütülebilir kod bulundurmaktadırlar. Diğer yandan güçlü gezginlik olarak tanımlanabilen ikinci türde transfer edilen bu programlar yürütülebilir kod ve yürütme durumunu beraber bulundurmaktadırlar. Bu programlar diğer bir makineye transfer edilince yürütmeye kaldıkları yerden devam edebilirler.

Zayıf gezginlikte, gezgin kodun transferinin tetiklenmesi açısından iki tip bulunur. Bunlar çekme ve itme tipleridir. İtme tipinde etmenler spesifik bir iş gerçekleştirmesi gerektiğinde yönetim gerekli kodu etmene yollar. Bunun tam tersi şekilde, çekme tipinde kod bir etmende gerektiği zaman etmen yönetici ile bağlantı kurar ve kodu yöneticiden indirir. Bunlardan başka güçlü gezginliğe sahip olan gezgin etmenler gerektiği zaman bir makinede çalışıp daha sonra bir başkasına göç etme ve orada çalışmasına kaldığı yerden devam etme yetisine sahiptir [15]. Doğasında buna uygun olan Java teknolojisi gezgin kod paradigmasının gerçekleştirmelerinde yaygın şekilde kullanılmaktadır.

Goldszmidt and Yemini'nin Delegasyon ile Yönetimi uzaktan tarih planlaması ve delege edilmiş etmenlerin eklenmesi ile bir itme modelidir [14]. IETF alanında yönetim işlerinin dinamik olarak delege edilmesi DISMAN (DIStributed MANagement) çalışma grubu tarafından ele alınmıştır. 1999 yılında bu çalışma grubu birkaç MIB tanımladı. Bunlardan en belirgin olanı Script MIB'di. Script MIB 2001 yılında standart hale geldi [16]. Schönwälder Script MIB ve uygulamalarını çalışmasında detaylı şekilde anlatır [11]. Script MIB yönetim fonksiyonlarının delege edilmesi için taban sağlayan bir SNMP MIB'idir. SNMP bugünün IP ağlarında hemen her yerde bulunduğu ve hemen her cihaz tarafından desteklendiğinden,

bir Script MIB gerçekleştirmesi ve bir script ortamı SNMP etmenlerine eklenerek bu tip bir delegasyonun kullanımı kolay hale gelmektedir.

Script MIB kullanılarak pekçok yönetim fonksiyonu dağıtık şekilde delege edilebilir. Script MIB başlıca üç tipte yönetim uygulamasında kullanılabilir. İlkinde, Script MIB etmenlerde kullanılabilir. Bu durumda bazı yönetim işleri, yönetilen birimin kaynaklarına lokal erişimi olan etmenlere scriptler halinde gönderilerek delege edilebilir. Bunlar bazı istatistiksel veri toplama ve hesaplama olabilir. Fakat bütün servisler scriptler halinde gerçekleştirilemez çünkü bazı servisler çok karmaşıktır ve bazı ciddi kurulum sürecine ihtiyaç duyar. İkinci tipte, scriptler orta seviye yöneticilerde zayıf ve güçlü dağıtık yönetimde kullanılır. Script MIB tasarımının ana amaçlarından birisi de yönetim işlerini scriptler halinde orta seviye yöneticilere gönderme fikri ile orta seviye yöneticilerin bulunduğu dağıtık yönetim sistemlerinde kullanım idi. Orta seviye yöneticilerde, Script MIB muhtelif amaçlar için kullanılabilir. Yalnız bunların dördü diğerlerine kullanılabilirlik yönüyle baskın çıkmaktadır ki bunlar dağıtık gözlem için scriptler, servis testi için scriptler, servis yönetimi ve kontrolü için scriptler ve hata kotarımı için scriptlerdir. Script MIB'in üçüncü uygulanım tipi ise Kooperatif Yönetim'de kullanımıdır. Fakat bu tip kullanım Script MIB'in bunu desteklemekte gerekli kabiliyetlerinin tam olmaması yönüyle çok sınırlıdır [11]. Script MIB ile ilgili daha çok şey söylenebilir. Fakat bu noktada Script MIB ile ilgili ayrıntılar 3. bölüme havale edilmektedir.

2.1.3.2 Dağıtık Nesneler

Bu ikinci tip güçlü dağıtık hiyerarşik yönetim paradigmaları dağıtık nesne teknolojilerini temel alınarak geliştirilmiştir. Bu çerçevede pekçok yaklaşım ele alınabilir fakat burada sadece ikisi incelenecektir; CORBA (Common Object Request Broker Architecture) ve Java RMI (Remote Method Invocation).

CORBA dağıtık uygulamaların birbirleri arasında uyumluluk problemleri yüzünden OMG (Object Management Group) tarafından standart hale getirilen bir yapıdır. CORBA özellikle telekomünikasyon dünyasında çok başarılı oldu. Bununla birlikte, IP dünyasında SNMP nesneleri kolayca nesnelere uyarlanabildiğinden kısmen başarılı oldu yalnız telekomünikasyon dünyasındaki ile kıyaslanabilecek seviyede değil.

Java RMI kullanarak yönetim uygulamaları dağıtık nesne yönelimli

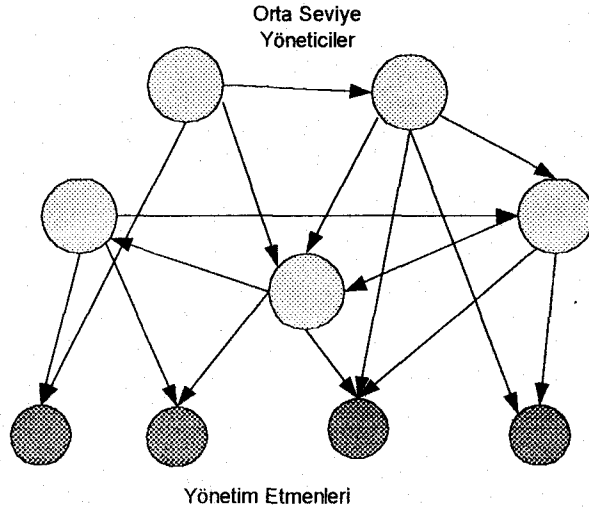
uygulamalar olarak programlanabilir. Bu durumda herşey nesnedir ve bütün nesneler birbirleri ile haberleşebilir. Ayrıca nesnelerin dizi haline getirilmesi (serialization) kullanılarak nesnelerin durumları da transfer edilebilir. Bununla birlikte dağıtık nesneler SNMP yönetim nesnelerine uyarlanabilir.

Dağıtık nesnelerin ağ yönetiminde kullanımı ile ilgili örnek vermek gerekirse Barotto, Souza and Wesphall'ın çalışması ele alınabilir [17]. Bu çalışmada bir grup yüksek performanslı istasyon CORBA kullanılarak yönetilmektedir. Yönetim nesneleri yönetilen elemanların içine yerleştirilmiştir ve ilgili bir yönetim nesnesi de bunlar ile CORBA aracılığıyla haberleşmektedir. Bu çalışmada herhangi bir hiyerarşi bulunmaması dolayısı ile güçlü dağıtık hiyerarşik yönetime örnek olmasa da yine de CORBA'nın bu alanda kullanımı ile ilgili bir fikir vermektedir.

Java tabanlı yönetim de güçlü dağıtık yönetim uygulamalarının geliştirilmesinde etkilidir. 1999 yılında FMA (Federated Management Architecture) Sun Microsystems tarafından ortaya çıkarıldı. FMA bütün etmenlerin Jini (Jini Sun Microsystems'in tak-kullan çözümüdür) kabiliyetli olmasını gerektirmektedir. FMA Java RMI'ı yönetim uygulamalarına genişleterek statik ve dinamik servisler tanımlar. Java tabanlı çözümlerde bütün etmenlerin Java desteklemesi zorunludur [10].

2.1.4 Güçlü Dağıtık Kooperatif Ağ Yönetimi

Hiyerarşik dağıtık yönetim paradigmalarında etmen ve orta seviye yönetici “niçin” bu işi yaptığını bilmez, sadece “nasıl”ı bilir. “nasıl” bilgisi etmenlere ve orta seviye yöneticilere gönderilirken yöneticiler niçin'i kendilerine saklar. Fakat kooperatif paradigmalarda yönetici etmene “niçin” içeren amacı yollarken etmenin nasıl'ı bilmesi beklenir. Bu kavram ele alınırsa, kooperatif paradigmalardaki etmenler akıllı olarak isimlendirilirler.



Şekil 2.5. Güçlü dağıtık kooperatif yönetim

Yakın zamanlara kadar kooperatif teknolojiler ağ yönetiminde kullanılmıyordu. Kökenler Çoklu-Etmen Sistemlerinden (Multi-Agent Systems) gelmektedir. Genelde akıllı etmenler dört özelliğe sahip olmak zorundadır. Bunlar özerklik, sosyal kabiliyet, tepkin (reaktif) ve proaktifliktir [18].

Kooperatif etmenler ile ilgili bir uygulama örneği vermek gerekirse Boudaoud, Goukia ve Guessoum'un çalışması güvenlik yönetimi konusunda bu yaklaşımı aktif şekilde kullanmaktadır [19]. Bu çalışmada bir güvenlik politikası yönetici etmeni intranet yönetici etmenleri ile haberleşen bir ekstranet yönetici etmenini kontrol etmektedir. Herbir intranet yönetici etmeni kendi alanında lokal etmenlere sahiptir. Bir alandaki intranet yönetici etmenleri ve lokal etmenlerin hepsi birbirleri arasında yardımlaşma için haberleşirler. Buradaki akıllı etmenler üç fonksiyonu icra ederler ki bunlar ağın güvenlik yönetimi işlerini gerçekleştirmek için süzme, etkileşim ve düşünmektir.

2.1.5 Ağ Yönetimi Mimarilerinin Ipsec Geçitleri Yönetimi İçin İncelenmesi

Bu bölümde bu kısma kadar mevcut ağ yönetimi mimarilerini tek tek incelendi. Burada asıl problem olan IPSec geçitlerinin yönetimi için mevcut ağ yönetimi mimarilerinin ele alınması gerekmektedir. Bahsedilen dört mimari de kendine has pekçok özelliği içermektedir. Bu noktada bu özelliklerden IPSec geçitlerinin yönetim ihtiyacına en fazla örtüşen mimari seçilecektir. IPSec geçitlerinin yönetiminde en

önemli hususlar ölçeklenebilirlik, tek hata noktası olmama, çok fazla ağ trafiği oluşmaması, yönetimin mümkün olduğunca otomatikleşmesi, esneklik ve çok kompleks olmamasıdır. Çizelge 2.1'de IPSec geçitlerinin yönetim gereksinim özellikleri ile yukarıda bahsedilen dört mimari kıyaslanmıştır. Burada şimdi herbir mimari tek tek ele alınacaktır.

İlk olarak merkezi yönetimi ele alınacaktır. Merkezi yönetimin olumlu bir yanı çok kompleks olmamasıdır. Bilindiği üzere dağıtık sistemler yetenekli olmalarına paralel olarak bir o kadar da komplekstirler. Esneklik noktasında ise yönetim bir noktada olduğundan bir derece esnek olabileceği düşünülebilir. Yani yönetim fonksiyonları merkezde değişiklik yapılabilir bir yapı varsa zaman içinde değiştirilebilir. Merkezi yönetimin negatif yönleri ele alınırsa daha önce bahsedilen en önemli problem ölçeklenebilirlik problemidir. Tek bir noktadaki yönetici yönetimi gereken düğüm sayısı arttıkça yetersiz hale gelmekte ve önemli pekçok yönetim faaliyeti yürütülemez hale gelmektedir. Burada bir diğer husus da merkezi yönetimin bütün ağı tek noktadan kontrol etmesi dolayısıyla yönetim trafiğinin bütün ağı boydan boya geçmesi gerekmektedir ki bu da gereksiz bir trafiğe sebep olur. Ayrıca merkezi yönetim açık şekilde tek hata noktası oluşturmakta ve yönetim merkezi ile haberleşme koptuğunda yönetilen bütün etmenler tek başlarına kalmaktadırlar. Yönetimin otomasyonu noktası ele alındığında merkezi yönetimde bu tür mekanizmalar kurulsa bile, yönetilecek düğüm sayısı arttığında bu mekanizmalar ile tek noktadan yönetim pek mümkün değildir. Bütün bu olumsuzluklar ele alındığında merkezi yönetim IPSec geçitlerinin yönetiminde yetersiz kalmaktadır.

Zayıf dağıtık hiyerarşik yönetim ikinci olarak ele alınırsa bunun merkez yönetimin pekçok problemini çözdüğü görülmektedir. Bu mimaride ölçeklenebilirlik büyük ölçüde çözülmüş durumdadır. Yine yönetim trafiği düşünüldüğünde çok optimize olmasa da merkezi yönetimden daha avantajlı görülmektedir. Yine tek hata noktası konusu burada kısmen çözülmüştür. Fakat orta seviye yöneticiler çok kabiliyetli olmadıklarından üst seviye yönetici ile bağlantı koptuğunda pekçok yönetim işi ortada kalmaktadır. Yönetimin otomasyonu da diğer noktalarda olduğu gibi merkezi yönetimden bir adım ileridedir. Fakat zayıf dağıtık hiyerarşik yönetimde orta seviye yöneticilere atanan işler statik olduğundan esneklik düşüktür. Kompleksliği ele alırsak da bahsedildiği üzere dağıtıklık seviyesi arttıkça bu da artmaktadır ki bu yüzden bu mimari merkezi yönetime göre daha komplekstir. Zayıf

dağıtık hiyerarşik yönetim bahsedilen bu yönleri ile ele alınırsa IPsec geçitlerinin yönetimine daha iyi bir aday olarak görünmektedir. Yalnız burada şunu da belirtmek gerekir ki zayıf dağıtık hiyerarşik yönetim için güçlü bir altyapı bulunmamaktadır. IPsec geçitleri IP alanında oldukları için telekomünikasyon dünyasının çözümleri burada kullanılabilir değildir. IP dünyasında da bu mimaride başarılı olmuş sadece RMON vardır ki IPsec cihazlarının yönetimi sadece gözlemden oluşmaması dolayısıyla RMON yeterli değildir. SNMPv2'nin getirdiği çözümler de bahsedildiği üzere yetersizdir. [18]

Üçüncü olarak ele alınması gereken mimari güçlü dağıtık hiyerarşik mimaridir. Bu mimari zayıf dağıtık hiyerarşik yönetimin hemen bütün olumlu yanlarını bulundurmakla beraber onun zayıf olduğu noktaların pekçoğunda da güçlüdür. Mesela gezgin kod ve dağıtık nesneler esneklik konusunda pekçok problemi çözmekle beraber orta seviye yöneticilere kendi başlarına pekçok işi yapabilme yeteneği sağlayarak tek hata noktası meselesinde de daha fazla yetkinlik oluşturmaktadır. Ayrıca yönetim fonksiyonlarının otomatikleştirilmesi noktasında da bu mimari yeterli görünmektedir. Karmaşıklık meselesine gelirsek de bu mimarinin öncekilerden daha kompleks olduğu görünmektedir [10]. Bütün bu noktalardan ele alındığında IPsec geçitlerinin yönetimi konusunda bu mimari öncekilerden daha iyi bir aday gibi görünmektedir.

Son olarak güçlü dağıtık kooperatif yönetimi ele alınacaktır. Bu mimari diğerlerinden doğa itibari ile çok farklıdır. Güçlü yönlerini ele alırsak diğer mimariler arasında otomasyon ve ölçeklenebilirlik yönünden en güçlüsü budur. Ayrıca esneklik yönünden de diğerlerinden ileridedir. Pekçok karar etmenlerde verildiği için trafik de çok fazla değildir. Yalnız bu sistemlerin global görüş ve global değerlendirme noktalarında eksikleri vardır ve komplekstirler. Ayrıca yönetim etmenlerinin de akıllı olması gerektiğinden yönetilen cihazlara da çok fazla yük düşmektedir [18]. Bu mimarinin amacı yönetim yükünü olabildiğince dağıtmaktır. Yalnız bu her zaman istenen bir şey değildir. Ayrıca bu tür bir mimari kullanıldığında bu mimariye uygun özel etmenlerin de yönetilen cihazlara konması gerekmektedir ki bu da her zaman istenen ve olası bir şey değildir. Bütün bu yönler ele alındığında bu mimari çok yetenekli olmasına rağmen IPsec geçitlerinin yönetimi için iyi bir aday olarak görünmemektedir.

Çizelge 2.1. Ağ yönetimi mimarilerinin karşılaştırılması

Özellikler	Merkezi Yönetim	Zayıf dağıtık hiyerarşik yön.	Güçlü dağıtık hiyerarşik yön.	Güçlü dağıtık kooperatif yön.
Ölçeklenebilirlik	Kötü	Orta	İyi	Çok iyi
Esneklik	Orta	Orta	Çok iyi	Çok iyi
Trafik yoğunluğu	Kötü	Orta	İyi	Çok iyi
Delegasyon seviyesi	Kötü	Orta	İyi	Çok iyi
Tek hata noktası olmama	Kötü	Orta	İyi	Çok iyi
Otomasyon	Kötü	Orta	İyi	Çok İyi
Karmaşıklık	Çok İyi	İyi	Orta	Kötü
Gerçekleştirilebilirlik	Çok İyi	İyi	Orta	Kötü

Güçlü dağıtık hiyerarşik yönetim mimarisinde kullanılan teknolojileri ele alırsak en çok göze çarpanlar daha önce de bahsedildiği üzere Java teknolojileri, CORBA ve Script MIB'dir. Java çözümleri özellikle Jini genelde bütün ortamların Java yetenekli olmasını gerektirmesi yönüyle IPsec geçitlerinin yönetiminde şu an için uygulanması zor görünen bir teknolojidir. Fakat yalnız yönetimler arası haberleşme ve dağıtım Java ile yapılır orta seviye yöneticiler etmenler ile SNMP gibi daha standart protokoller ile konuşursa bu çözüm uygun görünmektedir. CORBA da IPsec geçitlerine CORBA konuşabilen etmenler konulmadan aynı şekilde kullanılırsa olabilir görünse de CORBA kompleksliği yönüyle dezavantajlıdır. Son olarak Script MIB'i ele alırsak Script MIB her türlü script ortamını desteklemesi, yönetim scriptlerinin kontrolü için standart SNMP protokolünü kullanılması, yine orta seviye yöneticinin SNMP yetenekli olması dolayısıyla kullanılan script ortamının yönetilen cihazlar ile kolayca SNMP üzerinden haberleşme yapabilmesi ve kompleks olmaması gibi yetenekleri dolayısıyla çok avantajlı durmaktadır. SNMPv3 kullanıldığında SNMP yeterince güvenli olduğundan bu teknoloji IPsec geçitlerinin yönetimi için en iyi aday olarak görünmektedir.

Bu çalışmada IPsec/VPN geçitlerinin yönetimi için bu mimarilerden güçlü dağıtık hiyerarşik yönetim ve teknoloji olarak da SNMP Script MIB seçilmiş ve bu altyapıyı kullanan bir yönetim sistemi geliştirilmiştir. İlerleyen bölümlerde bu konu daha ayrıntılı olarak işlenmiştir

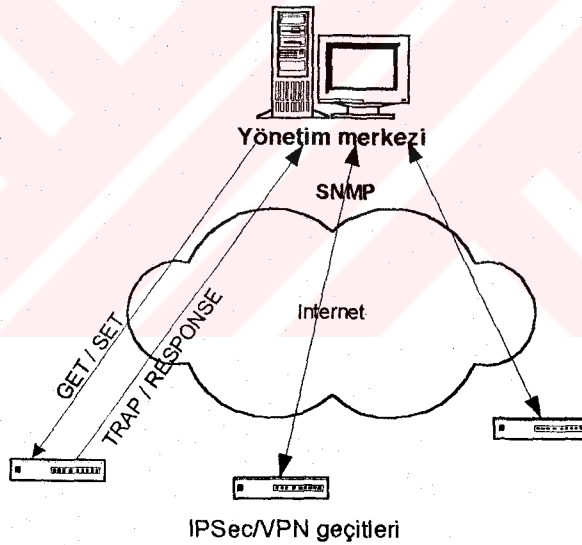


2.2 Konuyla İlgili Çalışmalar

Bu bölümde şimdiye kadar IPSec/VPN geçitlerinin yönetimi ile ilgili yapılan bazı çalışmalar özetlenecektir.

2.2.1 Merkezi Yönetim

IPSec/VPN geçitlerinin yönetiminde şimdiye kadar en çok uygulanan yönetim tarzıdır. Özellikle endüstride pekçok VPN geçit cihazı belirli bir gelişmişlik seviyesindeki politika konfigürasyon ve yönetim araçlarını beraberinde bulundurmaktadır. Mesela Cisco'nun Security Policy Manager'ı [20], Indus River'ın Policy Vision4'u ve Xedia'nın VPN Manager'ı [36] buna örnektir. Bu tip klasik VPN yönetim araçlarında politikalar merkezi olarak tanımlanır ve otomatik olarak ilgili cihazlara dağıtılır.



Şekil 2.6. IPSec geçitlerinin merkezi yönetimi

Pekçok IPSec/VPN geçitleri yönetim sistemleri merkezi yönetimi SNMP'yi kullanarak gerçekleştirmektedir. SNMP bilindiği üzere IP dünyasının standart yönetim mimarisidir. Bu noktada SNMP'nin IPSec geçitlerinin yönetimindeki kullanımı klasik SNMP yönetiminden çok farklı değildir. Yönetici GET ve SET mesajlarını etmenlere gönderir ve etmenlerden TRAP'ları alır. Şekil 2.'de IPSec geçitlerinin merkezi yönetimi gösterilmiştir. Herbir IPSec geçidinde SNMP etmeni önceden tanımlanmış MIB'ler ile çalışmaktadır. Halihazırda IETF tarafından bazı



IPSec için hazırlanmış bazı konfigürasyon ve gözlem MIB'leri bulunmaktadır. Bunlar IPSec politika konfigürasyonu için hazırlanmış "IPSec Policy Configuration MIB" [6], IPSec güvenlik birlikleri için düşük seviye gözlem ve durum nesnelerini tanımlayan "IPSec Monitoring MIB"[4], IPSec bağlantılarının gözlenimi, akımı ve düzeltimi için yüksek seviye nesnelerini tanımlayan "IPSec Flow Monitoring MIB" [5] ve IKE protokolü IPSec güvenlik birliklerinin oluşturulmasında kullanılma durumu için gözleme ve durum nesnelerini tanımlayan "Internet Key Exchange (IKE) Monitoring MIB" [7]'dir. Bu MIB'lerin hepsi hala draft seviyesindedir ve standart hale gelmemiştir.

Klasik SNMP merkezi yönetiminde tek bir alanda bir yönetici gerekli güvenlik politikalarını ve GB tanımlarını IPSec geçitlerine dağıtır. Dağıtım, ilgili MIB politika nesnelerine politika bilgisi içeren SET isteklerinin gönderimi tarzında gerçekleşir. Bu senaryoda güvenlik politikaları sadece yönetici tarafından yollanır. Yönetici bu işlemi tetikleyebilir veya etmen bunu yöneticiden güvenlik politikasının alınması gerektiğinde bir trap mesajı göndererek başlatabilir.

SNMP çatısında bahsedildiği üzere yönetim genelde merkezidir. Dağıtık yönetimin SNMP ile kullanılma durumunda bu yöneticiler arası SNMP mesajları ile gerçekleşebilir. Bu durumda güvenlik politikası ve yönetim parametreleri herbiri bir alanı kontrol eden yöneticiler arasında karşılıklı değiş tokuş edilebilir. IPSec geçitlerinin yönetiminde SNMP halihazırda kullanılmasına rağmen bu konuda pek fazla akademik çalışma bulunmamaktadır. Bu yönetim dünyasının IPSec geçitleri gibi kompleks cihazlar için daha kabiliyetli yönetim teknikleri araması gerçeğinin sonucu olabilir.

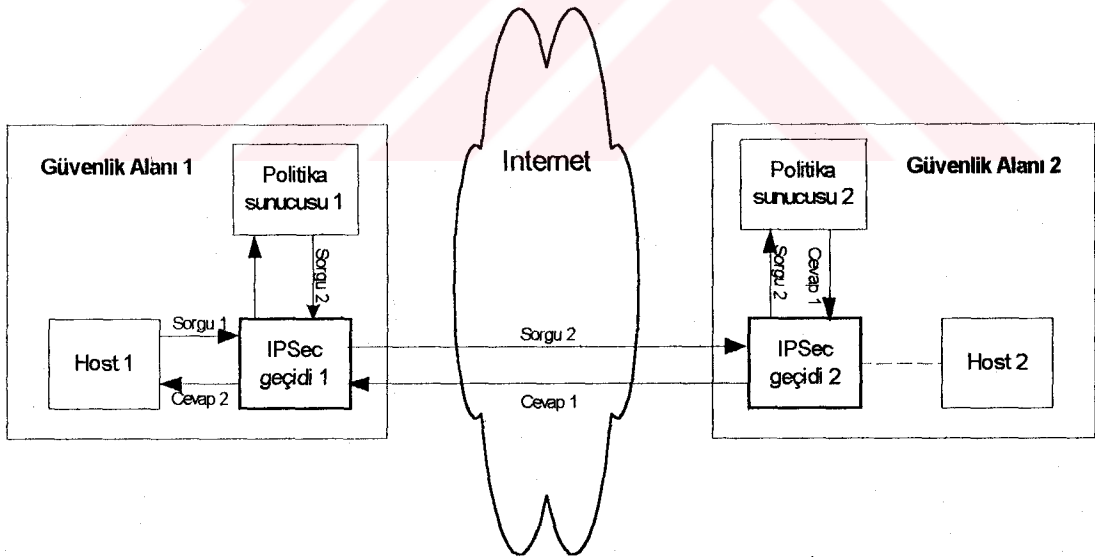
IPSec geçitlerine benzer cihazlar olan güvenlik duvarı cihazlarının merkezi yönetimi ile ilgili bazı çalışmalar vardır. Bunlardan birinde [30] güvenlik duvarı cihazlarının güvenli ve esnek yönetimi için bir global yönetici tasarlanmış ve gerçekleştirilmiştir. SNMP kullanılarak yapılan yönetim sisteminde bir güvenlik duvarı MIB'ı geliştirilmiştir. Bu çalışmada bahsedilen sistemin gerekleri ve gerçekleştirmesi ile ilgili ayrıntılar da verilmiştir. Yönetim sistemi tamamen klasik bir SNMP tabanlı merkezi yönetim yapısını taşımaktadır. Bir yönetici üzerinde SNMP etmeni taşıyan güvenlik duvarı cihazlarını SNMP ile yönetmektedir. Yönetim haberleşmesinin güvenliğinde SNMPv3 yerine SNMP/IPSec kullanılmıştır. Yönetim tarafında yöneticiye birden fazla istemci bağlanarak yöneticiyi kullanabilmektedir.

Burada istemci-sunucu yapısını gerçekleştirmek için de Web tabanlı bir sistem kullanılmıştır.

IPSec geçitlerinin merkezi yönetiminde de bahsedilen güvenlik duvarı yönetim sistemi ile hemen hemen aynı yapı kullanılmaktadır. Bu tür sistemler, geliştirilmeleri kolay olduğu için genellikle daha çok tercih edilmektedir. Fakat birinci bölümde açıklandığı üzere cihaz sayısı artınca merkezi yönetim pekçok yönden yetersiz kalmaktadır.

2.2.2 SPP (Security Policy Protocol)

IPSec geçitlerinin güvenlik yönetimi konusunda eksikliğin giderilmesine yönelik IETF tarafından SPS (Security Policy System-Güvenlik Politikası Sistemi) adı altında bir sistem geliştirildi. SPP'de bu sistem için geliştirilen protokoldür [21]. SPS'nin mimarisi tamamen dağıtık bir mimaridir. Bu sistemde farklı alanları kontrol eden politika sunucuları çalışmaktadırlar. Bir güvenlik alanındaki düğümlerin ve ağların güvenlik politikası bilgisinin bulunması, erişilmesi ve işlenmesi için gerekli mekanizmaların sağlanması SPS'in ana hedefidir. Herbir güvenlik alanı, alanın



Şekil 2.7. SPS politika sorgusu

kaynaklarını ve bunlara erişim için gerekli politikaları tanımlayan güvenlik politikası veritabanını bulunduran bir Politika Sunucusunu barındırmaktadır. Politika Sunucuları SPP kullanarak Politika istemcileri ile konuşurlar. Bu istemciler genelde IPSec geçitleridir. Bu protokol kullanılarak düğümlere diğer düğümlerle aralarında

haberleşme yolu oluşması için, IPSec geçitleri ile GB kurulmasında gerekli politika bilgisini sağlar. Seçiciler ve GB'ler (gerekliyse) içeren gerekli politikalar haberleşmeyi tamamlamak için transfer edilir [22].

SPS politikaların bulunması için de bir sorgu mekanizması geliştirmiştir. Şekil 2. bu sorgu yapısını göstermektedir. Politika sorguları politika istemcileri tarafından bir alandaki politika sunucusuna verilir. Politika sunucusu sorgulanan politikalar ile alanındaki politikaları karşılaştırır ve gerekirse diğer politika sunucuları ile haberleşir ve en sonunda uygun politikalar ile cevap verir. SPS gerekli protokol mesaj değişim formatı ve süreci tanımlamıştır.

SPS ile ilgili belirgin iki akademik çalışma vardır. İlki SPS'in dağıtık yapısına odaklanır ve SPS'in sınırlamalarını tartışır. Bu çalışmada SPS'e bir iyileştirme olarak IPSec/VPN servisleri için merkezi global yönetim sunucusu kullanan yeni bir politika tabanlı hibrit yönetim mimarisi önerilmektedir. Burada ana odak SPS'in global politika eksikliğinin ve buna bağlı olarak da güvenlik politikalarının yetersizlik, doğrulama, dekorelasyon ve çözümleme problemlerinin olmasıdır. Güvenlik politika yöneticisinin görevi global politikaları uygulamak ve muhtelif fonksiyonlarla lokal politika sunucularının yönetimini sağlamaktır. Bu çalışma önerilen sistemin prototip tasarımı ve gerçekleştirmesini de içermektedir [23].

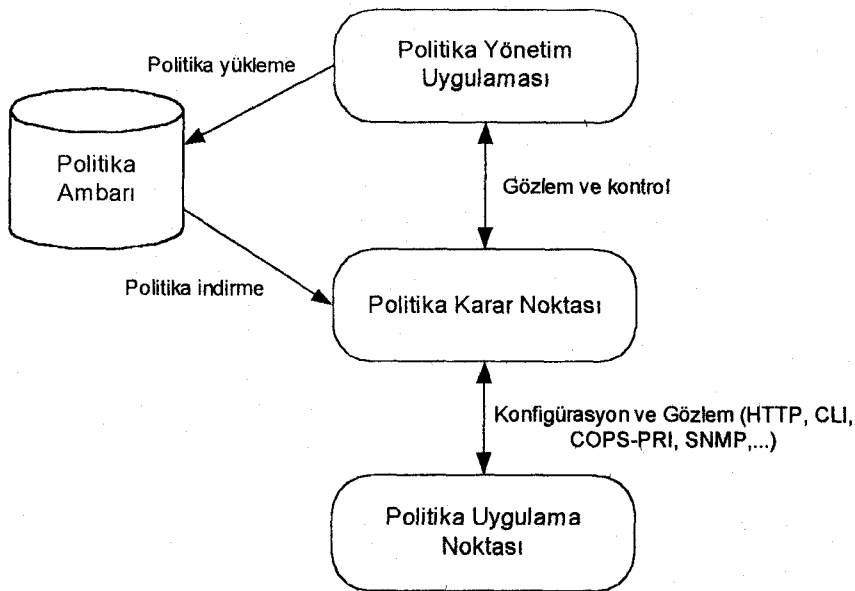
İkinci çalışma SPS ile ilgili değişik bir yaklaşıma sahiptir. Bu çalışmada IPSec geçitlerinin güvenlik yönetimi (özellikle güvenlik politikası yönetimi) için SNMP, COPS-PR (Common Open Policy Service for Policy Provisioning) ve LDAP (Lightweight Directory Access Protocol) gibi ağ yönetimi bilgisinin dağıtımında kullanılan klasik mekanizmalar ele alınmıştır [24]. SPS'in amaçlarına erişmesi noktasından bu mekanizmaların özellikleri ve uygunlukları detaylı şekilde tartışılmıştır. Daha sonra da bir SPS'i gerçekleştirmede takip edilen bir yaklaşım ve SPP açıklanmış ve tartışılmıştır.

SPS çok alanlı sistemlerin güvenlik politikalarının yönetiminde güçlü bir yapıya sahiptir. Fakat kolayca farkedilen bazı yetersizlikleri vardır. Öncelikle tamamen dağıtık bir yapıya sahip olmasının getirdiği bazı problemler vardır. Global bir politika kavramının olmayışı bunun sonucudur. Bu probleme yukarıda ilk bahsedilen çalışmada da [23] değinilmiştir. Herbir politika sunucusu kendi alanındaki politikalardan haberdardır ve sadece sorgu ile diğer politikalardan

haberdar olabilmektedir. Bu yaklaşım verimli değildir. Ayrıca politikaların tutarlılık kontrolü yoktur ve politikalar dağıtık olduğundan alanlar arası politikaların tutarlılıklarının kontrolü de olanaksız görünmektedir. Bunun yanısıra bu sistem IPSec geçitlerinin yönetimi konusunu bütün olarak ele almamakta ve sadece güvenlik politikalarının dağıtımını konusuna odaklanmaktadır. Bu haliyle SPS tek başına yeterli olmamakta ve ayrıca yönetim mekanizmalarına ihtiyaç duymaktadır. Son problem olarak da, SPS henüz olgunlaşmış bir yapıya sahip değildir. En son müsvedde (draft) üç yıl önce çıkmıştır ve ondan beri standartlaşma noktasında bir ilerleme olmamıştır. Standartlaşıp oturmamış bir yapının pratik olarak kullanımında değişik problemler de beraberinde gelebilmektedir.

2.2.3 Politika Tabanlı Ağ Yönetimi

Politika tabanlı ağ yönetimi metodolojisi IETF tarafından entegre servisler modeli kapsamında geliştirildi [25]. Bu ağ kaynaklarının rezervasyonunun kabul kontrolünün yönetimi için bir politika çatısında kullanılmak üzere önerilmişti. Daha sonradan bu yaklaşımın spesifik bir servis modelinden bağımsız olduğu ve değişik başka yerlerde de benzer şekilde uygulanabileceği anlaşıldı. Mesela bu IPSec'in uygulanmasında ve Ayrım Servisleri (Differentiated Services [26]) ortamında kolayca kullanılabilirdi. Bu sistem bunlardan başka daha genel yönetim problemlerinde de çok kullanışlıdır.



Şekil 2.8. IETF politika tabanlı ağ yönetimi mimarisi

IETF'nin henüz üzerinde anlaşılmış spesifik bir politika çatısı bulunmamaktadır. Buna rağmen pekçok temel kavramı üzerinde anlaşılmıştır. Bunlar politika yönetim uygulaması, PDP (Policy Decision Point - politika karar noktası), PEP (Policy Enforcement Point - politika uygulama noktası) ve politika ambarıdır (repository). Bunlar arasındaki ilişki Şekil 2.8'de gösterilmiştir. Bunlar tamamen politika tabanlı ağ yönetimi ile ilgili genel kavramlardır. Yoksa IETF'nin politika tabanlı ağ yönetimi konusundaki çalışmaları genelde gerçekleştirme ile ilgili dağıtım, platform, protokol veya dil gibi detayları içermemektedir.

Politika yönetim uygulamaları ağ operatörüne politikaları oluşturmak ve konuşlandırmak için gerekli arayüzü sağlar. Ayrıca politikaları politika ambarına depolar ve yönetilen ortamın durumunu gözler. Olası politika çatışmalarının tespiti için gerekli bazı basit kontroller bu uygulama tarafından gerçekleştirilir. Politika yönetim uygulaması ile politika karar noktası arasında herhangi bir protokol kullanılabilir. IETF'nin bu noktada herhangi bir tavsiyesi yoktur.

Politika kararları politika karar noktasında verilir. PDP politikaların indirilmesi ve yorumlanması gibi işleri gerçekleştirir. Ayrıca politika çatışmalarını kontrol eder, politika uygulama noktalarının isteklerini alır ve onlara ilgili kararlar ile cevap verir. Politika kurallarının değerlendirilmesinin tetiklenmesi değişik mekanizmalar ile gerçekleştirilebilir. IETF politika çatısı bu konuda açık birşey belirtmemektedir. Tetiklemeler özel sistem veya bileşen istekleriyle, olaylarla ve yoklama ile gerçekleştirilebilir.

PDP ağ eleman nitelikleriyle ilgili boolean ifadelerden oluşan politika durumlarına göre politika kararlarını verir. Eğer bir durum doğru olarak hesaplanmışsa ilgili faaliyet yürütülür. Bu faaliyetler genelde politikayı uygulayan hedef elemanların tekrar konfigüre edilmesi şeklindedir. Gerekli durumlarda PDP politika kurallarını PEP'in anlayabileceği daha spesifik parametrelere dönüştürür.

Politika kararlarının esas uygulandığı yerler politika uygulama noktalarıdır. Bunlar ağ elemanlarını bulunduran hedef birimlerdir. PEP bir karar durumu gerçekleştiğinde yürütülen politika faaliyetinin hedefi konumundadır. PEP, PDP ve politika ambarı fiziksel olarak aynı yerde olabilir. Buradaki ayrım tamamen fonksiyonallığa bağlı lojik bir ayrımdır. Bu yapıda her bir PDP birden fazla PEP'i kontrol edebilir. Yalnız, bir PEP sadece bir PDP tarafından kontrol edilebilir. Politika

kararının transferi için herhangi bir protokol kullanılabilir. Fakat IETF'nin tavsiyesi COPS/COPS-PR üzerinedir. Tabii HTTP, FTP (File Transfer Protocol) veya SNMP de kullanılabilir.

Politika ambarından bahsetmek gerekirse kısaca şu söylenebilir. Politika ambarı politika yönetim uygulaması tarafından da politikaların saklandığı ve PDP'ler tarafından politikanın alındığı bir veritabanıdır. Bu veritabanına erişim bir ambar erişim protokolleri kullanılarak gerçekleştirilir. IETF'nin bu noktada tavsiyesi LDAP'tır.

Politikanın uygulanabilirliği ona bir rolün tahsis edilmesi ile belirginleşir. Rol kavramı bütün politika çatısının tasarımında merkezidir. Rol, bir veya daha çok politikanın seçiminde kullanılan bir çeşit niteliktir. Bir politika idarecisi her bir kaynağa bir veya daha fazla rol atar ve her bir rol için politikaları belirler. Politika çatısı bu noktada bir rolle ilgili her bir kaynağın konfigürasyonundan sorumludur [29].

Politika tabanlı yönetim fikir olarak gerçekten çok iyidir. Ayrıca IPSec geçitlerinin yönetiminde de doğrudan uygulanabilir. Bu noktada politika tabanlı ağ yönetiminin ana dezavantajı Internet dünyasının üzerinde tam anlaştığı ve belirginleştirdiği genel bir politika çatısının olmamasıdır [27]. Ayrıca IETF tarafından politika dağıtımında kullanılması için geliştirilen COPS protokolü endüstride kabul görmemiştir ve sonuç olarak yakın zamanda IETF COPS'u ve politika tabanlı yönetimi ilerletme faaliyetlerini durdurmuştur. Bu konuda bir diğer husus da bu tür bir sistemin tam manasıyla gerçekleştirilmesinin zor ve kompleks olmasıdır. Ayrıca politika çatışmaları da üzerinde durulması gereken zor bir problemdir [28]. Özellikle bu yaklaşım insan faktörünü arka plana attığı için ve ağ cihazlarının bütün konfigürasyonuna politikalar çerçevesinde kendisi karar verdiği için bazı hataların oluşması muhtemeldir ki bu tür hatalar ciddi problemlere sebep olabilmektedir. IPSec geçitleri gibi güvenlik cihazları özellikle konfigürasyonunda çok dikkat gerektiren ve hata durumunda telafisi zor olan cihazlardır. Bu yüzden pek çok ağ idarecisi ve operatörler politika tabanlı yönetimi kullanmakta isteksizdirler.

3 IPSEC GEÇİTLERİ İÇİN SCRIPT MIB TABANLI DAĞITIK YÖNETİM SİSTEMİ

3.1 DISMAN Teknolojileri – Script MIB

DISMAN IETF altında kurulan ve DIStributed MANagement kelimelerinin kısaltmasından oluşan bir çalışma grubudur. Bu grup SNMP çatısı altında dağıtık ağ yönetimi teknolojileri geliştirmek amacı ile kurulmuştur. Şimdiye kadar bu grup Script MIB, Schedule MIB, Notification Log MIB gibi bazı MIB'leri geliştirmiştir. Burada Script MIB ele alınacaktır.

3.1.1 Script MIB

Script MIB [16] yönetim fonksiyonlarının delegasyonu için SNMP uyumlu bir yönetim bilgi tabanı (Management Information Base) tanımlar. Spesifik olarak bahsetmek gerekirse Script MIB şu yetenekleri sağlar:

- Yönetim scriptlerinin dağıtık yöneticilere transferi,
- scriptlerin buralarda başlatılması, askıya alınması, çalışmasına devam edilmesi ve çalışmasının iptal edilmesi,
- yönetim scriptleri için argümanların transferi,
- çalışan yönetim scriptlerinin gözlenmesi ve kontrolü,
- yönetim scriptlerinin çalışmasından üretilen sonuçların transferi

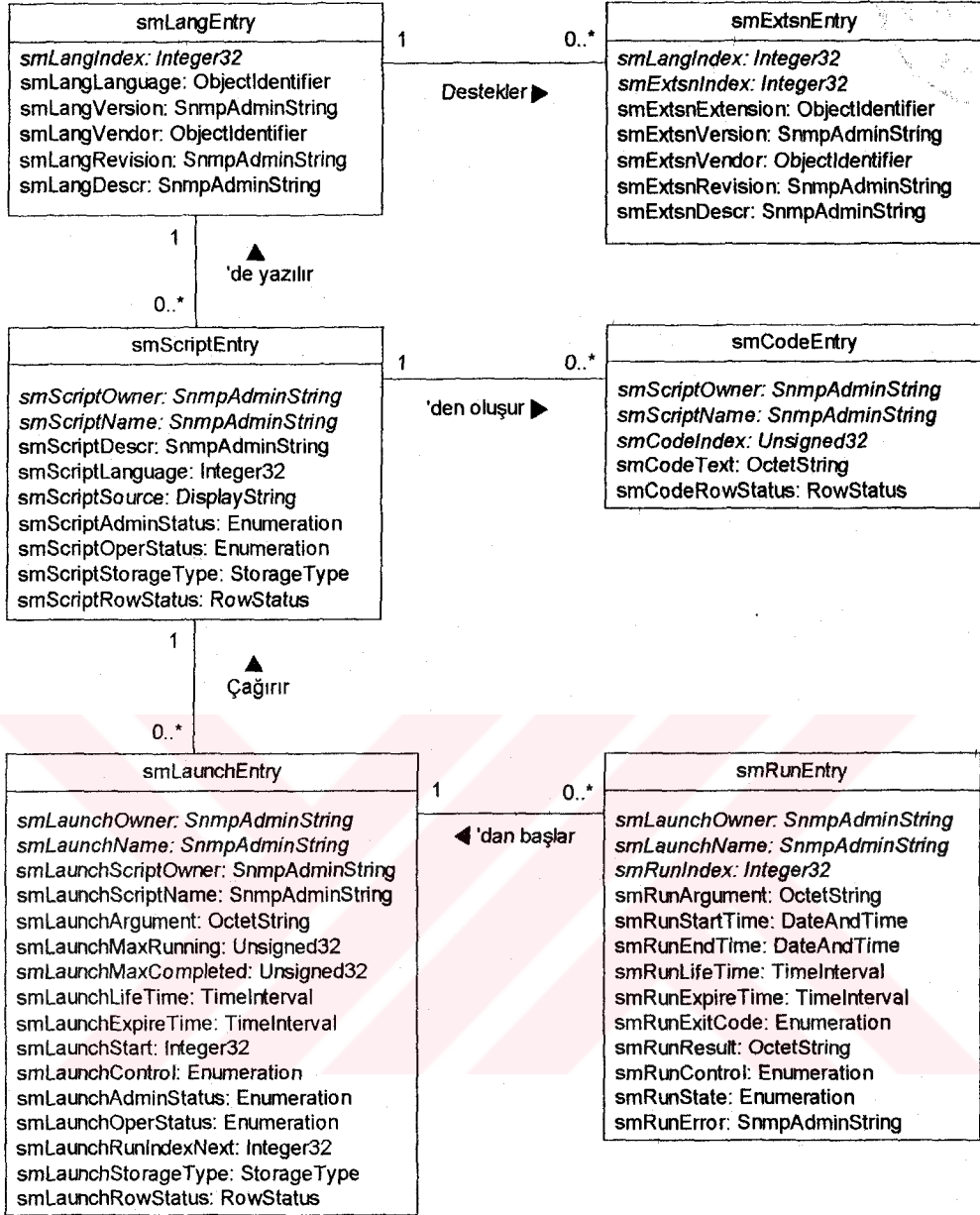
Script MIB yönetim fonksiyonlarının yürütülebilir ve ağ düğümlerine kurulabilir kod (script) yapısında tanımladığını varsayar. Ayrıca bu MIB spesifik dil veya çalışma ortamı hakkında herhangi bir önkabullenme yapmaz ve herhangi bir ortamla çalışabilir hatta gerçekleştirme ortamı uygun olursa derlenmiş yerli kod da kullanabilir.

3.1.1.1 Script MIB'in yapısı

Altı tablodan oluşan Script MIB'in mantıksal yapısı Şekil 3.1'de

gösterilmektedir. smLangTable ve smExtsnTable Script MIB gerçekleştirmesi tarafından desteklenen dil (language) ve dil uzantıları (language extensions) hakkında bilgi sağlar. Burada dile örnek vermek gerekirse Java ve Tcl buna örnek olabilir.

smScriptTable bir ağ elemanında kurulu Script MIB'ı destekleyen bütün scriptleri bulundurur. Bu tablo yöneticiye scriptleri kurmak veya kaldırmak ve scriptin durumunu okuyup değiştirmeyi sağlayan nesneleri içerir. Basit bir şekilde ifade etmek gerekirse bu tablo Script MIB gerçekleştirmesi tarafından sürdürülen script ambarına bir arayüzdür. Scriptler kalıcı şekilde kurulabilir, etmen tarafından herhangi bir yerden indirilebilir veya smCodeTable'a yazılarak etmene yollanabilir. Kalıcı scriptler silinemez, dinamik scriptler de sabit diske yazılarak güvence altına alınır. smScriptTable'daki her bir kayıt smLangTable'a scriptin dilini belirten bir referans bulundurur.



Şekil 3.1. Script MIB'in yapısı

smCodeTable SNMP set işlemi kullanılarak scriptlerin yollanma ve değiştirme kabiliyetini sağlar. SNMP mesajının limitlerini geçme durumlarında bir script smCodeTable'da birkaç satıra parçalanabilir. Sadece bu tablodaki scriptler okunabilir veya değiştirilebilir. Bu tablonun gerçekleştirilmesi opsiyoneldir.

smLaunchTable başlatılmaya hazır scriptleri tanımlar. Tablodaki nesnelerden biri başlatılacak scriptte verilecek argümanları tanımlar. smLaunchStart nesnesi de yöneticiye bir SNMP mesajı ile bir scripti çalıştırma olanağı verir. Bunlardan başka

scriptin ömrünü, script bittikten sonra sonuçların saklanma müddetini ve aynı scriptten aynı anda kaç tanesinin çalıştırılabileceğini tanımlayan nesneler de vardır.

smRunTable ise halihazırda çalışan veya henüz sona eren scriptleri listeler. Bu tablo yöneticiye çalışan scriptleri kontrol etmeyi ve durumunu görmeyi sağlayan nesneleri bulundurur. Ayrıntılı bahsetmek gerekirse smRunLifeTime scriptin sona ermesine kadar çalışabileceği süreyi kontrol eder, smRunExpireTime sona eren scriptin bu tabloda kalabileceği maksimum süreyi tanımlar, smRunControl ise yöneticiye çalışan scriptleri askıya alma, devam ettirme ve iptal etme olanakları sağlar. [11],[16]

3.1.1.2 Script MIB'in Haberleşme ve Kullanım Modeli

Bir ağ düğümündeki bir scriptin kontrolü ve çalıştırması için gerekli adımları tanımlayan Script MIB'in kullanımı aşağıda açıklanmıştır.

1. Bir scripti bir ağ düğümünde yürütmek isteyen bir yönetici öncelikle ilgili scriptin halihazırda kurulup kurulmadığını smScriptTable'a bakarak kontrol eder. Eğer ilgili script halihazırda kurulu ise yönetici 4. adımdan devam eder.
2. Yönetici bu noktada istediği scriptin bu düğümde çalıştırılıp çalıştırılamayacağını anlamak için hangi dillerin desteklendiğini smLangTable'ı ve gerekirse de smExtnTable'ı okuyarak görür.
3. Yönetici desteklenen dillere uygun bir scripti ambardan seçer ve smScriptTable'a yeni bir satır ekleyerek scripti kurar. Yönetici burada bir tercih yapmak durumundadır. Kurulan scripti ya SNMP ile yollar veya smScriptTable'a scriptin indirilebileceği URL'i yazar ve SNMP etmeninin scripti buradan indirmesini ister.
4. Script'i kurduktan sonra yönetici smLaunchTable'da bir satır oluşturur. Bu satır smScriptTable'daki scripte işaret etmektedir ve çalışma ortamının güvenlik parametrelerini tanımlamaktadır. Eğer daha önce aynı script kurulmuşsa bu tabloda bu script için bir veya birden fazla satır bulunabilir. Bu tabloda bir scripte onu değişik argümanlarla veya değişik güvenlik parametreleri ile çalıştırmak için birden fazla satır oluşturulabilir. Yönetici smLaunchTable'da gerekli düzenlemeleri yaptıktan sonra çalıştırma nesnesini set ederek scripti başlatabilir. Bu satırdaki başlatma nesnesi birden fazla set

edilerek aynı scriptten birden fazla aynı anda çalıştırılabilir. Herbir başlatmaya karşılık smRunTable'da ayrı bir satır otomatik olarak oluşturulur.

5. Çalışan scriptlerin hepsinin bilgisi smRunTable'da saklanır ve yönetici bu tabloyu kullanarak onları kontrol edebilir. Scriptlerin askıya alınması, devam ettirilmesi veya iptal edilmesi mümkündür. Ayrıca scriptlere sona erme süresi de bu tablodan atanabilir. Scriptin başlama zamanı, argüman ve çalışma durumu ile sona ermesinden sonra sona erme zamanı ve sonuç da bu tabloda bulunmaktadır. Script tarafından üretilen sonuçlar yönetici tarafından smRunExpireTime'da tanımlanan süreye kadar okunabilir. [11]

3.1.1.3 Script MIB'in Güvenliği

Script MIB için güvenlik gereksinimleri diğer MIB'lerden farklıdır. Normal MIB'ler genellikle sadece okuma, yazma ve bildirme izinlerini ayırt ederken Script MIB yürütme izinlerini de ayırt eder. Script MIB erişim güvenlik gereklerini erişim kontrolü [32] ile çalışması tasarlanan bir tablo indeksleme şemasıyla ve SNMPv3'ün mesaj güvenlik [31] tasarımlarıyla gerçekleştirir.

Scriptlerin çalışma ortamı güvenlik gerekleri smLaunchOwner'ın işletim sistemi ve çalışma ortamı güvenlik profillerine mapping yapılmasına izin verilerek sağlanır [11]. İşletim sistemi güvenlik profilleri çalışan bir script tarafından kullanılabilen işletim sistemi servislerini tanımlar. Çalışma ortamı profilleri ise güvenlik bir çalışma sistemi tarafından izin verilen operasyonlar setini tanımlar [33].

3.2 Yönetim Sistemi

Bundan önceki kısımlarda IPsec geçitlerinin yönetim ihtiyacı ve bu ihtiyacı karşılamakta kullanılabilecek mimariler ve çözümler incelendi. Bölüm 2.1.5'te bu ağ yönetimi mimarileri IPsec geçitlerinin yönetim ihtiyacını karşılama noktasında karşılaştırılmış ve bunlardan güçlü dağıtık hiyerarşik yönetim mimarisi ve Script MIB'in uygulanabileceği sonucu ortaya çıkmıştır.

Bu kısımda IPsec geçitlerinin yönetimi için önerilen Script MIB tabanlı dağıtık yönetim sistemi tanıtılmaktadır. Mimari ve sistem ayrıntılı bir şekilde açıklanmaktadır. Burada önerilen dağıtık sistem üç ana yönetim özelliğine sahiptir. Bunlar IPsec/VPN geçitlerinin konfigürasyon yönetimi, olay yönetimi ve

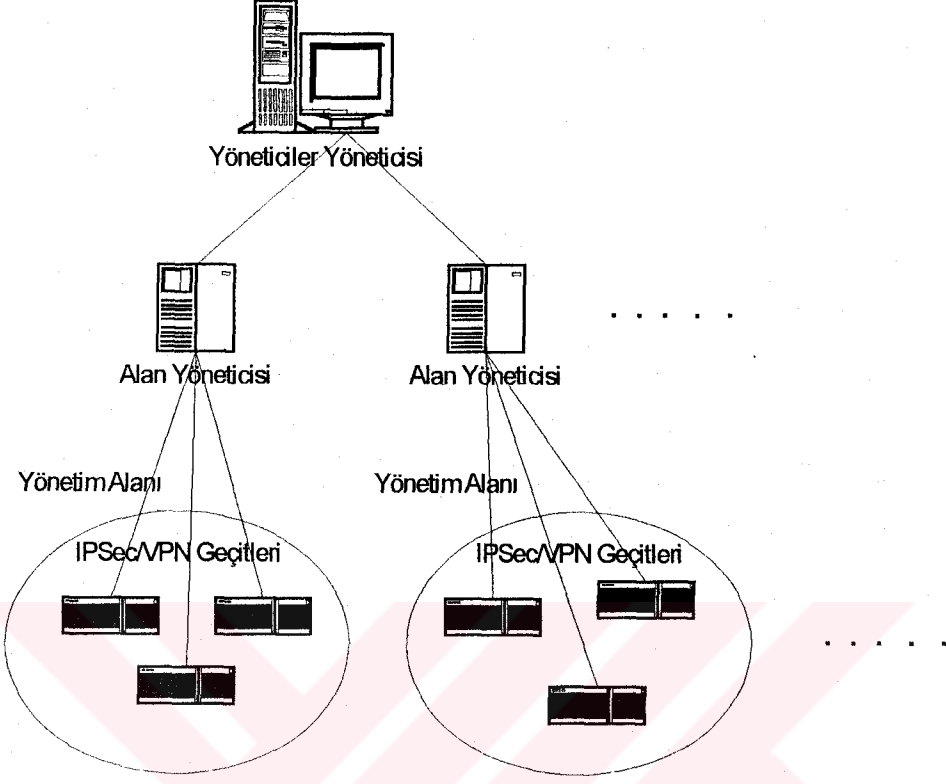
gözlenmesidir. Burada delegasyon mekanizmaları bütün bu özellikleri gerçekleştirmek için kullanılmaktadır. Hedef olarak alınan IPSec geçitlerinin gerekli MIB'leri gerçekleyen SNMPv3 etmeni bulundurduğu kabul edilmektedir. Daha iyi ifade etmek gerekirse burada tanımlanan sistem SNMP ile yönetilebilen IPSec/VPN geçitlerinin yönetimini yapmaktadır. Burada IPSec geçitlerinde çalışan SNMP etmenleri IYE (IPSec Yönetim Etmeni) olarak adlandırılmıştır.

Bundan önceki kısımlarda IPSec geçitlerinin yönetim ihtiyacı ve bu ihtiyacı karşılamakta kullanılabilecek mimariler ve çözümler incelendi. Bölüm 2.1.5'te bu ağ yönetimi mimarileri IPSec geçitlerinin yönetim ihtiyacını karşılama noktasında karşılaştırılmış ve bunlardan güçlü dağıtık hiyerarşik yönetim mimarisi ve Script MIB'in uygulanabileceği sonucu ortaya çıkmıştır.

Bu kısımda IPSec geçitlerinin yönetimi için önerilen Script MIB tabanlı dağıtık yönetim sistemi tanıtılmakta, mimari ve sistem ayrıntılı bir şekilde açıklanmaktadır. Burada önerilen dağıtık sistem üç ana yönetim özelliğine sahiptir. Bunlar IPSec/VPN geçitlerinin konfigürasyon yönetimi, olay yönetimi ve gözlenmesidir. Burada delegasyon mekanizmaları bütün bu fonksiyonallikleri gerçekleştirmek için kullanılmaktadır. Hedef olarak alınan IPSec geçitlerinin gerekli MIB'leri gerçekleyen SNMPv3 etmeni bulundurduğu farzedilmektedir. Daha iyi ifade etmek gerekirse burada tanımlanan sistem SNMP ile yönetilebilen IPSec/VPN geçitlerinin yönetimini yapmaktadır. Burada IPSec geçitlerinde çalışan SNMP etmenleri IYE (IPSec Yönetim Etmeni) olarak adlandırılmıştır.

Şekil 3.2'de görüldüğü üzere, açıklanan yönetim sisteminin mimarisi hiyerarşiktir. Sistemin en üstünde YY (Yöneticilerin Yöneticisi) bulunmaktadır. YY'nin ana sorumluluğu AY'lerin (Alan Yöneticisi) aktivitelerinin doğrudan ve IYE'lerin de dolaylı olarak kontrolüdür. Yönetim sistemindeki tek kullanıcı arayüzü YY'de bulunmaktadır. YY'leri ile IMA'lar arasında AY'ler bulunmaktadır. AY yönetim scriptleri olarak gerçekleştirilen delege edilmiş yönetim fonksiyonlarını yürütür. Bu işi yapabilmek için AY bir Script MIB gerçekleştirmesine ve de gerekli script ortamına ev sahipliği yapar. Her bir AY kendi alanında belirli bir sayıda IPSec/VPN geçidine sahiptir. Bunlar AY'nin sorumluluk alanını oluşturmaktadırlar. Buradaki alan kavramı topolojik (bölgesel) veya mantıksal (çalışma alanları) olabilir. Bu noktadaki karar tamamen IPSec geçitlerinin kullanım politikasına bağlıdır. IPSec geçitlerinin gruplanması ile ilgili kriterler Bölüm 1.4'te ayrıntılı şekilde ve örnekle

gösterilmiştir. Buradaki kritere göre kendi aralarında daha çok tünel oluşturulan IPSec geçitleri bir alanı oluşturur ve bu alana da bir AY tahsis edilir.



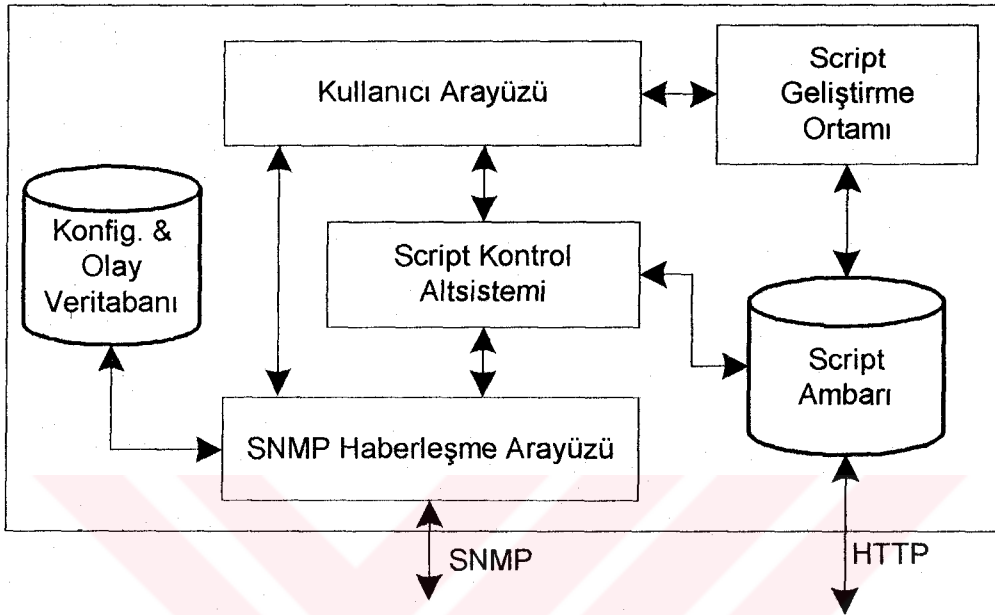
Şekil 3.2. Sistemin genel mimarisi

3.2.1 Yöneticilerin Yöneticisi

Yönetim sisteminin merkezi konumundaki YY temel olarak beş parçadan oluşmaktadır. Bunlar kullanıcı arayüzü, script kontrol altsistemi, script geliştirme ve test ortamı, script ve yönetim veri ambarları ile SNMP haberleşme altsistemidir. Bu sistemler arasındaki ilişkiler kabaca Şekil 3.3’de gösterilmiştir.

Kullanıcı arayüzünden öncelikle AY’lerin yönetimi (scriptlerin çalıştırılması, sonuçların alınması vs.) ve ikinci planda da doğrudan IYE’lerin yönetimi yapılabilmektedir. Bu YY’nin gerektiği zaman bir merkezi yönetim mimarisindeki bir yönetici edasıyla IYE’lerin konfigürasyonu, olayların toplanması ve gözlenmesi işlerini yapabildiği anlamına gelmektedir. Bu yetenek özellikle alanlar arası tünel yönetiminde kullanılabilmektedir. AY’ler sadece kendi alanlarındaki cihazlar ile ilgilendiklerinden dolayı alanlar arası faaliyetler YY’nin sorumluluğunda olmaktadır. YY bu işleri doğrudan kendi yapabileceği gibi istenirse birden fazla AY’yi paralel

olarak kontrol ederek onlara da yaptırabilir. YY'nin IYE'lere doğrudan müdahale edebilmesinin belki en pratik kullanılışı da herhangi bir alanın yöneticisi arızalandığı veya YY ona ulaşamadığında IYE'lerin kontrolsüz kalmaması ve acil yapılması gereken konfigürasyonların bu şekilde yapılabilmesidir.



Şekil 3.3. YY'nin altsistemleri

Script MIB hemen her türlü script ortamı (TCL, Perl, Java vs.) ile çalışabilmektedir. Burada önerilen sistemde ise Script MIB bir Java ortamını kullanmaktadır. Bu yüzden de YY'deki script geliştirme ortamı aslında gerekli kütüphanelerin entegre edildiği (Mesela SNMP kütüphaneleri) bir Java geliştirme ortamıdır. Java'yı Script MIB ortamında script dili olarak kullanma iki şekilde olabilmektedir [29]. Bunlar Java programları şeklinde scriptler ve Java nesneleri şeklinde scriptlerdir. Bu sistem farklı amaçlar için iki tipteki scriptleri de kullanmaktadır.

Script geliştirme ortamında geliştirilen yönetim scriptleri normalde AY'lere iki ayrı yöntemle gönderilebilir. YY scriptleri doğrudan AY'lere gönderebilir veya AY'yi tetikleyerek script ambarından çekmesini sağlayabilir. Burada bir tercih yapılmış ve ikinci yöntem seçilmiştir. Bunun sebebi ise ilk yöntemin YY'yi gönderme süreci ile daha fazla uğraştırması ve büyük scriptlerde bu işin daha da fazla olmasıdır. İkinci yöntemde ise scriptin ambardan alınması süreci ile AY ilgilenir.

3.2.2 Alan Yöneticileri

Bu mimaride AY kritik noktayı oluşturmaktadır. Tanımlanan YY yönetim işlerini merkezi olarak da gerçekleştirme yeteneğine sahip olduğu için AY'nin bu noktada işin içine girmesi YY'nin omuzlarından büyük bir miktar yükün alınması anlamına gelmektedir. Bu şekilde YY yüzlerce IPsec geçidi ile muhatap olmak yerine daha az sayıda AY ile muhatap olmaktadır. Bu işi YY bazı yönetim sorumluluklarını AY'lere yönetim scriptleri aracılığıyla delege etmek şeklinde gerçekleştirir. AY'ler bu sorumlulukları yerine getirebilecek teçhizata sahiptirler. Bir AY beş altsistemden oluşur: Bunlar SNMP haberleşme arayüzü, Script MIB gerçekleştirmesi, script çalışma motoru, olay sistemi ve ambardır.

AY'ler YY'nin yönetim sorumluluklarından üçünü gerçekleştirmek üzere delege edilmektedir. Bunların bazıları yönetim scriptlerini kullanarak gerçekleştirilirken geri kalanı da AY'nin içine gömülüdür. Yönetim sorumlulukları da bahsedildiği üzere konfigürasyon, olay yönetimi ve gözlemdir. AY halihazırda SNMP mesajları gönderip SNMP olaylarını alabilen bir SNMP yöneticisi gibi davranabilecek yeteneğe sahiptir. Yönetim scriptleri de bu yeteneği kullanabilecek şekilde hazırlanmaktadır.

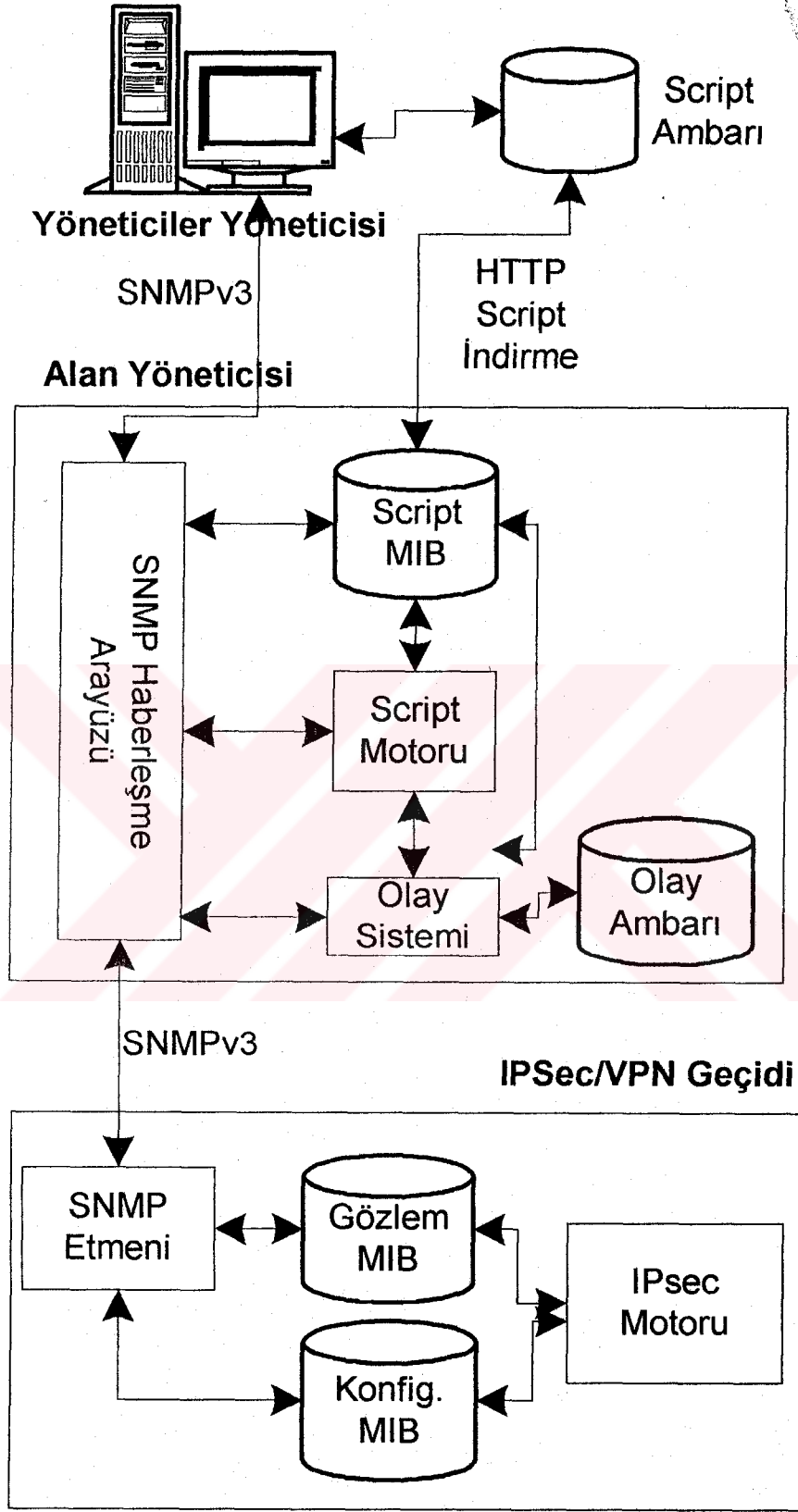
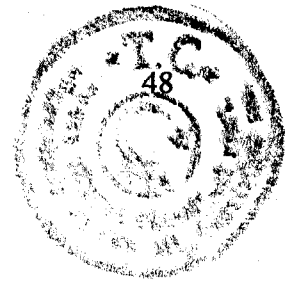
AY'lerde kullanılan script çalışma motoru ise aslında JVM'dir (Java Virtual Machine-Java Sanal Makinesi). Java programları olan scriptler için JVM doğrudan çalışma ortamı iken, Java nesneleri olan scriptler için bunların asıl çalışma ortamı JVM'de çalışan olay sistemi olduğundan, JVM onların dolaylı çalışma ortamıdır.

Olay sistemi AY'nin üzerinde çalışan ve en baştan ve kendi başına çalışan bir program olabileceği gibi Script MIB'e yollanan ve oradan çalıştırılan bir program da olabilir. Olay sistemi SNMP protokolünde standart tanımlı olan 162'nci portu dinleyen ve buraya gelen SNMP bildirilerini toplayıp değerlendiren bir sistemdir. Ayrıca üzerinde kayıt edilen filtreleri ve bu filtrelerin faaliyetlerini çalıştırma yeteneğine de sahiptir. Gelen herbir bildiri kurulu filtrelere sokulur ve bu herhangi bir filtreye bir bildiri takıldığında bu filtrenin faaliyet kodu çalıştırılır. Olay sistemi topladığı bildirilerin hepsini kaydedebileceği gibi sadece filtrelere takılanları kaydedebilir veya hiç kaydetmeyebilir. Bu nokta opsiyoneldir.

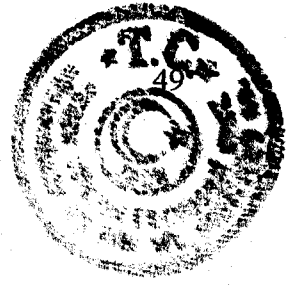
3.2.3 IPSec Yönetim Etmenleri

Buradaki mimaride, yönetilen düğüm olarak çok özel bir etmen gerekmemektedir. Aslında bu yönetim sisteminin çok güçlü olduğu özelliklerinden birisi bu sistemin sadece IPSec konfigürasyon ve gözlem MIB'leri içeren SNMPv3 destekleyen bir etmene ihtiyaç duymasıdır. Bu özelliği bulunduran herhangi bir IPSec/VPN cihazı tanımlanan yönetim sistemindeki AY'ler ve YY tarafından kolayca yönetilebilmektedir ve cihaz tarafında herhangi özel bir değişiklik gerekmemektedir.





Şekil 3.4. AY ve IYE'nin detaylı yapısı



3.2.4 Yönetim Script'leri

3.2.4.1 Kullanılan Script Ortamı

Bu çalışmada önerilen dağıtık yönetim sisteminin en kritik noktası yönetim scriptleridir. Bu scriptlerin belli başlı bazı özelliklerinin olması gerekmektedir. Öncelikle bunların hazırlanması kolay olmalıdır. Burada kullanımı önerilen Java dili diğer script dillerine göre nesne tabanlı olması, bir programlama dili olması yönüyle kullanımının ve öğrenilmesinin kolay olması, geniş standart kütüphaneleri ve buradaki kullanımıyla ilgili hazır SNMP kütüphanelerinin bulunması yönleriyle avantajlıdır. Ayrıca gözlem ve konfigürasyonda script olarak kullanılan Java programlarının Java'nın thread mekanizmalarının kullanabilmesi de büyük avantajları beraberinde getirmektedir.

Scriptler ile ilgili bir diğer husus çok uzun olmamalarıdır. Bu onların AY'ler tarafından transferinin hızlı olması noktasından önemli bir husustur. Yine kullanılan Java programları ele alındığında bunların uzunlukları çok fazla değildir. Tek başına çalışabilen ve tek bir dosya olan class dosyaları bytecode formatında olduğundan kendi kaynak kod dosyasından daha kısa olmaktadır. Örnek vermek gerekirse 634 satırlık bir Java dosyası 23KB civarı yer kaplarken derlenip class haline dönüştürüldüğünde yaklaşık 13KB yer kaplamaktadır ki böyle bir dosyanın AY tarafından transferi saniyeler mertebesinde daha fazla olmamaktadır. Ayrıca tünel kurulumu gibi genel amaçlı scriptler bir kez kurulduktan sonra tekrar tekrar kullanılmaktadır.

Script'lerin oluşturulmasında Java kullanılmasının bir diğer avantajı normal script ortamlarının kısıtlamalarının burada olmamasıdır. Daha açık ifade etmek gerekirse Java bir programlama dili olması yönüyle sınırlamaları çok azdır. Java ile hemen herşey yapılabilmektedir. Burada kullanılan sistemde de Java'nın bu yönlerinden istifade edilmektedir. Örnek vermek gerekirse scriptlerin belirli zamanlarda çalışmaya programlanması için genelde Schedule MIB kullanımı önerilirken Java kullanıldığında Java'nın kendi Timer mekanizması bu işi kolaylıkla halledebilmektedir.

3.2.4.2 Script Tipleri

Daha önce de belirtildiği gibi, YY üç tip yönetim özelliğini scriptler halinde delege eder. Bunlar konfigürasyon, olay ve gözlem yönetim scriptleridir. Konfigürasyon ve gözlem scriptleri kendi kendine çalışabilen Java programlarıdır. Olay scriptleri ise daha farklıdır. Olay scripti de bir Java programıdır fakat bu program sadece içinde faaliyet kodu içeren bir filtre nesnesini oluşturup olay sistemine kayıt ettirir, tek başına çalışıp bir iş gerçekleştirmez. Burada aslında olay scriptinin esas kısmı filtre ve faaliyet kodudur.

Konfigürasyon scriptleri birbirini takip eden birden çok SNMP mesajını gerçekleştirmek için tasarlanmıştır. Konfigürasyon için çarpıcı bir örnek tünel kurulumudur. Bu işin gerçekleştirilebilmesi için ilgili MIB kullanılarak pekçok SNMP mesajının tek tek gönderilmesi ve cevapların değerlendirilmesi gerekmektedir. Mesela önce bir GB tanımı eklenir, daha sonra gerekli IKE ve KH tanımları takip eder ve en sonunda da politika kuralları her iki trafik yönü için de eklenir. Bu konfigürasyon aralarında tünel kurulmak istenen iki IPSec geçidinde de uygulanması gereken bir prosedürdür. Konfigürasyon MIB'lerinin gerçekleştirilmesi ve tanımları her bir üretici için farklı olabilir. Fakat her durumda pekçok parametre ilgili tablolara kayıt eklenerek konfigüre edilmek zorundadır. YY pekçok sorumluluğa ve iş yüküne sahip olacağından bu gibi konfigürasyon ayrıntılarıyla doğrudan uğraşmak ona ek bir yük getirir. Bunun yerine buradaki çözümde gerekli tünel kurulum konfigürasyon scripti hazırlanır ve AY'lere yollanır. Her ne zaman bir tünelin kurulması gerekirse, kullanıcı gerekli tünel parametrelerini YY'ye verir, YY bu parametreleri doğrudan ilgili AY'ye yollayıp tünel kurma scriptini bu parametrelerle çalıştırır. Tünel kurma scriptini çalıştıran AY geri kalan işleri yapar. IPSec/VPN geçitlerinin konfigürasyonunda pekçok benzer senaryo bulunmaktadır. Ayrıca bu şekilde dinamik kod gönderilmesi ile SNMP konfigürasyonunun otomatik hale getirilmesi, farklı üreticilerin farklı IPSec konfigürasyon MIB'i ile muhatap olunması durumunda yeni script kullanılarak cihazın AY'ler tarafından konfigüre edilebilmesi avantajına sahiptir.

Olay scriptlerinin bildiri filtreleri olarak davranması amaçlanmıştır. AY'ler halihazırda kendi alanlarında bulunan IPSec geçitlerinden gelen bildirimleri toplayabilen bir Olay Sistemine sahiptir. YY'nin bu bildirimleri değerlendirmesi gerekmektedir. Bütün IPSec geçitlerinin bildirimleri ile uğraşmak yerine YY bu

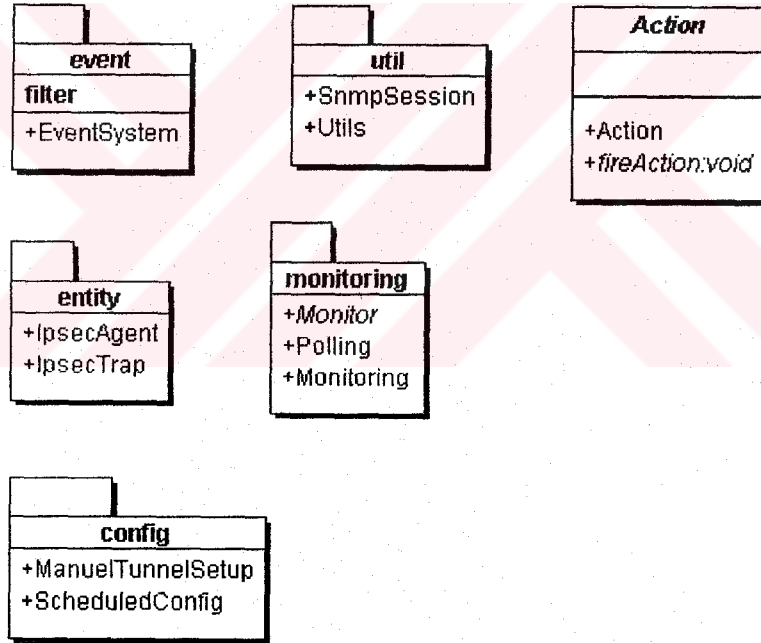
şekilde süzölmüş ve AY tarafından gönderilen olaylar ile ilgilenir. YY bildiri filtrelerini faaliyetler ile script halinde AY'lere gönderir. Geri kalan iş AY'nin üzerindedir. AY bu scripti çalıştırarak filtreyi olay sistemine kaydeder. Olay sistemi ise üzerinde kayıtlı olan filtreleri gelen bildirilere uygulayarak bu filtrelere bir olay takıldığında olayın takıldığı filtrenin faaliyetini gerçekleştirir. Faaliyetler YY'yi bildiri ile haberdar etmek veya cihazların konfigürasyon parametrelerini değıştirmek gibi şeyler olabilir.

Gözlem scriptleri olay ve konfigürasyon scriptlerinin karışımı gibi bir özellik gösterir. Fonksiyonallite yönüyle bunlar konfigürasyon scriptlerine benzerdirler ve içlerinde faaliyet tanımlanması yönüyle de olay scriptlerine benzerlik göstermektedirler. Bu scriptler cihazlardaki belirli parametreleri belirli aralıklarla gözler. Daha önceden belirtilen bir durum olduğunda ise faaliyet tetiklenir. Olay faaliyetlerinde olduğu gibi bu faaliyetler YY'yi haberdar etmek veya cihazların parametrelerini konfigüre etmek olabileceğı gibi sadece bir jurnal düşmek gibi herhangi başka bir şey de olabilir. Genellikle gözlenen parametreler IMA'lardaki SNMP MIB (mesela IPsec akış gözlem MIB'i-IPsec flow monitoring MIB [5] gibi) sayaçlarıdır. Kullanıcı gözlem scriptlerini faaliyeti ile bir taslak gibi hazırlar ve ne zaman bu scripti çalıştırmak isterse gerekli parametreler ile bunu çalıştırır. Gözlem scriptinin parametreleri mesela gözlenecek sayacın OID'i (Object Identifier), periyot, süre ve ilgili faaliyetin tetikleneceğı eşik değeri olabilir.

4 ÖRNEK UYGULAMA YORUMLAR VE KARŞILAŞTIRMA

4.1 Örnek uygulamalar

Daha önce de belirtildiği üzere önerilen dağıtık yönetim sisteminin kritik noktası scriptlerdir. Bunlar hemen hemen bütün yönetim fonksiyonlarını eda etmeleri yönüyle burada öncelikli olarak ele alınacaktır. Sistemin geri kalan kısmı daha standart bir yapıya sahiptir. Bu bölümde geliştirilen yönetim scriptleri tanıtılacaktır. Yönetim scriptlerinin geliştirilmesinde AdventNet'in [39] ve Jasmin [38] projesinin SNMP API'leri kullanılmıştır.



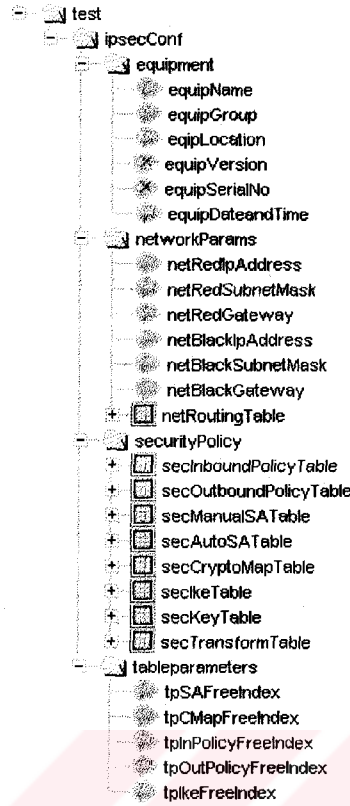
Şekil 4.1. Geliştirilen scriptlerin paket yapıları

Bahsedilen üç tip yönetim scripti için de örnek uygulamalar geliştirilmiştir. Bunlar daha önce de belirtildiği üzere Java dilinde yazılmış programlardır. Geliştirilen projedeki paket yapıları Şekil 4.1'de gösterilmiştir. Şekilde beş paket ve bir de sınıf vardır. Action sınıfı genel bir sınıf olduğundan hiçbir pakete konmamıştır. config paketi ManuelTunnelSetup ve ScheduledConfig

adında iki adet konfigürasyon scriptinden oluşmaktadır. event paketi EventSystem adındaki olay sistemini oluşturan script ve bir de filtreleri içeren filter paketinden oluşmaktadır. monitoring paketi ise Monitor adında soyut bir sınıf ile Polling ve Monitoring adında iki adet konfigürasyon scriptinden oluşmaktadır. Diğer iki paketteki sınıflar script değildir. entity paketinde IPSec etmenini tanımlayan bir IPSecAgent ve bildirileri tanımlayan IPSecTrap sınıfları bulunmaktadır. util paketinde ise SNMP kütüphanelerine erişimi sağlayan SnmpSession sınıfı ile scriptler tarafından ortak kullanılan bazı metotları içeren Utils sınıfı bulunmaktadır.

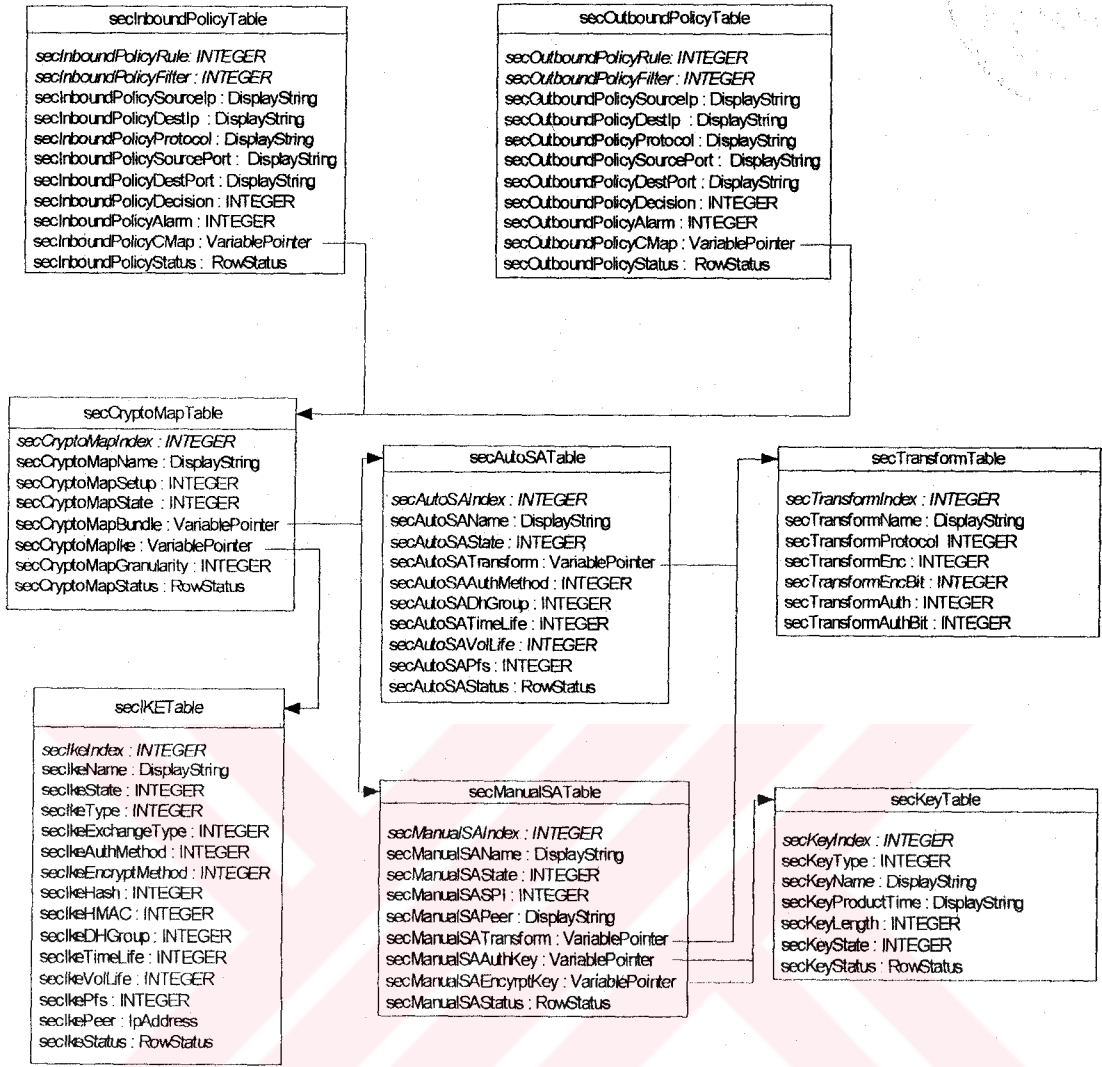
4.1.1 IPSec Konfigürasyon Test MIB'i

Konfigürasyon scriptlerinin yazılmasında bir IPSec konfigürasyon MIB kullanımı gerekmektedir. Burada test için yazılmış sade bir IPSec konfigürasyon MIB kullanılmıştır. Henüz müsvette (draft) durumunda olan IPSec Policy Configuration MIB [6] biraz daha kompleks bir yapıya sahip olması ve daha fazla tablo içermesinden dolayı tercih edilmemiş ve bu iş için daha sade bir MIB yazılmıştır. Yazılan bu MIB Ek-1'de bulunabilir. Bu MIB AdventNet Agent Toolkit 4.0 [39] kullanılarak derlenmiş ve derlenme sonucu ortaya çıkan ve IPSec konfigürasyon MIB'i gerçekleyen SNMP etmeni yazılan scriptlerin test edilmesinde kullanılmıştır.



Şekil 4.2. IPSec Configuration Test MIB'in yapısı

MIB'in yapısı da kabaca Şekil 4.2.'de gösterilmiştir. Bu MIB'den kısaca bahsetmek gerekirse temelde dört grup bulunmaktadır. Bunlar sırasıyla “equipment”, “networkParams”, “securityPolicy” ve “tableParameters” gruplarıdır. İlk grupta cihazla ilgili genel konfigürasyon parametreleri bulunurken ikinci grupta birkaç cihazın IP adresleri ve yönlendirme tablosu gibi bazı parametreler tanımlanmıştır. Burada esas odak noktasını oluşturan üçüncü grupta ise giren ve çıkan politika tabloları, manuel ve otomatik güvenlik birliği, ike tanımı, kript haritası, anahtar ve dönüşüm tabloları bulunmaktadır. Bu tabloların alanları, alan tipleri ve birbirleri ile ilişkileri Şekil 4.3.'de gösterilmiştir. Son grupta ise bu tabloların konfigürasyonunda yardımcı bazı parametreler bulunmaktadır.



Şekil 4.3. IPsec Configuration Test MIB'deki güvenlik parametre tabloları

4.1.2 Konfigürasyon Scriptleri

Örnek sistem için iki adet konfigürasyon scripti yazılmıştır. Bunlardan ilki iki IPsec geçidi arasında manuel güvenlik birliği kullanan bir tünel oluşturmaktadır. İkincisi ise bir takvime programlanmış konfigürasyon değişikliklerini yapmaktadır. Öncelikle tünel oluşturan scriptten söz etmek gerekirse bu script tek bir sınıftan oluşan bir Java programıdır. Bu sınıfın ismi "ManualTunnelSetup" tır. Bu sınıfın içinde static olarak tanımlanan main fonksiyonu çalışmasına bazı parametreler alır. Bunlar aralarında tünel kurulacak IPsec geçitlerinin IP adresleri, politika tabloları parametreleri, güvenlik birliği parametreleri ve de krypto haritası parametreleridir. Bu fonksiyon program ilk çalıştığında bu parametreleri alır ve

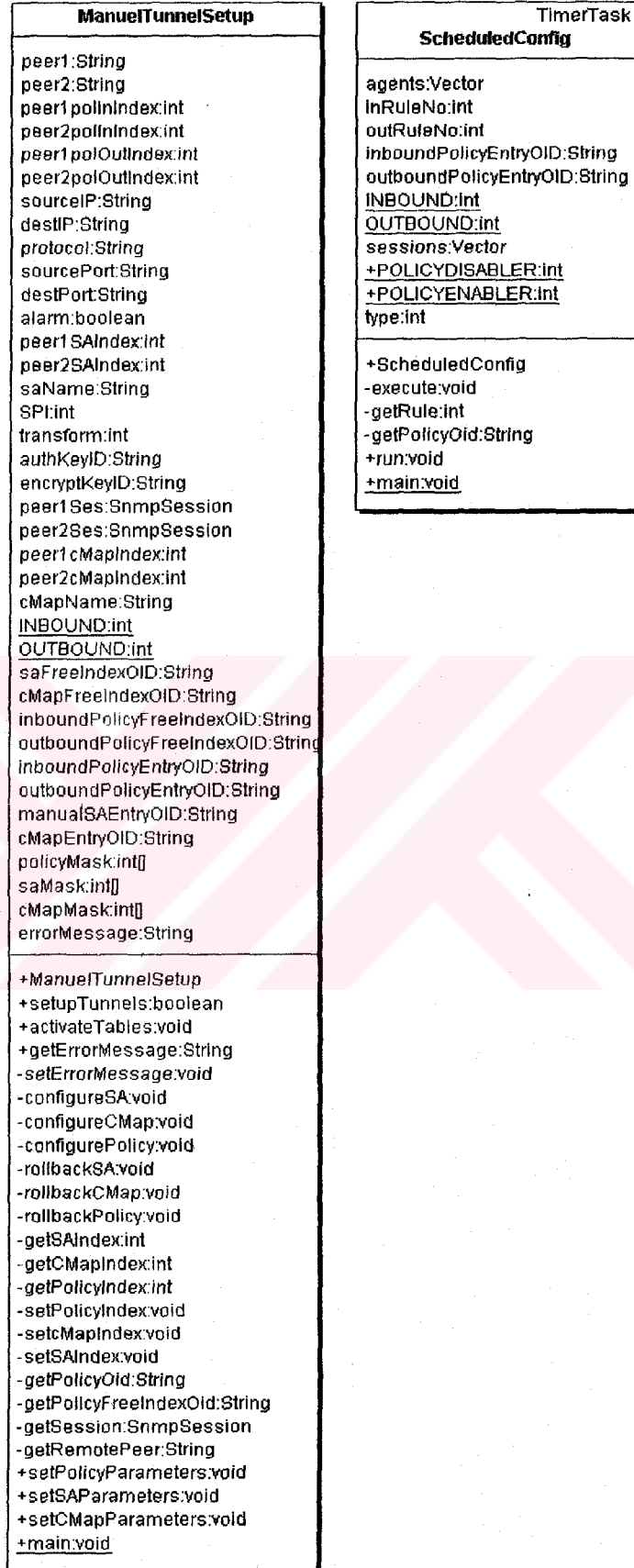
bunları kullanarak `ManualTunnelSetup` sınıfından bir nesne oluşturur. Ardından da bu nesnenin `setupTunnels` metodunu çağırarak iki IPSec geçidi arasında tünel oluşturulmasını sağlar. Bu metod herhangi bir hata oluşması durumunda `false` döndürmekte, tünel kurulumunun başarılı olması durumunda ise `true` döndürmektedir. `ManualTunnelSetup` nesnesi herhangi bir hata durumunda ilgili hata mesajını kendi içinde tutmakta ve bu mesaja `getErrorMessage` metodu kullanılarak erişilebilmektedir. Program haberleşmede veya SNMP işlemlerinde bir hata oluşması durumunda 6 sonucu ile programdan çıkmakta aksi takdirde ise 1 ile çıkmaktadır. Scriptin sonlanma sonucu Script MIB'de `smRunTable`'da çalışan scripte ait satırdaki `smRunExitCode` alanına yazılır. Scriptin çalışması sonucu üretilen sonuç metni ise (`println` ile basılan) `smRunResult` alanına yazılır.

Şimdi tünel kuran bu scriptin çalışması daha ayrıntılı incelenecektir. `setupTunnels` metodu önce IPSec geçitlerinden birisinin gerekli konfigürasyonunu yapar, daha sonra diğer geçidin konfigürasyonunu yapar. Her iki işlemde de herhangi bir problem oluşmazsa en son her iki cihazda yapılan konfigürasyonlar sırasıyla aktif hale getirilir ve böylece istenen tünel konfigürasyonu gerçekleştirilmiş olur. Eğer bu adımlardan herhangi birisinde herhangi bir hata oluşursa, o ana kadar yapılan bütün konfigürasyonlar cihazlardan geri alınır. Bu scriptin kaynak kodu Ek-2'de yer almaktadır.

Konfigürasyon scriptleri ile ilgili bir diğer uygulama ise `ScheduledConfig` isimli sınıftır. Bu sınıf da yine bir programdır. Tünel kuran scriptteki gibi yine `main` metodu parametre olarak `ScheduledConfig` sınıfından bu parametreleri kullanarak iki adet sınıf oluşturur. Bu scriptte iki adet zamana programlanmış konfigürasyon işi vardır. `ScheduledConfig` sınıfı Java'nın `TimerTask` sınıfından türetilmiş bir sınıftır ve `run` metodunu gerçeklemektedir. `main` metodu bu programlanmış işleri `ScheduledConfig`'den oluşturduğu iki farklı nesneyi Java'nın `Timer` sınıfından türettiği iki nesneye verir ve yine kendisine verilen zaman parametreleri ile bu iki `Timer` nesnesini programlar. Burada zamana programlanma periyodiktir. Yani `Timer` nesnesi yönetim scriptini belirli zamandan başlayarak belirli sabit aralıklarla çalıştırır. Bu script yönetim tarafından durdurulana kadar çalışmaya devam eder. Burada zamana programlanan scriptlerin birisi programlanma zamanı gelince kendi alanındaki bütün IPSec geçitlerinin politika tablolarından belirli bir kuralı aktif hale getirirken diğeri aynı kuralı pasif hale getirir. Ek-2'de bu scriptin kaynak kodu

verilmiştir. Buradaki koddaki default parametreleri düşünürsek bu uygulama bir iş yerinde her gün iş çıkış saati olan 17:15'te belirli bir kuralı aktif hale getirirken aynı zamanda hergün işe başlama saati olan 8:15'te aynı kuralı pasif duruma getirmektedir. Burada örnek olarak aktif hale getirilen belirli bir trafiği açan bir kural olabilir. Her iki konfigürasyon scriptinin UML (Unified Modelling Language) sınıf diyagramı Şekil 4.4'te gösterilmiştir.

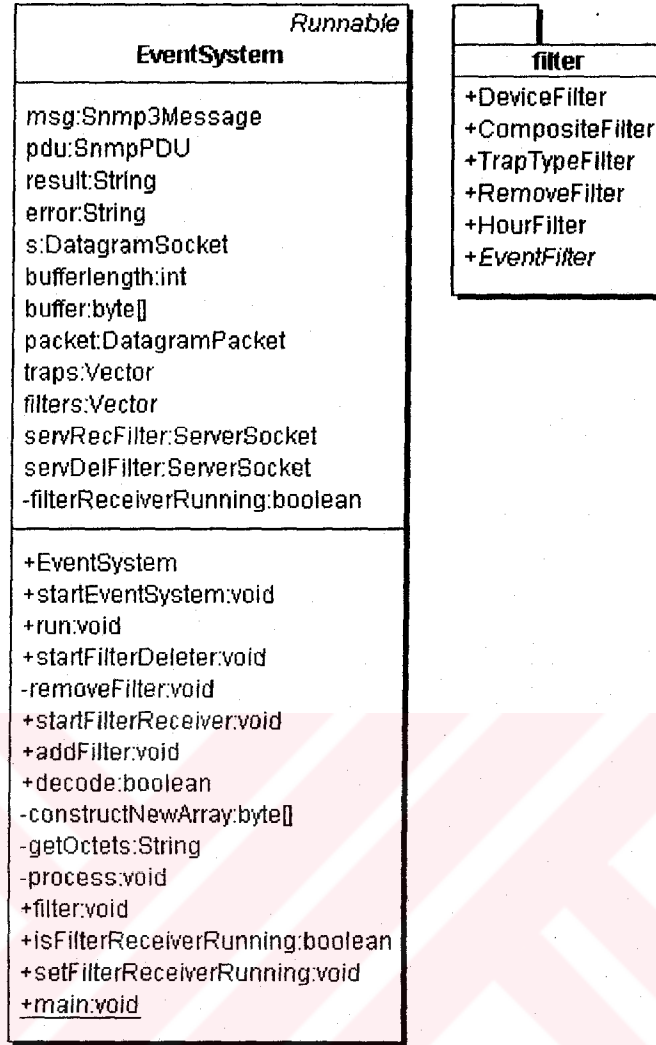




Şekil 4.4. Konfigürasyon script'lerinin UML sınıf diyagramı

4.1.3 Olay Scriptleri

Olay scriptleri bir önceki bölümde de bahsedildiği üzere belirli bir olay sistemine göre hazırlanmış içlerinde filtre ve faaliyet bilgisi bulunduran Java nesneleridir. Bu scriptler konfigürasyon scriptleri gibi bir program olarak AY'lere yollansa da bu programların çalışma mantıkları konfigürasyon scriptlerinkinden farklıdır. Script çalıştığında yaptığı sadece filtre nesnesi oluşturup bunu olay sistemine kaydettirmekten ibarettir. Bu kısa işi yaptıktan sonra sonlanırlar. Aslında burada esas işi yapan olay sistemidir. Olay sistemi IPsec geçitlerinden gelen bildirileri toplar ve bunları kendisine kaydettirilen filtreleri kullanarak değerlendirir. Burada uygulama olarak geliştirilen olay sisteminden ve olay scriptlerinden bahsedilecektir.



Şekil 4.5. event paketinin UML sınıf diyagramı

Öncelikle olay sisteminden bahsetmek gerekirse olay sistemi konfigürasyon scriptleri gibi tek başına çalışan bir Java programıdır. Buradaki olay sisteminin en önemli özelliği bunun kendisinin de aslında bir konfigürasyon scripti gibi yönetim tarafından AY'ye delege edilebilir yapıda olmasıdır. Olay sistemi **EventSystem** adında tek bir sınıftan oluşur ve içinde de bir **main** metodu bulunmaktadır. Bu **main** metodu önce **EventSystem** sınıfının bir nesnesini oluşturur ve bu nesne üzerinde çalışan iki adet thread oluşturur. Bu threadlerden birisi filtre kaydı için soketten **EventFilter** sınıfı tipinden filtre kabul ederken diğeri ise kayıtlı filtreleri silmek için yine soketten **String** tipinden filtre ismi dinlemektedir. Program bu iki threadi oluşturduktan sonra yine sonsuz döngüde IPsec geçitlerinden SNMP bildirilerini dinleyen bir **startEventSystem** metodunu çağırarak olay sistemini başlatır.

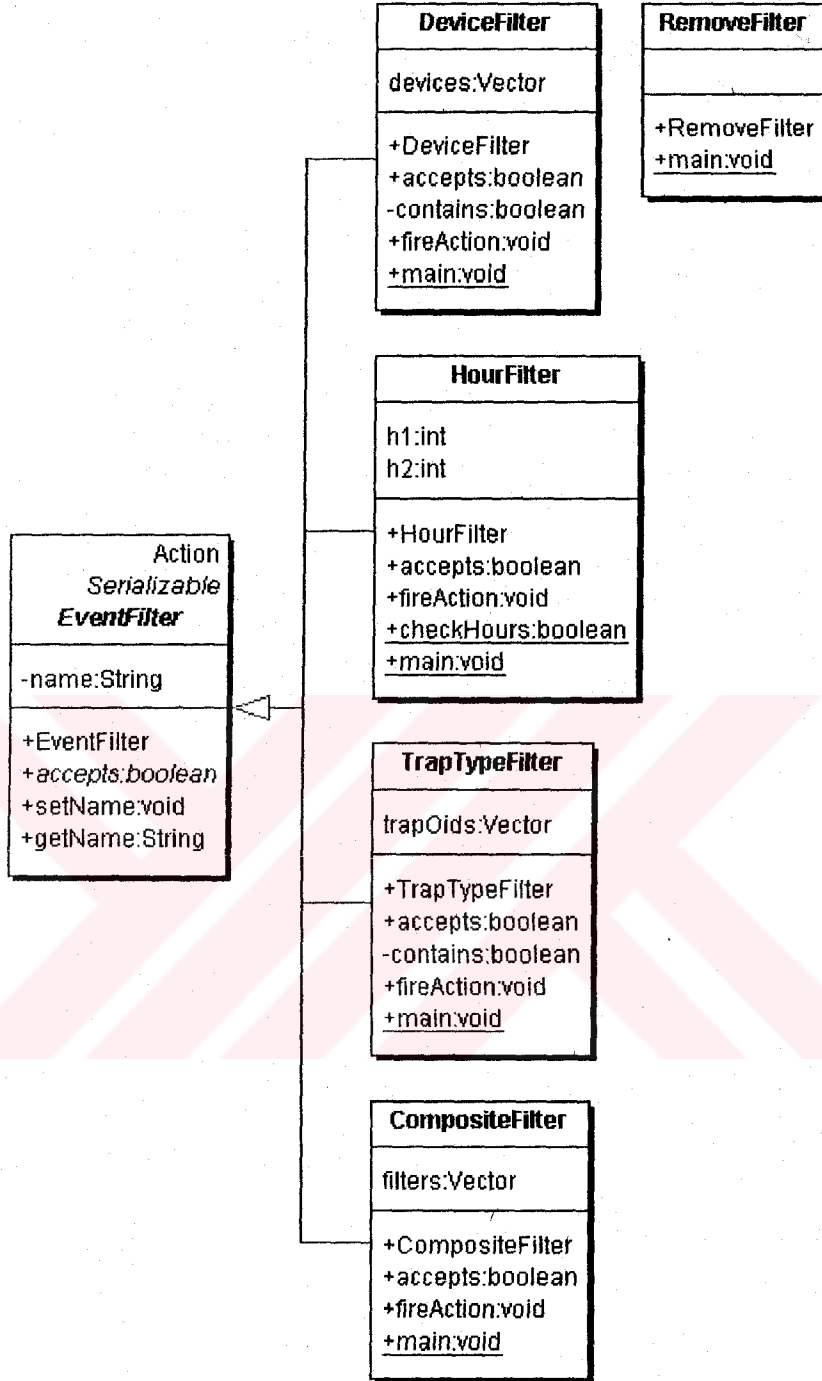
EventSystem UML sınıf diyagramı Şekil 4.5'te gösterilmiştir ve kaynak kodu da Ek-2'de bulunmaktadır. Olay sisteminden bir filtrenin silinmesi için de RemoveFilter isiminde bir script yazılmıştır. Bu script kendine parametre olarak verilen filtre ismini soket bağlantısı ile olay sistemine ulaştırarak filtrenin sistemden silinmesini sağlamaktadır.

Olay sistemi filtre olarak EventFilter tipinden türetilmiş filtre nesnelerini kullanmaktadır. EventFilter nesnesi ise Action soyut sınıfından türemiştir. EventFilter soyut bir sınıftır ve Action sınıfından kalıtımla gelen fireAction metodu haricinde name isimindeki bir parametre, bu parametre için get ve set metotları ile soyut bir accepts metodu tanımlar. EventSystem filtre olarak EventFilter'den türeyen nesneleri kullandığı için bütün filtrelerin accepts(), fireAction(), getName() ve setName() metotları ile name parametresine sahip olduğu garantilenmiş olur ki EventSystem de bütün filtrelerin bu metotları ve parametresi ile muhatap olur. EventSystem bir SNMP bildirisi aldığı anda bundan edindiği bilgilerle IpsecTrap sınıfından bir nesne oluşturur. Artık olay sisteminde bir IPsec geçidinden gelen bir SNMP bildirisi bir IpsecTrap nesnesi ile temsil edilecektir. Daha sonra olay sistemi bu nesneyi kendisine kayıtlı bütün filtrelerden geçirir ve herhangi bir filtreye takılırsa o filtrenin fireAction metodu çağırılarak ilgili faaliyetin yapılması sağlanır. Buradaki filtre ve faaliyet mantığını formülize edilirse bu şu şekilde ifade edilebilir:

IF (CONDITION) THEN ACTION

(4.1.)

Bütün bu filtre ile ilgili sınıflar ve bunlardan türetilen filtre sınıflarının UML sınıf diyagramı Şekil 4.6.'de verilmiştir.



Şekil 4.6. Filtre sınıflarının UML sınıf diyagramı

EventFilter'dan dört adet olay scripti geliştirilmiştir. Bunlar DeviceFilter, TrapTypeFilter, HourFilter ve CompositeFilter'dır. DeviceFilter bildirilerin gönderen IPSec geçidinin IP adresine göre filtrelenmesini sağlar. Mesela YY bu filtreyi kullanarak AY'ye şöyle birşey yaptırabilir:

- Eğer A veya B IP adresli IPSec geçitlerinden bir bildiri gelirse YY'ye bildiri gönder.

HourFilter olay sistemi tarafından alınan bir bildirinin alınış saatinin belirli bir saat aralığında olması durumunda filtrelenmesini sağlar. Bu filtre kullanılarak AY'ye YY tarafından şu şekilde bir iş yaptırılabilir:

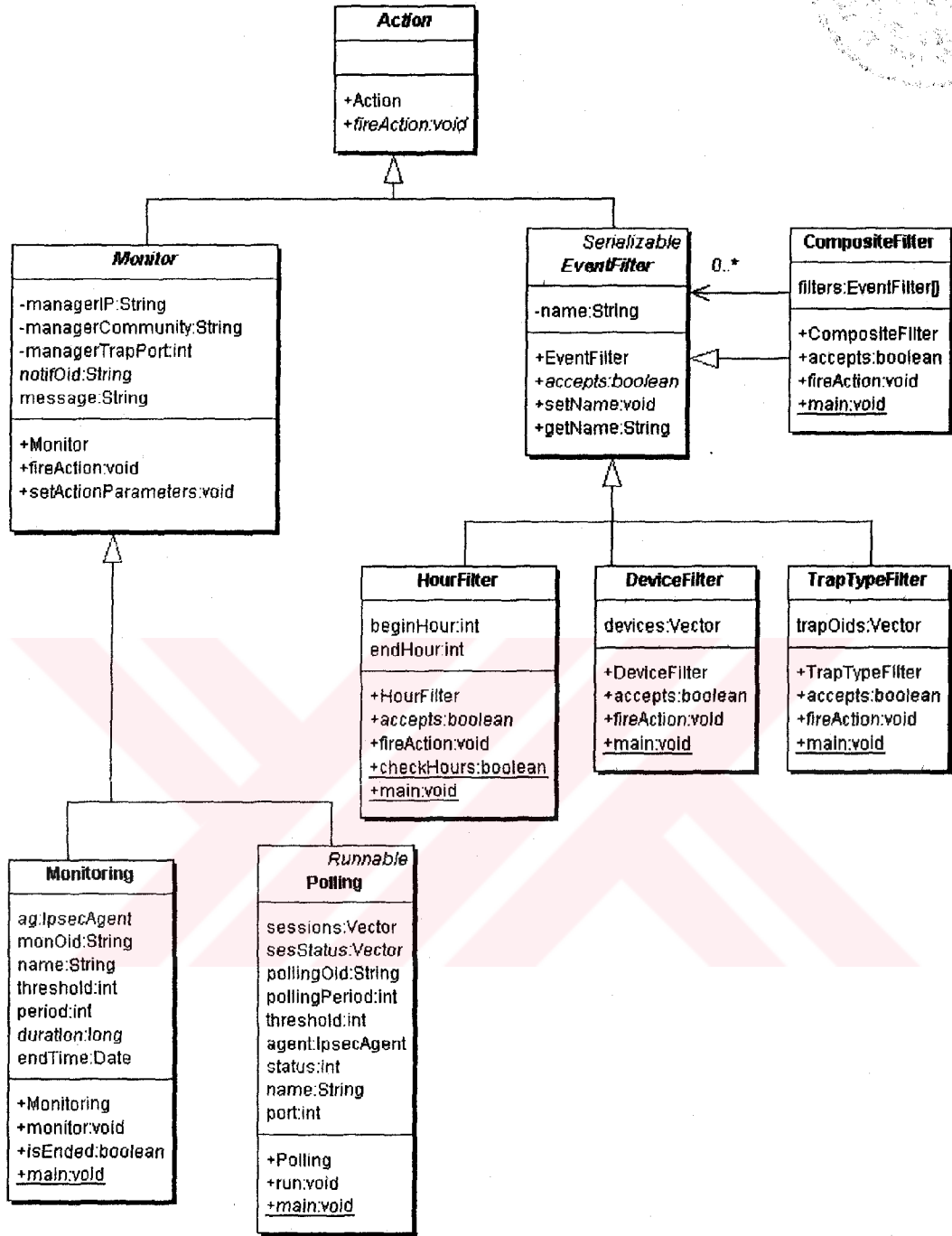
- Eğer gece 1:00 ile 5:00 arasında herhangi bir bildiri gelirse kayıt düş ve sesli uyarı yap.

TrapTypeFilter ise alınan bildirinin tipine göre filtrelenmesini sağlar. Yine bu filtre kullanılarak YY AY'ye şu şekilde bir iş yaptırabilir:

- Eğer C veya D veya E tipinde bir bildiri gelirse YY'ye bildiri gönder.

Burada bahsedilen üç filtre de çok genel filtrelerdir ve tek başlarına çok fazla birşey ifade edemeyebilirler. Bu noktada esas etkili filtre CompositeFilter filtresidir. Bu filtre birden fazla EventFilter tipindeki filtreyi kullanır ve süzme işini bulundurduğu filtrelerin hepsini kullanarak gerçekleştirir. Bu filtre filtrelemede kullandığı diğer filtrelerin faaliyetlerini kullanmaz kendisi ayrıca faaliyet tanımlar. Bu filtre kullanılarak YY tarafından AY'lere delege edilebilecek bazı işler şunlar olabilir:

- Eğer 212.45.33.23 IP adresli IPSec geçidinden ikeSysFailure (IETF IPSec flow monitoring MIB'de tanımlanmış bir bildiri) gelirse YY'ye bildiri gönder.
- Eğer 212.33.55.22 IP adresli veya 212.45.33.23 IP adresli IPSec geçidinden 8:00 ile 17:00 arasında ikeNegFailure (IETF IPSec IKE monitor MIB'de tanımlanmış bir bildiri) veya ipSecProtocolFailure (IPSec flow monitoring MIB'de tanımlanmış bir bildiri) gelirse YY'ye bildiri gönder ve kayıt düş.
- Herhangi bir IPSec geçidinden 0:00 ile 7:00 arasında bir bildiri gelirse admin@company.com.tr adresine email at.



Şekil 4.7. Action sınıfından türeyen gözlem ve olay scriptleri

4.1.4 Gözlem Scriptleri

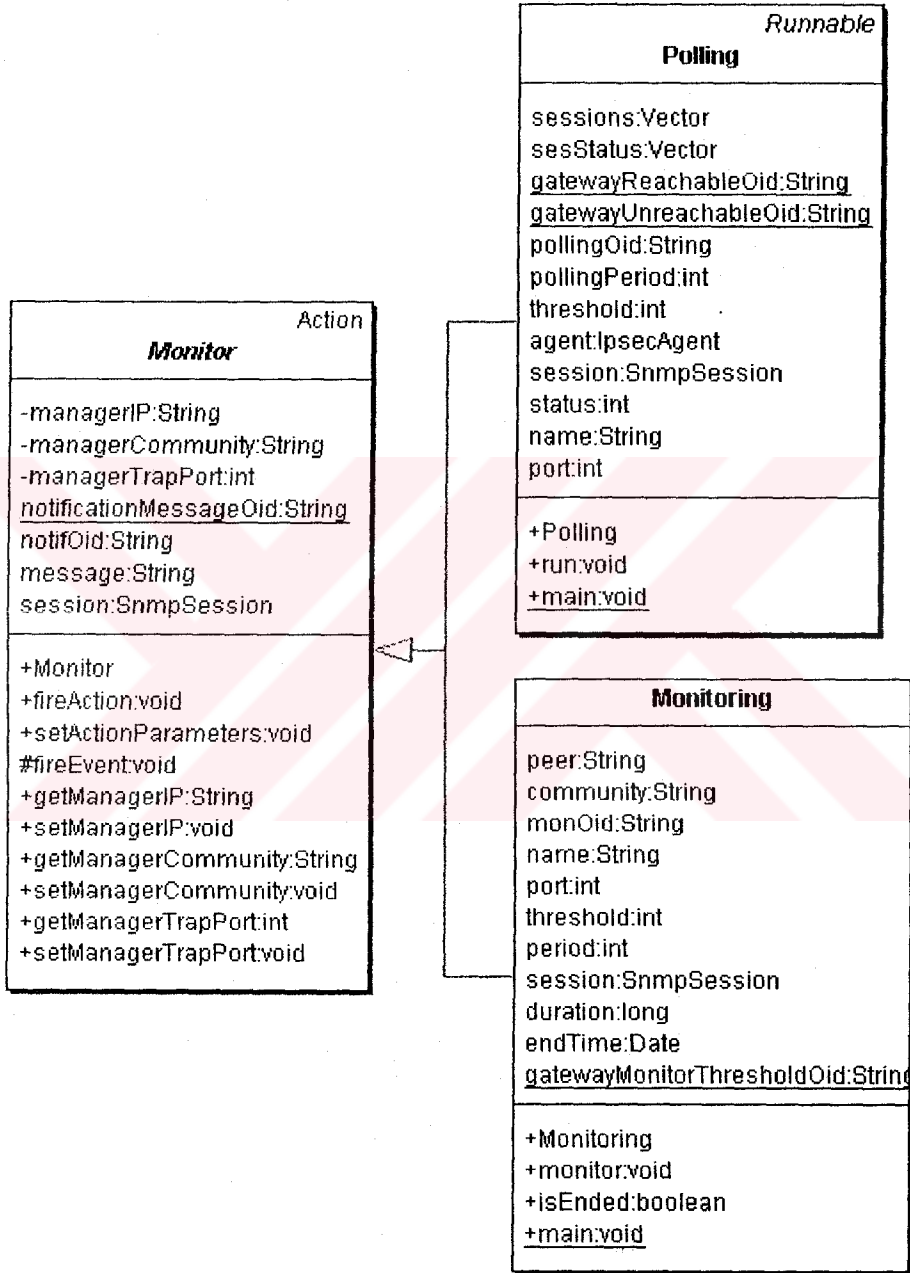
Gözlem scriptleri bahsi geçtiği üzere tek başına çalışabilmesi yönüyle konfigürasyon scriptlerine benzeyen, üzerilerinde belirli bir durumda tetiklenmek üzere bir faaliyet bulundurmaları yönüyle de olay scriptlerine benzemektedirler.

Burada uygulama olarak geliştirilen gözlem scriptleri de bu noktalarda belirginliklerini göstermektedirler. Gözlem için geliştirilen iki ayrı script vardır. Bunlardan ilki Polling, ikincisi ise Monitoring adını taşımaktadırlar. Gözlem scriptlerinin UML sınıf diyagramı Şekil 4.8'de gösterilmiştir. Bu her iki script de Monitor adındaki soyut sınıftan türetilmişlerdir. Monitor sınıfı ise olay scriptlerinin türediği EventFilter soyut sınıfının türediği Action sınıfından türemiştir. Bu sınıfta gözlem scriptleri için ortak olabilecek YY'ye SNMP bildirisi gönderme faaliyeti ve bu iş için kullanılacak YY ile ilgili bazı parametreler tanımlanmıştır. Action sınıfından türeyen olay ve gözlem sınıfları Şekil 4.7.'de gösterilmiştir.

Polling scripti AY'ye kayıtlı bütün IPsec geçitlerine belirli aralıklarla bağlanarak SNMP get işlemi yapar. Bu script parametre olarak eşik değeri ve yoklama periyodu alır. Yoklanması istenen her bir geçit için bir thread kaldırılır. Bu SNMP get işlemi için kullanılan API (Application Programming Interface)'lerin ilgili fonksiyon çağırıldığında bloklayarak çalışması yüzünden her bir IPsec geçidi için periyodu tutturmak açısından zorunlu olmaktadır. Her bir cihaz belirlenen periyotlarla yoklanır ve eğer eşik değeri miktarı kadar yoklama seferinde etmeden cevap alınamaz ise bu durumda faaliyet gerçekleştirilir ve bu cihaz ulaşılamaz olarak etiketlenir. Buradaki faaliyet IPsec geçidine ulaşamadığını belirten bir SNMP bildirisinin YY'ye yollanması şeklindedir. Bundan sonraki yoklamalarda faaliyet gerçekleştirilmez. Ancak etmene devam eden yoklamalar sonucunda ulaşım olması durumunda yine faaliyet gerçekleştirilir. Bu faaliyet de IPsec geçidine ulaşıldığını belirten bir SNMP bildirisinin YY'ye yollanması şeklindedir. Bu script sonsuz döngüde YY tarafından durdurulana kadar çalışmaya devam eder.

Monitoring scripti ise AY tarafından IYE'da bulunan sayaçların gözlenmesi için YY tarafından delege edilebilen bir programdır. Bu script de parametre olarak gözlem yapılacak IPsec geçidinin IP adresi, gözlenecek parametrenin OID'i, gözlem periyodu, gözlem süresi ve eşik değerini alır. Monitoring nesnesi oluşturulup monitor() metodu çağırıldıktan sonra bu program belirtilen aralıklarla, belirtilen cihazdan yine belirtilen süre boyunca ilgili sayacı okur ve bir önceki ile arasındaki farkı alır. Burada eğer bazı okumalarda cevap alınamazsa, cevap alınınca alınan değerle bir önceki değerin arasındaki farkı, cevap alınamayan okuma süresine bölerek ortalama bir değer oluşturur. Her durumda eğer iki okuma arasında ortaya

çıkan değer eşik değerini aşıyorsa bu durumda faaliyet tetiklenir. Buradaki uygulamada faaliyet yine Polling scriptindeki gibi bir SNMP bildirisinin YY'ye yollanması şeklindedir. Bu script parametre olarak verilen süre boyunca çalışır ve ardından sonlanır.



Şekil 4.8. Gözlem scriptlerinin UML sınıf diyagramı

Burada bahsedilen her iki gözlem scriptinin de faaliyetleri YY'yi bir SNMP bildirisi ile haberdar etmek şeklindedir. Her iki sınıf da Monitor sınıfından türediği ve `fireAction()` metodunu tekrar tanımlamadığı (override) için bu böyle



olmuştur. Yoksa olay scriptlerinde olduğu gibi gözlem scriptlerinde de hemen her türlü faaliyet (email atma, bildiri yollama, kayıt düşme, sesli uyarma vs.) tanımlanabilmektedir.

Gözlem scriptlerinin bu bölümde anlatılan uygulamalar ile sınırlı olduğu düşünülmemelidir. Buradakiler sadece örnektir. IPSec geçitlerinin gözlemi için ihtiyaca göre çok daha gelişmiş scriptler yazılabilir. Bunlar birden fazla parametreyi aynı anda gözleyip bunlar üzerinde belli bazı algoritmaları kullanarak istatistiksel veri elde etmek şeklinde olabileceği gibi başka türlü de olabilir.

4.2 Sistemin Avantajları ve Analizler

Sistemin uygulanması ve öngörülen tipte yönetim scriptlerinin yazılması ve bunların denenmesi sonucunda sistemin tahmin edildiği gibi merkezi yönetim ve diğer çözümlere göre pekçok avantaja sahip olduğu gerçeği ortaya çıkmıştır. Bu kısımda sistemin avantajları ayrıntılı şekilde açıklanacaktır.

4.2.1 Dağıtık Yönetim

Burada önerilen dağıtık yönetim sisteminin en önemli avantajlarından birisi halihazırda kullanılan ve üzerinde standart SNMP protokolünden başka bir yönetim protokolü bulundurmeyen IPSec yani SNMP ile yönetilebilen IPSec geçitlerine yazılım değişikliği yapmadan dağıtık yönetim imkanı sağlamasıdır. Bir kurum VPN kurmak için bu cihazlardan aldığı zaman daha sonra bunların yönetimi için dağıtık yönetim uygulaması gerektiğinde cihazların yazılımının değişmesi istenen bir şey olmamaktadır. Bu durumda önerilen sistem doğrudan cihazların yönetim yeteneklerini kullanarak onlara dağıtık yönetim imkanı sunmaktadır. Ayrıca bazı üreticilerin yaptığı gibi SNMP ile konfigürasyona izin verilmese dahi bu sistem olay ve gözlem işini de dağıtarak başarı ile gerçekleştirebilmesi yönüyle bu tip cihazların yönetim ihtiyacının belirli bir kısmını halletmektedir.

4.2.2 Standartlar Kullanılması

Burada geliştirilen dağıtık yönetim sistemi tamamen standartlaşmış çözümler kullanmaktadır. YY ile AY arasında YY'nin AY'ye kontrol mesajları ve AY'nin

YY'ye gönderdiği bildiriler SNMPv3 protokolünü kullanmaktadır. Aynı şekilde AY'lerin IYE'lere gönderdiği mesajlar ve alınan cevaplar SNMPv3 protokolünü kullanmaktadır. Ayrıca AY'ler scriptleri YY'nin script ambarından HTTP ile çekmektedir. Bütün bu mesajlaşmaların hepsi standart protokoller ile gerçekleşmektedir. Ayrıca Script MIB'de IETF'de standartlaşmış bir MIB'dir ki bu da pek çok şeyi kolaylaştırmaktadır.

4.2.3 Kod Dağıtımı

Dağıtık yönetim sistemlerinde bulunması gereken en önemli özelliklerden birisi de kod dağıtımına izin vermesidir. Bu gerçekleştirilen sistemlerin zaman içerisinde yeni ihtiyaçları karşılamak ve bu sistemlerde daha sonradan ortaya çıkabilecek hataları düzeltmek açılarından büyük önem arz etmektedir. Merkezi yönetim sistemleri tek noktadan düzeltmeler ile bu ihtiyaçları giderebilse dahi diğer dağıtık çözümler bu özelliklere sahip olmadıkları sürece gerekli esnekliği sağlamış olmazlar. Script MIB tabanlı dağıtık yönetim sistemi çözümü bu noktada kod dağıtımına izin vermesi, hatta yönetim fonksiyonlarının hepsinin dağıtılabilen kodlardan oluşması bahsedilen diğer çözümlerden avantajlı konuma geçmesini sağlamaktadır.

4.2.4 Ölçeklenebilirlik

Dağıtık yönetim sistemlerinin ortaya çıkmasının belki de en önemli sebebi ölçeklenebilirliktir. Tek noktadan yönetim belirli bir aşamaya kadar yeterli gibi görünse de yönetimi gereken cihaz sayısı arttığında yetersizlikler baş göstermektedir. Bu problem IPsec geçitlerinin yönetiminde de aynı şekildedir. Birinci bölümde de bahsedildiği üzere bu cihazların yönetim ihtiyacı belirli bir sayıdan sonra dağıtık yönetimi gerektirmektedir. Belirli bir sayıda IPsec geçidi merkezi yönetim ile idare edilebilse bile bu cihazların sayısının artması durumunda ölçeklenebilirlik problemleri ortaya çıkacaktır.

Bu yönden Script MIB tabanlı yönetim sisteminde cihazların sayısı ne kadar artarsa artsın hiyerarşik yönetim söz konusu olduğundan ve YY IYE'ler ile doğrudan muhatap olmadığından sistem bu cihazların yönetimini kaldırabilmektedir.

Bu noktada Bölüm 4.1'de anlatılan uygulamaları hesaba katarak burada

önerilen sistemin merkezi yönetimle karşılaştırması yapılacaktır. Herbir scriptin kurulması ve işletilmesi YY'ye belirli bir maliyet getirmektedir. Karşılaştırma için YY'nin scriptte yapılan işi doğrudan kendisi yapması ile bir AY'ye yaptırmasını kıyaslamak gerekmektedir.

Burada hesaplamalarda maliyet olarak bahsedilenler işlemin yapılma süresi + CPU meşguliyeti olarak ele alınmıştır. Ortalama değer olarak SNMP ile bir tabloya bir satır eklemenin maliyeti 2 birim, bir veya iki parametrenin okunması veya yazılması işleminin maliyeti de 1 birim ve bir bildirinin işlenmesi de 2 birim olarak hesaplanacaktır.

4.2.4.1 Konfigürasyon Scriptlerinin Ölçeklenebilirliği

Öncelikle konfigürasyon scriptleri ele alınacaktır. Konfigürasyon scriptlerinden tünel scriptini ele alırsak:

K : Bir scriptin YY tarafından AY'ye kurulmasının YY'ye maliyeti (Bu maliyet smScriptTable'a bir satır eklemek şeklindedir ki bu işin ardından AY belirtilen URL'den scripti kendi alır)

B : Bir scriptin her seferinde çalıştırılması için smLaunchTable'a bir satır eklenip smLaunchStart parametresinin set edilmesi maliyeti

S : Bu scriptin çalıştırılmasından sonra sonuç değerini okuma maliyeti (Bu maliyet smRunTable'ın bir satırından iki parametre okumak şeklindedir)

Bu durumda bu scriptin n kez çalıştırılması durumunda toplam maliyet (TMS):

$$TMS = K + n*(B + S) \quad (4.1.)$$

Bu maliyet birimi olarak ifade edilirse:

$$K = 2 \text{ birim}, B = 2+1 = 3 \text{ birim ve } S = 1 \text{ birim.}$$

$$TMS = 2 + n*(3+1) = 2 + 4n \quad (4.2.)$$

Bu noktada manuel tünel kuran scriptin başarılı olarak çalışmasının maliyetini hesaplanırsa:

Bir IPsec geçidinin tablolarının ayarlanmasının maliyeti:

$$\text{İki politika kaydı} + \text{bir GB kaydı} + \text{bir KH kaydı} = 2 + 1 + 1 = 4 \text{ birim}$$

Tabloların aktif hale getirilmesinin maliyeti: 1 birim

Bu durumda iki cihazda konfigürasyon yapmanın toplam maliyeti TMK:

$$TMK = 2 * (4 + 1) = 10 \text{ birim}$$

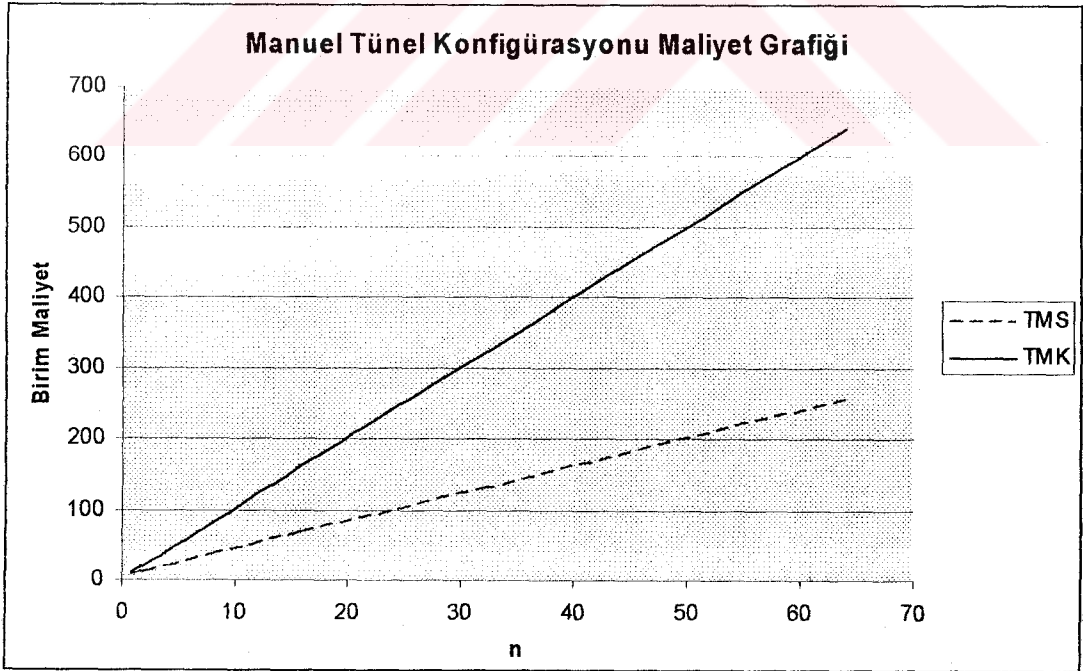
Bu işlemin n kez yapılmasının maliyeti ise:

$$TMK = 10 * n \quad (4.3.)$$

Bu noktada manuel tünel kurmanın script çalıştırma ile yapılması durumunda YY'ye maliyeti TMS ile aynı işlemin YY tarafından doğrudan yapılmasının maliyeti TMK değerlerini n'e bağlı bulduk. Bu maliyetlerin n'e bağlı grafiği Şekil 4.9.'da gösterilmiştir. Bu grafikte görüldüğü üzere YY'ye maliyeti yönüyle AY'ye script gönderilmesi ile tünel konfigürasyonunun yaptırılması her durumda bu işin doğrudan YY tarafından yapılmasından karlı olmaktadır. Burada delegasyon verimliliğini hesaplanırsa:

$$V = TMK / TMS \quad (4.4.)$$

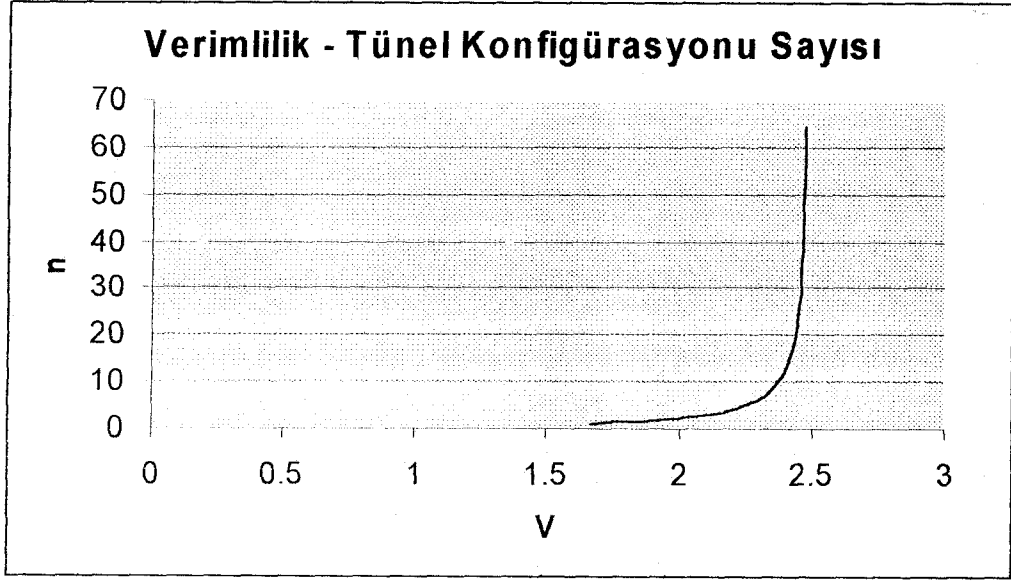
$$V = (10*n) / (2 + 4*n) \quad (4.5.)$$



Şekil 4.9. Manuel tünel konfigürasyon maliyet grafiği

Manuel tünel konfigürasyonu için delegasyon verimliliği Şekil 4.10'de gösterilmiştir. Bu şekilde görüldüğü üzere yapılması kurulan tünel sayısı arttıkça

verimlilik artmakta ve en limitte 2.5'e ulaşmaktadır.



Şekil 4.10. Tünel konfigürasyonu sayısı-verimlilik grafiği

Burada ise konfigürasyonu zamana programlama scriptinin yine AY'lerde çalıştırılması ile bu scriptin yaptığı işin doğrudan YY tarafından yapılması kıyaslanacaktır. Bu scriptin yaptığı işin bütün cihazlarda uygulandığı düşünülerek aşağıda modelleme gerçekleştirilecektir.

K : Scriptin bir AY'ye kurulması maliyeti

B : Scriptin bir AY'de çalıştırılması maliyeti

S : Bu scriptin sonucunun bir günde okunması maliyeti

m : Toplam AY sayısı

g : Yönetilen toplam geçit sayısı

T : Bu scriptin çalıştırıldığı toplam gün sayısı

p : Konfigürasyon değişikliğinin yapılma periyodu (saat olarak)

$$TMS = (K + B) * m + (m * S) * T \quad (4.6.)$$

$$K = 2 \text{ birim}, B = 2+1=3 \text{ birim}, S = 1 * 24/p \text{ birim}$$

$$TMS = 5 * m + m * T * 24/p \quad (4.7.)$$

Buradaki scriptin yaptığı iş YY tarafından yapılması durumunda:

G : Yapılmak istenen zamana programlanmış konfigürasyonun bir günlük maliyeti

$$TMK = T * G * g \quad (4.8.)$$

$G = (1+1) * 24/p = 48/p$ (Politikada bir kuralın aktif yapılması ve pasif yapılması)

$$TMK = T * g * 48/p \quad (4.9.)$$

TMK ve TMS fonksiyonları ortaya çıktı. Programlanan konfigürasyonun periyodunu 1 gün, toplam AY sayısını 4 ve yönetilen toplam IPsec geçidi sayısını da 100 alırsak:

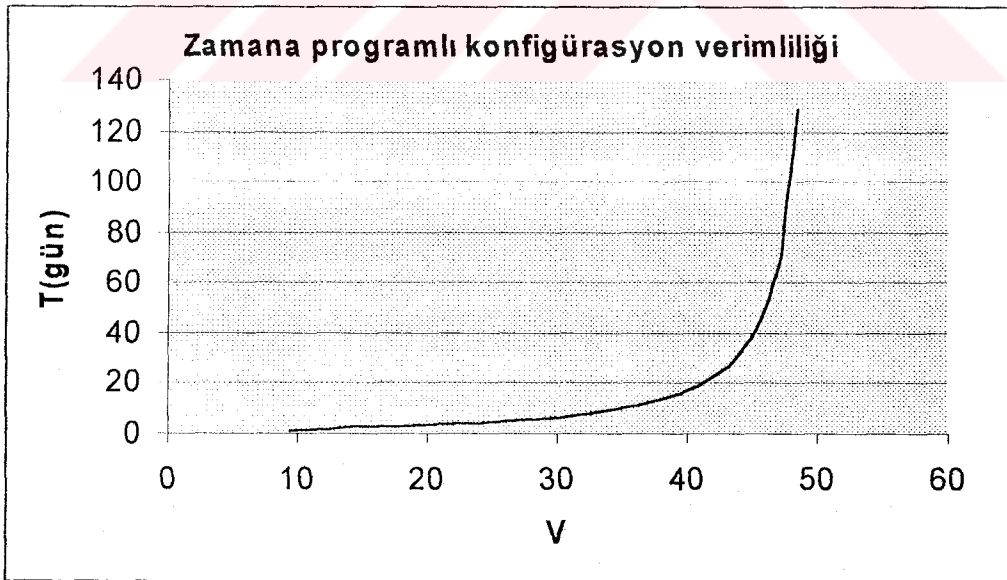
$$p = 24 \text{ saat}, m = 4, g = 100$$

$$TMS = 5*4 + 4*T*24/24 = 20 + 4 * T \quad (4.10.)$$

$$TMK = T * 100 * 48/24 = 200*T \quad (4.11.)$$

Çıkarılan formüllerden (4.11.) ve (4.12.)'un zamana bağlı grafiği Şekil 4.12.'de gösterilmiştir. Burada yine delegasyon verimliliğini (4.5.)'e göre hesaplamak gerekirse:

$$V = TMK / TMS = (200*T) / (20 + 4*T) \quad (4.12.)$$



Şekil 4.11. Zamana programlı konfigürasyon verimlilik grafiği

Şekil 4.11.'de (4.12.)'de çıkarılan formülün zamana bağlı grafiği çizilmiştir. Grafikte görüldüğü üzere verimlilik limitte 50'ye ulaşmaktadır.

Cihaz sayısı ve AY sayısı harf olarak bırakılırsa:

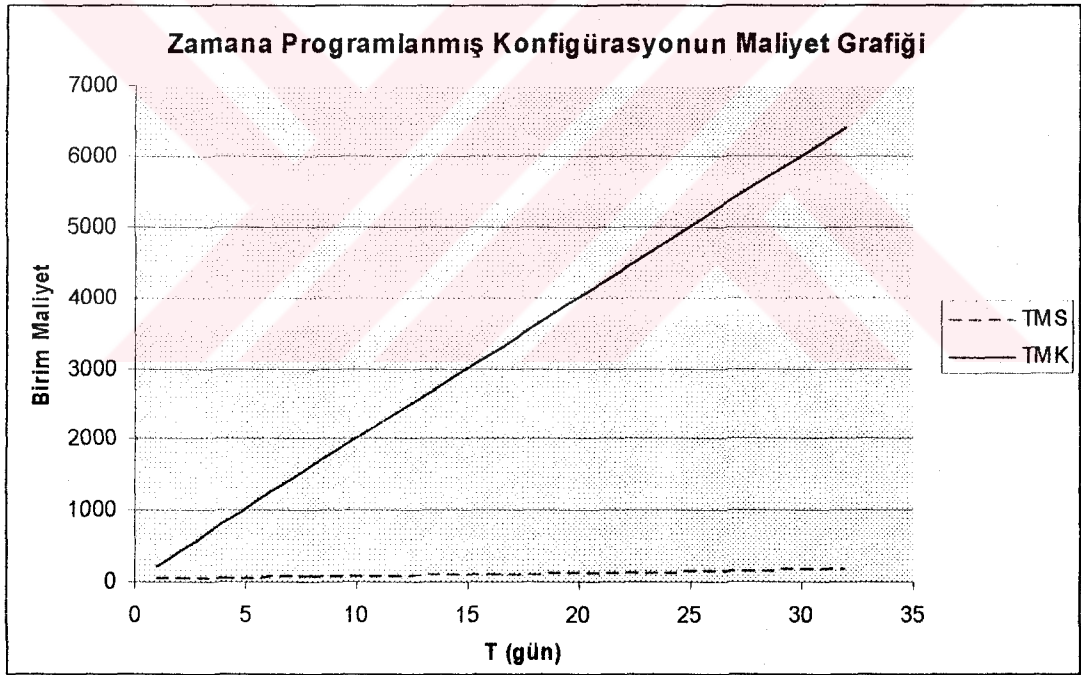
$$V = (2 * T * g) / (m * (T + 5)) \quad (4.13.)$$

Formül (4.13.) incelendiğinde delegasyon yapmak IPsec geçit sayısının artması ile artmakta fakat AY'lerin artması ile de azalmaktadır. Yalnız bu formülde bu değerleri tek başlarına ifade etmek doğru olmamaktadır. Burada önemli nokta g/m oranını yüksek tutmakla gerçekleşmektedir. Formül bu açıdan tekrar değerlendirilirse:

O : Bir AY'ye düşen ortalama IPsec geçidi sayısı (Alanlardaki ortalama geçit sayısı g/m)

$$V = (2 * T * O) / (T + 5) \quad (4.14.)$$

Bu durumda verimlilik O ile yani bir AY'ye düşen ortalama IPsec geçidi sayısı ile doğru orantılı olarak artmaktadır.



Şekil 4.12. Zamana programlanmış konfigürasyon maliyet grafiği

4.2.4.2 Olay Scriptlerinin Ölçeklenebilirliği

Bu kısımda olay scriptlerinin AY'lere delege edilmesi ile YY'nin üzerinden ne kadar yükün kalktığı ve bu işin verimliliği analiz edilecektir.

Bu iş için yönetilen bütün cihazlardan gelen SNMP bildirilerinin miktarı ve bunların işlenmesi maliyetini modellemek gerekmektedir.

g : Yönetilen IPSec geçidi sayısı

b : Bir geçidin günde ortalama ürettiği bildiri sayısı

p : Bir bildirinin değerlendirilmesinin maliyeti (süzülüp ilgili bir faaliyetin tetiklenmesi)

Bu durumda bütün cihazlarda bir günde üretilen ortalama bildiri sayısı:

$$\mathbf{TBS = g * b} \quad (4.1.)$$

Bütün bu bildirilerin işlenmesinin günlük maliyeti:

$$\mathbf{BM = TBS * p = g * b * p} \quad (4.2.)$$

T gün kadar bu iş yapıldığında toplam bildiri işleme maliyeti:

$$\mathbf{TBM = BM * T} \quad (4.3.)$$

Merkezi yönetim olması durumunda BM yükünün tamamı merkezi yönetimin üstüne düşmektedir. Bunun yerine bu bildirilerin işlenmesi işi AY'lere delege edilmesi durumu düşünülür ve bu iş için kullanılan olay scriptlerinin faaliyet olarak yine AY'ye bildiri yolladığı kabul edilirse bunun merkezi yönetime maliyet durumu şu şekilde olmaktadır:

K : Scriptin bir AY'ye kurulması maliyeti

B : Scriptin bir AY'de çalıştırılması maliyeti

F : Delege edilen ortalama olay scripti sayısı

m : Toplam AY sayısı

k : Bir filtreye günlük ortalama takılan ve bunun sonucunda faaliyet tetiklenen bildiri sayısı (Faaliyet olarak YY'ye bildiri gönderilmesi kabul edilirse bu sayı YY'ye ulaşan bildiri sayısına eşit olmaktadır)

T : Delegasyonun sürdüğü toplam gün sayısı

Bu durumda olay scriptlerinin kurulması ve çalıştırılması maliyeti:

$$\mathbf{OSM : (K + B)*m*F} \quad (4.4.)$$

YY'nin bildiri işlenmesinin günlük maliyeti:

$$\mathbf{BM = k * p} \quad (4.5.)$$

Bu işlemlerin YY'ye güne bağlı toplam maliyeti:

$$TSM = OSM + T \cdot BM = (K + B) \cdot m \cdot F + k \cdot p \cdot T \quad (4.6.)$$

Bu noktada yukarıda hesaplanan TBM ile TSM oranlanırsa bu noktada delegasyon verimliliği hesaplanabilir. Bu iş için yine konfigürasyondaki gibi bazı parametrelere değer verilecektir. Daha önce p 2 birim maliyet olarak belirlenmişti. Bu durumda:

$$K = 2, B = 3 \text{ ve } p = 2 \text{ için,}$$

$$TBM = g \cdot b \cdot 2 \cdot T$$

$$TSM = 5 \cdot m \cdot F + 2 \cdot k \cdot T$$

Burada cihaz sayısına yine 100, AY sayısına da yine 4 verirsek ve cihaz başına da 5 olay scripti kurulduğu düşünülünce:

$$TBM = 200 \cdot b \cdot T$$

$$TSM = 20 \cdot 5 + 2 \cdot k \cdot T = 100 + 2 \cdot k \cdot T$$

Bu durumda verimlilik:

$$V = TBM / TSM = 200 \cdot b \cdot T / (100 + 2 \cdot k \cdot T) \quad (4.7.)$$

Bu çıkan verimlilik formülü cihaz başına üretilen günlük ortalama bildiri sayısı b ile doğru orantılı, filtreye takılan bildiri sayısı k ile de ters orantılıdır. Bu noktada yine scriptlerin kurulma maliyetini temsil eden paydadaki 100 rakamı ihmal edilirse (çünkü bu iş bir kez yapıldıktan sonra değişiklikler haricinde çok sık yapılmaz) bu durumda b/k oranı verimliliği belirleyen unsur olmaktadır. Etkili filtreler yazılıp k oranı düşük tutulması durumunda verimlilik artacaktır. Verimliliğin yaklaşık bir değeri hakkında bir fikir edinmek için ortalama değerler ile verimliliği hesaplanırsa:

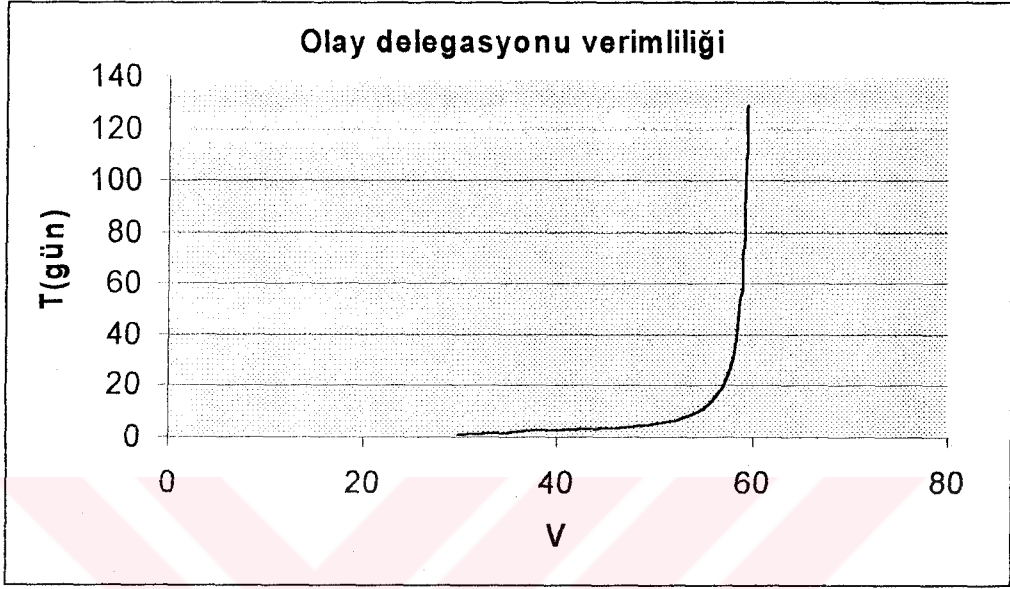
$$b = 30, k = 50 \text{ olsun bu durumda:}$$

$$V = 200 \cdot 30 \cdot T / (100 + 2 \cdot 50 \cdot T) = 6000 \cdot T / (100 + 100 \cdot T)$$

$$V = 60 \cdot T / (1 + T) \quad (4.8.)$$

Şekil 4.13'te olay delegasyonu verimliliği ile zaman arasındaki grafik çizilmiştir. Limitte verimin 60 olduğu görülmektedir. Tabii daha önce de bahsedildiği gibi verim sabit değildir. Bu işin uç noktası düşünülürse eğer tamamen inefektif olay scriptleri AY'lere delege edilirse bu durumda AY'ler gelen her bir

bildiri için YY'ye bildiri yollamak durumunda olacaklardır ki bu durumda delegasyonun hiçbir anlamı kalmamakta ve bu işin verimliliği de yaklaşık bir olmaktadır. Demek ki verimlilik 1'den yukarı doğru herhangi bir değere doğru uzanacaktır. Yine de kötü filtreler yazılsa dahi bu işin verimli olduğu aşikardır.



Şekil 4.13. Olay delegasyonu verimliliği-zaman grafiği

4.2.4.3 Gözlem Scriptlerinin Ölçeklenebilirliği

Bu kısımda olay scriptlerinin AY'lere delege edilmesi ile YY'nin üzerinden ne kadar yükün kalktığı ve bu işin verimliliği analiz edilecektir. Bir önceki bölümde bahsedilen Polling scripti analiz edilirse:

Bu işin tek bir yönetim merkezinden yapılması durumunda bunun yönetime maliyeti şu şekilde modellenenbilmektedir:

p : IPsec geçitlerinin yoklanma periyodu (saniye olarak)

g : Toplam geçit sayısı

Y : Bir cihazın bir kez yoklanmasının maliyeti

Bir günde cihazların toplam yoklanması maliyeti:

$$TGM = g * Y * 24 * 60 * 60 / p \quad (4.1.)$$

Y = 1 ve T gün kadar bu iş yapıldığında toplam yoklama maliyeti:

$$TGM = T * g * 24*60*60/p$$

(4.2.)

Aynı işin AY'lere delege edilmesi şeklinde yapılmasının maliyeti de şu şekilde hesaplanmaktadır.

K : Scriptin bir AY'ye kurulması maliyeti

B : Scriptin bir AY'de çalıştırılması maliyeti

m : Toplam AY sayısı

H : Scriptin çalışması sonucu günde oluşan ortalama yoklama bildirisi sayısı

D : Bir bildirinin işlenmesinin maliyeti

$$TSM = (B + K)*m + T * D * H \quad (4.3.)$$

K = 2, B = 3, D = 2 durumunda:

$$TSM = 5 * m + T * 2 * H \quad (4.4.)$$

Bu noktada delegasyon verimliliği:

$$V = (T * g * 24*60 * 60) / (p*(5 * m + 2 * T * H)) \quad (4.5.)$$

Hergün ortalama bir cihaza iki kez erişimin kopup geldiği kabul edilirse bu durumda $H = g * 4$ olmaktadır. Ayrıca her bir cihazın ortalama 10 saniyede bir yoklandığı kabul edilirse ve bu durumda yine delegasyon verimliliği hesaplanırsa:

$$V = TGM / TSM = (T * g * 24*60 * 60/10) / (5 * m + T * 2 * g * 4)$$

$$V = (8640 * g * T) / (5 * m + 8 * g * T) \quad (4.6.)$$

Burada yine $m = 4$ ve $g = 100$ alıp verimlilik zamana göre düşünülürse:

$$V = 8640 * 100 * T / (20 + 800 * T) = 8640 * 5 * T / (1 + 160 * T) \quad (4.7.)$$

Bu noktada paydadaki 1 rakamı ihmal edildiğinde verimliliğin 270 çıktığı görülmektedir. Tabii ki bu rakam yine kabullenmelere göre ortaya çıkmaktadır. Örneğin burada yoklama periyodu olarak 10 saniye kabul edilmiştir. (4.28.)'deki verimlilik formülü incelenirse yoklama işi delegasyon ile yapıldığında bunun verimliliğinin periyot ile ters orantılı olması manidardır. Ayrıca yine verimlilik yönetilen toplam geçit sayısı ile doğru orantılıdır. Yani eğer IPsec geçitlerinin yoklanması saatte bir yapılan bir aktivite olsa bu durumda bu işin AY'lere delege

ettirilmesinin çok fazla bir anlamı olmamaktadır. Fakat şu gözardı edilmemelidir ki IPSec geçitleri gibi güvenlik cihazları ağ altyapılarının kritik yerlerinde bulunmaları yönüyle pratikte çok sık gözlenmesi gereken cihazlardır. Bu durumda cihaz sayısının da fazla olması durumunda bu işin AY'lere delege edilip yaptırılması hem YY'nin trafiğinin azalması hem de işlem gücü yönünden yükünün azalması açısından gayet faydalıdır.

Monitoring scripti birçok cihaz için çalıştırılması durumunda Polling scripti ile benzerlik göstermesi yönüyle burada ayrıca analiz edilmeyecektir.

4.2.5 Hata Kotarma

Genelde merkezi yönetim sistemlerinin en büyük problemlerinden birisi de bu konuda olmaktadır. Yönetim merkezi bir adet olduğundan yönetim merkezinin cihazlarla haberleşmesinin kopması durumunda veya arıza durumlarında yönetilen cihazlar başıboş kalmaktadırlar. Burada önerilen dağıtık yönetim sisteminin bir özelliği de sistemde tek hata noktası bulunmamasıdır. Yani örneğin YY ile AY'lerin bağlantısı koparsa veya YY arızalanırsa AY'ler kendilerine delege edilmiş işleri devam ettirirler. Aynı şekilde YY'nin herhangi bir AY ile bağlantısı koparsa veya bir AY arızalanırsa o AY'nin yönettiği alandaki cihazların yönetimini YY doğrudan ele alıp gerekli konfigürasyon, gözlem veya olay yönetimi işlerini bir merkezi yönetici edası ile gerçekleştirebilir. YY'nin merkezi yönetim özelliklerine de sahip olması yönüyle doğrudan geçitlere müdahale edebilmesi sisteme ayrıca esneklik getirir..

4.2.6 Tekrar Kullanılabilirlik

Bu sistemde AY'lere delege edilen scriptler tekrar kullanılabilir özelliktedirler. YY aynı scripti farklı parametreler ile çalıştırabilmektedir. Mesela geliştirilen scriptlerden ManualTunnelScript ve ScheduledConfig tekrar kullanılabilirliği yüksek scriptlerdir. Hatta bunlar içlerinde hata bulunmadığı veya farklı MIB'ler ile muhatap olunmadığı müddetçe kullanılabilirler. Benzer şekilde yine geliştirilen olay scriptleri DeviceFilter, TrapTypeFilter, HourFilter ve CompositeFilter ile olay sisteminden filtre silen RemoveFilter scriptleri tekrar tekrar kullanılabilirler. Özellikle bunlardan CompositeFilter tekrar kullanılabilirliği ve etkinliği en yüksek olandır. Gözlem filtreleri Polling ve



Monitoring de aynı şekilde farklı parametreler ile tekrar tekrar çalıştırılabilir. Yalnız Polling scriptinin bir AY'de fonksiyonalite yönünden bir kez çalıştırılması gerekmektedir.

4.2.7 AY'lerin Esnek Yapısı

AY'lerin en önemli özelliği çok esnek olmalarıdır. Bu esneklik bütün yönetim fonksiyonalitelerinin script halinde olmasından kaynaklanmaktadır. Bunlardaki olay sistemi dahi Script MIB tarafından çalıştırılabilecek bir Java programı şeklinde yazılabilmektedir ve uygulamada da EventSystem ile örneği gösterilmiştir. AY'ler başlangıçta üzerlerinde gerekli kütüphaneler, veritabanı, Script MIB ve script ortamından başka bir şeye gereksinim duymazlar. Gerekli her türlü yönetim görevi YY tarafından script kodları aracılığı ile AY'lere yaptırılmaktadır. Tabii bu da AY'lerin kullanım açısından büyük esneklikler sağlamaktadır. Pek çok dağıtık ve merkezi yönetim sistemi bu esnekliğe sahip değildir. Genelde yönetim sisteminin kodu yazılmıştır ve sabittir. Herhangi bir değişiklik için üretici tarafından yenileme gerekmektedir. Fakat burada böyle bir şey söz konusu değildir. Hazır müsvette olarak yazılan scriptlerin ufak değişiklikleri ile zaman içinde ortaya çıkan değişik gereksinimler kolayca karşılanabilmektedir.

4.2.8 MIB Bağımlılığı Olmaması

Script MIB tabanlı dağıtık yönetim sistemi SNMP ile yönetilebilen hemen her IPsec geçidini yönetebilecek esnekliktedir. Değişik üreticilerin özel MIB'lerine adaptasyon kolayca sağlanabilmektedir. Farklı cihazların yönetimi için farklı scriptler kullanılarak bu kolayca gerçekleştirilebilmektedir. Bu noktada herhangi bir üreticiye veya bir MIB'e bağımlılık söz konusu olmamaktadır. Bu özellikle sabit kodlu yönetim sistemleri için problem olabilmektedir.

Çizelge 2.1'de, geliştirilen IPsec/VPN geçitleri için Script MIB tabanlı dağıtık yönetim sisteminin diğer çözümlerle kıyaslanması tablo halinde gösterilmiştir.



Çizelge 4.1. Çözümlerin kıyaslama tablosu

Kıyaslama ölçüsü	Script-MIB Tabanlı Dağıtık Yönetim Sistemi	Merkezi Yönetim	SPS	Politika- tabanlı yönetim
Dağıtık Yönetim	İyi	-	İyi	İyi
Standartlar Kullanılması	İyi	İyi	Kötü	Kötü
Kod Dağıtımı	Var	Yok	Yok	Yok
Ölçeklenebilirlik	İyi	Kötü	İyi	İyi
Hata Kotarma	İyi	Kötü	Kötü	İyi
Tekrar kullanılabilirlik	İyi	İyi	İyi	İyi
Yazılım yenileme esnekliği	İyi	İyi	Kötü	Kötü
MIB bağımlılığı	Yok	Yok	-	-
Özel etmen gereksinimi	Yok	Yok	Var	Var
Geniş kapsamlı yönetim	İyi	İyi	Kötü	İyi
Kullanım kolaylığı	İyi	İyi	İyi	Çok iyi

5 SONUÇLAR VE ÖNERİLER

Günümüz bilgi sistemleri günden güne daha çok eleman içeren ve kompleks bir hal almaktadır. Yönetilmeyen bir sistemin, cihazın ve hatta yazılımın istenen faydaları sağlaması düşünülemez. Bu yüzden bilgisayar ağlarının ve bilgi sistemlerinin gelişmesine paralel olarak yönetim sistemleri ve teknolojileri de gelişmiştir. Özellikle IP alanında standart hale gelen SNMP protokolü bugün hemen her türlü ağ cihazında, kişisel bilgisayarlarda ve iş istasyonlarında bulunmaktadır. Ağ elemanlarının yanında ağların güvenliğini sağlayan güvenlik cihazlarının da çok önemli yönetim gereksinimleri vardır. Bu çalışmanın konusu olan IPSec/VPN geçitleri de günümüzde kullanımı günden güne artan çok önemli güvenlik cihazlardır. Pekçok büyük ticari şirket, devlet kurumları ve askeri birimler Internet altyapısını kullanarak kendi özel ağlarını bu güvenlik sistemlerini kullanarak oluşturmaktadır.

IPSec geçitleri de diğer cihazlar ve sistemler gibi ciddi yönetim gereksinimi olan cihazlardır. Günümüzde bu cihazların yönetimi için ya standart konsol bağlantısı, ya SSH ile telnetten bağlantı veya cihazın üreticisine has merkezi bir yönetim sistemi kullanılmaktadır. Bu çözümlerin dışında başka bazı çözümler de önerildi ise de bu çözümlerin bir kısmı kabul görmedi, bir kısmı da henüz olgunlaşmamış kompleks çözümlerdir. Bu noktada, halihazırdaki SNMP tabanlı merkezi yönetim çözümlerine alternatif olabilecek ve onların yetersizliklerini çözebilecek bir dağıtık yönetim sisteminin geliştirilmesi gerekliliği üzerine buradaki çalışma gerçekleştirilmiştir.

Önce IPSec geçitlerinin yönetim gereksinimleri ortaya konmuş ve ardından bu gereksinimleri karşılamak için mevcut ağ yönetim mimarileri ayrıntılı olarak incelenmiştir. Bu ağ yönetim mimarileri IPSec geçitlerinin yönetim gereksinimlerini karşılama noktasından karşılaştırılmış ve bunlardan güçlü dağıtık hiyerarşik mimarinin kullanışlı ve etkili bir çözüm olduğu görülmüştür. Bunları müteakiben IPSec geçitlerinin yönetimi konusunda yapılmış çalışmalar gözden geçirilmiş ve bunların problemli tarafları tartışılmıştır. Halihazırdaki çözümlere alternatif olan IPSec/VPN geçitleri için Script MIB tabanlı dağıtık yönetim sistemi bu noktada tanıtılmış ve bu sistemin altyapısında kullanılan Script MIB teknolojisinden kısaca bahsedilmiştir.

Bu çalışmada önerilen yönetim sistemi mimari olarak güçlü hiyerarşik dağıtık mimari kullanmakta ve delegasyon mekanizmasını da Script MIB aracılığıyla gerçekleştirmektedir. Bu yönetim sisteminde, yönetilen IPSec geçitleri yönetim alanlarına ayrılmaktadır. Herbir alan bir Alan Yöneticisi (AY) tarafından yönetilmektedir. Herbir IPSec geçidinin üzerinde IPSec geçidinin SNMP ile yönetimini sağlayan IPSec Yönetim Etmeni (IYE) bulunmaktadır. Mimarinin en üstünde da Yöneticiler Yöneticisi (YY) bulunmaktadır. YY altında bulunan AY'lere yönetim fonksiyonlarını ve işlerini delege etmektedir. YY delegasyon işini hazırlanan scriptleri Script MIB'e kurmak ve daha sonra bunları uzaktan çalıştırmak şeklinde gerçekleştirmektedir. YY AY'ye delege ettiği işlerin sonucunu istediği zaman öğrenebilmektedir.

YY' lerin AY'lere delege ettikleri yönetim fonksiyonları üç ana başlık altında toplanabilmektedir. Bunlar konfigürasyon, olay ve gözlem yönetimidir. Bu scriptler aslında Java dilinde yazılmış program ve nesnelerdir. Bu delegasyon mekanizmalarının daha iyi analiz edilebilmesi için her üç tipten de örnek uygulamalar geliştirilmiştir. Bu uygulamaların analiz edilmesi sonucu yönetimin delege edilmesinden ne oranda fayda sağlandığı ortaya çıkmıştır. Konfigürasyon için geliştirilen scriptlerden manuel tünel kuran scriptin delege edilmesinin aynı işin YY tarafından delege edilmeden yapılması ile kıyaslandığında delegasyon verimliliği 2,5 civarında çıkmaktadır. Bir diğer konfigürasyon scripti olan zamana programlı konfigürasyon scriptinin ise delegasyon verimliliği 50 civarında çıkmaktadır. Olay scriptlerinde ise delegasyon verimliliği verilen örnekte 60 civarındadır. Gözlem scriptlerinden yoklama yapan scriptte ise örnek değerler ile delegasyon verimi 270 civarında çıkmaktadır. Bu sonuçlar değerlendirilirse şu sonuçlara varılabilmektedir. Yoklama, zamana programlı konfigürasyon gibi çok tekrar eden yönetim işlerinde verim yüksektir. Benzer şekilde etkili olay filtreleri oluşturulduğunda da yine delegasyondan çok yüksek verim alınabilmektedir. Fakat manuel tünel kurma gibi haberleşme tekrarının az olduğu ve kısa sürede sonuçlanan scriptlerde verim daha düşük olmaktadır. Çünkü bu scriptlerin çalıştırılması ve sonucunun alınmasının da belirli bir maliyeti vardır. Olay, gözlem ve zamana programlı konfigürasyon scriptlerinde scriptlerin kurulup çalıştırmasının bir maliyeti olsa da bu iş bir kez yapıldığı ve script daha sonra çalışmaya devam ettiği için bu scriptlerin kullanımı zaman içinde yüksek fayda sağlamaktadır.

Burada sunulan sistemin merkezi ynetime karřı, SNMP ile ynetilebilen IPSec geitlerine daėıtık ynetim altyapısı sunması, standartları kullanması, kod daėıtımı ynyle esnek olması, leklenebilirlik zellikleri, hata kotarımında başarısı, tekrar kullanılabilir ve yeniliklere kolay adapte olabilir esnek yapısı gibi avantajları ile ciddi manada stnlk saėlamaktadır.

Bu sistemin de tabii ki bazı eksik noktaları vardır. Olay ynetimi, gzlem ve bazı konfigrasyon durumları iin pekok esneklik ve kolaylıklar saėlasa da cihazların konfigrasyonu noktasında YY'yi kullanan kullanıcıya ok iř dřmektedir. rnek vermek gerekirse manuel tnel kuran script YY'nin ykn hafifletse de bu scriptin alışması iin gerekli parametreleri veren kullanıcının ykn pek hafifletememektedir. Bu aıdan dřnldėnde kullanıcı her bir tnel iin gerekli btn parametreleri saėlamak ve btn gvenlik haberleřmesi tasarımı ayrıntılarını yapmak durumundadır. Bu noktada IPSec geitlerinin tnel fonksiyonlarını kullanan aė kaynakları modellenerek bunlara gvenlik seviyeleri verip st seviye politikalar ile otomatik tnel konfigrasyonu oluřturulması ve bunların AY'lere daėıtılması gibi bazı zmler gerekmektedir. Bu sistemin daha da geliřtirilmesi iin bu noktalardaki eksikler tamamlanmasına ncelik verilebilir.

Bu sistemin geliřtirilmesi ile ilgili bir diėer neri de řu noktada olabilir. Bu sistem tamamen SNMP tabanlı ynetim yapmaktadır. Fakat zellikle konfigrasyon iin SNMP'ye alternatiflerin arandıėı bu sıralarda AY'lere konfigrasyon iin XML tabanlı zmleri destekleme zellikleri eklenerek daha geniř bir alanda ynetim kabiliyeti kazandırılabilir.

KAYNAKLAR

- [1] Kent, S., Atkinson, R.: Security Architecture for the Internet Protocol. RFC-2401, IETF, Nov.1998.
- [2] M. Subramanian, "Network Management, Principals and Practice", Addison Wesley, 2000.
- [3] A.Pathan, "The Functional Areas of Network Management", <http://nww.evitech.fi/n/lassease/NetworkManagement/FCAPS.htm>, 2000.
- [4] T. Jenkins, J.Shriver, "IPsec Monitoring MIB", Internet draft, April 2003.
- [5] C. Madson, L. Temoshenko, C. Pellecuru, B. Harrison, S. Ramakrishnan, "IPsec Flow Monitoring MIB", Internet draft, March 2003.
- [6] M. Baer, R. Charlet, W. Hardaker, R. Story, C. Wang, "IPsec Policy Configuration MIB", Internet draft, March 2003.
- [7] T. Jenkins, J.Shriver, "Internet Key Exchange (IKE) Monitoring MIB", Internet draft, April 2003.
- [8] ISO/IEC 7498-4, ISO Open Systems Interconnection, Basic Ref. Model, Pt. 4: Management Framework, 1989.
- [9] CCITT (now ITU-T). *Recommendation X.700. Data Communication Networks Management Framework for Open Systems Interconnection (OSI) for CCITT Applications*. ITU, Geneva, Switzerland, September 1992.
- [10] J.P. Martin-Flatin, S. Znaty, and J.P. Hubaux. "A Survey of Distributed Enterprise Network and Systems Management". *Journal of Network and Systems Management*, 7(1):9-26, 1999.
- [11] J.Schönwälder, J.Quittek and C.Kappler, "Building Distributed Management Applications with the IETF Script MIB", IEEE Journal on Selected Areas In Communications, Vol. 18, No. 5, May 2000.
- [12] K. McCloghrie. "The SNMP Framework". The Simple Times, 4(1):9-10, 1996.
- [13] K. McCloghrie (Ed.). RFC 1909. An Administrative Infrastructure for SNMPv2. IETF, February 1996.

[14] G. Goldszmidt and Y. Yemini. "Distributed Management by Delegation". In Proc. 15th Int. Conf. on Distributed Computing Systems (ICDCS'95), Vancouver, Canada, May 1995. IEEE Press, New York, NY, USA, 1995.

[15] A. Fuggetta, G.P. Picco, and G. Vigna. "Understanding Code Mobility". IEEE Transactions on Software Engineering, 24(5):342-361, 1998.

[16] D. Levi, J. Schoenwaelder, "Definitions of Managed Objects for the Delegation of Management Scripts", RFC 3165, August 2001.

[17] Barotto, A. M.; Souza, A.; Westphall, C. B. "Distributed Network Management using SNMP, JAVA, WWW and CORBA", Journal of Network and Systems Management, Plenum Publishing Corporation. Meddletown, USA, Vol. 8, No. 4. Pg. 483-497, December 2000.

[18] J.P. Martin-Flatin, S. Znaty and J.P. Hubaux, "A Survey of Distributed Network and Systems Management Paradigms". Technical Report SSC/1998/024, SSC, EPFL, Lausanne, Switzerland, March 1997.

[19] K. Boudaoud, H. Labiod, Z. Guessoum and R. Boutaba, "Network Security Management with Intelligent Agents", In proceedings of 2000 IEEE/IFIP Network Operations and Management Symposium (NOMS'2000), Honolulu, Hawaii, 10-14 April 2000.

[20] Cisco Secure Policy Manager 3.0, http://www.cisco.com/warp/public/cc/pd/sqsw/sqppmn/prodlit/cspmg_ds.htm, 2002.

[21] L.A. Sanchez, M.N. Condell, Security Policy Protocol, Internet Draft, July 2000.

[22] M. Baltuta, A. Lioy and D. Mazzocchi, "Security Policy System: Status and Perspective", Proceedings of the IEEE International Conference on Networks 2000 (ICON 2000), pages 278-284, September 2000.

[23] Baek, M. Jeong, and J. Park, "Policy-based Hybrid Management Architecture for IP-based VPN", Proceedings of the 2000 IEEE/IFIP Network Operations and Management Symposium, pp. 987-988, 2000.

[24] M. Baltatu, A. Lioy, D. Lombardo, D. Mazzocchi, "Towards a Policy System for IPsec: issues and an experimental implementation", In Proceedings of the IEEE International Conference on Networks 2001, Hong Kong, October 2001.



- [25] R. Yavatkar, D. Pendarakis, and R. Guerin, "A Framework for Policy-based Admission Control", RFC 2753, Intel, IBM, U. of Pennsylvania, January 2000.
- [26] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. "An Architecture for Differentiated Services". RFC 2475, Torrent Networking Technologies, EMC Corporation, Sun Microsystems, Nortel UK, Bell Labs Lucent Technologies, Lucent Technologies, December 1998.
- [27] P. Cremonese, M. Esposito, S. Giordano, M. Mondini, Simon Pietro Romano, and Giorgio Ventre "A Framework for Policy-Based Management of QoS Aware IP Networks", NETWORKING 2002: 1105-1110.
- [28] P. Flegkas, P. Trimintzios, G. Pavlou, I. Andrikopoulos, C. F. Cavalcanti, "On Policy-based Extensible Hierarchical Network Management in QoS-enabled IP Networks", Proceedings of the IEEE Workshop on Policies for Distributed Systems and Networks (Policy '01), Bristol, UK, M. Sloman, J. Lobo, E. Lupu, eds., pp. 230-246, Springer-Verlag, January 2001.
- [29] P. Martinez, M. Brunner, J. Quittek, J. Schönwälder, F. Strauß, S. Mertens, T. Klie, "Using the Script MIB for Policy-based Configuration Management", IEEE/IFIP Network Operations and Management Symposium 2002, Florence, April 2002.
- [30] Mi-Joung Choi, Hae-Young Lee, Ji-Young Kong, Young-Mi Shin, and James Won-Ki Hong, "Design and Implementation of a Global Manager System for Firewall Devices", Proc. of KNOM 2001 Conference, Daejeon, Korea, pp. 109-115, May 2001.
- [31] U. Blumenthal and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", Lucent Technologies, RFC 3414, December 2002.
- [32] B. Wijnen, R. Presuhn and K. McCloghrie, "View-based access control model (VACM) for the Simple Network Management Protocol (SNMP)", Lucent Technologies, BMC Software, Cisco Systems, RFC 3415, December 2002.
- [33] J. Schönwälder and J. Quittek, "Secure management by delegation within the Internet management framework," in Proc. 6th IFIP/IEEE Int. Symp. Integrated Network Management, Boston, pp. 687-700, May 1999.

- [34] Alan Luchuk, "Schedule MIB", Simple Times, November 1999.
- [35] D. Levi, J. Schoenwaelder, "Definitions of Managed Objects for Scheduling Management Operations", Nortel Networks, TU Braunschweig, RFC 3231, January 2002.
- [36] Z. Fu, and S. F. Wu. "Automatic Generation of IPSEC/VPN Policies in an Intra-Domain Environment," 12th International Workshop on Distributed Systems: Operations & Management (DSOM 2001), Nancy, France, pp. 279-290, October 2001.
- [37] P. Martin-Flatin. "Push vs. Pull in Web-Based Network Management". In Proc. 6th IFIP/IEEE International Symposium on Integrated Network Management (IM'99), Boston, MA, SA, May 1999, pp. 3-18. IEEE Press, 1999.
- [38] TU Braunschweig, NEC C&C Research Laboratories, <http://www.ibr.cs.tu-bs.de/projects/jasmin/>. Jasmin - A Script MIB Implementation, 1999.
- [39] AdventNet Agent Toolkit 4.0, <http://www.adventnet.com>, 2000.
- [40] O. Adalier, H. Gümüşkaya, "Centralized Policy-Based Management of VPN Devices Using SNMPv3", 5th Computer Networks Symposium (BAS2000), pp. 113-121, Ankara, Bilkent University, 2000.



ÖZGEÇMİŞ

Mücahit Mutlugün, 21 Aralık 1977 tarihinde Bursa'da doğdu. İlk, orta ve lise eğitimini Bursa'da tamamladı. Bilkent Üniversitesi Mühendislik Fakültesi Elektrik-Elektronik Mühendisliği bölümünden 2000 senesinde mezun oldu. 2000 senesi temmuz ayından beri TÜBİTAK Ulusal Elektronik ve Kriptoloji Araştırma Enstitüsü'nde araştırmacı olarak görev yapmaktadır.



EKLER

Ek-1: Test için Geliştirilen IPSec Configuration MIB

```
-- File Name : IPSEC-CONFIGURATION-MIB
--IPSEC-CONFIGURATION-MIB { iso org(3) dod(6) internet(1) private(4)
enterprises(1) 1 }
IPSEC-CONFIGURATION-MIB DEFINITIONS ::= BEGIN
    IMPORTS
        MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
        IpAddress, Integer32, Counter32, Gauge32, TimeTicks, Counter64,
        enterprises
            FROM SNMPv2-SMI
        DisplayString, RowStatus, DateAndTime
            FROM SNMPv2-TC
        IpAddress
            FROM RFC1155-SMI
        DisplayString
            FROM RFC1213-MIB;

    test MODULE-IDENTITY
        LAST-UPDATED          "00102000000Z"
        ORGANIZATION          "GYTE THESIS PROJECT"
        CONTACT-INFO          "    Mucahit Mutlugun  (Editor)
                                Postal: Tubitak UEKAE
                                    P.O.Box 21
                                    41470 Gebze KOCAELI
                                    Tel: +90 648 13 38
                                    Fax: +90 648 11 00
                                    E-mail: mucahit@uekae.tubitak.gov.tr"

        DESCRIPTION
            "The MIB module for IPSec gateways"

    ::= { enterprises 1 }

-- TEXTUAL-CONVENTIONS
    State ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
        STATUS      current
        DESCRIPTION "Any state"
        REFERENCE   ""
        SYNTAX      INTEGER {
                                Off(1),
                                On(2)
                            }

    org OBJECT IDENTIFIER
        ::= { iso 3 }

    dod OBJECT IDENTIFIER
        ::= { org 6 }

    internet OBJECT IDENTIFIER
        ::= { dod 1 }

    private OBJECT IDENTIFIER
        ::= { internet 4 }
```

```

enterprises OBJECT IDENTIFIER
    ::= { private 1 }

ipsecConf OBJECT IDENTIFIER
    ::= { test 1 }

ipsecTrap OBJECT IDENTIFIER
    ::= { test 2 }

equipment OBJECT IDENTIFIER
    ::= { ipsecConf 1 }

networkParams OBJECT IDENTIFIER
    ::= { ipsecConf 2 }

securityPolicy OBJECT IDENTIFIER
    ::= { ipsecConf 3 }

tableparameters OBJECT IDENTIFIER
    ::= { ipsecConf 4 }

traps OBJECT IDENTIFIER
    ::= { ipsecTrap 1 }

-- Equipment Parameters Group

equipName OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION  "Name of the IPsec gateway"
    ::= { equipment 1 }

equipGroup OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION  "Name of the group of IPsec gateway"
    ::= { equipment 2 }

equipLocation OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION  "Location of the IPsec gateway"
    ::= { equipment 3 }

equipVersion OBJECT-TYPE
    SYNTAX      DisplayString ( SIZE ( 0 .. 255 ) )
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION  "Version of the software working on the IPsec gateway"
    ::= { equipment 4 }

equipSerialNo OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS   read-only

```

```

STATUS          current
DESCRIPTION      "Serial no of IPsec gateway"
::= { equipment 5 }

```

```

equipDateandTime OBJECT-TYPE
SYNTAX            DateAndTime
MAX-ACCESS        read-write
STATUS            current
DESCRIPTION        "Current date and time of the system"
::= { equipment 6 }

```

---Network Parameters Group

```

netRedIpAddress OBJECT-TYPE
SYNTAX          IPAddress
MAX-ACCESS      read-write
STATUS          current
DESCRIPTION      "IP Address of red interface"
::= { networkParams 1 }

```

```

netRedSubnetMask OBJECT-TYPE
SYNTAX            IPAddress
MAX-ACCESS        read-write
STATUS            current
DESCRIPTION        "Subnet Mask of red interface"
::= { networkParams 2 }

```

```

netRedGateway OBJECT-TYPE
SYNTAX          IPAddress
MAX-ACCESS      read-write
STATUS          current
DESCRIPTION      "Gateway of red interface"
::= { networkParams 3 }

```

```

netBlackIpAddress OBJECT-TYPE
SYNTAX            IPAddress
MAX-ACCESS        read-write
STATUS            current
DESCRIPTION        "IP address of black interface"
::= { networkParams 4 }

```

```

netBlackSubnetMask OBJECT-TYPE
SYNTAX            IPAddress
MAX-ACCESS        read-write
STATUS            current
DESCRIPTION        "Subnet Mask of black interface"
::= { networkParams 5 }

```

```

netBlackGateway OBJECT-TYPE
SYNTAX          IPAddress
MAX-ACCESS      read-write
STATUS          current
DESCRIPTION      "Gateway of black interface"

```

```
::= { networkParams 6 }
```

--Routing Table

```
netRoutingTable      OBJECT-TYPE
    SYNTAX             SEQUENCE OF NetRoutingEntry
    MAX-ACCESS         not-accessible
    STATUS              current
    DESCRIPTION
        "routing Table"
    ::= { networkParams 8 }
```

```
netRoutingEntry      OBJECT-TYPE
    SYNTAX             NetRoutingEntry
    MAX-ACCESS         not-accessible
    STATUS              current
    DESCRIPTION
        "routing Entry"
    INDEX               { netRoutingIndex }
    ::= { netRoutingTable 1 }
```

```
NetRoutingEntry ::= SEQUENCE {
    netRoutingIndex  INTEGER,
    netRoutingInterface  DisplayString,
    netRoutingDestIP  IPAddress,
    netRoutingMask  IPAddress,
    netRoutingNext  IPAddress,
    netRoutingType  INTEGER,
    netRoutingPro  INTEGER,
    netRoutingAge  INTEGER,
    netRoutingStatus  RowStatus
}
```

```
netRoutingIndex      OBJECT-TYPE
    SYNTAX             INTEGER
    MAX-ACCESS         read-create
    STATUS              current
    DESCRIPTION
        "Index of routing table"
    ::= { netRoutingEntry 1 }
```

```
netRoutingInterface  OBJECT-TYPE
    SYNTAX             DisplayString
    MAX-ACCESS         read-create
    STATUS              current
    DESCRIPTION
        "Interface name"
    ::= { netRoutingEntry 2 }
```

```
netRoutingDestIP     OBJECT-TYPE
    SYNTAX             IPAddress
    MAX-ACCESS         read-create
    STATUS              current
    DESCRIPTION
        "Destination IP of route"
    ::= { netRoutingEntry 3 }
```

```
netRoutingMask        OBJECT-TYPE
    SYNTAX             IPAddress
    MAX-ACCESS         read-create
    STATUS              current
```

DESCRIPTION

"Subnet Mask of route"

::= { netRoutingEntry 4 }

netRoutingNext OBJECT-TYPE
 SYNTAX IpAddress
 MAX-ACCESS read-create
 STATUS current
 DESCRIPTION

"Next IP address in the route"

::= { netRoutingEntry 5 }

netRoutingType OBJECT-TYPE
 SYNTAX INTEGER
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION

"Type"

::= { netRoutingEntry 6 }

netRoutingPro OBJECT-TYPE
 SYNTAX INTEGER
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION

"Pro"

::= { netRoutingEntry 7 }

netRoutingAge OBJECT-TYPE
 SYNTAX INTEGER
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION

"Age"

::= { netRoutingEntry 8 }

netRoutingStatus OBJECT-TYPE
 SYNTAX RowStatus
 MAX-ACCESS read-create
 STATUS current
 DESCRIPTION

"RowStatus"

::= { netRoutingEntry 9 }

--Security Policy Group

-- Inbound Policy Table

secInboundPolicyTable OBJECT-TYPE
 SYNTAX SEQUENCE OF SecInboundPolicyEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION

"Description"

::= { securityPolicy 1 }

secInboundPolicyEntry OBJECT-TYPE
 SYNTAX SecInboundPolicyEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION

```

"Row Description"
INDEX      { secInboundPolicyRule,
secInboundPolicyFilter }
::= { secInboundPolicyTable 1 }

```

```

SecInboundPolicyEntry ::= SEQUENCE {
    secInboundPolicyRule  INTEGER,
    secInboundPolicyFilter INTEGER,
    secInboundPolicySourceIp  DisplayString,
    secInboundPolicyDestIp  DisplayString,
    secInboundPolicyProtocol DisplayString,
    secInboundPolicySourcePort DisplayString,
    secInboundPolicyDestPort DisplayString,
    secInboundPolicyDecision INTEGER,
    secInboundPolicyAlarm  INTEGER,
    secInboundPolicyCMap  VariablePointer,
    secInboundPolicyStatus RowStatus
}

```

```

secInboundPolicyRule      OBJECT-TYPE
    SYNTAX                 INTEGER
    MAX-ACCESS              read-create
    STATUS                  current
    DESCRIPTION

```

"Rule number of inbound policy"

```

::= { secInboundPolicyEntry 1 }

```

```

secInboundPolicyFilter    OBJECT-TYPE
    SYNTAX                 INTEGER
    MAX-ACCESS              read-create
    STATUS                  current
    DESCRIPTION

```

"Filter number of policy"

```

::= { secInboundPolicyEntry 2 }

```

```

secInboundPolicySourceIp  OBJECT-TYPE
    SYNTAX                 DisplayString
    MAX-ACCESS              read-create
    STATUS                  current
    DESCRIPTION

```

"Source IP Address of inbound policy"

```

DEFVAL                     { "*" }

```

```

::= { secInboundPolicyEntry 3 }

```

```

secInboundPolicyDestIp    OBJECT-TYPE
    SYNTAX                 DisplayString
    MAX-ACCESS              read-create
    STATUS                  current
    DESCRIPTION

```

"Destination Ip Address of inbound policy"

```

DEFVAL                     { "*" }

```

```

::= { secInboundPolicyEntry 4 }

```

```

secInboundPolicyProtocol  OBJECT-TYPE
    SYNTAX                 DisplayString ( SIZE ( 0 .. 255 ) )
    MAX-ACCESS              read-create
    STATUS                  current
    DESCRIPTION

```

"Higher level protocol"

```

DEFVAL                     { "*" }

```

```

::= { secInboundPolicyEntry 5 }

secInboundPolicySourcePort OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION   "secInboundPolicy source port number"
    DEFVAL       { "*" }
::= { secInboundPolicyEntry 6 }

secInboundPolicyDestPort OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION   "InboundPolicy destination port number"
    DEFVAL       { "*" }
::= { secInboundPolicyEntry 7 }

secInboundPolicyDecision OBJECT-TYPE
    SYNTAX      INTEGER
                {
                    discard(1),
                    pass(2),
                    ipsec(3)
                }
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION   "InboundPolicy decision"
    DEFVAL       { 1 }
::= { secInboundPolicyEntry 8 }

secInboundPolicyAlarm OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION   "InboundPolicy alarm"
::= { secInboundPolicyEntry 9 }

secInboundPolicyCMap OBJECT-TYPE
    SYNTAX      VariablePointer
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION   "This object points to a row at Crypto Map Table"
::= { secInboundPolicyEntry 10 }

secInboundPolicyStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION   "Status of inboundPolicyEntry"
::= { secInboundPolicyEntry 11 }

```

-- Outbound Policy Table

```

secOutboundPolicyTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SecOutboundPolicyEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "Description"
    ::= { securityPolicy 2 }

secOutboundPolicyEntry OBJECT-TYPE
    SYNTAX      SecOutboundPolicyEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "Row Description"
    INDEX       { secOutboundPolicyRule,
secOutboundPolicyFilter }
    ::= { secOutboundPolicyTable 1 }

SecOutboundPolicyEntry ::= SEQUENCE {
    secOutboundPolicyRule  INTEGER,
    secOutboundPolicyFilter  INTEGER,
    secOutboundPolicySourceIp  DisplayString,
    secOutboundPolicyDestIp  DisplayString,
    secOutboundPolicyProtocol  DisplayString,
    secOutboundPolicySourcePort  DisplayString,
    secOutboundPolicyDestPort  DisplayString,
    secOutboundPolicyDecision  INTEGER,
    secOutboundPolicyAlarm  INTEGER,
    secOutboundPolicyCMap  VariablePointer,
    secOutboundPolicyStatus  RowStatus
}

secOutboundPolicyRule OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "outboundPolicy rule number"
    ::= { secOutboundPolicyEntry 1 }

secOutboundPolicyFilter OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "outboundPolicy filter number"
    ::= { secOutboundPolicyEntry 2 }

secOutboundPolicySourceIp OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "OutboundPolicy source Ip"
    DEFVAL      { "*. *.*.*.*" }
    ::= { secOutboundPolicyEntry 3 }

secOutboundPolicyDestIp OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS   read-create
    STATUS       current

```



```
DESCRIPTION
    "outboundPolicy destination Ip"
DEFVAL      { "*" }
::= { secOutboundPolicyEntry 4 }

secOutboundPolicyProtocol OBJECT-TYPE
    SYNTAX      DisplayString
    MAX ACCESS   read-create
    STATUS       current
    DESCRIPTION  "OutboundPolicy higher level protocol"
DEFVAL      { "*" }
::= { secOutboundPolicyEntry 5 }

secOutboundPolicySourcePort OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION  "outboundPolicy source port"
DEFVAL      { "*" }
::= { secOutboundPolicyEntry 6 }

secOutboundPolicyDestPort OBJECT-TYPE
    SYNTAX      DisplayString
    MAX ACCESS   read-create
    STATUS       current
    DESCRIPTION  "outboundPolicy destination port"
DEFVAL      { "*" }
::= { secOutboundPolicyEntry 7 }

secOutboundPolicyDecision OBJECT-TYPE
    SYNTAX      INTEGER
    {
        discard(1),
        pass(2),
        ipsec(3)
    }
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION  "outboundPolicy decision"
DEFVAL      { "D" }
::= { secOutboundPolicyEntry 8 }

secOutboundPolicyAlarm OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION  "OutboundPolicy alarm"
::= { secOutboundPolicyEntry 9 }

secOutboundPolicyCMap OBJECT-TYPE
    SYNTAX      VariablePointer
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION  "This object points to a row at Crypto Map Table"
::= { secOutboundPolicyEntry 10 }
```



```
secOutboundPolicyStatus OBJECT-TYPE
    SYNTAX                RowStatus
    MAX-ACCESS             read-create
    STATUS                 current
    DESCRIPTION
        "Status of outboundPolicyEntry"
    ::= { secOutboundPolicyEntry 11 }
```

-- Manual SA Table

```
secManualSATable          OBJECT-TYPE
    SYNTAX                 SEQUENCE OF SecManualSAEntry
    MAX-ACCESS             not-accessible
    STATUS                 current
    DESCRIPTION
        "Manual SA Table"
    ::= { securityPolicy 3 }
```

```
secManualSAEntry          OBJECT-TYPE
    SYNTAX                 SecManualSAEntry
    MAX-ACCESS             not-accessible
    STATUS                 current
    DESCRIPTION
        "Row Description"
    INDEX                  { secManualSAIndex }
    ::= { secManualSATable 1 }
```

```
SecManualSAEntry ::= SEQUENCE {
    secManualSAIndex        INTEGER,
    secManualSASName        DisplayString,
    secManualSASState       INTEGER,
    secManualSASPI          INTEGER,
    secManualSAPeer         DisplayString,
    secManualSATransform    VariablePointer,
    secManualSAAuthKey       VariablePointer,
    secManualSAEncyrptKey   VariablePointer,
    secManualSASStatus      RowStatus
}
```

```
secManualSAIndex          OBJECT-TYPE
    SYNTAX                 INTEGER
    MAX-ACCESS             read-create
    STATUS                 current
    DESCRIPTION
        "SA table index"
    ::= { secManualSAEntry 1 }
```

```
secManualSASName          OBJECT-TYPE
    SYNTAX                 DisplayString
    MAX-ACCESS             read-create
    STATUS                 current
    DESCRIPTION
        "Name of SA"
    ::= { secManualSAEntry 2 }
```

```
secManualSASState         OBJECT-TYPE
    SYNTAX                 INTEGER
    {
        invalid-notused(0),
        invalid-used(1),
        passive(2),
    }
```

```

                                active(3)
                                }
MAX-ACCESS                     read-only
STATUS                         current
DESCRIPTION                    "State of SA"
 ::= { secManualSAEntry 3 }

secManualSASPI                 OBJECT-TYPE
SYNTAX                        INTEGER (256..2147483648)
MAX-ACCESS                     read-create
STATUS                         current
DESCRIPTION                    "SPI of SA"
 ::= { secManualSAEntry 4 }

secManualSAPeer                OBJECT-TYPE
SYNTAX                        DisplayString
MAX-ACCESS                     read-create
STATUS                         current
DESCRIPTION                    "Peer of SA"
DEFVAL                        { '01010101'h }
 ::= { secManualSAEntry 5 }

secManualSATransform            OBJECT-TYPE
SYNTAX                        VariablePointer
MAX-ACCESS                     read-create
STATUS                         current
DESCRIPTION                    "This object points to a row at Transform Table"
 ::= { secManualSAEntry 6 }

secManualSAAuthKey              OBJECT-TYPE
SYNTAX                        VariablePointer
MAX-ACCESS                     read create
STATUS                         current
DESCRIPTION                    "This object points to a row at Key Table"
 ::= { secManualSAEntry 7 }

secManualSAEncyrptKey           OBJECT-TYPE
SYNTAX                        VariablePointer
MAX-ACCESS                     read-create
STATUS                         current
DESCRIPTION                    "This object points to a row at Key Table"
 ::= { secManualSAEntry 8 }

secManualSAstatus               OBJECT-TYPE
SYNTAX                        RowStatus
MAX-ACCESS                     read-create
STATUS                         current
DESCRIPTION                    "Status of SAEntry"
 ::= { secManualSAEntry 9 }

-- Auto SA Table
secAutoSAtable                  OBJECT-TYPE
SYNTAX                        SEQUENCE OF SecAutoSAEntry
MAX-ACCESS                     not-accessible

```

```

STATUS          current
DESCRIPTION
    "SA Table"
::= { securityPolicy 4 }

secAutoSAEntry  OBJECT-TYPE
SYNTAX          SecAutoSAEntry
MAX-ACCESS      not-accessible
STATUS          current
DESCRIPTION
    "Row Description"
INDEX           { secAutoSAIndex }
::= { secAutoSatable 1 }

SecAutoSAEntry ::= SEQUENCE {
    secAutoSAIndex      INTEGER,
    secAutoSASName      DisplayString,
    secAutoSASState     INTEGER,
    secAutoSATransform  VariablePointer,
    secAutoSASAuthMethod INTEGER,
    secAutoSASDhGroup   INTEGER,
    secAutoSASTimeLife  INTEGER,
    secAutoSASVollLife  INTEGER,
    secAutoSASAPfs      INTEGER,
    secAutoSASStatus    RowStatus
}

secAutoSAIndex  OBJECT-TYPE
SYNTAX          INTEGER
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "SA table index"
::= { secAutoSAEntry 1 }

secAutoSASName  OBJECT-TYPE
SYNTAX          DisplayString
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION
    "Name of SA"
::= { secAutoSAEntry 2 }

secAutoSASState OBJECT-TYPE
SYNTAX          INTEGER
{
    invalid-notused(0),
    invalid-used(1),
    passive(2),
    active(3)
}
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "State of SA"
::= { secAutoSAEntry 3 }

secAutoSATransform OBJECT-TYPE
SYNTAX          VariablePointer
MAX-ACCESS      read-create
STATUS          current

```

DESCRIPTION

"This object points to a row at Transform Table"

::= { secAutoSAEntry 4 }

secAutoSAAuthMethod	OBJECT-TYPE
SYNTAX	INTEGER
	{
	not-in-manual(0),
	shared-key(1),
	DSS-signiture(2),
	RSA-signiture(3),
	RSA-encryption(4),
	RSA-encryption-revised(5)
	}
MAX-ACCESS	read-create
STATUS	current
DESCRIPTION	"Anth Method"

::= { secAutoSAEntry 5 }

secAutoSADhGroup	OBJECT-TYPE
SYNTAX	INTEGER
	{
	not-in-manual(0),
	modp768(1),
	modp1024(2),
	ec2n155(3),
	ec2n185(4),
	modp1536(5)
	}
MAX-ACCESS	read-create
STATUS	current
DESCRIPTION	"Dh Group"

::= { secAutoSAEntry 6 }

secAutoSATimeLife	OBJECT-TYPE
SYNTAX	INTEGER
MAX-ACCESS	read-create
STATUS	current
DESCRIPTION	"Time Life"

::= { secAutoSAEntry 7 }

secAutoSAVollLife	OBJECT-TYPE
SYNTAX	INTEGER
MAX-ACCESS	read-create
STATUS	current
DESCRIPTION	"Volume Life"

::= { secAutoSAEntry 8 }

secAutoSAPfs	OBJECT-TYPE
SYNTAX	State
MAX-ACCESS	read-create
STATUS	current
DESCRIPTION	

"Volume Life"

```
:: { secAutoSAEntry 9 }
```

```
secAutoSAStatus      OBJECT-TYPE
    SYNTAX             RowStatus
    MAX-ACCESS         read-create
    STATUS             current
    DESCRIPTION
        "Status of SAEntry"
 ::= { secAutoSAEntry 10 }
```

-- Crypto Map Table

```
secCryptoMapTable     OBJECT-TYPE
    SYNTAX             SEQUENCE OF SecCryptoMapEntry
    MAX-ACCESS         not-accessible
    STATUS             current
    DESCRIPTION
        "Crypto Map Table"
 ::= { securityPolicy 5 }
```

```
secCryptoMapEntry     OBJECT-TYPE
    SYNTAX             SecCryptoMapEntry
    MAX ACCESS         not-accessible
    STATUS             current
    DESCRIPTION
        "Row Description"
    INDEX              { secCryptoMapIndex }
 ::= { secCryptoMapTable 1 }
```

```
SecCryptoMapEntry ::= SEQUENCE {
    secCryptoMapIndex      INTEGER,
    secCryptoMapName       DisplayString,
    secCryptoMapSetup      INTEGER,
    secCryptoMapState      INTEGER,
    secCryptoMapBundle     VariablePointer,
    secCryptoMapIke        VariablePointer,
    secCryptoMapGranularity INTEGER,
    secCryptoMapStatus     RowStatus
}
```

```
secCryptoMapIndex     OBJECT-TYPE
    SYNTAX             INTEGER
    MAX-ACCESS         read-create
    STATUS             current
    DESCRIPTION
        "Index of CryptoMap"
 ::= { secCryptoMapEntry 1 }
```

```
secCryptoMapName       OBJECT-TYPE
    SYNTAX             DisplayString
    MAX-ACCESS         read-create
    STATUS             current
    DESCRIPTION
        "Name of CryptoMap"
 ::= { secCryptoMapEntry 2 }
```

```
secCryptoMapSetup      OBJECT-TYPE
    SYNTAX             INTEGER
    {
```

```

                                manuel(1),
                                automatic(2)
                                }
MAX-ACCESS                     read-create
STATUS                         current
DESCRIPTION                    "Setup of CryptoMap"
::= { secCryptoMapEntry 3 }

secCryptoMapState              OBJECT-TYPE
SYNTAX                        INTEGER
                                {
                                    invalid-notused(0),
                                    invalid-used(1),
                                    passive(2),
                                    active(3)
                                }
MAX-ACCESS                     read-only
STATUS                         current
DESCRIPTION                    "State of CryptoMap"
::= { secCryptoMapEntry 4 }

secCryptoMapBundle             OBJECT-TYPE
SYNTAX                        VariablePointer
MAX-ACCESS                     read-create
STATUS                         current
DESCRIPTION                    "This object points to a row at Manuel or Automatic SA Table"
::= { secCryptoMapEntry 5 }

secCryptoMapIke                OBJECT-TYPE
SYNTAX                        VariablePointer
MAX-ACCESS                     read-create
STATUS                         current
DESCRIPTION                    "This object points to a row at IKE Table"
::= { secCryptoMapEntry 6 }

secCryptoMapGranularity        OBJECT-TYPE
SYNTAX                        INTEGER
                                {
                                    not-in-manual(0),
                                    veryhigh(1),
                                    high(2),
                                    middle(3),
                                    low(4)
                                }
MAX-ACCESS                     read-create
STATUS                         current
DESCRIPTION                    "Granularity of CryptoMap"
::= { secCryptoMapEntry 7 }

secCryptoMapStatus             OBJECT-TYPE
SYNTAX                        RowStatus
MAX-ACCESS                     read-create
STATUS                         current
DESCRIPTION                    "Status of CryptoMap"
::= { secCryptoMapEntry 8 }

```

-- Ike Table

```

secIkeTable
  SYNTAX
  MAX-ACCESS
  STATUS
  DESCRIPTION
    ""
  ::= { securityPolicy 6 }

```

```

secIkeEntry
  OBJECT-TYPE
  SYNTAX
  MAX-ACCESS
  STATUS
  DESCRIPTION
    "Row Description"
  INDEX { secIkeIndex }
  ::= { secIkeTable 1 }

```

```

SecIkeEntry ::= SEQUENCE {
  secIkeIndex          INTEGER,
  secIkeName           DisplayString,
  secIkeState          INTEGER,
  secIkeType           INTEGER,
  secIkeExchangeType  INTEGER,
  secIkeAuthMethod     INTEGER,
  secIkeEncryptMethod  INTEGER,
  secIkeHash           INTEGER,
  secIkeHMAC           INTEGER,
  secIkeDHGroup        INTEGER,
  secIkeTimeLife       INTEGER,
  secIkeVolLife        INTEGER,
  secIkePfs            INTEGER,
  secIkePeer           IpAddress,
  secIkeStatus         RowStatus
}

```

```

secIkeIndex
  OBJECT-TYPE
  SYNTAX
  MAX-ACCESS
  STATUS
  DESCRIPTION
    "Index of Ike"
  ::= { secIkeEntry 1 }

```

```

secIkeName
  OBJECT-TYPE
  SYNTAX
  MAX-ACCESS
  STATUS
  DESCRIPTION
    "Name of Ike"
  ::= { secIkeEntry 2 }

```

```

secIkeState
  OBJECT-TYPE
  SYNTAX
    {
      passive(1),
      active(2)
    }
  MAX-ACCESS
    read-only

```

```

STATUS          current
DESCRIPTION      "State of Ike"
::= { secIkeEntry 3 }

```

```

secIkeType      OBJECT-TYPE
SYNTAX          INTEGER
{
    isakmp(1),
    ah(2),
    esp(3),
    ipcomp(4),
    any(5)
}

MAX-ACCESS      read-create
STATUS          current
DESCRIPTION      "Type of Ike"
::= { secIkeEntry 4 }

```

```

secIkeExchangeType OBJECT-TYPE
SYNTAX              INTEGER
{
    base(1),
    ip(2),
    ao(3),
    aggr(4),
    info(5),
    qm(6),
    ngm(7),
    none(8),
    any(9)
}

MAX-ACCESS      read-create
STATUS          current
DESCRIPTION      "ExchangeType of Ike"
::= { secIkeEntry 5 }

```

```

secIkeAuthMethod OBJECT-TYPE
SYNTAX            INTEGER
{
    psk(1),
    dss(2),
    rsas(3),
    rsae(4),
    rsaer(5)
}

MAX-ACCESS      read-create
STATUS          current
DESCRIPTION      "AuthMethod of Ike"
::= { secIkeEntry 6 }

```

```

secIkeEncryptMethod OBJECT-TYPE
SYNTAX               INTEGER
{
    des(1),
    triple-des(2),
    cast(3),
    blowfish(4),

```

```

aes(5),
null(6)
}
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION     "EncryptMethod of Ike"
::= { secIkeEntry 7 }

secIkeHash      OBJECT-TYPE
SYNTAX          INTEGER
{
    md5(1),
    sha1(2),
    tiger192(3),
    ripemd160(4),
    null(5)
}
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION     "Hash Algorithm of Ike"
::= { secIkeEntry 8 }

secIkeHMAC      OBJECT-TYPE
SYNTAX          INTEGER
{
    hmac-md5(1),
    hmac-sha1(2),
    hmac-tiger192(3),
    hmac-ripemd160(4),
    null(5)
}
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION     "HMAC of Ike"
::= { secIkeEntry 9 }

secIkeDHGroup   OBJECT-TYPE
SYNTAX          INTEGER
{
    modp768(1),
    modp1024(2),
    ec2n155(3),
    ec2n185(4),
    modp1536(5)
}
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION     "DHGroup of Ike"
::= { secIkeEntry 10 }

secIkeTimeLife  OBJECT-TYPE
SYNTAX          INTEGER
MAX-ACCESS      read-create
STATUS          current
DESCRIPTION     "secIkeTimeLife of Ike"

```



```
 ::= { secIkeEntry 11 }

secIkeVolLife          OBJECT-TYPE
    SYNTAX              INTEGER
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION          "secIkeVolLife of Ike"
 ::= { secIkeEntry 12 }
```

```
secIkePfs              OBJECT-TYPE
    SYNTAX              INTEGER
                        {
                            md5(1),
                            sha1(2),
                            null(3)
                        }
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION          "secIkePfs of Ike"
 ::= { secIkeEntry 13 }
```

```
secIkePeer            OBJECT-TYPE
    SYNTAX              IpAddress
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION          "secIkeVolLife of Ike"
 ::= { secIkeEntry 14 }
```

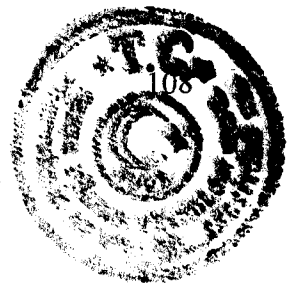
```
secIkeStatus          OBJECT-TYPE
    SYNTAX              RowStatus
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION          "Status of Ike"
 ::= { secIkeEntry 15 }
```

-- Key Table

```
secKeyTable           OBJECT-TYPE
    SYNTAX              SEQUENCE OF SecKeyEntry
    MAX-ACCESS          not-accessible
    STATUS              current
    DESCRIPTION          "Key Table"
 ::= { securityPolicy 7 }
```

```
secKeyEntry           OBJECT-TYPE
    SYNTAX              SecKeyEntry
    MAX-ACCESS          not-accessible
    STATUS              current
    DESCRIPTION          "Row Description"
    INDEX               { secKeyIndex }
 ::= { secKeyTable 1 }
```

```
SecKeyEntry ::= SEQUENCE {
    secKeyIndex        INTEGER,
    secKeyType         INTEGER,
```



```
secKeyName          DisplayString,
secKeyProductTime   DisplayString,
secKeyLength         INTEGER,
secKeyState          INTEGER,
secKeyStatus         RowStatus
}

secKeyIndex          OBJECT-TYPE
    SYNTAX            INTEGER
    MAX-ACCESS        read-only
    STATUS            current
    DESCRIPTION
        "Index of Key"
    ::= { secKeyEntry 1 }

secKeyType           OBJECT-TYPE
    SYNTAX            INTEGER
    {
        Psk(1),
        Private(2),
        Public(3),
        Authen(4),
        Encrypt(5),
        OwnPrivate(6),
        OwnPublic(7)
    }
    MAX-ACCESS        read-only
    STATUS            current
    DESCRIPTION
        "Type of Key"
    ::= { secKeyEntry 2 }

secKeyName           OBJECT-TYPE
    SYNTAX            DisplayString
    MAX-ACCESS        read-only
    STATUS            current
    DESCRIPTION
        "Name of Key"
    ::= { secKeyEntry 3 }

secKeyProductTime    OBJECT-TYPE
    SYNTAX            DisplayString
    MAX-ACCESS        read-only
    STATUS            current
    DESCRIPTION
        "Product Time of Key"
    ::= { secKeyEntry 4 }

secKeyLength         OBJECT-TYPE
    SYNTAX            INTEGER
    MAX-ACCESS        read-only
    STATUS            current
    DESCRIPTION
        "Length of Key"
    ::= { secKeyEntry 5 }

secKeyState          OBJECT-TYPE
    SYNTAX            INTEGER
    {
        Deleted(1),
        Valid(2)
```

```

    )
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "State of Key"
    ::= { secKeyEntry 6 }

```

```

secKeyStatus      OBJECT-TYPE
    SYNTAX         RowStatus
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "Status of Key"
    ::= { secKeyEntry 7 }

```

```

secTransformTable OBJECT-TYPE
    SYNTAX         SEQUENCE OF SecTransformEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Transform Table"
    ::= { securityPolicy 8 }

```

```

secTransformEntry OBJECT-TYPE
    SYNTAX         SecTransformEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Row Description"
    INDEX          { secTransformIndex }
    ::= { secTransformTable 1 }

```

```

SecTransformEntry ::= SEQUENCE {
    secTransformIndex      INTEGER,
    secTransformName       DisplayString,
    secTransformProtocol   INTEGER,
    secTransformEnc        INTEGER,
    secTransformEncBit     INTEGER,
    secTransformAuth       INTEGER,
    secTransformAuthBit    INTEGER
}

```

```

secTransformIndex OBJECT-TYPE
    SYNTAX          INTEGER
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "Index of Transform"
    ::= { secTransformEntry 1 }

```

```

secTransformName  OBJECT-TYPE
    SYNTAX         DisplayString
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "Name of transform"
    ::= { secTransformEntry 2 }

```

```

secTransformProtocol OBJECT-TYPE
    SYNTAX            INTEGER

```

```

{
    AH(1),
    ESP(2)
}
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
    "Protocol of transform"
::= { secTransformEntry 3 }

secTransformEnc OBJECT-TYPE
    SYNTAX      INTEGER
    {
        des(1),
        des3(2),
        idea(3),
        blowfish(4),
        rc4(5),
        aes(6),
        cast(7)
    }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Enc type of transform"
    ::= { secTransformEntry 4 }

secTransformEncBit OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Enc bit of transform"
    ::= { secTransformEntry 5 }

secTransformAuth OBJECT-TYPE
    SYNTAX      INTEGER
    {
        hmac-md5(1),
        hmac-sha(2)
    }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Auth type of transform"
    ::= { secTransformEntry 6 }

secTransformAuthBit OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Auth bit of transform"
    ::= { secTransformEntry 7 }

--Table parameters group

tpSAFreeIndex OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS  read-write
    STATUS      current

```

DESCRIPTION

""

```
DEFVAL { 1 }
::= { tableparameters 1 }
```

```
tpCMapFreeIndex OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-write
STATUS current
DESCRIPTION
```

""

```
DEFVAL { 1 }
::= { tableparameters 2 }
```

```
tpInPolicyFreeIndex OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-write
STATUS current
DESCRIPTION
```

""

```
DEFVAL { 1 }
::= { tableparameters 3 }
```

```
tpOutPolicyFreeIndex OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-write
STATUS current
DESCRIPTION
```

""

```
DEFVAL { 1 }
::= { tableparameters 4 }
```

```
tpIkeFreeIndex OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-write
STATUS current
DESCRIPTION
```

""

```
DEFVAL { 1 }
::= { tableparameters 5 }
```

```
notificationMessage OBJECT-TYPE
SYNTAX DisplayString
MAX-ACCESS accessible-for-notify
STATUS current
DESCRIPTION
```

"Message in notifications "

```
::= { traps 1 }
```

```
ipsecGatewayUnreachable NOTIFICATION-TYPE
```

```
STATUS current
DESCRIPTION
```

"ipsec Gateway cannot be reached by domain manager"

```
::= { traps 2 }
```

```
ipsecGatewayReachable NOTIFICATION-TYPE
```

```
STATUS current
DESCRIPTION
```

"ipsec Gateway, which could not be reached, is currently reached by domain manager"

```
::= { traps 3 }
```

ipsecGatewayMonitorThreshold NOTIFICATION-TYPE

STATUS current

DESCRIPTION

"ipsec Gateway, which could not be reached, is
currently reached by domain manager"

::= { traps 3 }

END



Ek-2: Geliştirilen yönetim scriptlerinin kaynak kodları

Konfigürasyon Scriptleri

ManualTunnelSetup.java

```
package ipsecman.config;

import java.io.*;
import java.util.*;

import disman.SnmpException;
import jmgmt.asnl.*;
import jmgmt.snmp.io.*;
import ipsecman.util.SnmpSession;
import ipsecman.entity.*;

/**
 * This class represents a manuel IPSec tunnel setup. Manager
 * delegates necessary tunnelling parameters and peer IPSec gateways.
 * When this program (script) is run it communicates with peers using
 * SNMP and set necessary tunnel parameters to IPSec SNMP agent's
 * tables. Configuration takes place in two level. In first level,
 * necessary table rows are created in Security Association, Crypto
 * Map Inbound and Outbound Policy Tables, and whenever any error
 * occurs, previous configurations are rolled back. In second level,
 * after all rows are created in two peers, those rows are activated
 * using related SNMP set operations.
 * @author Mucahit Mutlugun
 * @version 1.0
 */

public class ManuelTunnelSetup {
    String peer1;
    String peer2;
    /*Policy Parameters*/
    int peer1polInIndex;
    int peer2polInIndex;
    int peer1polOutIndex;
    int peer2polOutIndex;
    String sourceIP;
    String destIP;
    String protocol;
    String sourcePort;
    String destPort;
    boolean alarm;
    /*SA Parameters*/
    int peer1SAIndex;
    int peer2SAIndex;
    String saName;
    int SPI;
    int transform;
    String authKeyID;
    String encryptKeyID;
    SnmpSession peer1Ses;
    SnmpSession peer2Ses;
```

```

/*CMapParameters*/
int peer1cMapIndex;
int peer2cMapIndex;
String cMapName;
static final int INBOUND = 1;
static final int OUTBOUND = 2;

String saFreeIndexOID = "1.3.6.1.4.1.1.1.4.1.0";
String cMapFreeIndexOID = "1.3.6.1.4.1.1.1.4.2.0";
String inboundPolicyFreeIndexOID = "1.3.6.1.4.1.1.1.4.3.0";
String outboundPolicyFreeIndexOID = "1.3.6.1.4.1.1.1.4.4.0";

String inboundPolicyEntryOID = "1.3.6.1.4.1.1.1.3.1.1";
String outboundPolicyEntryOID = "1.3.6.1.4.1.1.1.3.2.1";
String manualSAEntryOID = "1.3.6.1.4.1.1.1.3.3.1";
String cMapEntryOID = "1.3.6.1.4.1.1.1.3.5.1";

int[] policyMask = {
    1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
int[] saMask = {
    1, 2, 4, 5, 6, 7, 8, 9};
int[] cMapMask = {
    1, 2, 3, 5, 8};
String errorMessage;

public ManuelTunnelSetup(IpsecAgent ial, IpsecAgent ia2) throws
    SnmpException,
    IOException {
    this.peer1 = ial.getIpAddress();
    this.peer2 = ia2.getIpAddress();
    peer1Ses = new SnmpSession(peer1, ial.getPort(),
                                ial.getCommunity());
    peer2Ses = new SnmpSession(peer2, ia2.getPort(),
                                ia2.getCommunity());
}

public boolean setupTunnels() {
    try {
        configureSA(peer1);
    }
    catch (Exception ex) {
        setErrorMessage(peer1 + " " + ex.getMessage());
        return false;
    }
    try {
        configureCMap(peer1);
    }
    catch (Exception ex1) {
        setErrorMessage(peer1 + " " + ex1.getMessage());
        rollbackSA(peer1);
        return false;
    }
    try {
        configurePolicy(INBOUND, peer1);
    }
    catch (Exception ex1) {
        setErrorMessage(peer1 + " " + ex1.getMessage());
        rollbackSA(peer1);
        rollbackCMap(peer1);
        return false;
    }
}

```

```
try {
    configurePolicy(OUTBOUND, peer1);
}
catch (Exception ex1) {
    setErrorMessage(peer1 + " " + ex1.getMessage());
    rollbackSA(peer1);
    rollbackCMap(peer1);
    rollbackPolicy(INBOUND, peer1);
    return false;
}

try {
    configureSA(peer2);
}
catch (Exception ex) {
    setErrorMessage(peer2 + " " + ex.getMessage());
    rollbackSA(peer1);
    rollbackCMap(peer1);
    rollbackPolicy(INBOUND, peer1);
    rollbackPolicy(OUTBOUND, peer1);
    return false;
}

try {
    configureCMap(peer2);
}
catch (Exception ex1) {
    setErrorMessage(peer2 + " " + ex1.getMessage());
    rollbackSA(peer1);
    rollbackCMap(peer1);
    rollbackPolicy(INBOUND, peer1);
    rollbackPolicy(OUTBOUND, peer1);
    rollbackSA(peer2);
    return false;
}

try {
    configurePolicy(INBOUND, peer2);
}
catch (Exception ex1) {
    setErrorMessage(peer2 + " " + ex1.getMessage());
    rollbackSA(peer1);
    rollbackCMap(peer1);
    rollbackPolicy(INBOUND, peer1);
    rollbackPolicy(OUTBOUND, peer1);
    rollbackSA(peer2);
    rollbackCMap(peer2);
    return false;
}

try {
    configurePolicy(OUTBOUND, peer2);
}
catch (Exception ex1) {
    setErrorMessage(peer2 + " " + ex1.getMessage());
    rollbackSA(peer1);
    rollbackCMap(peer1);
    rollbackPolicy(INBOUND, peer1);
    rollbackPolicy(OUTBOUND, peer1);
    rollbackSA(peer2);
    rollbackCMap(peer2);
    rollbackPolicy(INBOUND, peer2);
    return false;
}
```

```

try {
    activateTables(peer1);
    activateTables(peer1);
}
catch (Exception ex2) {
}

return true;
}

public void activateTables(String peer) throws
    SnmpException, IOException,
    SnmpResponseException, SnmpDecodeException {
    SnmpSession s = getSession(peer);
    Vector varbinds = new Vector();
    varbinds.add(new Varbind(new OID(manualSAEntryOID + "." +
        saMask[7] +
        "." + getSAIndex(peer)),
        new Asn1Value(1)));
    varbinds.add(new Varbind(new OID(cMapEntryOID + "." +
        cMapMask[4] + "." +
        getCMapIndex(peer)),
        new Asn1Value(1)));
    varbinds.add(new Varbind(new OID(getPolicyOid(INBOUND) + "." +
        policyMask[10]
        + "." +
        getPolicyIndex(INBOUND,
peer) +
        "." + 1),
        new Asn1Value(1)));
    varbinds.add(new Varbind(new OID(getPolicyOid(OUTBOUND) +
"." +
        policyMask[10]
        + "." +
        getPolicyIndex(OUTBOUND,
peer) +
        "." + 1),
        new Asn1Value(1)));
    Varbind[] response;
    response = s.setRequest(varbinds);
}

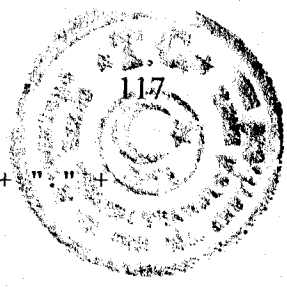
public String getErrorMessage() {
    return errorMessage;
}

private void setErrorMessage(String errorMessage) {
    this.errorMessage = errorMessage;
}

private void configureSA(String peer) throws
    SnmpException, IOException,
    SnmpResponseException, SnmpDecodeException {
    int saIndex;
    SnmpSession s = getSession(peer);
    String sindex = s.getRequest(saFreeIndexOID);
    saIndex = Integer.parseInt(sindex);
    setSAIndex(peer, saIndex);

    Vector varbinds = new Vector();

```



```
varbinds.add(new Varbind(new OID(manualSAEntryOID + "." +
                                saMask[1] +
                                "." + saIndex),
                                new Asn1Value(saName)));
varbinds.add(new Varbind(new OID(manualSAEntryOID + "." +
                                saMask[2] +
                                "." + saIndex),
                                new Asn1Value(SPI)));
varbinds.add(new Varbind(new OID(manualSAEntryOID + "." +
                                saMask[3] +
                                "." + saIndex),
                                new
Asn1Value(getRemotePeer(peer))));
varbinds.add(new Varbind(new OID(manualSAEntryOID + "." +
                                saMask[4] +
                                "." + saIndex),
                                new Asn1Value(transform)));
varbinds.add(new Varbind(new OID(manualSAEntryOID + "." +
                                saMask[5] +
                                "." + saIndex),
                                new Asn1Value(authKeyID)));
varbinds.add(new Varbind(new OID(manualSAEntryOID + "." +
                                saMask[6] +
                                "." + saIndex),
                                new Asn1Value(encryptKeyID)));
varbinds.add(new Varbind(new OID(manualSAEntryOID + "." +
                                saMask[7] +
                                "." + saIndex),
                                new Asn1Value(5))); //RowStatus

CreateAndWait
    Varbind[] response;
    response = s.setRequest(varbinds);
}

private void configureCMap(String peer) throws
    SnmpException, IOException,
    SnmpResponseException, SnmpDecodeException {
    int cMapIndex;
    SnmpSession s = getSession(peer);
    String cindex = s.getRequest(cMapFreeIndexOID);
    cMapIndex = Integer.parseInt(cindex);
    setCMapIndex(peer, cMapIndex);

    Vector varbinds = new Vector();
    varbinds.add(new Varbind(new OID(cMapEntryOID + "." +
                                    cMapMask[1] + "." +
                                    cMapIndex),
                                    new Asn1Value(cMapName)));
    varbinds.add(new Varbind(new OID(cMapEntryOID + "." +
                                    cMapMask[2] + "." +
                                    cMapIndex),
                                    new Asn1Value(1))); //Manuel Setup
    varbinds.add(new Varbind(new OID(cMapEntryOID + "." +
                                    cMapMask[3] + "." +
                                    cMapIndex),
                                    new Asn1Value(saName)));
    varbinds.add(new Varbind(new OID(cMapEntryOID + "." +
                                    cMapMask[4] + "." +
                                    cMapIndex),
                                    new Asn1Value(5))); //RowStatus

    CreateAndWait
```



```
Varbind[] response;
response = s.setRequest(varbinds);
}

private void configurePolicy(int table, String peer) throws
    SnmpException, IOException,
    SnmpResponseException, SnmpDecodeException {
    int polindex;
    SnmpSession s = getSession(peer);
    String pindex = s.getRequest(getPolicyFreeIndexOid(table));
    polindex = Integer.parseInt(pindex);
    setPolicyIndex(peer, table, polindex);

    Vector varbinds = new Vector();

    if ( ( (table == OUTBOUND) && (peer.equals(peer1))) ||
        ( (table == INBOUND) && (peer.equals(peer2))) ) {
        varbinds.add(new Varbind(new OID(getPolicyOid(table) +
            "." +
            policyMask[2]
            + "." + polindex + "."
            +
            1),
            new Asn1Value(sourceIP)));
        varbinds.add(new Varbind(new OID(getPolicyOid(table) +
            "." +
            policyMask[3]
            + "." + polindex + "."
            +
            1),
            new Asn1Value(destIP)));
    }

    if ( ( (table == INBOUND) && (peer.equals(peer1))) ||
        ( (table == OUTBOUND) && (peer.equals(peer2))) ) {
        varbinds.add(new Varbind(new OID(getPolicyOid(table) +
            "." +
            policyMask[2]
            + "." + polindex + "."
            +
            1),
            new Asn1Value(destIP)));
        varbinds.add(new Varbind(new OID(getPolicyOid(table) +
            "." +
            policyMask[3]
            + "." + polindex + "."
            +
            1),
            new Asn1Value(sourceIP)));
    }

    varbinds.add(new Varbind(new OID(getPolicyOid(table) + "." +
        policyMask[4]
        + "." + polindex + "." +
        1),
        new Asn1Value(protocol)));

    if ( ( (table == OUTBOUND) && (peer.equals(peer1))) ||
        ( (table == INBOUND) && (peer.equals(peer2))) ) {
        varbinds.add(new Varbind(new OID(getPolicyOid(table) +
            "." +
```

```

        policyMask[5]
        + "." + polindex + "."
+
        1),
        new Asn1Value(sourcePort)));
    varbinds.add(new Varbind(new OID(getPolicyOid(table) +
        "." +
        policyMask[6]
        + "." + polindex + "."
+
        1),
        new Asn1Value(destPort)));
    }
    if ( ( (table == INBOUND) && (peer.equals(peer1))) ||
        ( (table == OUTBOUND) && (peer.equals(peer2))) ) {
        varbinds.add(new Varbind(new OID(getPolicyOid(table) +
            "." +
            policyMask[5]
            + "." + polindex + "."
+
            1),
            new Asn1Value(destPort)));
        varbinds.add(new Varbind(new OID(getPolicyOid(table) +
            "." +
            policyMask[6]
            + "." + polindex + "."
+
            1),
            new Asn1Value(sourcePort)));
    }
    varbinds.add(new Varbind(new OID(getPolicyOid(table) + "." +
        policyMask[7]
        + "." + polindex + "." +
1),
        new Asn1Value(3))); //Decision:
    apply ipsec
        varbinds.add(new Varbind(new OID(getPolicyOid(table) + "." +
            policyMask[8]
            + "." + polindex + "." +
1),
            new Asn1Value( (alarm) ? 1 : 2)));
    varbinds.add(new Varbind(new OID(getPolicyOid(table) + "." +
        policyMask[9]
        + "." + polindex + "." +
1),
            new Asn1Value(cMapName)));
    varbinds.add(new Varbind(new OID(getPolicyOid(table) + "." +
        policyMask[10]
        + "." + polindex + "." +
1),
            new Asn1Value(5))); //RowStatus
    CreateAndWait
        Varbind[] response;
        response = s.setRequest(varbinds);
    }

    private void rollbackSA(String peer) {
        SnmpSession s = getSession(peer);
        int saIndex = getSAIndex(peer);

```

```

Vector varbinds = new Vector();
varbinds.add(new Varbind(new OID(manualSAEntryOID + "." +
                                saMask[7] +
                                "." + saIndex),
                                new Asn1Value(6)));

Varbind[] response;
try {
    response = s.setRequest(varbinds);
}
catch (Exception ex) {
    setErrorMessage(peer + " SA rollback failed " +
                    ex.getMessage());
}

}

private void rollbackCMap(String peer) {
    SnmpSession s = getSession(peer);
    int cmapIndex = getCMapIndex(peer);
    Vector varbinds = new Vector();
    varbinds.add(new Varbind(new OID(cMapEntryOID + "." +
                                    cMapMask[4] +
                                    "." + cmapIndex),
                                    new Asn1Value(6)));

    Varbind[] response;
    try {
        response = s.setRequest(varbinds);
    }
    catch (Exception ex) {
        setErrorMessage(peer + " CMap rollback failed " +
                        ex.getMessage());
    }

}

private void rollbackPolicy(int table, String peer) {
    SnmpSession s = getSession(peer);
    int policyIndex = getPolicyIndex(table, peer);
    Vector varbinds = new Vector();
    varbinds.add(new Varbind(new OID(getPolicyOid(table) + "." +
                                    policyMask[10] +
                                    "." + policyIndex + "." +
                                    new Asn1Value(6))));

    Varbind[] response;
    try {
        response = s.setRequest(varbinds);
    }
    catch (Exception ex) {
        setErrorMessage(peer +
            ( (table == INBOUND) ? " Inbound" :
              " Outbound") +
            " Policy rollback failed " +
            ex.getMessage());
    }

}

private int getSAIndex(String peer) {
    if (peer.equals(peer1)) {
        return peer1SAIndex;
    }
    else {
        return peer2SAIndex;
    }
}

```

```

    }
}

private int getCMapIndex(String peer) {
    if (peer.equals(peer1)) {
        return peer1cMapIndex;
    }
    else {
        return peer2cMapIndex;
    }
}

private int getPolicyIndex(int table, String peer) {
    if (peer.equals(peer1) && (table == INBOUND)) {
        return peer1polInIndex;
    }
    else if (peer.equals(peer1) && (table == OUTBOUND)) {
        return peer1polOutIndex;
    }
    else if (peer.equals(peer2) && (table == INBOUND)) {
        return peer2polInIndex;
    }
    else { //if (peer.equals(peer2) && (table == OUTBOUND))
        return peer2polOutIndex;
    }
}

private void setPolicyIndex(String peer, int table, int pindex)
{
    if (peer.equals(peer1) && (table == INBOUND)) {
        peer1polInIndex = pindex;
    }
    else if (peer.equals(peer1) && (table == OUTBOUND)) {
        peer1polOutIndex = pindex;
    }
    else if (peer.equals(peer2) && (table == INBOUND)) {
        peer2polInIndex = pindex;
    }
    else if (peer.equals(peer2) && (table == OUTBOUND)) {
        peer2polOutIndex = pindex;
    }
}

private void setCMapIndex(String peer, int cMapIndex) {
    if (peer.equals(peer1)) {
        peer1cMapIndex = cMapIndex;
    }
    else {
        peer2cMapIndex = cMapIndex;
    }
}

private void setSAIndex(String peer, int saIndex) {
    if (peer.equals(peer1)) {
        peer1SAIndex = saIndex;
    }
    else {
        peer2SAIndex = saIndex;
    }
}
}

```

```

private String getPolicyOid(int table) {
    if (table == INBOUND) {
        return inboundPolicyEntryOID;
    }
    else {
        return outboundPolicyEntryOID;
    }
}

private String getPolicyFreeIndexOid(int table) {
    if (table == INBOUND) {
        return inboundPolicyFreeIndexOID;
    }
    else {
        return outboundPolicyFreeIndexOID;
    }
}

private SnmpSession getSession(String peer) {
    if (peer.equals(peer1)) {
        return peer1Ses;
    }
    else if (peer.equals(peer2)) {
        return peer2Ses;
    }
    else {
        return null;
    }
}

private String getRemotePeer(String peer) {
    if (peer.equals(peer1)) {
        return peer2;
    }
    else if (peer.equals(peer2)) {
        return peer1;
    }
    else {
        return null;
    }
}

/*Those parameters are assumed to belong to outbound policy
of peer1*/
public void setPolicyParameters(String sourceIP, String destIP,
                                String protocol,
                                String sourcePort,
                                String destPort, boolean alarm)
{
    this.sourceIP = sourceIP;
    this.destIP = destIP;
    this.protocol = protocol;
    this.sourcePort = sourcePort;
    this.destPort = destPort;
    this.alarm = alarm;
}

public void setSAParameters(String name, int SPI,
                             int transform, String authKeyID,
                             String encryptKeyID) {
    this.saName = name;

```

```

    this.SPI = SPI;
    this.transform = transform;
    this.authKeyID = authKeyID;
    this.encryptKeyID = encryptKeyID;
}

public void setCMapParameters(String name) {
    this.cMapName = name;
}

public static void main(String[] argv) {
    String peer1 = null, peer2 = null, community1 = "public",
        community2 = "public";
    int port1 = 161, port2 = 161;
    String sip = "*. *.*.*", dip = "*. *.*.*", sport = "*",
        dport = "*",
        proto = "*";
    String saName = "saName";
    int spi = 1000, trans = 1;
    boolean alarm = false;
    String authKey = "1", encKey = "1";
    String cmapName = "cmapName";
    for (int i = 0; i < argv.length; i++) {
        if (argv[i].equals("-p1")) {
            peer1 = argv[i + 1];
        }
        if (argv[i].equals("-p2")) {
            peer2 = argv[i + 1];
        }
        if (argv[i].equals("-pr1")) {
            port1 = Integer.parseInt(argv[i + 1]);
        }
        if (argv[i].equals("-pr2")) {
            port2 = Integer.parseInt(argv[i + 1]);
        }
        if (argv[i].equals("-c1")) {
            community1 = argv[i + 1];
        }
        if (argv[i].equals("-c2")) {
            community2 = argv[i + 1];
            /*Policy arguments*/
        }
        if (argv[i].equals("-si")) {
            sip = argv[i + 1];
        }
        if (argv[i].equals("-di")) {
            dip = argv[i + 1];
        }
        if (argv[i].equals("-pr")) {
            proto = argv[i + 1];
        }
        if (argv[i].equals("-sp")) {
            sport = argv[i + 1];
        }
        if (argv[i].equals("-dp")) {
            dport = argv[i + 1];
        }
        if (argv[i].equals("-a")) {
            alarm = true;
            /*Manual sa arguments*/
        }
    }
}

```

```

    }
    if (argv[i].equals("-sn")) {
        saName = argv[i + 1];
    }
    if (argv[i].equals("-spi")) {
        spi = Integer.parseInt(argv[i + 1]);
    }
    if (argv[i].equals("-tr")) {
        trans = Integer.parseInt(argv[i + 1]);
    }
    if (argv[i].equals("-ak")) {
        authKey = argv[i + 1];
    }
    if (argv[i].equals("-ek")) {
        encKey = argv[i + 1];
        /*Cmap arguments*/
    }
    if (argv[i].equals("-cn")) {
        cmapName = argv[i + 1];
    }
}

IpsecAgent ia1 = new IpsecAgent(peer1, community1, port1);
IpsecAgent ia2 = new IpsecAgent(peer2, community2, port2);
ManuelTunnelSetup tunnel = null;
try {
    tunnel = new ManuelTunnelSetup(ia1, ia2);
}
catch (Exception ex) {
    System.out.println("Tunnel failed: " + ex.getMessage());
}
tunnel.setPolicyParameters(sip, dip, proto, sport, dport,
                           alarm);
tunnel.setSAParameters(saName, spi, trans, authKey, encKey);
tunnel.setCMapParameters(cmapName);
if (tunnel.setupTunnels()) {
    System.out.println("Tunnel setup succesfully");
    System.exit(1); //no error
}
else {
    System.out.println("Tunnel failed: \n Reason: " +
                       tunnel.getErrorMessage());
    System.exit(6); //runtime error
}
}
}

```

ScheduledConfig.java

```

package ipsecman.config;

import java.io.*;
import java.util.*;

import disman.SnmpException;
import ipsecman.entity.*;
import ipsecman.util.*;
import jmgmt.asnl.*;
import jmgmt.snmp.io.*;

/**

```

*This class represents a scheduled configuration script. Manager
 *deletes this script with appropriate scheduling parameters. When
 *the scheduled time comes script disable or enable a ipsec policy
 *rule indicated in parameters also.

*
 * @author Mucahit Mutlugun
 * @version 1.0
 */

```
public class ScheduledConfig
    extends TimerTask {
    Vector agents;
    int inRuleNo;
    int outRuleNo;
    String inboundPolicyEntryOID = "1.3.6.1.4.1.1.1.3.1.1";
    String outboundPolicyEntryOID = "1.3.6.1.4.1.1.1.3.2.1";
    static final int INBOUND = 1;
    static final int OUTBOUND = 2;
    Vector sessions = new Vector();
    public static final int POLICYDISABLER = 1;
    public static final int POLICYENABLER = 2;
    int type;

    public ScheduledConfig(int inRuleNo, int outRuleNo, int type)
        throws SnmpException, IOException {
        agents = Utils.getDomainAgents();
        Enumeration enum = agents.elements();
        while (enum.hasMoreElements()) {
            IpsecAgent ag = (IpsecAgent) enum.nextElement();
            sessions.add(new SnmpSession(ag.getIpAddress(),
                                           ag.getPort(),
                                           ag.getCommunity()));
        }
        this.inRuleNo = inRuleNo;
        this.outRuleNo = outRuleNo;
        this.type = type;
    }

    private void execute(SnmpSession s, int table, int status) {
        int policyIndex = getRule(table);
        Vector varbinds = new Vector();
        varbinds.add(new Varbind(new OID(getPolicyOid(table) + "." +
                                           12 + "." + policyIndex + "." + 1),
                                  new Asn1Value(status)));

        Varbind[] response;
        try {
            response = s.setRequest(varbinds);
        }
        catch (Exception ex) {
            System.out.println(s.getPeer().address.getHostAddress()
                               + ( (table == INBOUND) ? " Inbound" : " Outbound") +
                               " Policy status change failed " +
                               ex.getMessage());
        }
    }

    private int getRule(int table) {
        if (table == INBOUND) {
            return inRuleNo;
        }
        else {

```

```
        return outRuleNo;
    }
}

private String getPolicyOid(int table) {
    if (table == INBOUND) {
        return inboundPolicyEntryOID;
    }
    else {
        return outboundPolicyEntryOID;
    }
}

public void run() {
    Enumeration enum = sessions.elements();
    while (enum.hasMoreElements()) {
        SnmpSession ses = (SnmpSession) enum.nextElement();
        if (type == this.POLICYDISABLER) {
            execute(ses, INBOUND, 2);
            execute(ses, OUTBOUND, 2);
        }
        if (type == this.POLICYENABLER) {
            execute(ses, INBOUND, 1);
            execute(ses, OUTBOUND, 1);
        }
    }
}

public static void main(String[] argv) {
    int year = 2003, month = 6, day1 = 1, hour1 = 17,
        minutel = 15;
    int day2 = 1, hour2 = 8, minute2 = 15;
    int period = 24; //in hours
    int irule = 1, orule = 1;
    for (int i = 0; i < argv.length; i++) {
        if (argv[i].equals("-y")) {
            i++;
            year = Integer.parseInt(argv[i]);
            continue;
        }
        if (argv[i].equals("-mo")) {
            i++;
            month = Integer.parseInt(argv[i]);
            continue;
        }
        if (argv[i].equals("-d1")) {
            i++;
            day1 = Integer.parseInt(argv[i]);
            continue;
        }
        if (argv[i].equals("-h1")) {
            i++;
            hour1 = Integer.parseInt(argv[i]);
            continue;
        }
        if (argv[i].equals("-mil")) {
            i++;
            minutel = Integer.parseInt(argv[i]);
            continue;
        }
        if (argv[i].equals("-p")) {
            i++;
            period = Integer.parseInt(argv[i]);
            continue;
        }
    }
}
```

```

        i++;
        period = Integer.parseInt(argv[i]);
        continue;
    }
    if (argv[i].equals("-ir")) {
        i++;
        irule = Integer.parseInt(argv[i]);
        continue;
    }
    if (argv[i].equals("-or")) {
        i++;
        orule = Integer.parseInt(argv[i]);
        continue;
    }
}

Calendar c1 = new GregorianCalendar();
c1.set(year, month - 1, day1, hour1, minutel, 0);
Calendar c2 = new GregorianCalendar();
c2.set(year, month - 1, day2, hour2, minute2, 0);

ScheduledConfig configPolicy1 = null;
ScheduledConfig configPolicy2 = null;
try {
    configPolicy1 = new ScheduledConfig(irule, orule,
                                        ScheduledConfig.
                                        POLICYENABLER);
    configPolicy2 = new ScheduledConfig(irule, orule,
                                        ScheduledConfig.
                                        POLICYDISABLER);
}
catch (SnmpException ex) {
}
catch (IOException ex) {
}

Timer t1 = new Timer();
System.out.println(c1.getTime().toString());
t1.scheduleAtFixedRate(configPolicy1, c1.getTime(),
                      period * 60 * 60 * 1000);
System.out.println(c2.getTime().toString());
Timer t2 = new Timer();
t2.scheduleAtFixedRate(configPolicy2, c2.getTime(),
                      period * 60 * 60 * 1000);
}
}

```

Olay Scriptleri

EventSystem.java

```
package ipsecman.event;
```

```
/**
 *This class represents a delegated event system. It collects SNMP
 *traps from IPSec agents in its domain. In addition this system is
 *capable of filtering events. In order to achieve this, it can
 *receive filter objects in filter class types extending EventFilter.
 *Those filters are carrying filtering info and action code when
 *filter accepts a trap. Also deleting filter objects from this
 *system is also possible. Filters are registered to this system
 *using socket connection from port 5000. Also unregistering of

```

```
*filter objects can be done using port 5050 by sending name of
*filter to this port.
* @author Mucahit Mutlugun
* @version 1.0
*/
import java.io.*;
import java.net.*;
import java.util.*;
```

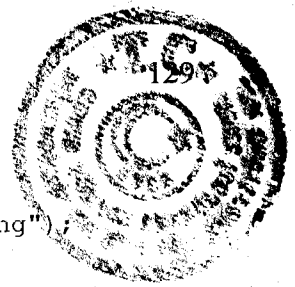
```
import com.adventnet.snmp.snmp2.*;
import ipsecman.event.filter.*;
import ipsecman.entity.*;
```

```
public class EventSystem
    implements Runnable {
    Snmp3Message msg;
    SnmpPDU pdu;
    String result;
    String error;
    DatagramSocket s;
    int bufferlength = 1000;
    byte[] buffer = new byte[bufferlength];
    DatagramPacket packet = new DatagramPacket(buffer,
buffer.length);
    Vector traps = new Vector();
    Vector filters = new Vector();
    ServerSocket servRecFilter;
    ServerSocket servDelFilter;
    private boolean filterReceiverRunning;

    public EventSystem() {
        try {
            s = new DatagramSocket(162);
            servRecFilter = new ServerSocket(5000);
            servDelFilter = new ServerSocket(5050);
        }
        catch (IOException ex) {
        }
    }

    public void startEventSystem() {
        while (true) {
            try {
                s.receive(packet);
                packet.getData();
                process();
            }
            catch (IOException ex) {
            }
        }
    }

    public void run() {
        if (!isFilterReceiverRunning()) {
            startFilterReceiver();
        }
        else {
            startFilterDeleter();
        }
    }
}
```



```
public void startFilterDeleter() {
    setFilterReceiverRunning(true);
    System.out.println("Filter Deleter is initializing");
    while (true) {
        try {
            Socket clientSocket = servDelFilter.accept();
            System.out.println("Incoming connection for filter
delete");
            DataInputStream is = new DataInputStream(
                new
BufferedInputStream(clientSocket.getInputStream()));
            ObjectInputStream objIn = new ObjectInputStream(is);
            try {
                String filterName = (String) objIn.readObject();
                removeFilter(filterName);
            }
            catch (ClassNotFoundException ex1) {
            }
            catch (IOException ex1) {
            }
        }
        catch (IOException ex) {
        }
    }
}

private void removeFilter(String filterName) {
    Enumeration enum = filters.elements();
    while (enum.hasMoreElements()) {
        EventFilter f = (EventFilter) enum.nextElement();
        if (f.getName().equals(filterName))
        {
            filters.remove(f);
            return;
        }
    }
    System.out.println("No filter with "+filterName+" is
found");
}

public void startFilterReceiver() {
    setFilterReceiverRunning(true);
    System.out.println("Filter Receiver is initializing");
    while (true) {
        try {
            Socket clientSocket = servRecFilter.accept();
            System.out.println("Incoming connection for filter
receiver");
            DataInputStream is = new DataInputStream(
                new
BufferedInputStream(clientSocket.getInputStream()));
            ObjectInputStream objIn = new ObjectInputStream(is);
            try {
                EventFilter filter = (EventFilter)
objIn.readObject();
                addFilter(filter);
            }
            catch (ClassNotFoundException ex1) {
            }
            catch (IOException ex1) {
            }
        }
    }
}
```

```

    }
    catch (IOException ex) {
    }
}

public void addFilter(EventFilter filter) {
    Enumeration enum = filters.elements();
    String filterName = filter.getName();
    while (enum.hasMoreElements()) {
        EventFilter f = (EventFilter) enum.nextElement();
        if (f.getName().equals(filterName)) {
            System.out.println("A filter with this name is
already defined");
            return;
        }
    }
    filters.add(filter);
}

public boolean decode(byte abyte0[]) {
    result = "PDU :\n" + getOctets(abyte0, abyte0.length);
    error = "";
    try {
        msg = new Snmp3Message(abyte0);
    }
    catch (Exception exception) {
        error = "Exception : " + exception.getMessage();
    }
    pdu = msg.getPDU();
    if (pdu != null) {
        try {
            if (!pdu.decode()) {
                error = "Decode error";
                return false;
            }
        }
        catch (Exception exception1) {
            error = "Exception : " + exception1.getMessage();
            return false;
        }
    }
    else {
        return false;
    }
    return true;
}

private byte[] constructNewArray(byte[] b, int length) {
    byte[] ret = new byte[length];
    for (int i = 0; i < ret.length; i++) {
        ret[i] = b[i];
    }
    return ret;
}

private String getOctets(byte abyte0[], int i) {
    StringBuffer stringBuffer = new StringBuffer();
    int j = 0;
    if (abyte0.length < i) {
        i = abyte0.length;
    }
}

```

```

    }
    for (int k = 0; k < i; k++) {
        if (j++ > 19) {
            j = 1;
            stringBuffer.append("\n");
        }
        String s = Integer.toString(abyte0[k] & 0xff, 16);
        if (s.length() < 2) {
            s = "0" + s;
        }
        stringBuffer.append(s + " ");
    }

    return stringBuffer.toString();
}

private void process() {
    byte[] result = packet.getData();
    byte[] b = new byte[packet.getLength()];
    for (int i = 0; i < packet.getLength(); i++) {
        b[i] = result[i];
    }
    if (decode(b)) {
        Vector v = pdu.getVariableBindings();
        SnmpVarBind var1 = (SnmpVarBind) v.elementAt(1);
        SnmpVarBind var2 = (SnmpVarBind) v.elementAt(2);
        IpsecTrap trp = new IpsecTrap();
        trp.setOID(var1.getVariable().toString());
        trp.setName("name");
        trp.setSourceIP(packet.getAddress().getHostAddress());
        trp.setMessage(var2.getVariable().toString());
        trp.setReceiveTime(new Date());
        System.out.println(trp.toString());
        filter(trp);
    }
}

public void filter(IpsecTrap trp) {
    Enumeration enum = filters.elements();
    while (enum.hasMoreElements()) {
        EventFilter ef = (EventFilter) enum.nextElement();
        if (ef.accepts(trp)) {
            ef.fireAction();
        }
    }
}

public boolean isFilterReceiverRunning() {
    return filterReceiverRunning;
}

public void setFilterReceiverRunning(boolean
filterReceiverRunning) {
    this.filterReceiverRunning = filterReceiverRunning;
}

public static void main(String[] args) {
    EventSystem trapRec = new EventSystem();
    Thread t1 = new Thread(trapRec);
    t1.start();
}

```

```

        Thread t2 = new Thread(trapRec);
        t2.start();
        trapRec.startEventSystem();
    }
}

```

EventFilter.java

```

package ipsecman.event.filter;

/**
 * @author Mucahit Mutlugun
 * @version 1.0
 */
import ipsecman.entity.IpsecTrap;
import java.io.Serializable;
import ipsecman.Action;

public abstract class EventFilter extends Action implements
Serializable{
    private String name;
    public EventFilter(String name) {
        this.name = name;
    }
    public abstract boolean accepts(IpsecTrap trp);

    public void setName(String name)
    {
        this.name = name;
    }

    public String getName()
    {return name;}
}

```

DeviceFilter.java

```

package ipsecman.event.filter;

/**
 * This class represents a device filter. It registers itself
 * to the event system.
 * @author Mucahit Mutlugun
 * @version 1.0
 */
import ipsecman.entity.IpsecTrap;
import java.net.Socket;
import java.io.*;
import java.io.ObjectOutputStream;
import java.util.Vector;

public class DeviceFilter extends EventFilter{
    Vector devices;
    public DeviceFilter(String name, Vector dev) {
        super(name);
        devices = dev;
    }
}

```

```

public boolean accepts(IpsecTrap trp)
{
    if (contains(trp.getSourceIP()))
        return true;
    else return false;
}
private boolean contains(String device)
{
    for (int i = 0; i < devices.size(); i++) {
        if (device.equals(devices.elementAt(i)))
            return true;
    }
    return false;
}

public void fireAction()
{
    System.out.println("Device filter action....\n");
}

public static void main(String[] argv) {
    Vector devices = new Vector();
    String name = "";
    for (int i = 0; i < argv.length; i++) {
        if (argv[i].equals("-n")) {
            i++;
            name = argv[i];
            continue;
        }
        devices.add(argv[i]);
    }

    DeviceFilter df = new DeviceFilter(name, devices);
    try {
        Socket sock = new Socket("127.0.0.1", 5000);
        ObjectOutputStream out = new
ObjectOutputStream(sock.getOutputStream());
        out.writeObject(df);
        out.flush();
        out.close();
    }
    catch (IOException ex) {
        System.out.println(ex.getMessage());
        System.exit(6);
    }
    System.exit(1);
}
}

```

TrapTypeFilter.java

```

package ipsecman.event.filter;

/**
 * This class represents a trap type filter. It registers itself
 * to the event system.
 * @author Mucahit Mutlugun
 *
 * @version 1.0
 */

```

```

import ipsecman.entity.IpsecTrap;
import java.net.Socket;
import java.io.*;
import java.io.ObjectOutputStream;
import java.util.Vector;

public class TrapTypeFilter extends EventFilter{
    Vector trapOids;
    public TrapTypeFilter(String name, Vector dev) {
        super(name);
        trapOids = dev;
    }

    public boolean accepts(IpsecTrap trp)
    {
        if (contains(trp.getOID()))
            return true;
        else return false;
    }
    private boolean contains(String device)
    {
        for (int i = 0; i < trapOids.size(); i++) {
            if (device.equals(trapOids.elementAt(i)))
                return true;
        }
        return false;
    }

    public void fireAction()
    {
        System.out.println("TrapTypeFilter action performed");
    }

    public static void main(String[] argv) {
        Vector trapOids = new Vector();
        String name = "";
        for (int i = 0; i < argv.length; i++) {
            if (argv[i].equals("-n")) {
                i++;
                name = argv[i];
                continue;
            }
            trapOids.add(argv[i]);
        }

        TrapTypeFilter tf = new TrapTypeFilter(name, trapOids);
        try {
            Socket sock = new Socket("127.0.0.1", 5000);
            ObjectOutputStream out = new
                ObjectOutputStream(sock.getOutputStream());
            out.writeObject(tf);
            out.flush();
            out.close();
        }
        catch (IOException ex) {
            System.out.println(ex.getMessage());
            System.exit(6);
        }
        System.exit(1);
    }
}

```

HourFilter.java

```

package ipsecman.event.filter;

/**
 * This class represents a hour filter. It filters traps between
 * certain hours. It registers itself to the event system.
 * @author Mucahit Mutlugun
 * @version 1.0
 */

import ipsecman.entity.IpsecTrap;
import java.net.Socket;
import java.io.*;
import java.io.ObjectOutputStream;
import java.util.Vector;
import java.util.Date;
import java.util.Calendar;
import java.util.GregorianCalendar;

public class HourFilter
    extends EventFilter {
    int h1, h2;
    public HourFilter(String name, int h1, int h2) {
        super(name);
        this.h1 = h1;
        this.h2 = h1;
    }

    public boolean accepts(IpsecTrap trp) {
        Date d= trp.getReceiveTime();
        GregorianCalendar c = new GregorianCalendar();
        c.setTime(d);
        int h = c.get(Calendar.HOUR_OF_DAY);
        if ((h <= h2) && (h >= h1))
            return true;
        else
            return false;
    }

    public void fireAction() {
        System.out.println("\tHour filter action....\n");
    }

    public static boolean checkHours(int h1, int h2)
    {
        if ((h1 < 0) || (h1 > 24))
            return false;
        if ((h2 < 0) || (h2 > 24))
            return false;
        if (h1 > h2)
            return false;
        return true;
    }

    public static void main(String[] argv) {
        Vector devices = new Vector();
        String name = "";
        int h1=11, h2=11;
        for (int i = 0; i < argv.length; i++) {
            if (argv[i].equals("-n")) {
                i++;
                name = argv[i];
            }
        }
    }
}

```

```

        continue;
    }
    if (argv[i].equals("-h1")) {
        i++;
        h1 = Integer.parseInt(argv[i]);
        continue;
    }
    if (argv[i].equals("-h2")) {
        i++;
        h2 = Integer.parseInt(argv[i]);
        continue;
    }
}

if (!HourFilter.checkHours(h1, h2))
    System.exit(-1);
HourFilter df = new HourFilter(name, h1, h2);
try {
    Socket sock = new Socket("127.0.0.1", 5000);
    ObjectOutputStream out = new
        ObjectOutputStream(sock.getOutputStream());
    out.writeObject(df);
    out.flush();
    out.close();
}
catch (IOException ex) {
    System.out.println(ex.getMessage());
    System.exit(6);
}
System.exit(1);
}
}

```

CompositeFilter.java

```

package ipsecman.event.filter;

/**
 * This class represents a composite filter. It includes other
 * filters. It registers itself to the event system.
 * @author Mucahit Mutlugun
 * @version 1.0
 */

import ipsecman.entity.IpsecTrap;
import java.net.Socket;
import java.io.*;
import java.io.ObjectOutputStream;
import java.util.Vector;
import java.util.Enumeration;

public class CompositeFilter
    extends EventFilter {
    Vector filters;
    public CompositeFilter(String name, Vector dev) {
        super(name);
        filters = dev;
    }

    public boolean accepts(IpsecTrap trp) {

```

```

Enumeration enum = filters.elements();
while (enum.hasMoreElements()) {
    if (!( (EventFilter) enum.nextElement()).accepts(trp))
        return false;
}
return true;
}

public void fireAction() {
    System.out.println("Composite filter action....\n");
}

public static void main(String[] argv) {
    Vector filters = new Vector();
    Vector devices = new Vector();
    Vector notifOids = new Vector();
    String name = "";
    DeviceFilter devf;
    TrapTypeFilter ttf;
    HourFilter hf;
    int h1 = 0, h2 = 24;
    for (int i = 0; i < argv.length; i++) {
        if (argv[i].equals("-n")) {
            i++;
            name = argv[i];
            continue;
        }
        if (argv[i].equals("-d")) {
            i++;
            devices.add(argv[i]);
            continue;
        }
        if (argv[i].equals("-o")) {
            i++;
            notifOids.add(argv[i]);
            continue;
        }
        if (argv[i].equals("-h1")) {
            i++;
            h1 = Integer.parseInt(argv[i]);
            continue;
        }
        if (argv[i].equals("-h2")) {
            i++;
            h2 = Integer.parseInt(argv[i]);
            continue;
        }
    }
    if (devices.size() != 0) {
        devf = new DeviceFilter(name, devices);
        filters.add(devf);
    }
    if (notifOids.size() != 0) {
        ttf = new TrapTypeFilter(name, notifOids);
        filters.add(ttf);
    }
    if (HourFilter.checkHours(h1, h2)) {
        hf = new HourFilter(name, h1, h2);
        filters.add(hf);
    }
}

```



```
}
CompositeFilter cf = new CompositeFilter(name, filters);
try {
    Socket sock = new Socket("127.0.0.1", 5000);
    ObjectOutputStream out = new ObjectOutputStream(sock.
        getOutputStream());
    out.writeObject(cf);
    out.flush();
    out.close();
}
catch (IOException ex) {
    System.out.println(ex.getMessage());
    System.exit(6);
}
System.exit(1);
}
```

RemoveFilter.java

```
package ipsecman.event.filter;

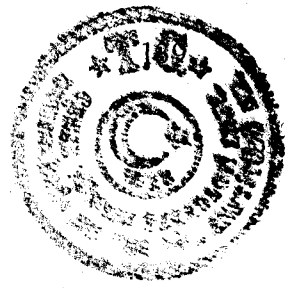
import java.util.Vector;
import java.net.Socket;
import java.io.ObjectOutputStream;
import java.io.IOException;

/**
 * This unregisters filters from EventSystem according to their
 * names
 * @author Mucahit Mutlugun
 * @version 1.0
 */

public class RemoveFilter {
    public RemoveFilter() {
    }

    public static void main(String[] argv) {
        Vector names = new Vector();
        String name;
        for (int i = 0; i < argv.length; i++) {
            if (argv[i].equals("-n")) {
                name = new String(argv[i + 1]);
                names.add(name);
            }
        }
        try {
            for (int i = 0; i < names.size(); i++) {
                String filterName = (String) names.elementAt(i);
                Socket sock = new Socket("127.0.0.1", 5050);
                ObjectOutputStream out = new
ObjectOutputStream(sock.getOutputStream());
                out.writeObject(filterName);
                out.flush();
                out.close();
            }
        }
        catch (IOException ex) {

```



```
        System.out.println(ex.getMessage());
        System.exit(6);
    }
    RemoveFilter removeFilter1 = new RemoveFilter();
    System.exit(1);
}
}
```

Action.java

```
package ipsecman;

/**
 * This class represents a base for action-enabled scripts.
 * @author Mucahit Mutlugun
 * @version 1.0
 */

public abstract class Action {
    public Action() {
    }

    public abstract void fireAction();
}
```

Gözlem Scriptleri

Monitor.java

```
package ipsecman.monitoring;

import com.adventnet.snmp.beans.*;
import com.adventnet.snmp.snmp2.SnmpOID;
import com.adventnet.snmp.snmp2.SnmpVar;

import ipsecman.Action;
import ipsecman.util.*;

/**
 * This class represents a base for monitoring scripts.
 * @author Mucahit Mutlugun
 * @version 1.0
 */

public abstract class Monitor extends Action{
    private String managerIP;
    private String managerCommunity;
    private int managerTrapPort;
    final static String notificationMessageOid =
".1.3.6.1.4.1.1.2.1.1";

    String notifOid;
    String message;
    SnmpSession session;
    public Monitor() {
    }
}
```

```

public void fireAction()
{
    fireEvent();
}

public void setActionParameters(SnmpSession ses, String message,
String notifOid)
{
    this.session = ses;
    this.message = message;
    this.notifOid = notifOid;
}

protected synchronized void fireEvent() {
    System.out.println("****.....Event is fired.....: ****\n"
+ message);
    SnmpTarget trapTarget = new SnmpTarget();
    trapTarget.setTargetHost(managerIP);
    trapTarget.setCommunity(managerCommunity);
    trapTarget.setTargetPort(managerTrapPort);
    trapTarget.addObjectID(notificationMessageOid);

    SnmpOID oid = new SnmpOID(notifOid);
    SnmpVar[] vb = new SnmpVar[1];
    SnmpVar v = null;
    try {
        v =
SnmpVar.createVariable(session.getPeer().address.toString() + " " +
                        message, (byte) 4);
    }
    catch (com.adventnet.snmp.snmp2.SnmpException ex1) {}
    vb[0] = v;

    try {
        trapTarget.snmpSendNotification(1, oid, vb);
    }
    catch (DataException ex) {
        System.out.println(ex.getMessage());
    }
}

public String getManagerIP() {
    return managerIP;
}

public void setManagerIP(String managerIP) {
    this.managerIP = managerIP;
}

public String getManagerCommunity() {
    return managerCommunity;
}

public void setManagerCommunity(String managerCommunity) {
    this.managerCommunity = managerCommunity;
}

public int getManagerTrapPort() {
    return managerTrapPort;
}

public void setManagerTrapPort(int managerTrapPort) {

```

```

        this.managerTrapPort = managerTrapPort;
    }
}

```

Polling.java

```

package ipsecman.monitoring;

/**
 * This class represents a polling script. It polls all agents in
 * domain manager's domain and triggers a trap if any agent cannot
 * be reached.
 * @author Mucahit Mutlugun
 * @version 1.0
 */
import java.io.*;
import java.util.*;
import ipsecman.entity.*;
import ipsecman.util.*;

public class Polling
    extends Monitor
    implements Runnable {
    Vector sessions = new Vector();
    Vector sesStatus = new Vector();
    final static String gatewayReachableOid =
".1.3.6.1.4.1.1.2.1.3";
    final static String gatewayUnreachableOid =
".1.3.6.1.4.1.1.2.1.2";
    String pollingOid = "1.3.6.1.2.1.1.3.0"; //RFC 1213 sysUpTime
    int pollingPeriod; //in milliseconds
    int threshold;

    IpsecAgent agent;
    SnmpSession session;
    int status = 0;
    String name;
    int port;
    public Polling(String name, IpsecAgent ag,
                    int period,
                    int threshold) throws
        disman.SnmpException, IOException {
        this.name = name;
        this.agent = ag;
        session = new SnmpSession(ag.getIpAddress(), ag.getPort(),
                                   ag.getCommunity());
        pollingPeriod = period * 1000;
        this.threshold = threshold;
    }

    public void run() {
        String result;
        while (true) {
            try {
                result = session.getRequest(pollingOid);
                if (status == threshold) {
                    setActionParameters(session, "ulasiliyor",
                                         gatewayReachableOid);
                    fireAction();
                }
            }
        }
    }
}

```

```

        status = 0;
        System.out.println(agent.getIpAddress() + name +
            " ulasiyor: " +
            (new Date()).toString());
    }
    catch (Exception ex) {
        System.out.println(agent.getIpAddress() + name +
            " ulasilamadi: " +
            ex.getMessage());
        if (status == threshold) {
            System.out.println("Durum:" + status);
            continue;
        }
        if (status < threshold) {
            status++;
            System.out.println("Durum:" + status);
            if (status == threshold) {
                setActionParameters(session,
                    ex.getMessage(),
                    gatewayUnreachableOid);
                fireAction();
            }
        }
    }
    try {
        Thread.sleep(pollingPeriod);
    }
    catch (InterruptedException ex1) {
    }
}

public static void main(String[] argv) {
    int period = 2; //Default is 2 in Seconds
    int threshold = 3;
    BufferedReader in;
    Vector hosts = new Vector();
    Vector comms = new Vector();
    Vector ports = new Vector();
    for (int i = 0; i < argv.length; i++) {
        if (argv[i].equals("-p")) {
            i++;
            period = Integer.parseInt(argv[i]);
        }
        if (argv[i].equals("-t")) {
            i++;
            threshold = Integer.parseInt(argv[i]);
        }
    }
    Vector agents = Utils.getDomainAgents();

    try {
        for (int i = 0; i < agents.size(); i++) {
            Polling p = new Polling("Thread" + i,
                (IpsecAgent)
                    agents.elementAt(i),
                    period, threshold);
            p.setManagerIP("127.0.0.1");
            p.setManagerCommunity("public");
        }
    }
}

```

```

        p.setManagerTrapPort(162);
        Thread t = new Thread(p);
        t.start();
    }
}
catch (Exception ex) {
    System.out.println(ex.getMessage());
    System.exit(6);
}
System.exit(1);
}
}

```

Monitoring.java

```

package ipsecman.monitoring;

import java.io.*;
import java.util.*;
import disman.*;
import ipsecman.util.*;

/**
 * This class represents a monitoring script. It monitors an SNMP
 * MIB counter and triggers a trap if threshold is reached.
 * @author Mucahit Mutlugun
 * @version 1.0
 */

public class Monitoring
    extends Monitor {
    String peer;
    String community;
    String monOid, name;
    int port, threshold, period; //in milliseconds
    SnmpSession session;
    long duration; //in milliseconds
    Date endTime;
    final static String gatewayMonitorThresholdOid =
        ".1.3.6.1.4.1.1.2.1.4";

    public Monitoring(String peer, String community, int port,
        String name,
            String oid,
            int threshold, int period, long duration)
        throws
            disman.SnmpException, IOException {
        this.peer = peer;
        this.community = community;
        this.port = port;
        if ( (name == null) || (oid == null) ) {
            throw new IllegalArgumentException("Name and oid are
                necessary");
        }
        this.name = name;
        this.monOid = oid;
        this.threshold = threshold;
        this.period = period * 1000;
        this.duration = duration;
        session = new SnmpSession(peer, port, community);
        Date beginTime = new Date();
    }
}

```

```

        long time = beginTime.getTime();
        time += duration;
        endTime = new Date(time);
        System.out.println(endTime.toString());
    }

    public void monitor() {
        int oldvalue = 0;
        int newvalue = 0;
        int counter = 1;
        int difference;
        try {
            String result = session.getRequest(monOid);
            System.out.println("Result is: " + result);
            oldvalue = Integer.parseInt(result);
        }
        catch (Exception ex) {
            System.out.println(ex.getMessage());
        }
        while (!isEnded()) {
            try {
                Thread.sleep(period);
            }
            catch (InterruptedException ex) {
            }

            try {
                String result = session.getRequest(monOid);
                System.out.println("Result is: " + result);
                newvalue = Integer.parseInt(result);
                difference = newvalue - oldvalue;
                System.out.println("Mean is: " + difference /
counter);
                if (difference / counter >= threshold) {
                    String message = "Threshold " + threshold +
                        " is exceeded at " + name;
                    setActionParameters(session, message,
gatewayMonitorThresholdOid);
                    fireAction();
                }
                oldvalue = newvalue;
                counter = 1;
            }
            catch (Exception ex) {
                System.out.println(ex.getMessage());
                counter++;
            }
        }
    }

    public boolean isEnded() {
        Date time = new Date();
        return time.after(endTime);
    }

    public static void main(String[] argv) {
        String peer = "127.0.0.1";
        String community = "public";
        int port = 161;
        String name = null, oid = null;
    }

```

```

int threshold = 100;
int period = 5; //second
long duration = 600000; //milliseconds
for (int i = 0; i < argv.length; i++) {
    if (argv[i].equals("-p")) {
        peer = argv[i + 1];
    }
    if (argv[i].equals("-pr")) {
        port = Integer.parseInt(argv[i + 1]);
    }
    if (argv[i].equals("-c")) {
        community = argv[i + 1];
    }
    if (argv[i].equals("-n")) {
        name = argv[i + 1];
    }
    if (argv[i].equals("-o")) {
        oid = argv[i + 1];
    }
    if (argv[i].equals("-t")) {
        threshold = Integer.parseInt(argv[i + 1]);
    }
    if (argv[i].equals("-pe")) {
        period = Integer.parseInt(argv[i + 1]);
    }
    if (argv[i].equals("-d")) {
        duration = 60000 * Integer.parseInt(argv[i + 1]);
    }
}
//minutes
}
try {
    Monitoring mon = new Monitoring(peer, community, port,
                                   name,
                                   oid, threshold, period,
duration);
    mon.setManagerIP("127.0.0.1");
    mon.setManagerCommunity("public");
    mon.setManagerTrapPort(162);
    mon.monitor();
}
catch (Exception ex) {
    System.out.println(ex.getMessage());
    System.exit(6);
}
System.exit(1);
}
}

```

Diğer Sınıflar

IpsecAgent.java

```
package ipsecman.entity;
```

```
/**
```

```
 * This class represents an Ipsec Agent entity.
```

```
 * @author Mucahit Mutlugun
```

```
 * @version 1.0
```

```
 */
```

```
public class IpsecAgent {
    private String ipAddress;
    private String community;
    private int port;
    public IpsecAgent(String ip, String community, int port) {
        this.ipAddress = ip;
        this.community = community;
        this.port = port;
    }
    public String getIpAddress() {
        return ipAddress;
    }
    public void setIpAddress(String ipAddress) {
        this.ipAddress = ipAddress;
    }
    public String getCommunity() {
        return community;
    }
    public void setCommunity(String community) {
        this.community = community;
    }
    public int getPort() {
        return port;
    }
    public void setPort(int port) {
        this.port = port;
    }
}
```

IpsecTrap.java

```
package ipsecman.entity;

/**
 * This class represents an SNMP type Ipsec trap entity.
 * @author not attributable
 * @version 1.0
 */

public class IpsecTrap {

    private String name=null;
    private String OID=null;
    private String sourceIP=null;
    private String message=null;
    private java.util.Date receiveTime=null;
    public IpsecTrap() {
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getOID() {
        return OID;
    }
}
```



```
public void setOID(String OID) {
    this.OID = OID;
}

public String getSourceIP() {
    return sourceIP;
}

public void setSourceIP(String sourceIP) {
    this.sourceIP = sourceIP;
}

public String getMessage() {
    return message;
}

public void setMessage(String message) {
    this.message = message;
}

public java.util.Date getReceiveTime() {
    return receiveTime;
}

public void setReceiveTime(java.util.Date receiveTime) {
    this.receiveTime = receiveTime;
}

public String toString() {
    return "sourceIP: " + sourceIP + "\nOID: " + OID +
"\nmessage: " + message +
"\nreceiveTime: " + receiveTime.toString();
}
}
```

Utils.java

```
package ipsecman.util;

import java.io.*;
import java.util.*;
import ipsecman.entity.IpsecAgent;

/**
 * Utils class for ipsecman
 * @author Mucahit Mutlugun
 * @version 1.0
 */

public class Utils {
    private static String agentsFileName = "DomainAgents.dat";
    public static Vector getDomainAgents() {
        BufferedReader in;
        Vector agents = new Vector();
        try {
            in = new BufferedReader(new FileReader(agentsFileName));
        }
        catch (FileNotFoundException ex2) {
            System.out.println("DomainAgents file not found");
        }
    }
}
```



```
        return agents;
    }

    String line;
    try {
        while ( (line = in.readLine()) != null) {
            int a = line.indexOf(" ");
            String addr = line.substring(0, a);
            System.out.println(addr);
            String rest = line.substring(a + 1, line.length());
            a = rest.indexOf(" ");
            String community = rest.substring(0, a);
            System.out.println(community);
            String port = rest.substring(a + 1, rest.length());
            IpsecAgent ag = new IpsecAgent(addr, community,
                Integer.parseInt(port));
        }
        in.close();
        return agents;
    }
    catch (IOException ex3) {
        System.out.println("Could not read Agents file");
        return agents;
    }
}

public String getAgentsFileName() {
    return agentsFileName;
}

public void setAgentsFileName(String agentsFileName) {
    this.agentsFileName = agentsFileName;
}
}
```