ВЕНЕТА ЙОСИФОВА

Подсигуряване на процеса на IPv6 маршрутизация с IPSec тунелиране

Защита на протоколи OSPFv3 и BGP4/4+ базирана на технологии със свободен софтуер

ИЗДАТЕЛСТВО

Венета Йосифова

Подсигуряване на процеса на IPv6 маршрутизация с IPSec тунелиране

Защита на протоколи OSPFv3 и BGP4/4+ базирана на технологии със свободен софтуер

София, 2015
Издателство

© 2015 Венета Йосифова
© 2015 Издателство .....

ISBN 987-954-07-3973-1

Съдържание

1. Първа глава – Увод 9
1.1. Актуалност на проблема, причини налагащи прехода от IPv4 към IPv6 9
1.2. Поставени цели и задачи 15
1.3. Ползи от реализацията 16
1.4. Структура 17

2. Втора глава – Особености на протокола IPv6 и интеграцията му с IPsec 18
2.1. Протокол IPv6 18
2.1.1. Подобренията в новата версия 19
2.1.2. Адресиране 21
2.1.3. Формат на заглавната част на протокола 32
2.2. Приложение на IPsec в процеса на удостоверяване между съседни маршрутизатори 36
2.2.1. Характеристика и особености на IPSec 36
2.2.2. Authentication Header 40
2.2.3. Encapsulating Security Payload 43
2.3. Предимствата на IPsec пред други методи за удостоверяване в процеса на маршрутизация 45

3. Трета глава – Анализ на протокола за външна маршрутизация BGP в частта установяване на съседство между маршрутизатори 49
3.1. Изследване на същностна на протокола BGP, необходимо при процеса на конфигурация 49
3.1.1. Автономни системи 49
3.1.2. Протоколът BGP 50
3.1.3. Начин на работа на BGP 52
3.2. Предимствата на IPsec пред вграденият механизъм за удостоверяване с пароли при защитата на BGP от намеса на „трети страни“ в процеса на комуникация 60

4. Четвърта глава – Протокол за маршрутизация OSPFv3 и начин на функциониране 70
4.1. Изследване на особеностите на версия 3 на протокол OSPF 70
4.2. Използвани методи и средства за защита, чрез приложение на IPsec в процеса на маршрутизация 92

5. Пета глава – Проектиране и реализиране на модел на топология, приложима за онагледяване на процеса на защита на IPv6 маршрутизация с протоколи BGP4/4+ и OSPFv3 чрез изграждане на IPSec тунели 98
5.1. Технологии ползвани за реализирането на задачата 98
5.1.1. Особености при инсталацията на необходимия „*open source*“ маршрутизиращ софтуер 101
5.1.2. Конфигуриране на средата за симулация с виртуални машини 104
5.2. Представяне на модела на топология и създаване на адресна схема 107
5.2.1. Създаване на адресна схема 108
5.2.2. Представяне на видовете роли в моделната топология 113
5.3. Реализиране на метод за защита на процеса на маршрутизация, гарантиращ изграждането на стабилна и защитена комуникация между маршрутизаторите 114
5.3.1. Конфигуриране на устройствата 115
5.3.2. Конфигуриране на маршрутизаторите с OSPFv3 117
5.3.3. Конфигуриране на маршрутизаторите с BGP4/4+ 119
5.3.4. Конфигуриране на маршрутизаторите с IPSec тунелиране 121

6. Шеста глава – Анализ и оценка на реализираната защитена среда за маршрутизация 127
6.1. Съпоставка между маршрутизациите в не защитена и защитена среда 127
6.1.1. Маршрутизация в незащитена с IPSec среда 127
6.1.2. Маршрутизация в защитена с IPSec среда 131
6.2. Оценка на реализираната защитена среда за маршрутизация 132

7. Седма глава – Заключение 135
7.1. Постигнати резултати 135
7.2. Насоки за бъдещо изследване 137

Използвана литература и ресурси 138

Приложение 1 150

Приложение 2 153

1. Първа глава – Увод

1.1 Актуалност на проблема, причини налагащи прехода от IPv4 към IPv6

След световния старт на протокола IPv6 на 6 юни 2012 г., глобалната употреба на IPv6 се е увеличила двойно [72]. През 2013 г. се отбелязава за трета поредна година двойно увеличение на използването на IPv6 в световен мащаб. Ако тези тенденции се запазят, повече от половината интернет потребители в целия свят ще имат IPv6 свързаност за по-малко от шест години. [30, 150, 159] Според данните дадени от Cisco *Visual Networking Index (VNI)*, които се занимават с прогнозиране и анализ на растежа и използването на IP свързаността на мрежите в света, се очаква до 2017 г. 42% от всички настолни и мобилни мрежови устройства да бъдат IPv6 съвместими [43].

Подсигуряването на процеса на маршрутизация е проблем с огромно значение, както за локалните мрежи, така и за глобалната мрежа като цяло. Защитеното маршрутизиране, се опитва да намали рисковете и да затрудни злонамереното отклоняване на трафика. Опасността, при незащитена мрежова комуникация, недобронамерени лица да могат да виждат трафика и той да може да бъде лесно модифициран или отклоняван, е голяма. Осигуряването на такава защита е сложен проблем, имайки се предвид, че е достатъчно сложно дори самото конфигуриране на маршрутизирането, така че да работи наистина добре. Причината е, че тази защита, малко или повече излиза извън полето на традиционната комуникационна сигурност, при която проблемите обикновено произтичат от логически грешки в кода или недобри протоколи. Когато става въпрос за маршрутизиращи протоколи, те изпълняват задачите, за които са създадени: да обменят информация за топологията, да намират най-добър път, да препращат пакети по оптималното разстояние до целта. Проблемите при тях са изцяло от наличието на злонамерени участници в мрежовата комуникация. Това важи, както за протоколите за вътрешно маршрутизиране, какъвто е OSPF, така и за тези за външно (между различни автономни системи), като BGP [1, 2]. Разбира се, не винаги

става въпрос за целенасочена атака, но известни реално случили се инциденти показват мащабите на проблема. Пример за това е „*Pakistan Telecom blocks YouTube*“ от 2008 година. Пакистанските власти нареждат на националния телеком да филтрира достъпа до *YouTube* само в рамките на страната. Но пакистанският телеком, при опита си да филтрира достъпа до *YouTube*, изпраща грешна информация (фалшиви интернет адреси) през BGP до интернет доставчика PCCW в Хонконг, който от своя стана препраща тази грешна маршрутизираща информация през глобалната мрежа, като блокира достъпа до сайта за два часа. Грешната информация, която телекома и следващите интернет доставчици предават към всички маршрутизатори в глобалната мрежа, казва че най-добрият път, през който да минава всичкият *YouTube* трафик е през Пакистан, където той се блокира. Друг пример е когато малайзийски интернет доставчик *DataOne* блокира *Yahoo*. През 2004 година префикса на дейта центъра на *Yahoo* е бил откраднат (hijacked) от *DataOne*. За този инцидент експерти по мрежова сигурност твърдят, че е бил злонамерен и от *DataOne* умишлено са се опитали да блокират трафика към *Yahoo*. Още един известен инцидент, показващ мащабите на проблема с не контролираното разпространението на маршрутизираща информация в глобалната мрежа, е случаят с изтичането на пълна BGP таблица с маршрути от бразилски интернет доставчик CTBC през 2008 година. Това е можело да причини открадване (*hijacking*) на голяма част от адресите на планетата. Но за щастие никой, който не е бил клиент или директно свързан с техните доставчици не е получил тези грешни маршрути. Другите успяват да филтрират грешната информация и да избегнат проблема. [89] За това имат заслуга и *BGPmon Network Solutions* (фирма, която се занимава със следене на BGP атаки и „кражби на адреси“ – *hijacking*), които са засекли заплахата и са информирали своите клиенти по света за проблема. [90] През 2010 година китайски ISP (интернет доставчик) отклонява около 10% от целия интернет трафик, включително такъв към големи западни компании и медии. Трафикът е бил пренасочван към Китай за около двайсет минути, като от ISP твърдят, че това се е дължало на грешка в конфигурация на системата. След анализ на данните от *Renesys* (друга компания, занимаваща се с кибер

сигурност, специализирана в мониторинг на интернет пътищата), потвърждават че най-вероятната причина, наистина е грешка.

Ако тези реално случили се атаки не са били достатъчно убедителни, има една атака, демонстрирана през август 2008 година, от двама изследователи по сигурността по време на най-голямата конференция за хакери DEF CON. Тази атака е окачествената като „най-голямата“ BGP открита атака. Чрез нея се демонстрира как някой атакуващ може да подслуша или промени незащитени данни на която и да е компания. Така атакуващият (разузнавателна агенция, корпоративен шпионин или кибер престъпник) може да пренасочи всичкия трафик на компанията през своята собствена мрежа и след това да я препрати към истинската ѝ крайна цел без законният ѝ притежател да може да разбере това [3, 4, 5, 6, 91]. Пет години не беше известно дали някой се е възползвал от тази възможност. Но през май 2013 година се случва точно това. Мистериозно и временно (в продължение на няколко месеца) пренасочване на интернет трафик към Исландия и Беларус [92], точно както бе показано от двамата специалисти по сигурността на DEF CON’2008. В този конкретен случай нито изпращача, нито получателя разбира, че данните им са имали множество „междинни спирки“, като некриптираните може да са били копирани и променяни.

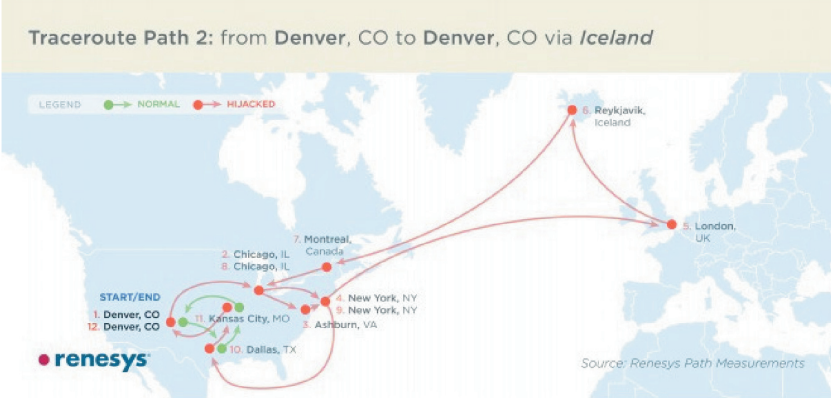
Използваната техника не атакува бъг или пропуск в BGP протокола, а се възползва от факта, че BGP се базира на доверяването, че получаваната информация е коректна. Един BGP маршрутизатор се доверява, че другата страна праща своя истински най-добър път до група от адреси. Когато се търси в маршрутната таблица адресът на получателя, се избира от този префикс, при който съвпадението е най-късо (тоест най-конкретния префикс). Възползвайки се от това, всеки който има контрол над BGP маршрутизатор, може да анонсира адресна информация, която да бъде предпочетена пред другите. Само минути са необходими за предаването ѝ в глобалната мрежа.

Обикновено, когато някой иска да препрати отклонения трафик към правилната посока, той се връща като бумеранг отново, тъй като другите маршрутизатори все още ще смятат неговия адрес за най-добър. При сегашната техника, когато трафикът е минал през маршрутизатора на атакуващия, се препраща директно към

истинския получател и междинните устройства не получават предупреждение. Атакуващият променя единствено полето, съдържащо адреса на получателя в хедъра на пакетите, но не и това на изпращача. Така тази атака за подслушване и прихващане на трафик от тип „човека по средата“ („*man-in-the-middle*“) е почти невъзможно да бъде открита за дълъг период от време.

Случилото се през 2013 година е засегнало трафик на правителствени и финансови компании, според експерти по киберсигурност. Пренасочването се е случвало всекидневно в продължение на месеци. Конкретни примери са отклоняването на трафика от Мексико към Русия и Беларус, преди да достигне целта си. Наблюдавано е и удължаване на пътя на трафика при комуникация между Чикаго и Иран. [73]

Когато пренасочването към Беларус е спряло, се е заменило с такова към Исландия. Това е показано и на Фигура 1.1. Според Doug Madory, експерт в *Renesys*, фактът, че последното отклоняване към Беларус е съвпаднало с първото към Исландия, с разлика от минути едно от друго, е показателно, че не става въпрос за грешка от страна на исландския доставчик. [73] И все пак не е ясно дали доставчиците на двете страни са замесени или техните мрежи са били също жертва на атака.



Фиг. 1.1 Пренасочване на трафика към Исландия. Трите зелени (по-светли) стрелки около Канзас показват нормалния път, а червените (по-тъмни), този който е реално изминат. [73]

Трудно е да се определи и дали тази атака е осъществена на национално ниво или е дело на престъпна организация. По думите на Madory, на единия ден, цел на атаката са финансови компании, на другия – чужда правителствена информация. Разузнавателните организации на Русия и Китай могат да бъдат заинтересовани от такава информация, но също така трябва да се има предвид, че и Националната агенция по сигурността (*National Security Agency – NSA*) на САЩ вече има директен достъп до оптичните кабели навсякъде по света, според документи, изтекли от Edward Snowden. [73, 74]

Още преди пет години, в свой доклад Европейската комисия предрича, че в настоящото десетилетие има между 10% и 20% вероятност телекомуникационните мрежи да бъдат засегнати от значителни аварии с потенциални икономически загуби възлизащи на 193 млрд. евро. Причините могат да бъдат, както природни бедствия и хардуедни повреди, така и вследствие на човешка дейност като тероризъм или кибератаки. В същото време африкански ИТ специалисти изчисляват, че 80% от всички компютърни станции на континента, включително и правителствени компютри, са засегнати от компютърни вируси. Те представляват потенциална опасност, да се превърнат в най-голямата бот мрежа в света. С един милион такива компютри може да се генерира толкова трафик, че да се блокира достъпа едновременно на 500 от най-големите компании в света. А с 10 милиона такива хоста, може да се блокира мрежовата инфраструктура на по-големите западни държави. [151]

Като последен пример през 2014 година станахме свидетели и на кражба на виртуалната валута *Bitcoins*, като предполагаемата сума е около 83 000 долара, използвайки същия метод за отклоняване (*hijacking*) [92, 93].

Всичко това показва, че злонамерено отклоняване, блокиране или промяна на трафика (*hijacking*) се е случвало и ще се случва и за в бъдеще, но при при липса на удостоверяване между участниците в комуникацията и предаване на некриптирани данни е много по-опасно и трябва да се обърне сериозно внимание.

За да се намали рискът от тези проблеми и да се предложи един по-съвременен и ефективен начин за решаването им, *Internet*

Engineering Task Force (IETF) стандартизират протокола IPSec да бъде част от протокола IPv6. Така защитата предлагана от IPSec е на IP ниво в TCP/IP протоколния стек, като по този начин се осъществява защита и за всички протоколи от по-високите нива. Протоколът IPSec е реализиран да използва AH (A*uthentication Header*) и ESP (*Encapsulating Security Payload*). Това позволява освен аутентикация (удостоверяване), да се извършва и криптиране на данните. По този начин, дори и някой да се опитва, примерно да „прослушва“ (*sniffing*) трафика в мрежата, за да прочете прихванатите данни, ще му се наложи да прилага някакъв метод на „грубата сила“ (*Brute-force attack*) към ESP ключа [7, 8].

Когато е бил създаван протоколът IPv4, никой не е предполагал, колко масово, навсякъде по света за всякакви устройства ще бъде използван. На теория при 32 бита, съществуват около 4 млрд. адреси, което изглежда много, но на практика реално използваемите са доста по-малко [11, 12]. Адресите IPv4 са се използвали още преди да съществуват трите, по-късно петте регионални регистратори (*Regional Internet Registries – RIR*), които да регулират раздаването на адресите по петте континента. Запасът от IPv4 адреси, който притежава IANA (*Internet Assigned Numbers Authority*, департамент в *Internet Corporation for Assigned Names and Numbers* – ICANN) е изчерпан от 1 февруари 2011, като на 3-ти февруари петте RIR получават последните блокове с дължина 8 (/8) (всеки с по 16,8 милиона IPv4 адреса) [9]. На 14 септември 2012 година европейският RIR – RIPE NCC, започва да раздава адреси от това адресно пространство. Това означава, че всеки локален член на RIPE NCC ще може да заяви и получи, еднократно, префикс /22 (1024 адреса), само ако вече имат разпределен дял от IPv6 адресното пространство [10]. Адресът вече е с дължина 128 бита, тоест 2^128 – около 3.4×10^38 адреса. Те не само са достатъчни за всяко устройство, което има достъп до глобалната мрежа, да получи собствен адрес, но в IPv6 е оптимизиран и са отстранени недостатъци на предшественика му. Най-важното подобрение е, че IPSec е вграден като част от IPv6, и тази защита е задължителна да бъде поддържана от всички IPv6 реализации в близко бъдеще [13], особено що се отнася до защита на трафика

между маршрутизаторите. По тази причина удостоверяването е било премахнато от OSPFv3 и вместо това се ползва IPv6 *Authentication Header* и *Encapsulating Security Payload* (ESP) от IPv6 [16]. Тези виртуални тунелни интерфейси (*Virtual Tunnel Interface – VTI*), грижещи се за *site-to-site* криптирането, защитават всички видове IPv6 трафик – *unicast* и *multicast* [17]. Други подобрения са: премахването на необходмостта да се използва NAT (*Network Address Translation*), способност за автоматична конфигурация и по-лесна администрация без DHCP, без повече колизии при частните адрес, опростено и по-ефективно маршрутизиране на пакетите, по-опростен формат на заглавната част на пакета (*header*), истинско управление на качеството на услугите (*Quality of Service – QoS*), наречено още *flow labeling* (етикиране на потоците), по-гъвкави опции и разширения (*options* и *extensions*) на протокола [14, 15].

Всички тези обстоятелства, описващи предимствата на протокола IPv6 пред IPv4, както и факта, IPv4 адресно пространство в периода 2015 г. – 2020 г. ще бъде напълно изчерпано, налагат прехода към новата версия [88]. А процесът на подсигуряване на защитена маршрутизация, на фона на все по-достъпната и мощна техника за закупуване от крайния потребител се превръща в един още по-актуален за практическо решаване проблем.

1.2 Поставени цели и задачи

Целта на настоящата работа е да се представи решение на конкретна задача, поставяща проблема по защита на процеса на IPv6 маршрутизация с протоколи BGP4/4+ и OSPFv3 чрез изграждане на IPSec тунели. От така поставената цел произтичат следните задачи:

1. Ще се направи изследване на свойствата на IPSec, приложими в процеса на аутентикация (удостоверяване) между съседни маршрутизатори, в контекста на протокола IPv6. Както и ще се докажат предимствата му пред други методи за удостоверяване в процеса на маршрутизация.

2. Ще се анализира протокола за външна маршрутизация BGP в частта установяване на съседство между маршрутизатори. Ще

се обясни защо вграденият механизъм за удостоверяване с пароли не е достатъчен за защита от намеса на „трети страни“ и защо се налага приложение на IPSec.

3. Ще се разгледа протокола OSPF. Ще се обясни функционирането му, като се наблегне на особеностите на версия 3, пригодена за работа в IPv6 среда. Детайлно ще се опишат използваните методи и средства за защита в процеса на маршрутизация – установяване на съседство, обмен на маршрути и други и по какъв начин ще се приложи IPSec.

4. На базата на проучванията от предните задачи ще се предложи и реализира метод за защита на процеса на маршрутизация, който да гарантира изграждането на стабилна и защитена комуникация между маршрутизаторите по протоколи BGP4/4+ и OSPFv3. За целта ще се реализира мрежа от маршрутизатори с изградени IPSec тунели между тях, реализирани на виртуални машини и ще се проведат тестове за сигурността в процеса на маршрутизация по BGP4/4+ и OSPFv3 в тази среда.

5. Проведените тестове ще послужат за анализ и оценка на реализираната технология, по-специално ефективността на защитата от измами с IP адреси (*IP Spoofing*) и други атаки, свързани с маршрутизацията.

1.3 Ползи от реализацията

Ползите от реализацията са изследването на проблема за сигурността при обмена на информация между маршрутизатори в дадена топология. И намирането на решение на конкретна задача, базирано на подобренията, свързани със сигурността при IPv6 (IPSec). Всичко това е приложено при подсигуряването на маршрутизацията с протоколи за динамична маршрутизация BGP4/4+ и OSPFv3.

Това решение ще може да послужи и помогне за решаването на проблеми, възникнали в различни реални ситуации. Както в процеса на обучение, така и на системни администратори и инженери, които искат да тестват своите нови конфигурации, като прилагането на IPSec при IPv6 базираните протоколи за маршрутизация, преди те

да бъдат пренесени на реални устройства. Причината за това е, че симулираната мрежова топология, с избраните „*open-source*“ технологии, позволяват да се експериментира със сложни мрежови конфигурации, правейки възможно максимално да се приближат до начина, по който работят реалните устройства в компютърните мрежи.

1.4 Структура

Първа глава въвежда в предметната област, описва целта, задачите и структурата на настоящата работа. Втора глава представя протоколът IPv6. Изследва свойствата на IPSec, приложими в процеса на удостоверяване между съседни маршрутизатори. Представя новата версия на IP протокола и интеграцията му с IPSec. Анализира предимствата на IPsec пред други методи за удостоверяване в процеса на маршрутизация. Трета глава разглежда протокола за външна маршрутизация BGP в частта установяване на съседство между маршрутизатори. Анализира предимствата на BGP протокола при използването на IPSec пред вграденият механизъм за удостоверяване с пароли при защитата от намеса на „трети страни“ в процеса на комуникация. В четвърта глава се разглеждат функционалните характеристики на протокола за маршрутизация OSPFv3. Изследват особеностите на версия 3 на OSPF, както и използваните методи и средства за защита, чрез прилагане на IPSec в процеса на маршрутизация. Пета глава е практическа и е свързана с проектиране и реализация на модел на топология, приложима за онагледяване на процеса на защита на IPv6 маршрутизация с протоколи BGP4/4+ и OSPFv3, чрез изграждане на IPSec тунели. В нея се реализира метод за защита на процеса на маршрутизация, гарантиращ изграждането на стабилна и защитена комуникация между маршрутизаторите. Шеста глава дава анализ и оценка на реализираната защитена среда за маршрутизация. В частност, се показва ефективността на защитата от *IP Spoofing* и други атаки, свързани с маршрутизацията, на база на проведени тестове.

2. Втора глава – Особености на протокола IPv6 и интеграцията му с IPsec

В тази глава ще бъдат разгледани някои нови характеристики, предоставени от IPv6 (*Internet Protocol version 6*). По-сериозно внимание ще се обърне на новите, подобрени начини за защита в IPv6, а именно, използването на *Authentication Header* (AH) и *Encapsulating Security Payload* (ESP). В главата ще се представи как тези подобрения в защитата могат да предотвратят някои видове мрежови атаки, съществуващи в момента.

2.1 Протокол IPv6

Досегашната версия на *Internet Protocol* (IP), който е и основният протокол, използван за изпращане на информация през глобалната мрежа е IPv4. Той датира от 70-те години на миналия век. Неговите основни недостатъци, от сегашна гледна точка, са малкото количество на адреси и проблемите свързани със сигурността. Протоколът IPv4 разполага с 32-битово адресно пространство, което вече на практика е изчерпано. Относно сигурността, единствената защита предоставяна от самия протокол, е полето *Security option*, което отговаря за специфични изисквания на DoD (*Department of Defence, USA*). Поради тези недостатъци, както и желанието за подобряване на бързодействието, по-лесното конфигуриране и управление на мрежите, *Internet Engineering Task Force* (IETF), още в началото на 90-те години на миналия век започва да разработва IPv6. Техните разработки са дефинирани в различни стандартизационни документи IETF – RFCs (примерно 1752, 2460, 2462, 2406 и др.).

Съкращението IPv6 идва от *Internet Protocol Version* 6, наричан още IPng (*Internet Protocol next generation*), тоест протокол от следващо поколение. Това е най-новата версия на IP, описана от комитета по стандартизация IETF, заменяща досега масово използваната версия IPv4. Тъй като IPv6 е наследник на предишната версия, той е проектиран като надграждащ протокол, което ще

позволи едновременното съществуване и използване на двете версии за определено време. Протоколът IPv6 е проектиран така, че да позволява глобалната мрежа да се разширява постоянно, както по отношение на броя на участващите устройства, така и по отношение на обема на предавания трафик. [14]

Но тъй като все още е невъзможно да се говори за изцяло преминаване към IPv6 и за спиране на употребата на IPv4, съществуват различни механизми за транзит на адреси както на едната версия, така и на другата. Някой от тях са: *Dual stack*, *Manual tunnel*, *6to4 tunnel*, *ISATAP tunnel*, *Teredo tunnel*. При IPv6 тунелирането механизмите обикновено представляват енкапсулация на IPv6 пакета в IPv4, тоест добавя се IPv4 хедър и така се предава през мрежа ползваща единствено старата версия. Ако ръчно се настройва подобна съвместимост, и двата крайни маршрутизатора се конфигурират с адреси от двете версии [43, 24].

2.1.1 Подобренията в новата версия

Въпреки че увеличаването на адресното пространство е основно подобрение, с което се характеризира новата версия на IP, има още важни технологични промени, които значително подобряват протокола.

Преди всичко протоколът IPsec е част от IPv6 (така нареченото *Native IPsec support*) и по този начин той вече притежава вграден механизъм за защита и удостоверяване. Тази интеграция на двата протокола може да се прилага както към хостове (*hosts*) така и към защитени шлюзове (*security gateways*). Поддръжката на *Native IPsec* (на IPv6) е налична в Линукс ядра след 2.6.x (*Linux 2.6.x kernels*) и при Cisco IOS след 12.4. Тук ядрата на операционните системи поддържат два вида бази от данни. Едната база е *Security association database* (SAD), съдържаща по едно SA (*Security Association*) за всеки интерфейс. Другата е *Security Policy Database* (SPD), която съдържа информация кой трафик трябва да бъде криптиран, в какъв режим работи – транспортен или тунелен (*transport* или *tunnel*), както и крайните точки в мрежата, които комуникират. При IPv6,

IPsec е реализиран, като се прилагат два хедъра – AH (*Authentication Header*) използван за удостоверяване и ESP (*Extension Header*) за конфиденциалност. Формата на хедъра на IPv6 е опростен, но същевременно са добавени полезни опции и разширения. Тъй като, на практика, IPv4 IPsec се поддържа от всички съвременни клиентски и сървърни операционни системи, IPv6 с IPsec може да бъде приложено от системните администратори по всяко време, без да променят приложенията, които използват. Благодарение на новото адресиране колизиите при частните адреси се елиминират. Също така отпада и нуждата от използването на NAT (*Network Address Translation*). [13, 14].

Още едно подобрение при IPv6 е възможността за динамично конфигуриране. То може да бъде или *stateful* или *stateless*. При *stateful* динамичното конфигуриране се ползва *Dynamic Host Configuration Protocol* (DHCPv6), при който динамично, от определено множество адреси, се раздават IP адресите, които ще бъдат използвани за конфигуриране на новозакачил се възел към мрежата. При *stateless* автоматичното конфигуриране се ползва автоматично конфигуриране (*auto-configuration*).

До тук казаните неща ще бъдат по-подробно разгледани нататък в тази глава. Друго, което също е важно да се отбележи е, че има вече истински *Quality of Service* (QoS). Това свойство може да бъде използвано чрез полето *Flow Label* на IPv6 *header*. QoS осигурява предимство, на пакети, които трябва да достигнат целта си при определени времеви изисквания, като например предаването на видео или звук в реално време, без да има накъсвания на потока. [19] От друга страна, това позволява и на доставчиците на мрежови услуги, да продават на някои свои клиенти трафик, гарантирайки, че той ще бъде предават с предимство пред останалите. Идея, която предизвиква противоречиви реакции, заради максимата, че всеки трябва да има равноправен достъп до глобалната мрежа. [154]

От всичко изброено до тук е видно, че новата версия Ipv6, позволява едно по-опростено и по-ефективно маршрутизиране. [4]

2.1.2 Адресиране

Както вече се каза и в първа глава, с увеличаването на адресното пространство могат да се адресират до 2^128 или 3.4x10^38 различни IP адреси, което предоставя 655570793348866943898599 (6.5x10^23) адреса за всеки квадратен метър от повърхността на Земята. [15, 21]

IPv6 дефинира 128-битови адреси, представяни като осем 16-битови числа, разделени с двоеточия. 16-битовите числа са записани като 4 цифри в шестнайсетичен формат. Форматът е: *hhhh: hhhh: hhhh: hhhh: hhhh: hhhh: hhhh: hhhh*. Където *hh* представлява един байт. За да се онагледи казаното, ще се представи следния пример.

Пример за IPv6 адрес е 3FFE:2900:D005:0000:02AA:00FF: FE28:9C5A

Десетично	**Шестнайсетично**	**Двоично**
0	0	0000
1	1	0001
2	2	0010
3	3	0011
...	...	...
15	F	1111

Таблица 1. Съответствие между десетично, шестнайсетично и двоично представяне на числата

Като се вземе предвид Таблица 1, показваща преобразуването на числата от десетична, двоична и шестнайсетинча бройна система, примерният IPv6 адрес може лесно да бъде представен в двоична форма, като ще придобие следния вид:

0011111111111110001010010000000011010000000001010000000000000000
0000001010101010000000001111111111111110001010001001110001011010

Ако този 128-битов адрес се раздели на по 16-битови блокчета, ще се получи следното:

00111111 11111110 00101001 00000000 11010000 00000101 00000000 00000000 00000010 10101010 00000000 11111111 11111110 00101000 10011100 01011010

При това представяне ясно се вижда, че всяка част, разделена с двоеточие, представлява два байта. От тук всеки 16-битов блок, ако се преобразува в шестнайсетичен формат и се раздели с двоеточие, ще се получи горния адрес.

Представянето на IPv6 адреса може да бъде опростено, чрез премахването на всички водещи нули от всеки 16-битов блок. Задължително е обаче всеки блок да има поне една цифра. Така адресът може да се запише във вида:

3FFE:2900:D005:0:2AA:FF:FE28:9C5A

В този пример може да се демонстрира още едно опростяване, което е позволено. В адрес, където има един или повече последователни блокове, състоящи се само от нули, може да се запишат като „ :: “ две последователни двоеточия. Но това се допуска само за една единствена последователност от нулеви блокове в адреса, а ако има друга, нулите се изписват. [44]

По този начин адресът придобива окончателно вида:

3FFE:2900:D005::2AA:FF:FE28:9C5A

Протоколът IPv6 може да бъде използван като адрес в URL полето на браузърите, точно както и IPv4. Единствената разлика и особеност, за която трябва да се внимава, е наличието на двоеточия, което се възприема от браузарите като разделител за оказване на порт. Затова адресът се поставя в квадратни скоби.

Пример за един адрес е този на СУ „Св. Климент Охридски“.

HTTP://[2001:67C:20D0::22]/

Разбира се и при IPv6 могат да се оказват портове към адреса. Формата е точно същият както при IPv4 и се добавя след затварящата квадратна скоба. Например, ако се иска, порт 80, да бъде указан към горния адрес, единственото, което трябва да се направи е да се допише „ :80 “, с което примера добива следния вид:

HTTP://[2001:67C:20D0::22]:80/

Типове IPv6 адреси

Важно е да се обърне внимание и на различните видове IPv6 адреси. При тази версия на IP, те са три вида: *Unicast*, *Multicast* и *Anycast*.

При **Unicast** адресите е характерно, че те идентифицират един конкретен интерфейс. По този начин, пакети адресирани към *Unicast* адрес се доставят до точно определен интерфейс на даден маршрутизатор в топологията. С други думи, *Unicast* адресите се ползват при комуникация между един източник и само един получател. Подробно ще бъдат разгледани различните типове *Unicast* адреси по-нататък в изложението.

При **Multicast** адресите, IPv6 префикса е FF00::/8. Този тип адреси идентифицират множество интерфейси. По този начин пакети, адресирани до *Multicast* адрес, се доставят до тези интерфейси, които се идентифицират от този адрес. Тоест *Multicast* адресите се ползват при комуникация с един източник и множество получатели (множество интерфейси).

Има три вида *Multicast* адресиране:

– *Multicast* адрес за всички възли в мрежата (**All Nodes Multicast Address**) – той идентифицира група от всички IPv6 възли, като обхвата за *interface-local* е FF01:0:0:0:0:0:0:1, а за *link-local* обхвата е FF02:0:0:0:0:0:0:1.
– *Multicast* адрес за всички маршрутизатори в мрежата (**All Routers Multicast Address**) – той идентифицира група от всички маршрутизатори, като обхвата за *interface-local* е FF01:0:0:0:0:0:0:2, а за *link-local* обхвата е FF02:0:0:0:0:0:0:2.
– *Multicast* адрес (ограничен) запазен специално за *Neighbour Discovery Protocol* в линк. (**Solicited Node Multicast Address**) – той задължително се изчислява за всеки възел в мрежата като функция от съответния *Unicast* или *Anycast* адреси. Този адрес има вида: FF02:0:0:0:0:1:FFXX:XXXX. Формира се, като се вземат най-малко значещите 24 бита от IPv6 адреса (*Unicast* или *Anycast*) и се добавят към префикса FF02:0:0:0:0:1:FF00::/104, което дефакто е

интервала от FF02:0:0:0:0:1:FF00:0000 до FF02:0:0:0:0:1:FFFF:FFFF. Това изчисляване на *Solicited Node Multicast* се извършва за всички конфигурирани *Unicast* или *Anycast* адреси на интерфейс, ръчно или автоматично. Пример за такъв адрес: ако IPv6 адресът е 4037::01:800:200E:8C6C, то съответният му *Solicited Node Multicast* адрес е FF02::1:FF0E:8C6C.

При **Anycast** адресите особеността е, че те се конфигурират на повече от един интерфейс, които обикновено принадлежат на повече от едно устройство в мрежата. По този начин, пакет изпратен към *Anycast* адрес се маршрутизира до най-близкият интерфейс с този адрес, според метриката, определяща дистанция за конкретния използван протокол. По този начин, от множество получатели се определя точно един. Адресното пространство, което използват *Anycast* адресите, е същото като на *Unicast* (използват кой да е от неговите адресни формати). Единственото, което е необходимо, за да се направи един *Unicast* адрес *Anycast*, е да се конфигурира на повече от един интерфейс.

Има един задължителен *Anycast* адрес, наречен *Subnet-Router Anycast* адрес. Префикса, който идентифицира връзка в мрежата, може да бъде наречен подмрежов префикс (*subnet prefix*) на *Anycast* адрес. Този *Anycast* адрес е същият като *Unicast* адресът за интерфейс от връзката (линка), като за частта от адреса, определяща мрежата, се запазва (n бита), а частта определяща интерфейса (*interface identifier*), тоест (128 – n) бита са само нули. Така пакет изпратен към *Subnet-Router Anycast* адрес се доставя до един маршрутизатор в подмрежата. Този тип адреси може да бъде използван за определяне множество то маршрутизатори, принадлежащи към организация, доставяща мрежови услуги. Друга употреба на *Anycast* адрес е да са определят маршрутизатори в частна подмрежа или домейн.

Концепцията за *Unicast* и *Multicast* съществува и при протокола IPv4, въпреки че там те са реализирани различно. Но *Anycast* е уникален за IPv6. Той работи като комбинация от другите два. От една страна *Unicast* адресирането се ползва за изпращане на информация до един конкретен получател, от друга страна

Multicast се ползва за изпращане на данни към група получатели. При *Anycast* обаче данните се получават от точно определен получател, който е част от група от кандидат получатели.

Важно е да се обърне внимание, че IPv6 адрес винаги идентифицира интерфейс, не възел (устройство) в мрежата. Устройствата се определят от *Unicast* адресите, зададени на интерфейсите на конкретното устройство. Следва да бъдат разгледани различните видове *Unicast* IPv6 адреси.

– Неопределен *Unicast* адрес (**Unspecified Address**) – Този тип има префикс ::/128, тоест адрес от вида 0:0:0:0:0:0:0:0. Този адрес никога не се задава на възел в мрежата и никога не се използва като адрес на получател. Той единствено указва липса на адрес. Примерна употреба на този адрес е задаването му като адрес на изпращача в IPv6 пакета, изпратен от даден хост преди той да е научил собствения си адрес. Тези адреси никога не се маршрутизират от маршрутизаторите в мрежата. Подобен е на адрес 0.0.0.0 в IPv4 протокола.
– Локалния адрес на хоста (**Loopback Address**) – Този вид адрес има префикс ::1/128, това е адреса 0:0:0:0:0:0:0:1. Този адрес позволява на устройството да изпраща пакети до себе си. Еквивалентен е на 127.0.0.1 адреса при IPv4 протокола. Пакетите, адресирани до този адрес, никога не се изпращат към прилежащия линк и не се препращат от IPv6 маршрутозаторите в мрежата.
– Локални адреси за връзката-линка (**Link-local Address**) – Тези адреси имат префикс FE80::/10. Въпреки че пространството FE80::/10 е запазено за Link-local адресите, 54 бита трябва да бъдат нули. Следователно префикса става FE80::/64. Това е показано на Фигура 2.1.

```
|   10     |
|  bits    |         54 bits         |          64 bits           |
+----------+-------------------------+----------------------------+
|1111111010|           0             |       interface ID         |
+----------+-------------------------+----------------------------+
```

Фиг. 2.1 *Link-Local IPv6 Unicast* адреси [78]

Тези адреси се използват между съседите в един линк и при процеса за откриване на съседи (*Neighbor Discovery*) и маршрутизатори, който от своя страна определя как възлите в IPv6 подмрежата си взаимодействат с другите хостове и маршрутизатори. Когато съседи от един линк (подмрежа) си комуникират те използват този тип адреси. Обхвата на *Link-local* адресите (тоест където те са уникални и не се дублират) е локалната връзка (подмрежа). Затова, когато една мрежа няма маршрутизатор, тези адреси се ползват за комуникацията между хостовете. Важно е да се отбележи, че за конфигурирането на *Link-local* адресите не се ползва DHCP, те се задават автоматично, дори и когато никакъв друг *Unicast* адрес не е наличен за интерфейса. [86, 87]

– Следващият вид е **Site-local Address**. Има префикс FEC0::/10. Не много широко използван и не получил единодушен отговор какво представлява точно „*site*“. Премахнат е от RFC3879 през 2004 година. Тези адреси са ползвани за комуникация между възли в един и същ сайт на вътрешната мрежа на организация. Не се маршрутизира през BGP протокола.
– Уникалните локални адреси (**Unique Local Address**) имат префикс FC00::/7. Те са създадени, за да заменят *Site-local* адресите. Проблемът налагащ това е, че въпреки *Site-local* адресите да са частни, те може да се дублират, ако в една организация има повече от един сайт. Това налага да се ползват зони с ID-та, за да се определи сайта, към който принадлежи получателя. Всичко това излишно усложнява администрирането на мрежата, което е довело до въвеждането на нов тип частни адреси, уникални за всички сайтове в организацията. Тези адреси се дефинират в RFC4193 през 2005 година [85]. Тези адреси са еквивалентни на частните адреси при IPv4 протокола. Те не могат да бъдат маршрутизирани в глобалната мрежа.
– Глобалните адреси (**Global Unicast Address**) – На този етап IANA е ограничила раздаването на глобалните адреси до префикса 2000::/3 (**001**0 0000 0000 0000) до 3FFF::/3 (**001**1

1111 1111 1111). Но дефакто адресното пространство на *Global Unicast*, дефинирано в RFC4291, притежава цялото IPv6 адресно пространство с изключение на FF00::/8. При сравнение с IPv4, *Global Unicast* адресите са еквивалентни на публичните адреси. Те са глобално маршрутизируеми и се достъпват от тези участници в глобалната мрежа, които поддържат IPv6. Формата на *Global Unicast* адреса е показан на Фигура 2.1.

```
|          n bits          |   m bits  |        128-n-m bits          |
+--------------------------+-----------+------------------------------+
| global routing prefix    | subnet ID |        interface ID          |
+--------------------------+-----------+------------------------------+
```

Фиг. 2.2 *Global Unicast* адреси [78]

Префикса *Global Routing Prefix* представлява мрежовата част на адреса, която доставчикът е дал на клиента. Към момента всички RIR раздават адреси с префикс /48, т.е. n = 48. Следващите m бита (16 бита) са определени за *Subnet ID*, което се ползва от мрежовите администратори за идентификация на определена подмрежа в границите на организацията. Всички *Global Unicast* адреси, които не започват с **000**, имат 64-бита резервирани за интерфейс ID, тоест n + m = 64. Еквивалент на *Interface ID* в IPv4 е хостовата част от адреса. Ползва се, защото един хост може да има повече от един интерфейс и всеки от тях да е конфигуриран с един или повече IPv6 адреси. Тези, които започват с **000**, нямат такова ограничение за полето, съдържащо интерфейс ID. Примерни за *Global Unicast* адреси, които започват с 000 са IPv6 адреси с вградени IPv4 адреси. Има дефинирани два типа адреси, които пренасят IPv4 адреси в най-младшите 32-бита на адреса. Единият е „*IPv4-Compatible IPv6 address*“, другият е „*IPv4-mapped IPv6 address*“.

– **IPv4-mapped IPv6 address** – Има префикс ::FFFF:0:0/96. Този адресен тип се използва за представяне адресите на IPv4 мрежови възли като IPv6 адреси. Формата на адреса е показан на Фигура 2.3.

```
|                80 bits                | 16 |      32 bits        |
+---------------------------------------+----+---------------------+
|0000..............................0000|FFFF|    IPv4 address     |
+---------------------------------------+----+---------------------+
```

Фиг. 2.3 Формат на “*IPv4-mapped IPv6*“ адреси [78]

Така адресът 0:0:0:0:0:FFFF:*w.x.y.z* или ::FFFF:*w.x.y.z се ползва за представяне на възел поддържащ единствено протокол IPv4 като* IPv6 възел. Тези адреси обаче се използват само за вътрешно адресиране и никога не се записват като адрес на източник или получател в IPv6 пакета.

– **IPv4-compatible IPv6 address** – Имат префикс 0000::/96. На този момент този тип вече е премахнат от RFC4291, защото сегашният транспортен механизъм на IPv6, не поддържа тези адреси.

Други съвместими адреси са:

– **6to4 address** – Има префикс 2002::/16. Адресите 6to4 се използват за комуникация между възли поддържащи както IPv4 така и IPv6. Този тип адреси се формират от комбинацията на глобалния префикс 2002::/16 с 32-битовия публичен IPv4 адрес, получавайки се 48-битов префикс. Тази транспортна технология е описана в RFC3056.
– **Teredo address** – Тези адреси имат префикс 2001:0000::/32. Използва се при комуникация между възли, поддържащи IPv4 и IPv6, но единият или двата са зад IPv4 NAT (*Network Address Translation*) устройство. Тези адреси се формират от *Teredo* префикс и публичния IPv4 адрес от *Teredo* сървър в комбинация от други елементи. Тази транспортна технология е описана в RFC4380.
– **ISATAP address** – Адресите ISATAP (*Intra-Site Automatic Tunnel Addressing Protocol*) се използват при комуникация между възли, поддържащи IPv4 и IPv6, в частните мрежи. Те използват локално конфигуриран ID на интерфейс ::0:5EFE:*w.x.y.z, където w.x.y.z е публичен или частен Unicast* IPv4 адрес. Този ISATAP интефейс ID може да бъде комбиниран с всеки 64-битов префикс, който е валиден IPv6 *Unicast* адрес (*Link-local* адресни префикси (FE80::/64),

Site-local префикси и *Global* префикси). Тази транспортна технология е описна в RFC4214.

По-детайлна информация за внедряването на IPv6 и за различните типовете IPv6 адресиране, форматът им и техният обсег на видимост, може да се получи от следните източници [21, 44, 45, 75, 76, 77, 78, 81, 152, 159].

Сега ще бъдат разгледани възможностите за конфигуриране на IPv6 адреси. Единият вариант е статичен – може да се зададе статично конфигуриран *Global Unicast* адрес на хост, чрез команди въведени в командния ред или през графично приложение в зависимост от използваните операционни системи и възможностите, които предлагат.

Вторият начин е автоматично/динамичното конфигуриране на IPv6 адреси.

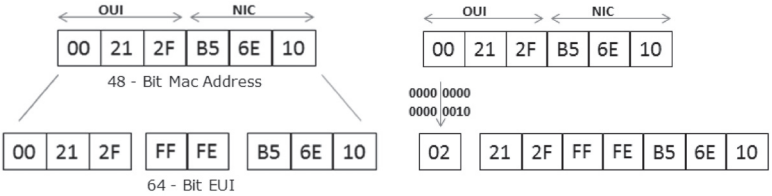
– Автоматично конфигуриране (*Stateless Address Autoconfiguration – SLAAC*) – При този метод не е необходим DHCPv6 сървър. Ползва се ICMPv6 съобщения (тип *Router Advertisement*). Така дадено устройство получава префикс, дължина на префикса и шлюз (*gateway*) по подразбиране от IPv6 маршрутизатор.
– Динамично конфигуриране (*Stateful*) с протокола DHCPv6. Хостът може да пренебрегне *Router Advertisement* съобщенията и да получи всичката необходима информация от DHCPv6 сървъра или да вземе частична конфигурация от ICMPv6 RA и от DHCPv6. Раздаваната адресна конфигурация включва *Global Unicast* адрес, дължина на префикса, шлюз (*gateway*) по подразбиране както и адрес на DNS сървър.

Който и от двата варианта да се ползва, актуалната част за *Interface ID* на *Global Unicast* адреса не се изпраща. Това налага хоста сам да определи своят 64-битов *Interface ID*. Това става чрез:

– *Manual interface ID assignment* – генериране на случайно 64 битово число, според статичен адресен план. Другият вариант е използване на процеса EUI-64 за определяне на

Interface ID. Като цяло е по-добре обвързването на MAC адреса с IP адреса, защото може да се отрази на много места, ако се промени в последствие и по този начин е по-скалируема конфигурацията.

– *EUI-64 interface ID assignment – Extended Unique Identifier* (EUI) позволява на хост сам да си създаде уникален 64-битов IPv6 Interface ID. Тази възможност е ключово предимство пред IPv4, тъй като вече няма нужда човек да се грижи за ръчно конфигуриране или ползване на DHCP. Тук IPv6 EUI-64 се получава от 48-битовия физически MAC адрес, който се разделя на две равни части. Първите 24-бита са OUI (*Organizationally Unique Identifier*) на производителя, вторите са специфични за мрежовата карта (NIC). За да се получи 64-битовия EUI, трябва да се добавят между двете половини 16 бита, които са точно определени: 0xFFFE. Тези 16 бита са специално резервирани от IEEE точно за генерирането на EUI-64 от EUI-48 MAC адрес. Това е показано на Фигура 2.4.



Фиг. 2.4 Образуване на EUI-64 от EUI-48 MAC адрес [36]

Последната стъпка в този процес е свързана с най-левият 7-ми бит, така нареченият *universal/local* (U/L) бит. Ако е 0, това означава, че е локално администриран, а ако е 1, адресът е глобално уникален. Обаче при глобалните уникални адреси, присвоени от IEEE, за OUI частта този бит винаги е установен в 0. А при локално създадените адреси, които са конфигурирани ръчно от администратора, този бит е 1. Затова се налага този бит да се инвертира, когато се създава EUI-64 адрес като *interface ID* в IPv6, както е показано на Фигура 2.4 в дясно. Това инвертиране на

„u“ бита се прави, за да е по-лесно за системните администратори ръчно да конфигурират локални идентификатори, когато няма налични хардуерни възможности. По този начин глобалните си остават глобални, а локалните локални. На този момент този бит не се ползва за нищо конкретно, но в бъдеще при по-нататъшно развитие на технологиите, може да влезне в реална употреба, позволявайки на този интерфейсен идентификатор да определя видимостта глобална или локална.

След като процесът по установяване на 64 битовият интерфейсен идентификатор приключи, полученият *Interface ID* се комбинира със съответния IPv6 префикс, за да се създаде *Global Unicast* адрес или *Link-local* адрес. За глобалния се ползва префикса получен от *Router Advertisement* при автоматичното конфигуриране на адреси (S*tateless Address Autoconfiguration – SLAAC*). За Link-local се ползва префикса FE80::/64. [36, 84, 88]

Важно е да се обърне внимание на протокола **Neighbour Discovery**. Този протокол замества ARP (*Address Resolution Protocol*) протокола, ползван при IPv4 за намиране на физическия адрес на устройство по неговия мрежов адрес (*Address Resolution*). При новата версия на IP за тази задача се грижи ICMPv6 ND (N*eighbor Discovery)*, което е разширение на ICMPv6 протокола. По този начин ICMPv6 включва четири нови съобщения като част от NDP (*Neighbor Discovery Protocol*).

- *Router Solicitation message* – това съобщение се праща от хост до всички маршртизатори, за да получи динамично конфигурация чрез SLAAC.
- *Router Advertisement message* – съдържа адресната конфигурация, поискана с RS. Праща се от маршрутизатор до всички хостове.
- *Neighbor Solicitation message* и *Neighbor Advertisement message* – ползват се за намирането на физическия адрес на дадено устройство. За определяне на адреса на получателя се праща NS, а отговора се получава в NA. Може да се ползва и за откриване на дублиращи се адреси. Когато устройството получи *Global Unicast* или *Link-local* адрес, то праща NS съобщение до всички. Ако същият адрес вече

се ползва, съответното друго устройство ще отговори с NA съобщение, че адресът е зает.

В обобщение може да се каже, че всеки възел в мрежата (хост или маршрутизатор) използва *Neighbor Discovery* за определяне на адреса от каналния слой (*Data Link-layer*) на съседите си. Протоколът ND управлява обмена на съобщения между възлите в мрежата, пренасящи данни, необходими за автоматичната конфигурация на хостове и за преноса на другите пакети в локалната връзка.

Още един положителен момент при IPv6 е, че при изпълнение на командата *ping*, не се губи първият пакет, защото тук не се ползва ARP и таблицата е предварително попълнена от *Advertisements*. При *point-to-point* връзките физическия адрес е еднозначно определен, поради това че участниците във връзката са само два и пакетите имат единствен възможен получател. [79] [82, 83, 88]

2.1.3 Формат на заглавната част на протокола

В следващите редове ще се изследват особеностите на новият формат на заглавната част (*Header*, хедър) на IPv6 протокола.

При IPv6, IP *Header* е опростен и е дефиниран с точно определена дължина от 40 байта. При проектирането на IPv6, някои полета са премахнати, преименувани или преместени при допълнителните нови полета IPv6 *Extension Headers*. Разликите са показани във Фигура 2.5 и Фигура 2.6. Както се вижда от фигурите, първата разлика е, че полето IHL (*Internet Header Length*), вече не се ползва, понеже IPv6 *Header* е с фиксирана дължина (40 байта), тоест полето *Options* от IPv4, което е премахнато тук, вече не варира от 0 до 40 октета. Следващото поле *Type of Service* е преименувано на *Traffic Class.*

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Фиг. 2.5. Формат на заглавната част на IPv4 [22]

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Фиг. 2.6 Формат на заглавната част на IPv6 [23]

Функциите на полето *Total Length* се изпълняват от *Payload Length*. Тъй като в IPv6 фрагментирането на пакетите се извършва само от възела източник, но не и от междинните маршрутизатори по пътя, полетата *Identification*, *Flags*, и *Fragment Offset*, които отговарят за контрола на фрагментирането, са преместени в подобни полета в допълнителния хедър *Fragment Extension Header* [23]. Функционалността, която се изпълняваше от *Time to Live*, сега се извършва от полето *Hop Limit*. Функциите на полето *Protocol* се изпълняват от *Next Header Type*. Полето *Header Checksum* е премахнато, като така не се губи време да се изчислява на всяко препредаване на пакетите. Опасността от незасечени грешки, поради липсата на пресмятане на контролната

сума при IP протокола е минимална, защото такава се пресмята при повечето енкапсулации. За IPv6 това се нарича *Upper-Layer Checksums*, като примерно за протоколите като TCP, UDP има така нареченият „*pseudo-header*“, който имитира IP *Header*, за да пресметне контролната сума за 128-битовите IPv6 адреси [23, 24]. Полето *Options* също е премахнато, тази функционалност е преминала в допълнителните хедъри IPv6 Extension Headers. Това е направено, за да може само информацията, която е необходима за маршрутизирането да се обработва, което повишава ефективността [24, 19].

Трябва да се обърне внимание и на това, че при процеса на енкапсулация за слой n-1 данните и контролната информация на слой n – PDU(n) (протоколен блок от данни – *Protocol Data Unit*), са обикновени данни (*payload*). И докато при IPv4 *Header*, той е следван от данните (*payload*) от транспортния слой TPDU (*Transport Protocol Data Unit*), които обикновено включват протоколите TCP, UDP или ICMP, то след IPv6 *Header* следват допълнителни хедъри (*Extensions Headers*). Едва след тях следват протоколите от транспортния слой (техните *header* и *data*). Наименованието им и стойността, която отговаря за всеки от тях, и която се записва в полето *Next Header*, са дадени на Фигура 2.7.

Next Header Value	Next Header
0	Hop-by-Hop Options header
4	Internet Protocol
6	Transmission Control Protocol
17	User Datagram Protocol
43	Routing header
44	Fragment header
45	Inter Domain Routing Protocol
46	Resource Reservation Protocol
50	Encapsulating Security Payload
51	Authentication header
58	Internet Control Message Protocol
59	No next header
60	Destination Options header

Фиг.2.7 Възможните стойности в полето *Next Header* [24]

Шестте *Extensions Headers*, дефинирани за IPv6 и предназначението им са следните:

- *Routing Header* – много подобен на опциите *Loose Source* и *Record Route* при IPv4. Използва се, за да определи маршрутизаторите, през които ще се мине, за да се достигане крайната цел, като за целта се изреждат един или повече междинни възли, които да бъдат посетени.
- *Authentication Header* (AH) – Този *header* отговаря за сигурността и се грижи за удостоверяването (*authentication*) и целостта (*integrity*) на данните.
- *Encapsulating Security Payload* (ESP) – И този *header* е свързан със сигурността, като той се грижи за удостоверяването и криптирането (*encryption*) на данните.
- *Fragmentation Header* – Изпълнява подобна функционалност като опцията свързани с фрагментирането на пакетите при IPv4.
- *Destination Options Header* – Този *header* съдържа множество от опции, които да бъдат обработвани само от крайния възел – получател (*destination*). Използва се примерно при *Mobile* IPv6. *Mobile* IPv6 е IETF е стандарт, който дава възможност за роуминг на мобилните устройства в мрежа с IPv6. При него опцията „*Home Address*“, се пренася от *Destination Options Header*. „*Home Address*“ се ползва, когато се изпрати пакет от мобилното устройство, докато то не е в дома, за да информира получателя за своя домашен адрес. Така при промяна на местоположението си запазва адреса от предишната връзка [25, 26].
- *Hop-by-Hop Options Header* – Множество от опции, които трябва да бъдат обработени от всеки възел по пътя, който пакетът минава до крайната си точка. Такива може да бъдат функции за дебъгване [19, 23, 24].

При изследването на възможностите на IPsec, ще бъде обърнато внимание на два от представените хедъри. Това са *Authentication Header* и *Encapsulating Security Payload* (ESP).

2.2 Приложение на IPsec в процеса на удостоверяване между съседни маршрутизатори

Използването на IPsec е от изключителна важност за защита на процеса на маршрутизация. Причината е, че маршрутизиращите протоколи са създадени да извършват добре работата си, за която са били проектирани, но не и да бъдат защитени от злонамерени атаки. Грешките и дупките в сигурността, обикновено се дължат на пропуски в конфигурацията. Затова настоящата работа ще се придържа към принципа „*Don't Secure Routing Protocols, Secure Data Delivery*“ (защита на преноса на данните, а не на самите маршрутизиращи протоколи) [2], като ще се покаже как с помощта на IPv6 и вграденият в него IPsec ще се защити процесът на маршрутизация от неправомерно прочитане на конфигурационна информация или от пренасочване по грешни маршрути (*hijacking*) [3].

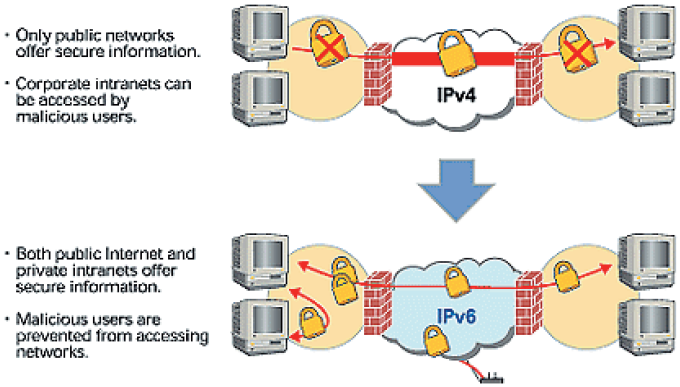
2.2.1 Характеристика и особености на IPsec

От всичко, казано до тук, може да се заключи, че едно от най-важните подобрения в IPv6 е това, че *IP Security* (IPsec) е вече част от самият протокол.

По дефиниция, сигурност в компютърните мрежи (*network security*), е аспект от мрежовата администрация, който е фокусиран върху подсигуряването на това, че данните, връзките и устройствата в една мрежа се използват от упълномощени лица по удостоверен начин. Най-общо, сигурността трябва да гарантира: [153]

- Наличност и достъпност на ресурсите (*Availability*) – мрежовите компоненти, информацията и услугите са налични, когато са необходими.
- Конфиденциалност (*Confidentiality*) – услугите и информацията са видими само от лица, които са били оторизирани за тях.
- Цялост на данните (*Integrity*) – информацията не е била променена, подменена, разрушена или открадната преди да достигне получателя, за който е била предвидена.

За да отговори на тези изисквания, IPsec използва *Authentication Header* (AH) и *Encapsulating Security Payload* (ESP). Както е известно, IPsec се е ползвал и е бил пригоден да работи и с IPv4, като опция, позволявайки реализирането на *Virtual Private Networks* (VPN). При IPv6, чрез IPsec *virtual tunnel interface* (VTI), вече се предоставя *end-to-end* или *site-to-site* защитеност на комуникацията, което означава, че предаваните данни са защитени от началния възел (източника) до получателя, независимо от къде минават пакетите. При използването на IPsec се криптира всеки пакет, така че да може да бъде приложен върху целия IP трафик, за разлика от широко използвания SSL, който работи само на горния слой – на TCP [8]. Използването на тази, вградена в IPv6, IPsec енкапсулация защитава всички видове трафик – *unicast* и *multicast*. С други думи, IPsec VTI позволяват на IPv6 маршрутизаторите да работят като такива отговарящи за сигурността (*security gateways*), установяващи IPsec тунели с други *security gateway* маршруризатори и криптиращи трафикът на вътрешните мрежи, когато минава през публичния IPv6 интернет. [17]



Фиг. 2.8 Съпоставка между IPv4 и IPv6 *site-to-site* защитеността [8]

При IPv4 такава защитеност обикновено се прилага само при граничните маршрутизатори на отделните мрежи [19]. Това може да се види на Фигура 2.8. Това, което предоставя IPsec, е защита на мрежово ниво в TCP/IP протоколния стек, което означава, че всички

приложения, които използват IPv6, също ще бъдат защитени с него благодарение на енкапсулацията на данните от по-горните нива в IPv6 пакетите. Точно поради тази причина, че е на по-ниско ниво, той осъществява защитата на всички протоколи от по-високи нива като TCP, HTTP и други. Тази гъвкавост и прозрачност на протокола IPsec прави възможно да се използва за защита в най-различни ситуации и да се конфигурира за всякакви нужди [8]. Докато при IPv4 се защитават публичните мрежи, то при IPv6, се защитава и трафика между участниците в частните, вътрешни мрежи.

За да се осигури защитеното предаване на данните, IPsec използва *Authentication Header* (AH) и *Encapsulating Security Payload* (ESP). Те могат да бъдат използвани заедно или поотделно, в зависимост от целите, които се иска да постигнат. И AH и ESP могат да бъдат в един от двата режима на работа.

- Тунелен режим (*tunnel mode*) – При този режим, IPsec се прилага върху целия IP пакет. IPv6 хедъра и AH или ESP се прилагат към оригиналния пакет и така се предава.
- Транспортен режим (*transport mode*) – При този режим, протоколът се прилага към протоколите от транспортния слой (примерно TCP, UDP, ICMP), като към IPv6 хедъра и AH или ESP *header*, следват данните и заглавната част от транспортния протокол. [63]

И AH и ESP използват SA и *Internet Key Exchange* (IKE) при установяването и управлението на *Security Associations*. IKE определя процесът на договаряне на параметрите необходими за установяване на ново асоцииране (SA), като предаване на секретни ключове, използването на определени криптографски алгоритми и др. По същество IKE представлява протоколен стандарт за обмен на ключове, който се използва заедно с IPsec. Нищо, че IPsec може да бъде конфигуриран и без IKE – ръчно, с използването на IKE има някои допълнителни предимства, като повече гъвкавост и по-лесно конфигуриране. Той е хибриден протокол, който реализира *Oakley key exchange* и *Skeme key exchange* във архитектурата *Internet Security Association Key Management Protocol* (ISAKMP). Тоест, ISAKMP, *Oakley* и *Skeme* са протоколи по сигурността, реализирани от IKE. Тази функционалност е много подобна на модела със защитен шлюз (*security gateway*), където се ползва

IPv4 IPsec защитеност [8]. Ако и двете защити AH и ESP се прилагат над трафика на данните, тогава се създават две или повече SA, за да се представи защитеност на трафика. За защитата на обикновена, двустранна комуникация между два хоста или между два маршрутизатора (gateways) трябват две SA, по едно за всяка посока. *Security Association* се определя уникално по три компонента: *Security Parameter Index* (SPI, поле в AH/ESP хедъра), IP адреса на целта (*destination address*) и защитен протокол AH или ESP. [27]

Концепцията *Security Association* (SA) е фундаментална за IPsec, като се използва при всички IPv6 реализации, както и при тези на IPv4, които реализират AH, ESP или и двете. За да може да се използва IPsec, системата генерира два вида бази от данни. Едната е *Security association database* (SAD), съдържаща по едно SA – *security association* за всеки интерфейс и една глобална база за виртуалните връзки (*virtual link*). И в случаите, когато IPsec е конфигуриран за област (*Area*), пак за всеки интерфейс в тази област, използващ конфигурацията на IPsec за областта, ще притежава свой SA запис в SAD. По-конкретно, всеки SA в SAD е запис базиран на избрания тип защита (AH и ESP), адреса до целта и *Security policy index* (SPI). Този индекс – SPI е число, създадено от потребителя (администратора на системата), отговарящо на изисквания в мрежовият план, който се следва в организацията. Избраните стойностите на SPI трябва да се приложат върху цялата мрежа. Политиките по сигурността (*Security Policies*) са правила, които са заложени в реализирането на IPsec и които казват как да се обработват различните видове пакети получени от маршрутизатора.

Другата база *Security Policy Database* (SPD) се използва за съхранение на политиките за сигурност за съответното устройство. И тук те се генерират системно (автоматично). Тези политики помагат на маршрутизаторите да определят кои пакети трябва да приемат и кои не. Примерно за всеки пакет, който пристигне и има ESP хедър се търси в базата с политиките за сигурност SPD, за да се провери дали съответния пакет отговаря на политиките. Ако не се намери политика, по която да бъде приет, той не се приема и се отхвърля. [65, 66].

2.2.2 Authentication Header

Първият хедър *Authentication Header* отговаря за сигурността и се грижи за автентичността (*authentication*) и целостта (*integrity*) на целия IPv6 пакет. AH осигурява и така наречената защита *Anti-replay*.

Автентичността или удостоверяването на данните е свързано, примерно с това, че получателят, като получи IP пакет от даден адрес, записан в IP хедъра, да може да е сигурен, че този пакет наистина идва от този адрес. Целостта на данните, е свързана с това, че получателят, щом получи IP пакет, той може да бъде сигурен, че съдържанието на пакета не е било променяно по пътя от изпращача до получателя. Защитата *Anti-replay*, се отнася до това, че няма да се приеме пакет, който не е част от поредицата пакети с нарастващи номера на пакетите. [27]

На фигура 2.9 е показан формата на *Authentication Header*.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Payload Len  |          RESERVED             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Security Parameters Index (SPI)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Sequence Number Field                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                Authentication Data (variable)                 |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Фиг. 2.9 Формат на *Authentication Header* [28]

В хедъра на IPv6 се съдържа поле *Next Header*, което показва следващият *Extension Header*, например AH, или вид транспортен протокол. При AH, *Next Header* е 8-битово поле, което идентифицира типа на данните, които следват след AH. Стойността е номер на протокол, определен от IANA. Например номер 6 е заделен за TCP. Полето *Payload Length* съдържа дължината на AH. Полето *Security Parameters Index* (SPI) съдържа индекса за идентифициране на *Security Association*. Полето *Sequence Number* е брояч, гарантиращ последователността на пакетите, които пристигат. Стойността му е 0, когато се установява първоначално комуникацията между изпращача и получателя. След което започва да се увеличава с единица, когато или изпращачът, или получателят предават данни

един към друг. Ако получателя засече повтарящ се пореден номер, пакетът се отхвърля, благодарение на *Anti-replay* защитата. Това предотвратява атака, при която данните се променят и се препращат модифицирани. Това често е ползвано при *Session Replay* атаките във версия 4 на IP протокола. При тези атаки, атакуващият прихваща предаването между оторизиран потребител и сървъра, след което модифицира данните и препраща променените пакети все едно те идват от потребителя в истинската потребителска сесия [19]. Полето *Authentication Data* е с променлива дължина и съдържа *Integrity Check Value* (ICV), което се грижи за автентичността и целостта на данните. Алгоритъмът за удостоверяване на автентичността, избран когато *Security Association* е установено между изпращача и получателя, определя дължината на ICV, правилата за сравнение и необходимите стъпки в този процес. Тази стойност е изчислена върху пакетът от изпращача и се верифицира от получателя, който сравнява стойността на полето с резултата получен от собственото му изчисление. ICV се изчислява върху полетата от IP хедъра, които остават непроменени по време на предаването. Това са AH със стойност на *Authentication data* равна на 0 и потребителските данни на IP пакета. Полета от заглавната част на IPv6, които могат да се променят като *Hop Limit*, *Traffic Class* и *Flow Label*, не участват в изчислението на ICV. Получателят на IP пакет с AH хедър, преизчислява стойността на ICV с алгоритъма по удостоверяване на автентичността и ключа, договорени в SA. Ако ICV е същото, тоест съвпадат двете стойности, получателят може да е сигурен, че данните са автентични и не са били модифицирани. Това поле трябва изрично да включва *padding* (допълнителни битове), така че да се гарантира, че дължината на AH хедъра е цяло кратно на 32 бита (IPv4) или 64 бита (IPv6) [19, 24, 28].

Работеща реализация на AH трябва да поддържа един от двата задължителни за прилагане алгоритми: „*HMAC with MD5*“ и „*HMAC with SHA-1*“. HMAC е съкращение от *Hash Message Authentication Code*, което е техника, която извършва удостоверяването на съобщенията, използвайки хеш функция (тоест еднопосочна функция, която съпоставя и изчислява уникален код на базата на входните данни). Преди да се приложи хеш функцията, към съобщението е добавен и секретния ключ на изпращача. MD5

е съкращение от *Message Digest* 5, а SHA-1 идва от *Secure Hash Algorithm* 1. MD5 е хеш функция и извежда 128-битов изход, а SHA-1 е хеш функция, която извежда 160-битов изход. При тези еднопосочни функции алгоритъмът получава на входа си низ или текст в явен вид и произвежда на изхода друг низ или криптиран текст, от който не може да се получат първоначалните входни данни, тоест тези алгоритми представляват необратими функции. Важното при тях е, че при еднакви входни данни, се получават еднакви изходни данни, с фиксиран размер и много по-малки от оригиналните. При изчислението на хеша от двете страни (изпращача и получателя) се проверява и гарантира целостта на данните, които са били предавани (*data integrity*).

Употребата на *Authentication Header* може да предотврати атаки като *IP Spoofing* (ползваща измамни IP адреси). При *IP Spoofing*, атакуващия генерира IP пакети с различни адреси (и не съвпадащи с този на неговия компютър). Това може да се ползва за различни злонамерени действия. Примерно при *rlogin* (програма за *Unix*, позволяваща отдалечен достъп до друг хост), която се използва главно в частни мрежи, където информацията за потребителите и паролите им бива достъпна за всички *Unix* машини. Този модел разчита на сигурността на другите машини и самата инфраструктура на мрежата. При тази ситуация, при която на един компютър се позволява достъп до друг компютър, без да се изисква някаква допълнителна защита, съществува опасност за сигурността. Ако IP адресът на един от тези компютри в тази мрежа е *spoofed* (с цел измама е използван от друг), атакуващият може да осъществи достъп до, който и да е друг компютър в мрежата. IP *Spoofing* се прилага и при други видове атаки като *TCP Session Hijacking*. При тази атака, по време на съществуваща и установена сесия, атакуващият, чрез подмяна на IP адреса на своите пакети може да претендира, че е част от тази сесия. Друга атака, където се ползва IP *Spoofing* е при така наречената *Man-in-the-Middle* атака. Където атакуващият прекъсва по средата комуникацията между две устройства и чрез подмяна на адресите им в пакетите може да ги обработва и препраща все едно са дошли от истинските участници в комуникацията.

2.2.3 Encapsulating Security Payload

Вторият хедър *Encapsulating Security Payload* осигурява сигурността, като се грижи за автентичността (*authentication*) и криптирането, както и за целостта (*integrity*) на енкапсулираните данни. И тук също както при AH има *Anti-replay* защита. Трябва да се отбележи, че при ESP, алгоритъмът за автентичността, се прилага само към данните, които се криптират. Това означава, че полетата от IP хедъра не са защитени, освен ако ESP не е в тунелен режим. За да бъдат защитени и полетата от IP хедъра, може да се ползва в комбинация с *Authentication Header*. Криптирането на данните гарантира конфиденциалността им. Така получателят на IP пакета е сигурен, че никой друг не е чел съдържанието на IP пакета, освен информацията необходима на маршрутизаторите.

При ESP опцията за удостоверяване на автентичност и опцията за гарантиране на конфиденциалност не са задължителни едновременно, но поне една от тях трябва да бъде избрана. На Фигура 2.10 е даден формата на ESP.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ----
|               Security Parameters Index (SPI)                 | ^Auth.
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |Cov-
|                      Sequence Number                          | |erage
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | ----
|                    Payload Data* (variable)                   | |   ^
~                                                               ~ |   |
|                                                               | |Conf.
+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |Cov-
|               |     Padding (0-255 bytes)                     | |erage
+-+-+-+-+-+-+-+-+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |   |
|                               |  Pad Length   | Next Header   | v   v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|                 Authentication Data (variable)                |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Фиг. 2.10 Формат на *Encapsulating Security Payload* [29]

И при *Encapsulating Security Payload*, както при *Authentication Header*, има поле *Security Parameters Index*, което се използва за определяне на *Security Association*. Аналогично е и полето *Sequence Number*, използвано за *Anti-replay* защитата. Полето *Payload Data* може да бъде с променлива дължина и съдържа данните описани от полето *Next Header*. Данните, които са в полето *Payload Data* са криптираните данни. Типът им се намира в полето *Next Header*,

което определя дали данните ще са примерно целия IP пакет, ако е използван тунелен режим или транспортни данни (от протоколи като TCP, UDP, ICMP), ако е в транспортен режим. Други две полета, които са показани на Фигура 2.10, са *Padding* и *Pad Length*. Полето *Padding* съдържа допълнителни битове, които могат да са необходими за криптиращия алгоритъм, а *Pad Length* съхранява броя на тези допълнителни байтове. Полето *Authentication Data* (което е с променлива дължина) съдържа *Integrity Check Value* (ICV), който се грижи за автентичността и целостта на данните. Тази стойност е изчислена върху пакета от изпращача и се сравнява от получателя, със стойността която той е получил при собственото му изчисление. ICV се изчислява върху целия ESP пакета минус полето *Authentication Data*, с което се осигурява удостоверяването, както е показано и на Фигура 2.10. Самите данни, които са в *Payload Data* се намират в частта, която се шифрова с криптиращ алгоритъм и ключове, договорени от SA, осигурявайки конфиденциалност. В транспортен режим само транспортни данни се криптират, а в тунелен режим се криптира целият IP пакет. Това прави предаваната информация конфиденциална.

ESP е проектиран да използва симетрични алгоритми за криптиране, където един ключ се използва както за криптиране, така и за декриптиране на данните. Според спецификацията на ESP (RFC2406), пълноценната ESP реализация трябва да поддържа следните задължителни за реализиране алгоритми: *Data Encryption Standard* (DES) в *Cypher Block Chaining* (CBC) режим, HMAC с MD5, HMAC с SHA-1, *NULL Authentication algorithm*, *NULL Encryption algorithm*. Понеже алгоритмите за криптиране и автентичност при ESP са по избор, дава се възможност да не се избере нищо (NULL) за алгоритъм, само че трябва да се има предвид, че може само единият да бъде договорен като NULL (*Authentication* или *Encryption* алгоритъм), но другият трябва да бъде наличен [29]. Въпреки че DES е определен по спецификация, ESP не е зависим от точно определен алгоритъм, и затова DES е заменен с много по-сигурните алгоритми за криптиране *Triple-DES* или *Advance Encryption Standard* (AES).

Главното преимущество да се ползва ESP за криптиране с цел конфиденциалност на данните, е да се намали риска от

злонамерените последици от така наречената атака *sniffing* (буквално „подушване“ или следене на трафика). Докато чувствителна информация, като пароли примерно, се предава в явен вид, то всеки ползващ инструменти като *TCPdump* и *windump*, може да я достъпи и прочете. С криптирането на IP пакетите това би затруднило многократно този вид изтичане на конфиденциална информация.

2.3 Предимствата на IPsec пред други методи за удостоверяване в процеса на маршрутизация

От както съществуват протоколи за маршрутизация, въпросът за сигурността е бил важен, защото самите протоколи не са проектирани да гарантират защитеност. Тяхната основна задача е правилно и оптимално да изпълняват дейностите по маршрутизиране и те се справят с тази си задача. Затова се разработват, пробват и предлагат различни методи за защитаване на процеса на работа на маршрутизиращите протоколи. За някои протоколи като OSPFv2 има добавени възможности за удостоверяване на автентичността чрез предаване на пароли в явен текст или с MD5 HMAC и пресмятане на контролни суми. Обаче това не решава проблема с атаки като *sniffing* на пакети. Тъй като в IPv6 вече се използва удостоверяващ хедър AH като част от неговия протоколен стек, това налага конфигурирането на големи ключове. Тоест функционалната разлика с предишния начин на работа е, че сега не просто се извършва удостоверяване на пакетите, но и се добавя криптиране на данните с използването на ESP. Така че тази основна разлика оказва влияние и върху предишните версии на протоколите EGRP, BGP, OSPF, използващи удостоверяване за защита на процеса на маршрутизация. При тях това, че се удостоверява комуникацията, не означава, че някой по средата на пътя не може да я прихване и прочете. Тоест, ако някой прави *sniffing* на трети слой, то OSPF комуникацията между два маршрутизатора ще бъде все още видима. Това включва свободен достъп примерно до обмена на служебна информация като LSA (*link-state advertisement*) и целия трафик обменян между двете устройства. В OSPFv3 (OSPF за IPv6) с възможността да се

криптира трафика, вече не съществува ефективен начин злонамерен атакуващ да получи достъп по поверителни данни и ще се наложи да приложи някакъв вид *brutoforce* атака към ключа на ESP [7]. Освен това прилагането на IPsec опростява използването на OSPFv3, защото го освобождава от реализираният в самият него механизъм за защита [32].

Друг метод за удостоверяване в процеса на маршрутизация е двойното удостоверяване (*Double Authentication*) [34]. Това представлява схема на разпространение на симетрични ключове с цел намаляване на атаките, свързани с обновленията при протоколите със следене състоянието на връзката (Link *State Routing*), представител на които е OSPF. При *Double Authentication* всеки маршрутизатор използва два симетрични ключа. Първият код за автентичност може да бъде верифициран от всеки друг маршрутизатор, без този от който идват обновленията на таблицата. Вторият може да бъде използван от неговите съседи за проверка на целостта (*integrity*) на данните и на първия код. Това ще позволи да се провери дали първият код и служебните данни по обновленията са променени, преди да достигнат при следващите съседи [34]. Но този метод не решава проблемите със съюзяването на съседи (примерно ако са част от бот мрежа, контролирана от хакер). Такива съюзили се съседни маршрутизатори могат да фалшифицират ключовете.

И за *Border Gateway Protocol* (BGP), съществуват различни разработени методи за защита на процеса на маршрутизация. Като се вземе предвид, че BGP е протокол за външна маршрутизация (*Exterior Gateway Protocol* – EGP) и е предназначен за работа в различни автономни системи, които са под различен административен контрол, опасността и мащабите на атаките, при неговото компрометиране, стават значителни. Такива примери вече бяха показани в първа глава.

Един от методите за защита на процеса на предаване на пакети по протокола BGP е S-BGP. Тази архитектура, ползва три основни механизми за сигурност. Първият е *Public Key Infrastructure* (PKI), тоест инфраструктура, ползваща публични ключове за проверка на автентичността на електронно съобщение с помощта на публичен ключ, който се регистрира от регистриращ орган (*Registration Authority*), при което титуляра получава сертификат за собственост

от сертифициращ орган (*Certificate Authority*). Тук, при тази архитектура, чрез верифициращ орган (*Vertification Authority*), се удостоверява собствеността на блока от IP адреси, които принадлежат на автономната система (*Autonomous System* – AS), тяхното идентифициране и идентифицирането на маршрутизатори (с BGP), които имат право да представляват тези AS. Вторият много важен механизъм е т.нар. Атестиране (*Attestation*), който е свързан с пренасянето на маршрутизиращата информация в BGP UPDATE, защитена с цифров подпис. Тези подписи заедно със сертификатите от S-BGP PKI позволяват на получателя на BGP UPDATE да потвърди, че префиксите на адресите и информацията за пътищата е автентична. Третият механизъм ползва IPsec, за да осигури целостта на данните и автентичността на BGP участниците, предаващи си контролен трафик. Тъй като IPsec е реализиран на мрежово ниво, той защитава целостта и на транспортния слой (TCP връзките ползвани между BGP участниците). Ако контролният трафик трябва да бъде конфиденциален, IPsec предоставя решение, без да се налагат каквито и да било други промени по BGP [41, 42].

При BGP използването на криптографските механизми с публични ключове е твърде скъпо и значително намалява производителността на маршрутизаторите. Съществуват подобрения, които могат да решат този проблем. D. Nicol, S. Smith и M. Zhao виждат, че при големи натоварвания времето за конвергенция (уеднаквяване на маршрутизиращата информация) при обикновения S-BGP е много по-голямо от това на BGP. Те наблюдават влиянието от много силно кеширане и предизчислителни оптимизации за S-BGP и успяват да намалят времето за конвергенция да доближи това на BGP. Обаче тези оптимизации са нереалистични и прекалено консумиращи памет. Затова те, използвайки структурата на BGP процеса, проектират оптимизации, които намаляват забавянето, свързано с криптографските задачи, чрез намаляване на разходите по подписването на частните ключове, прилагано върху много съобщения. Те наричат този метод *Signature-Amortization* (S-A) и стигат до извода, че той предоставя резултати за конвергирането, близки или по-добри в сравнение с много оптимизираният S-BGP, но без действията, консумиращи огромно количество памет [37, 38].

Друг метод за защита на предаваните съобщения на BGP е базиран на симетрични криптиращи системи. Y. Hu, A. Perrig

и M. Sirbu предлагат схема, наречена *Secure Path Vector* (SPV), за защита на обновяванията на BGP съобщенията. Тази схема използва криптографски хеш функции като хеш вериги („*hash chains*“), хеш дървета („*hash trees*“), както и еднократно подписване („*one time signatures*“) с основно предимство, че се генерира много бързо, като една двойка ключове се използва за удостоверяване на много съобщения. Това решение е по-бързо от SBGP, затова и като допълнение може да предостави по-добра защита и от *replay* атаките [35].

Всички тези методи са предвидени да защитават само BGP протокола. В „*Routing Protocol Security Using Symmetric Key Based Techniques*“ [30] авторите предлагат механизъм на защита на BGP и OSPFv2, който ползва симетрични ключове, които са нарекли *Distributed−KD* (*distributed key distribution protocol*) и *Central−KD* (*centralized key distribution protocol*). С тях правят експерименти, сравнявайки ги с протоколите S-BGP и SPV и по емпиричен път доказват преимуществата на своите в бързодействието при генерирането и верифицирането на подписи. Но ако се вземе предвид, че IPsec е част от протокола IPv6 и че предоставя решение на проблема по сигурността, тоест автентичността, конфиденциалността и целостта на предаваните пакети между маршрутизаторите, то това решава основният проблем, благодарение на това, че новите версии на OSPF версия 3 и BGP са преработени да работят с IPv6 [39, 40, 24]. Така това се явява едно универсално решение и за двата протокола, които ще бъдат използвани за реализацията на задачата в настоящата работа.

В тази глава беше представена новата версия на IP протокола, както и новите типове IPv6 адреси, чрез които се предава служебен трафик, който може да бъде обект на атаки, ако не се защити. Бяха изследвани свойствата на IPSec, приложими в процеса на аутентикация между съседни маршрутизатори, в контекста на протокола IPv6. Също така бяха анализирани предимствата му пред други методи за удостоверяване в процеса на маршрутизация. В следващата глава ще бъде разгледан протокола за външна маршрутизация BGP 4/4+ и механизмите му за зашита от намеса на „трети страни“ в комуникацията.

3. Трета глава – Анализ на протокола за външна маршрутизация BGP в частта установяване на съседство между маршрутизатори

3.1 Изследване на същностна на протокола BGP, необходимо при процеса на конфигурация

Преди да се премине към анализ на BGP, ще се въведе едно основно понятие за протокола. Това са така наречените Автономии системи.

3.1.1 Автономни системи

Под автономна система (АС) се разбира такава система, която се намира под единен административен контрол. Такива системи могат да бъдат корпоративни мрежи или интернет доставчици. Всяка АС ползва BGP или за получаване на информация за други АС, или за разпространение на информация за собствената си мрежа, която трябва да бъде достъпна отвън [114]. Протоколът BGP използва *Prefixes* и *Autonomous System Paths* (*AS Paths*), за да определи най-краткият път до получателя, където се намира конкретния префикс.

Номерът на автономната система – *Autonomous System Number* (ASN) се задава, така че да бъде глобално уникален за публичните автономни системи. Използва се за идентифициране на самата АС, така че BGP да знае коя е точно локацията на получателя. Дължината на ASN е 16-бита, но има и нови 32-битови номера, които са били създадени от IANA (*Internet Assigned Numbers Authority*), когато запаса от свободни 16-битови е започнал да се изчерпва. Съществуват два вида, публични и частни. Публичните ASN се разпределят от IANA между петте RIR (ARIN, LACNIC, RIPE NCC, AFRINIC и APNIC. За Европа това е *Reseaux IP Europeens* – RIPE), които ги раздават по определени критерии на крайни клиенти, подобно на IP адресите [113, 121, 123]. Частните

АС заемат определен интервал от ASN пространството. Има и една част от специални номера които е резервирана за документални и други нужди и не се раздава.

– Публичните ASN са глобално уникални и могат да бъдат обявявани в глобалната мрежа през BGP. Използват се за уникално идентифициране на мрежи или системи от мрежи, които да са видими за външния свят. Интервалът на публичните 16-битови ASN е от 1 – 23455 и от 23457 – 64511, а за 32-битовите е 131072 – 4199999999.
– Частните ASN не се предават към глобалната мрежа и не трябва да се обявяват от BGP протокола. Обикновено се ползват от доставчиците на интернет в частни мрежи, но това означава, че те трябва да се грижат да не ги предадат при BGP обновленията в глобалната мрежа. За да се предпазят от подобни грешни обновления много доставчици на интернет филтрират пътищата, съдържащи частни номера. Интервалът на частните 16-битови ASN е 64512 – 65534, а на 32-битовите е 4200000000 – 4294967294.

3.1.2 Протоколът BGP

Протоколът BGP (*Border Gateway Protocol*) е маршрутизиращ протокол, спадащ към групата на протоколите, предназначени за външна маршрутизация, т.е. *Exterior Gateway Protocol* (EGP). Той извършва *interdomain* маршрутизация. Противоположна на *intradomain* маршрутизацията, при която областта на действие на протокола се ограничава в един домейн или автономна система. Проектиран е да осигурява маршрутизиране без опасност от зацикляне.

Най-новата версия на BGP е версия 4, описана в RFC4271 от 2006 година. По-ранните версии се приемат от общността на мрежовите оператори като остарели и почти не се поддържат. Настоящият документ RFC4271 е минал през повече от двайсет работни варианта и се базира на предходния RFC1771, при който се въвежда BGP4. Важно допълнение към BGP4 е MBGP (*Multiprotocol Extensions for BGP*), разширение което позволява

поддържането на протокола IPv6 от BGP, описан в RFC4760. Среща се и с означението BGP4/4+. Това допълнение към протокола позволява успоредно да се ползват различни адресни фамилии. При BGP без *Multiprotocol Extensions* се поддържат само IPv4 *Unicast* адреси. При MBGP (BGP4/4+) се поддържат IPv4 и IPv6 *Unicast* или *Multicast* адреси за двата. [155]

Протоколът BGP, макар че изпълнява функциите на 3 слой на OSI модела, работи като приложен процес (7 слой), базиран на TCP съединение на порт 179. От гледна точка на своята реализация BGP е по-прост от OSPF, защото не се грижи за нещата, за които отговаря TCP. От друга страна, конфигурирането на BGP е по-сложно от това на протоколите за вътрешна маршрутизация. Неправилното конфигуриране може да причини зацикляния и различни проблеми в засегнатите мрежи [122]. Съществуват много примери за грешки в конфигурациите, причинили проблеми в глобалната мрежа, някои бяха посочени още в първа глава.

Маршрутизиращият алгоритъм, който ползва BGP е така нареченият „*path-vector routing algorithm*“, който е комбинация от *distance-vector* алгоритъма за маршрутизиране и *AS-path* атрибута (списък от автономни системи, през които трябва да се мине от източника до дестинацията), с чиято помощ могат да се откриват зацикляния в мрежата. По правилата на този алгоритъм се обменя информацията между BGP-„говорещите“ устройства (маршрутизаторите). При обмен на обновления се изпращат данни за мрежата, атрибути специфични за пътя, списък с номерата на автономните системи, през които трябва да се мине за достигане до мрежата на получателя. В тези обновяващи съобщения (които ще бъдат разгледани по-подробно малко по-нататък) се съдържа и *AS-path* атрибута.

Въпреки че протоколът BGP е за външна маршрутизация, той има две разновидности:

- *Internal BGP* (iBGP) – за работа вътре в една автономна система (AS)
- *External BGP* (eBGP) – за работа извън една AS и свързващ автономните системи една с друга. Това е и основната характеристика за BGP, затова е познат и като „*interdomain routing protocol*“ както беше вече казано в началото.

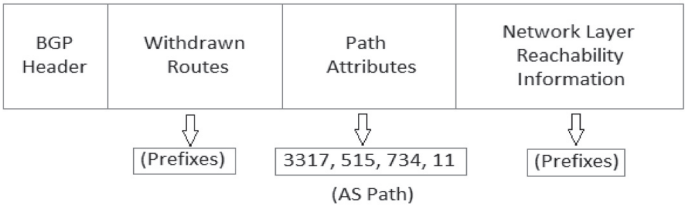
Тези имена се отнасят за един и същ протокол, само областта на работа е различна. Главната задача на BGP е да свързва локални мрежи с външни мрежи, да осигурява достъп до глобалната мрежа или да прави възможна връзката с други организации. Когато се свързват външни организации с локалната, се ползва eBGP. Обаче съществуват много случаи, при които многото мрежи в рамките на една организация или автономна система стават толкова сложни, че се налага използването на протокол като BGP, за да е възможно ефективното администриране. Тогава се казва, че се ползва iBGP [100, 102, 114, 122].

3.1.3 Начин на работа на BGP

При BGP няма процес, който да открива пътища, нито BGP изпраща собствената си маршрутизираща таблица постоянно, както е при други протоколи. След първоначално включване маршрутизаторът предава своята таблица и от там нататък всичко зависи само от получаването и пращането на обновления при промяна на маршрути. По-конкретно, когато се започва комуникация чрез BGP по установена TCP връзка, се обменят първо **OPEN** съобщение и се договарят параметрите на връзката. При BGP с *Multiprotocol Extensions* се ползва т.нар. *Capability Advertisement* процедура, за да се определят способностите на другия участник в комуникацията относно поддържането на *Multiprotocol Extensions* (основно използването на IPv6). Според стандартизиращия документ на BGP-4, описан в RFC4271, ако се получи OPEN съобщение с поне един неразпознат *Optional Parameter*, връзката трябва да се прекъсне. При MBGP (по-конкретно RFC5492) се дефинира *Optional Parameter* и нови правила, които позволяват на BGP да предава *Capability* в OPEN съобщенията. Така всички, които поддържат този стандартизиращ документ, позволяват да се установи връзка, дори и ако има неразпознат *Capability* параметър [106, 130].

След като връзката вече е установена, цялата маршрутизираща таблица, на всеки един от участниците в установената комуникация, се предава към другите участници. По-късно, когато

настъпят промени по топологията, този процес не се повтаря, а се препращат само съобщения с обновления. Тези съобщения се наричат **UPDATE** и съдържат информация за добавени / обновени (*advertises*) или премахнати (*withdraws*) пътища. На Фигура 3.1 е показана опростена схема на формата на UPDATE съобщението.



Фиг. 3.1 Схематично представяне на BGP UPDATE съобщението

Полето *Withdrawn Routes* съдържа списък с адресни префикси за премахнати пътища, които трябва да бъдат изтрити от BGP таблиците. Всеки маршрут, който трябва да бъде премахнат от установена BGP сесия, се изпраща до съседите по този начин.

Полето *Network Layer Reachability Information* също съдържа списък от адресни префикси. Това са префиксите на новите пътища, които са били обявени като достъпни. Тази информация е била взета от локалната Adj-RIB-Out база на изпращача и се добавя към базата Adj-RIB-In на съседите [124].

Полето *Path Attributes* съдържа атрибутите (представени чрез кодове), които по същество представляват метриката, използвана в процеса по избор на път (*decision process*). Тези атрибути са във всяко UPDATE съобщение, с изключение на тези, носещи само *withdrawn* пътища. Има 19 вида BGP маршрутини атрибута определени от IANA, като ще се обърне внимание на най-основните от тях:

- ORIGIN – показва произхода, т.е. от къде е било получено UPDATE съобщението. Използва се за определяне на предпочитания маршрут.
- AS_PATH – съдържа списък с номера на автономни системи, през които минава маршрута. Често има повече от възможен маршрут. В тези случаи BGP ползва AS_PATH, за да избере един най-добър. Най-лесно е да избере най-

краткия, тоест този с най-малко транзитни автономни системи. Например, ако има подобен случай AS1-AS2-AS3 и AS4-AS5-AS6-AS7, ще се избере AS1-AS2-AS3.

– NEXT_HOP – Това е IP адресът на следващият маршрутизатор, който е част от пътя до получателя.
– MULTI-EXIT-DISC – съкратено от *Multiple Exit Discriminator*, това е метрика, използвана за определяне на път при наличието на множество изходни точки. Това прави възможно да се каже на другата автономна система, че от няколко изходни точни, една е предпочитана.
– LOCAL-PREF – указва предпочитание за един конкретен път пред другите в автономна система.
– ATOMIC-AGGREGATE – указва, че локалният *decision* процес предпочита по-малко специфичен път до целта, пред някой, който е по-специфичен (такъв който се предполага, че би трябвало да бъде избран).
– AGGREGATOR – указва, че няколко пътища са били обобщени / събрани в един по-общ.
– WEIGHT – принуждава *decision* процеса да избере път, чрез указване на тегло.

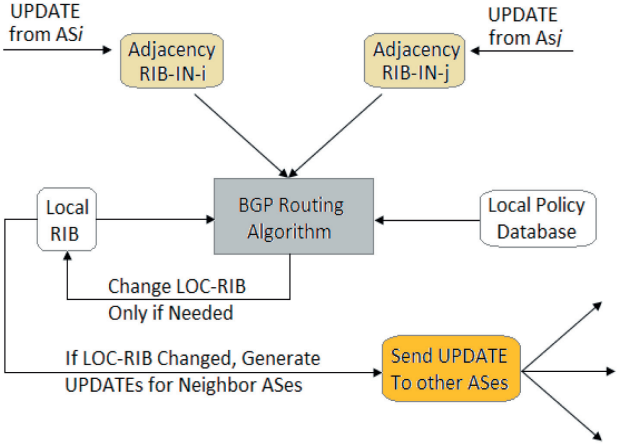
Важен аспект на AS_PATH е неговата роля при справянето с зацикляния при протокола BGP. Отхвърля се всеки път, който съдържа номера на локалната автономна система, защото това показва, че тя вече е посетена и би се получило зацикляне [122].

При *BGP-4 Multiprotocol Extensions*, който поддържа IPv6, има няколко допълнителни неща към UPDATE съобщението. Специфичните атрибути за IPv4 са само три: NEXT_HOP (съдържа IPv4 адрес), AGGREGATOR (съдържа IPv4 адрес) и NLRI (съдържа IPv4 адресен префикс). Следователно, за да се позволи използването на BGP4/4+, трябват само две неща. Едното е възможност да се асоциира NEXT_HOP с конкретния мрежов протокол и второто е също NLRI да се асоциира с конкретен мрежов протокол. Затова UPDATE съобщението е идентично със това на BGP без MBGP, като само се добавят два нови типа атрибути.

– MP_REACH_NLRI – съкращение от *Multiprotocol Reachable NLRI*. Този атрибут има поле *Address Family Identifier* (AFI). С негова помощ се идентифицира, адресът, пренасян в NEXT HOP, към кой мрежов протокол принадлежи, както и семантиката на NLRI.
– MP_UNREACH_NLRI – съкращение от *Multiprotocol Unreachable*. Това е опционален атрибут, използван за премахнатите *withdraw* пътища.

Маршрутизаторите, ползващи BGP, но не поддържащи MGBP, ще игнорират информацията в тези атрибути и няма да я предадат към други. Ако от два маршрутизатора, само един поддържа MBGP, комуникацията ще бъде възможна както преди по *Unicast* IPv4 [106, 107, 127, 128].

Фигура 3.2 представя схематично процеса, по който BGP обработва UPDATE съобщенията.



Фиг 3.2 Обработване на UPDATE съобщение [97]

Обновленията се записват в така наречената *Routing Information Base* (RIB) база. Тя е различна от маршрутизиращата таблица, в която се записва само един маршрут към даден получател. В RIB обикновено има множество от всички възможни маршрути,

на базата на които маршрутизаторът избира един, който ще бъде използван и записан в маршрутизиращата таблица. В случай, че даден път трябва да бъде премахнат от маршрутизиращата таблица, може веднага да бъде заменен с друг от RIB базата. Ако е единствен и в RIB, тогава той се изтрива и от там и не се изпращат обновяващи съобщения към другите маршрутизатори. Това е единствения случай, в който се изтриват записи от RIB.

Така постъпилите промени, пристигащи от автономните системи, първо се записват в съответни Adj-RIB-In, след което BGP алгоритъма (*Decision* порцесът) установява, дали има нужда да се обнови локалната база Loc-RIB, на базата на съществуващите записи и на зададените политики. Ако е настъпила промяна в Loc-RIB, маршрутизаторът създава UPDATE съобщение, което да бъде препратено към съседните автономни системи [119, 125, 126].

Когато BGP поддържа множество протоколи, в това число IPv6, се ползва още една допълнителна RIB база. В този случай е налице разделението на *Unicast Routing Information Base* и на *Multicast Routing Information Base*, понеже RIB трябва да може да поддържа всяка маршрутизираща информация, която получава. Новата *Multicast Routing Information Base* е разклонение на старата, поддържаща само IPv4 Unicast префикси. При MGBP, *Unicast RIB* съдържа *Unicast* префиксите, а *Multicast RIB* се използва само за *Multicast* обработки [127, 128, 129].

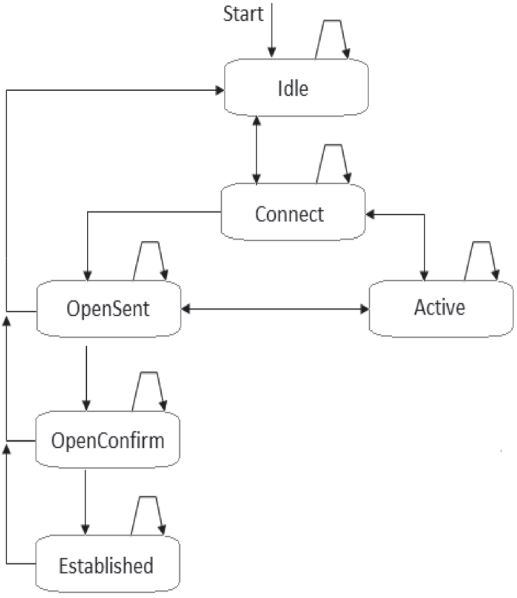
Следващият вид съобщения са **NOTIFICATION**. Те служат за информиране при възникване на грешки. След изпращането на съобщението връзката се прекратява. Последният вид съобщения са **KEEPALIVE**, изпращат се периодично, за да поверява състоянието на връзката [97, 104, 108, 117, 119, 125].

Протоколът BGP ползва краен автомат (**Finite-State Machine – FSM**) при взимането на решения, необходими при взаимодействието с другите BGP участници. Крайните автомати са математически модели на прости сметачни машини. Автоматът може да се намира в едно от множеството състояния, които притежава.

Има няколко сесийни атрибута, които са задължителни. Тези атрибути са обяснени в следващите редове:

- *State* – Състояние на автомата. Сесийният атрибут „състояние“ служи за показване на текущото състояние на BGP автомата. За BGP крайните състояния са: *Idle*, *Connect*, *Active*, *OpenSent*, *OpenConfirm* и *Established*.
- *ConnectRetryCounter* – Показва броя на пътите, които BGP се опитва да установи сесия.
- *ConnectRetryTimer* – Съответства на *ConnectRetryTime*, отчита кога ще изтече времето преди да се опита да установи сесия.
- *ConnectRetryTime* – Този параметър определя времетраенето на BGP *ConnectRetry* таймера в секунди.
- *HoldTimer* – Съответства на *HoldTime*, отчита кога ще изтече времето.
- *HoldTime* – Това време се предава заедно с OPEN съобщението и информира съседа, за периода в който се очаква да се получи отговор (UPDATE, KEEPALIVE или NOTIFICATION съобщение). Ако такъв отговор не бъде изпратен за това време, се приема, че той не е достъпен на този момент. По подразбиране е 180 секунди.
- *KeepaliveTimer* – Съответства на *KeepaliveTime*, отчита кога ще изтече времето.
- *KeepaliveTime* – По подразбиране 60 секунди. Това означава, че на всеки 60 секунди се праща KEEPALIVE съобщение, за да се провери дали TCP връзката е все още активна. Ако за времето, за което *HoldTimer* изтече, не се получи такова съобщение се приема TCP сесията е прекъсната и няма връзка към съседния маршрутизатор.

На Фигура 3.3 е показано действието на крайния автомат (FSM).



Фиг. 3.3 Блок схема на *Finite State Machine* – BGP

За да се установи BGP сесия, автоматът трябва да мине през следните състояния:

– *Idle* – Това състояние настъпва при т.нар. „стартиращо събитие“. Случва се при създаването на нова конфигурация на BGP процеса или при рестартиране на процеса. Протоколът BGP винаги стартира в това състояние, което инициализира BGP процеса с всички необходими ресурси, пуска таймера *ConnectRetry* и инициализира TCP връзка към съседния маршрутизатор. След всичко това се преминава в следващото състояние *Connect*, в което започва да чака за TCP връзка от съседа към него. Ако настъпи грешка се преминава отново в състояние *Idle* и маршрутизаторът издава друго „стартиращо събитие“.
– *Connect* – В това състояние BGP процеса чака за завършването на познатия *three-way handshake* за установяване на TCP връзка (едната страна праща SYN, другата отговаря

със SYN-ACK и накрая първата потвърждава с ACK). При това има три възможни последващи варианта:

1) ако всичко мине благополучно – Ако успешно е установена TCP връзка, BGP процеса премахва таймера *ConnectRetry* и изпраща OPEN съобщение към съседа. Освен това се преминава към следващото състояние *OpenSent*. В този случай *HoldTimer* е 4 минути.
2) ако е възникнал проблем – При неуспешна TCP връзка, BGP процеса рестартира *ConnectRetry* таймера и преминава към състояние *Active*.
3) връщане отново към *Idle* – При всеки друг случай се преминава отново към *Idle* състояние.

Ако таймерът *ConnectRetry* изтече, докато е още в състояние *Connect*, чакайки за TCP отговор, той се рестартира и се прави нов опит за инициране на TCP връзка. В това състояние остава, докато не изтече времето на таймера.

– *Active* – Тъй като това състояние се достига при неуспешна TCP връзка при *Connect*, то вече е разгледано. Важно е да се отбележи, че в състояние *Active,* ако се получи „стартиращо събитие“, то се пренебрегва.
– *OpenSent* – Това състояние показва, че OPEN съобщението е било изпратено и се очаква съседния маршрутизатор да отговори също с OPEN съобщение. От това състояние има три възможни разклонения:
  1) преминаване към *OpenConfirm* – Ако е било получено OPEN съобщение и не са открити грешки в него. Тук се договарят и *HoldTime* и се настройват *Keepalive* таймерите. Щом се премине към *OpenConfirm* се изпраща KEEPALIVE съобщение.
  2) преминаване към *Active* – Ако отсрещната страна прекъсне TCP връзката, преди да е изпратила OPEN съобщение, BGP връзката ще се затвори и *ConnectRetry* таймера ще се рестартира.
  3) преминаване към *Idle* – Ако е възникнала грешка при OPEN съобщението, ще се изпрати NOTIFICATION съобщение и ще се премине в начално състояние.

– *OpenConfirm* – В това състояние се чака или KEEPALIVE или NOTIFICATION съобщение. Ако се получи първото, то BGP преминава към *Established* състояние. Ако е второто, се преминава към начално Idle състояние. Освен това, ако изтече *HoldTimer* или се получи „събитие за спиране“, отново се преминава към Idle състояние и се изпраща NOTIFICATION съобщение към съседния маршрутизатор.
– *Established* – Това състояние показва, че BGP сесията е напълно установена. Двете страни могат да обменят KEEPALIVE, UPDATE и NOTIFICATION съобщения. Всеки път, когато се получи KEEPALIVE или NOTIFICATION, *HoldTimer* се рестартира, а при NOTIFICATION се преминава към *Idle*. При всеки друг случай („**стартиращо събитие**“ се игнорира) се праща NOTIFICATION към съседа и се отива към *Idle*. [95, 96, 126, 131, 132]

3.2 Предимствата на IPsec пред вграденият механизъм за удостоверяване с пароли при защитата на BGP от намеса на „трети страни“ в процеса на комуникация

В тази подточка от разглежданата главата, ще бъдат представени механизмите за сигурност, които могат да се приложат. Много от предимствата на IPsec вече бяха засегнати във втора глава, където се изтъкнаха предимствата на IPv6 и вграденият му механизъм за защита – IPsec. Тук ще се обърне внимание на проблемите, срещани при защитата на комуникацията при BGP с използването на пароли и защо използването на IPsec е предпочитания метод.

Важността от защита на BGP се видя и от примерите, които бяха дадени в първа глава. Мащабите на потенциалните проблеми са толкова значителни, защото протоколът „BGP е централната нервна система, към която практически всички доставчици на интернет са свързани“ [100]. Затова е обект и на атаки. Атакуващите знаят, че ако успеят да намерят слабост в конфигурирането на

BGP, могат потенциално да доведат до дестабилизиране на цялата глобална мрежа.

Вградения механизъм за защита е чрез размяна на пароли при удостоверяване. В RFC2385 „*Protection of BGP Sessions via the TCP MD5 Signature Option*“ се дефинира как парола може да бъде хеширана с MD5 алгоритъма и добавена към BGP пакетите. Механизмът за пресмятане на MD5 е криптографска функция, която заменя обикновеното изчисление на контролна сума при TCP пакетите в BGP. Аутентициращия механизъм TCP MD5 е бил разработен за защита на BGP сесиите от получаването на измамни (*spoofed*) TCP пакети (това е била главната мотивация), които биха могли да повлияят на BGP данните или да прекратят самата TCP връзка. Той осигурява начин за пренос на хеширания от MD5 резултат в TCP сегмент. Този хеш се изчислява на базата на информация, известна само на крайните точки в процеса на комуникация. Без да се знае паролата, е почти невъзможно да се конструира пакета от получателя [94]. Това гарантира удостоверяването и целостта на съобщенията. Тази опция се използва от маршрутизиращия протокол, за да се подсигури защита на ниво сесия срещу внедряването на фалшифицирани TCP сегменти в съществуващия TCP трафик (примерно *TCP Reset*). Но има много проблеми, които са свързани със самия хеширащ (MD5) алгоритъм, който може да бъде успешно атакуван. *Message Authentication Codes* (MAC), използвани от TCP MD5, са определени като твърде слаби. От една страна заради използваната хеш функция и от друга, поради начина на управление на ключовете (*key-management*). Използваният ключ при изчисление на хеша се пази локално. Липсата на автоматизирано управление на ключове означава, че ключове са пароли, които често никога не се променят. Но е препоръчително, като добра практика, да се задават различни пароли за двойките от различните сесии, което пък е трудно за поддържане. Тъй като протоколът BGP е базиран върху TCP и използва TCP MD5 опцията за защита на TCP сесиите, всички уязвимости на BGP могат да бъдат обвързани със сигурността на протоколът TCP в частност или с компрометирането на отделни маршрутизатори и данните които управляват [100, 101, 104, 110, 111, 115].

Тези заплахи могат да бъдат разделени като:

– заплахи свързани с удостоверяването – ако атакуващ се представя за легитимно устройство;
– заплахи свързани с конфиденциалността (*confidentiality*) – прихващане и прочитане на информацията при комуникация на две устройства;
– заплахи свързани с целостта на данните (*integrity*) – вмъкване на фалшифицирано BGP съобщение в потока разменян между BGP устройства или неоторизирано прочитане на информация (атака „*man in the middle*“);
– атаки срещу TCP сесията – това включва прекъсване на TCP сесията с фалшиво детерминиране от BGP (пращане на фалшиви пакети) или промяна на пътя, причинявайки зацикляния;
– DoS (*Denial of Service*) и DDoS (*Distributed Denial of Service* – тоест такива, изпълнявани разпределено от няколко страни) атаки – при тях примерно са възможни следните атаки: изпращане на голямо количество SYN, консумиращи паметта на устройство (SYN *flooding*); пращане на голямо количество пакети към порт 179, затормозявайки процесорът на устройството и др.

Някои от тях могат да бъдат решени с вградения механизъм за удостоверяване, но други – не. Тъй като вграденият механизъм позволява удостоверяване и проверка за целостта на данните, това позволява на получателя да знае от кого идва съобщението и дали е било променено. Тоест при атаки свързани с *hijacking* на сесия или *replay* атаки, както и при неоторизирано участие в BGP сесия, тази защита ще свърши работа [98]. Що се отнася до TCP, атаки насочени към този протокол също са възможни и засягат BGP процеса. Някои такива атаки включват наводняване с TCP RST и TCP SYN (т.нар. *flooding* атаки). Атаката с TCP RST позволява на атакуващия да причини прекъсване на връзката между маршрутизаторите. При TCP RST пакета трябва да има пореден номер в определен интервал, което прави тази атака по-вероятна от други свързани с TCP. При наводняване с TCP SYN атакуваният маршрутизатор се принуждава да изразходва наличните си ресурси (като памет и

процесорно време), което може да причини прекъсване на връзката или блокиране на работата на самото устройство. Атаки, свързани с TCP SYN, SYN ACK, ACK, RST/FIN/FIN-ACK, могат да бъдат отблъснати с помощта на MD5. Но атаки като DoS и DDoS, които просто препълват TCP порт 179, ще бъдат успешни. Причината е, че пакетите, получени на порт 179, се обработват от BGP процеса, който в много случаи не ползва достатъчно мощен процесор да се справи с тези атаки [103, 109].

Друг проблем при защитата с TCP MD5 е липсата на възможност за криптиране на данните. Така при атаки, свързани с конфиденциалността на данните, не може да се осигури никаква защита. За разлика от вградения начин, IPsec предлага криптиране на данните с помощта на ESP (както беше вече подробно разгледано във втора глава). Въпреки че не е инструмент, създаден специално за BGP, конфиденциалността и удостоверяването, които се осигуряват от IPsec тунелирането, са много надеждна защита от външни атаки, при установяването на съседство между маршрутозаторите. Удостоверяването при IPsec е много по-надеждно в сравнение с MD5 TCP, а криптирането на данните дава защита от spoofing атаки, дори и да са от изпълнени от локален сегмент [97, 99].

Още едно предимство на IPsec е по отношение на TCP. Тъй като IPsec е част от IPv6, който е протокол от мрежов слой, а TCP е протокол от транспортния слой, се осигурява автоматична защита на по-високо ниво от по-ниското, без да се налагат промени по BGP.

Четирите вида съобщения, които бяха разгледани, OPEN, KEEPALIVE, NOTIFICATION и UPDATE, могат да станат обект на уязвимост за BGP процеса. Те могат да попаднат под нерегламентирано наблюдение, могат да бъдат *spoofed* (подменени с фалшиви, при които не е ясен истинският изпращач), модифицирани, препращани отново (*replayed*) или изтривани. Тоест, някой отвънка може да праща лъжливи BGP съобщения, за да наруши нормалното функциониране на връзката. Процесът на маршрутизация може да се наруши с фалшиви UPDATE съобщения. При наличие на грешка в хедъра на BGP съобщението,

маршрутозаторът ще отхвърли това съобщение и ще прекъсне връзката с другото устройство. Ако нападател успее да премахне KEEPALIVE съобщенията от потока с данни, тогава отсрещната страна ще прекрати връзката, понеже ще мисли, че устройството вече не е достъпно. Затова е задължителна допълнителна защита, поне като вградената TCP MD5. Както вече беше казано, BGP има механизъм за прекратяване на връзката по всяко време, чрез NOTIFICATION съобщенията. Ако се подмени NOTIFICATION съобщението, това може да причини по естествен начин прекратяване на връзката от страна на получателя. Когато се препращат съобщения (*replayed*) с дадена информация на по-късен етап, това може да доведе до премахване (*withdrawn*) на валидни пътища, а невалидни да бъдат обявени (*announced*). Тоест BGP свързаността може да бъде нарушена (по неправомерен начин) с използването на UPDATE съобщения, съдържащи фалшива или неактуална маршрутизираща информация. По-конкретно, ако съдържа фалшифицирани атрибути ATOMIC_AGGREGATE, NEXT_HOP в AS_PATH и *Withdrawn Routes* или NLRI. За тези атаки, TCP MD5 не може да осигури защита [103, 109].

В RFC5925 се представя нова TCP опция за аутентициране – *TCP Authentication Option* (TCP-AO), която заменя опцията за подписване TCP MD5 *Signature option*, описана в RFC2385 (TCP MD5). Въпреки че TCP MD5 е много по-широко използван в сравнение с новия механизъм за защита TCP-AO, няма как да не се направи съпоставка на преимуществата на IPsec и пред новата по-добра TCP защита TCP-AO.

Новата опция TCP-AO специфицира употребата на по-силни кодове за удостоверяване *Message Authentication Codes* (MACs), които вече позволяват защитата на т.нар. *long-lived* TCP връзки (като при BGP) и осигуряват защита от *replay* атаки. В RFC5925 за TCP-AO и неговия съпътстващ документ RFC5926 „*Cryptographic Algorithms for TCP-AO*“, се описват необходимите стъпки за подобряване на слабостите при MAC и обмена на секретни ключове. Тези стъпки задължават поддържането на два MAC алгоритъма. Единият е HMAC-SHA-1-96 (описан в HMAC, RFC2104), а другият е AES-128-CMAC-96 (описан в NIST-SP800-

38B). Изследователите по криптография казват, че тези два MAC алгоритъма са достатъчно сигурни. Освен това при TCP-AO се позволява добавянето на допълнителни MAC в бъдеще. Заради поддържането на няколко MAC алгоритъма се казва, че TCP-AO има т.нар. алгоритмична гъвкавост. Още едно подобрение е осигуряването на по-лесен механизъм за координация на ключове (чрез *Master Key Tuple* (MKT) – статични или външни), както и по-ясно формулирани препоръки за външния обмен на ключове. Това дава възможност за замяна на един ключ с друг в една и съща връзка. В същото време цялостен криптографски обмен на ключове не е възможен в рамките на TCP, защото при TCP SYN липсва достатъчно свободно място и за управление на такива договаряния (това е описано по-подробно в раздел 7.6 на документа RFC5925).

При TCP-AO идентификаторът на опцията е 29, а при TCP MD5 е 19, но въпреки това не е възможно и двата механизма да се използват едновременно. За дадена връзка може да се ползва само един от двата. TCP-AO поддържа IPv6 и е напълно съвместим с предложените изисквания за подмяна на TCP MD5. Проблем при използването на TCP-AO се явява фактът, че всички, които ползват сега TCP MD5, не могат да мигрират направо, защото TCP MD5 не поддържа каквито и да било промени в алгоритъма на сигурността след като веднъж връзка е установена [105, 110, 118].

Връзки, които се знае, че използват TCP-AO, могат да бъдат атакувани с изпращането на сегменти с невалидни MAC кодове. Атакуващите трябва да знаят само *ID* на връзката и стойността на полето *Length* на TCP-AO, за да повлияят на процесорния капацитет на устройството. Това е подобно на атаките към IPsec, където IP адресът и SPI (*Security Parameter Index*) са видими. Разликата е, че за IPsec, целият SPI (32 бита) е произволен, докато при маршрутизиращите протоколи само порта на източника (16 бита или по-малко) е произволен. Като резултат теоретично е по-лесно за атакуващия да подмени (*spoof*) TCP-AO сегмент, отколкото при IPsec защитата.

Използването на всеки алгоритъм за сигурност може да представи възможност за *Denial-Of-Service* (DoS) атака към процесора, където нападателят изпраща неверни, произволни

сегменти, така че получателят да изразходва значителни процесорни усилия, за да ги отхвърли. В IPsec такива атаки са намалени чрез използване на полетата с голям SPI и *Sequence Number*, които частично валидират сегментите, преди процесорът да валидирала ICV (*Integrity Check Value*).

При TCP-AO, двойката сокети извършва повечето от функцията на IPsec SPI и *IPsec Sequence Number*, за да се избегне *replay* атаки. Така TCP вече може да се предпази от повторения (*replay*) на сегменти с автентични данни с различни контролни битове (като SYN, FIN, ACK), но дори и автентични, те могат да повлияят на прозореца, контролиращ претоварването (*congestion control*) с пакети. При този вид атака TCP-AO не може да се справи, затова когато е необходима такава защита, се препоръчва използването на IPsec.

В стандартизиращия документ RFC5925 „*TCP Authentication Option*“ се казва, че TCP-AO предлага подобна защита като IPsec, но не е проектиран да замени употребата на IPsec или IKE (*Internet Key Exchange Protocol*), когато е необходима по-висока степен на защита или се налага по-сложно за управление на сигурността. В RFC5925 пише: „В действителност ние препоръчваме използването на IPsec и IKE, особено когато е желателно да могат да се договарят параметри и сесийни ключове, или когато е необходима възможност за промяна на ключа. TCP-AO е предназначен за използване, само когато използването на IPsec не би било възможно“ [112, 118].

Много препоръки в последно време сочат използването на IPsec тунелирането като подходящ механизъм за защита на BGP сесията. IPsec не е специфична защита за BGP, но е част от последната версия на интернет протокола (IPv6) и затова осигурява сигурност още на ниво мрежов слой. Тук характеристиките на IPsec, които бяха разгледани във втора глава, като криптиране и удостоверяване, както и предоставянето на ключови услуги за управление на поддържането на дългосрочни сесии, прилягат към нуждите за защита на BGP. Протоколът *Internet Key Exchange* (IKE) се занимава с проблемите на динамичното договаряне на ключове за сесия. Протоколът *IPsec Authentication Header* (AH) и протокол *Encapsulating Security Payload* (ESP) реализира сигурността на

пакетите, за да се гарантира поверителността и автентичността на данните, целостта и конфиденциалността на данните, също и защита от *replay* атаки. Всички тези услуги работят съвместно. Когато се установява IPsec защита, тя е между конкретните комуникиращи чрез IPv6. IPsec все повече се превръща в доминиращо средство за създаване на сигурна комуникация между устройства, тъй като то е повсеместно, добре познато и по-лесно за конфигуриране, освен това предлага и цялостни решения за сигурността на BGP. Както е показано в Таблица 3.1, IPsec осигурява всеобхватна защита, включително и ограничавайки, до известна степен DoS атаки, в сравнение с MD5. Решения като MD5 и TCP-AO се използват, защото те са лесни за реализиране, а конкретно що се отнася до MD5, то представлява исторически наложилият се като вграден метод за защита. Затова автономните системи ще продължат да използват тези методи, докато не се приложи по-силна и надеждна защита, каквато представлява IPsec. От съвременна гледна точка TCP MD5 служи само като краткосрочна мярка и не трябва да бъде прилагано като дългосрочно решение, ако има възможност на устройствата, на които се конфигурира да поддържат по-добри решения като IPv6 (IPSec) [116].

	Integrity	Confidentiality	Replay Prevention	DOS Prevention
MD5 Integrity	yes	no	yes	no
≈				≈
IPsec (AH)	yes	no	yes	yes
IPsec (ESP)	yes	yes	yes	yes

Таблица 3.1 BGP *Peer Session Security Solutions*
Изискванията за сигурността са в колоните. Сравнението е на база защитата осигурявана между автономните системи. [116]

Основен проблем при атаките, причиняващи „отказ от услуга“ чрез предизвикване на физическо прекъсване на връзка от транспортната мрежа или с наводнена (*flooding*) връзка с огромно количество трафик (ставайки неизползваеми) е възможността атакуващият да пренасочи данни от трафика по такъв начин, че да преминават през възли, контролирани от него. Протоколи като IPsec предоставят ограничени форми на превенция на DoS

атаките, но нито един адекватен отговор на *flooding* атаките. Един метод за защитата срещу тези атаки е да се попречи на маршрутизаторите да извършват повече обработка, отколкото е необходимо, чрез изолиране на входящия трафик в множество опашки, основани на приоритет. Съобщения, които влияят върху процеса на маршрутизация (BGP UPDATE и NOTIFICATION съобщенията или други изпращани към TCP порт 179) ще бъдат поставени в опашката с по-висок приоритет, която ще има по-голям достъп до процесора на маршрутизатора, всеки друг трафик ще се поставя в опашка с по-нисък приоритет, като ще се обработва, когато са налични ресурси [116, 110]. Това консумира ресурси на устройството, но и трите метода TCP-AO, TCP MD5 и IPsec влияят на производителността на маршрутизатора и това не може да се избегне.

Още един щрих към защита на BGP е необходимостта да се съгласуват системни администратори на различни автономни системи. При BGP, когато се създава връзка, обикновено тя е към зона на мрежова администрация под външен контрол и само по себе си, да се конфигурира да работи е труден и време-отнемащ процес. Това се дължи на факта, че различните организации имат различни политики за сигурност и спазват различни процедури, които обикновено се поддържат от много администратори, примерно при по-големите доставчици на интернет. Затова се случва така, че да се прави компромис с удостоверяването или с конфиденциалността на данните. А и двете, силно криптиране и надеждно удостоверяване, се налага да има, за да се защити BGP сесия. [133]

В настоящата глава беше разгледан и анализиран протокола за външна маршрутизация BGP в частта установяване на съседство между маршрутизатори. Представен е начинът му на работа и различните видове служебни съобщения, които се налага да бъдат защитени при предаване. Беше обяснено защо механизъм за удостоверяване с пароли не е достатъчен за защита от намеса на „трети страни“ и защо се налага приложение на IPSec. На база на изследването направено до тук, се вижда че никой от двата TCP вградени механизма не предлага нищо повече от IPsec, а в същото

време те не предлагат конфиденциалност. В следващата глава ще ще бъде анализиран протоколът за вътрешна маршрутизация OSPFv3, като ще се покаже че при него IPSec на IPv6 е единствено средство за защита. Затова то се явява като универсално решение както за OSPFv3, така и за BGP4/4+ едновременно, което е нужно за реализацията на задачата в настоящата работа.

4. Четвърта глава – Протокол за маршрутизация OSPFv3 и начин на функциониране

В тази глава ще се обясни функционирането на протокола за маршрутизация OSPF, като се наблегне на особеностите на версия 3, пригодена за работа в IPv6 среда.

4.1 Изследване на особеностите на версия 3 на протокол OSPF

Протоколът за маршрутизация OSPF се класифицира като маршрутизиращ протокол със следене на състоянието на връзката (*Link-state*), използващ алгоритъма на Дийкстра (Edgar W. Dijkstra) за намиране на най-кратък път в граф (*Shortest Path First – SPF*). Това важи както за OSPFv2, така и за OSPFv3. Този алгоритъм не означава, че избира пътя с най-малко междинни маршрутизатори между изпращача и получателя. При него всеки път си има цена, и тя е определяща за това дали един маршрут ще бъде избран или не. Самият алгоритъм и доказателства свързани с него са материя, която се разглежда и изследва от дискретната математика и излизат извън обхвата на настоящата работа.

При тези протоколи конвергенцията (*convergence*) се постига на няколко стъпки. Конвергенция означава да се постигне уеднаквена картина върху мрежовата топология от всички участници (всички маршрутизатори) [47]. Стъпките са следните. На първо място всеки маршрутизатор научава кои са неговите директно свързани мрежи. Това става чрез обмен на *hello* пакети, с които става „запознаването“ с другите пряко свързани *Link-state* маршрутизатори. След което всеки маршрутизатор създава свой собствен пакет *Link State Packet* (LSP), който включва информация за съседите му, като: идентифициращ номер на съседа (*neighbor ID*), вид на връзката, примерно *Serial* или *Ethernet* (*link type*), и пропускателната способност (*bandwidth*). След като се създаде този пакет, той се изпраща до всичките му преки съседи, които записват информацията и я препращат към техните съседи, докато

всички участници в комуникацията не получат и запишат еднаква информация за мрежата. Едва след като всички маршрутизатори получат всички LSP пакети, всеки от тях конструира мрежова карта на топологията, чрез която в последствие се определя и най-добрият път, до който и да е друг в мрежата. Под директно свързани мрежи се има предвид, мрежи, които са прилежащи към някой от интерфейсите на маршрутизатора. *Link-State* означава информация за състоянието на връзката. За да се установи кои са директно свързаните мрежи, работещи с *link-state* протокол, се ползват *Hello* пакет. Всеки маршрутизатор е длъжен да изпрати такъв пакет към своите интерфейси и ако от другата страна работи същия протокол, те започват да разменят *hello* пакети, през определен интервал от време, показващ че връзката не се е прекъснала. Установяването на такова съседство се нарича *adjacency*. Стъпката, при която се изпращат LSP пакетите, съдържащи състоянието и информация за пряко свързаните, се случва при две условия: при начално инициализиране на маршрутизатор или когато има промяна в топологията. Базата от данни, която всеки маршрутизаор попълва с информацията от LSP пакетите се използва за конструирането на картата на топологията. При наличието на цялата тази информация, и като се премахне информацията която се повтаря (понеже съседния маршрутиразтор също изпраща своята директна връзка с този), се построява SPF дърво за всички мрежи, които са достъпни (не само пряко свързаните). Най-добрият път до всяка от мрежите в топологията се намира, чрез изчисляването на най-ниската цена за всеки маршрут, сумирайки цените на всяка връзка по пътя до всяка мрежа. Накрая SPF алгоритъмът определя най-кратките пътища и ги записва в маршрутизиращата таблица. Тази таблица има както директно свързаните, така и мрежите до които има достъп с техните изчислени цени. Преимуществото на тези алгоритми е, че маршрутизаторът може да определи самостоятелно най-добрият път до всяка мрежа в топологията, а конвергенцията се постига бързо. Това бързодействие изисква повече памет и по-добра процесорана производителност на маршрутизатора. Освен това при първоначално стартиране на процеса при *link-state* протоколите се консумира значителна част от *bandwidth* поради

множеството предавани служебни пакети. В останалата част от тази глава ще се разгледа как е реализиран този процес при протокола OSPF и ще се обърне по-конкретно внимание на разликите при протокола OSPFv3, предназначен за работа с IPv6.

Исторически погледнато, развитието на OSPF започва още през 1987 година. Първата версия е била само за експериментални цели и никога не е ползвана. Версия 2 на OSPF, която е настоящата версия, работеща с IPv4, е от 1991 година, като е обновена през 1998 с официалния RFC2328. Година по-късно е публикуван и RFC2740 за OSPFv3 за IPv6. Сегашната версия на стандартизационния документ е RFC5340 „*OSPF for IPv6*“ и е от 2008 година. Както беше разгледано в първа и втора глава, предстоящото масово навлизане на IPv6 след изчерпването на IPv4 адресно пространство ще наложи и на маршрутизиращите протоколи да преминат към версии, пригодени за работа с IPv6.

Много от характеристиките на OSPFv2 са наследени и от OSPFv3. Затова ще бъдат разгледани особеностите на новия протокол. Някои от нещата, които се запазват и при двете версии са това, че и двата са *Link-state Interior Gateway Protocol* (IGP), тоест *Link-state* протоколи за вътрешна маршрутизация, ползват алгоритъма SPF, запазен е и принципа с назначаването на контролен маршрутизатор, областите (*OSPF areas*) и други. Но формата на OSPF пакетите и LSA са различни (най-малкото защото трябва да поддържат 128-битови IPv6 адреси). Има и други промени и подобрения, които не позволяват новата версия OSPFv3 да е обратно съвместима с предходната. Но трябва да се каже, че от IETF са доразработили протокола и през 2010 година публикуват RFC5838 „*Support of Address Families in OSPFv3*“, така че да се даде възможност да се поддържат и двете адресни фамилии IPv4 и IPv6. Последното обновяване на RFC5838 е от месец юли 2013 година, публикувано в RFC6969. Това е много подобно на *MultiProtocol Border Gateway Protocol* (MP-BGP), който функционира и като IPv4 и като IPv6 маршрутизиращ протокол [49, 50, 51].

Двете версии на OSPF имат еднакъв протоколен номер 89. Този номер за OSPFv3 се записва в полето *Next Header* на IPv6 пакета, докато при OSPFv2 в полето *Protocol* на IPv4 пакета.

Има малка промяна в терминологията, която трябва да се изясни, за да не стават обърквания по-нататък в изложението. При IPv6 употребата на *link* е подобна по значение със *subnet* или *network* за OSPF с IPv4. Но трябва да се обърне внимание, че *Network-LSA* остава непроменено, а едновременно с това е създаден нов *Link-LSA* с различна употреба.

В началото ще се разгледат няколко важни особености при версия 3 на OSPF, а след това по-подробно ще се обърне внимание на принципа на работа на OSPF протокола и промените при формата на пакетите и формата на LSA при OSPFv3.

Това, че OSPFv3 работи с връзка (*Per-Link*), а не с подмрежа (*Per-Subnet*), не е единствено терминологична промяна. На един интерфейс вече могат да се конфигурират повече от един IPv6 адрес. Една връзка (*link*) може да принадлежи на няколко подмрежи. Тоест, два интерфейса може да са свързани с една мрежова връзка, но да комуникират по различни IPv6 подмрежи (префикси). IPv6 използва термина *link*, за да означи комуникационната способност или средата, върху която устройствата могат да комуникират на мрежово ниво.

Друга промяна, която е резултат от преминаването на OSPF към работа с IPv6, е премахването на т.нар. *Addressing Semantics* OSPF от протоколните пакети и основните LSA типове. IPv6 адресите вече не са представени в OSPF пакетите, с изключение на LSA *payload*, носен от пакетите *Link State Update*. *Router-LSA* и N*etwork-LSA* вече не съдържат мрежов адрес (има нов тип LSA), а само информация за топологията. OSPF *Router ID*, *Area ID* и *LSA Link State ID* остават 32-битови и записани като десетични числа разделени с точки (IPv4 нотация) и не им се присвояват IPv6 адреси. Освен това, съседните маршрутизатори вече се определят не от IP адреса, а само от *Router ID*, независимо от вида на връзката.

Още една промяна е добавянето на допълнителен *Flooding Scope* (област на разпространение на LSA съобщенията). Дефинирани са три *flooding scopes*:

– *Link-local scope* – LSA се разпространяват само в локалната връзка (*local link*) и никъде другаде. Използва се от новия *link-LSA*;

– *Area scope* – LSA се разпространяват само в една OSPF област (*area)*. Ползва се от *router-LSA*, *network-LSA*, *inter-area-prefix-LSA*, *inter-area-router-LSA* и *intra-area-prefix-LSA*;
– *AS scope* – LSA се разпространяват в автономната система. Ползва се от *AS-external-LSA*.

Още една разлика с OSPFv2 е употребата на IPv6 *Link-Local* адреси за откриване на съседи (*neighbor discovery*), *auto-configuration* и др. по единична връзка (*single link*). Както вече беше казано, тези адреси не се маршрутизират, а са за локална употреба. От всички OSPF интерфейси, без тези на виртуалните връзки (*virtual links*), OSPF пакетите се изпращат посредством асоциирания към интерфейса *Link-Local Unicast* адрес, като адрес на източник (*source address*). Маршрутизаорът научава *Link-Local* адресите на всички останали маршрутизатори, свързани към неговите връзки (*links*) и използва тези адреси като *next-hop*, за да препрати пакетите.

Важна разлика е и поддържането на няколко инстанции на OSPF протокола на едина връзка (*Multiple Instances per Link*). Например това може да се използва, ако *Network Access Point* (NAP) сегмент е споделен между няколко доставчика. Всеки от тях ползва различни OSPF домейни и трябва да си останат такива, дори и да няма налични повече от една физически връзки. При OSPF за IPv4 това се е постигало с използването на различно удостоверяване, зададена в полето *authentication*, а сега се пускат колкото инстанции на протокола са необходими. Друг пример за употреба е, ако една връзка се иска да принадлежи на няколко OSPF области. Поддържането на няколко инстанции за една връзка се постига с *Instance ID*, записан в хедъра OSPF пакета и в структурата от данни за OSPF интерфейса (*Interface Data Structure* [48]).

Управлението на неразпознатите LSA типове също е променено. При OSPFv2 винаги се отхвърлят, а при OSPFv3 или се обработват все едно имат *link-local flooding scope* или се записват и разпространяват по същия начин както, ако са били разпознати, но игнорирайки ги от собствените си SPF алгоритми. Предишният начин на работа е причинявал проблеми, когато

получаващият маршрутизатор е поддържал повече опции от другите, принадлежащи на връзката, и е можел да обработи правилно съобщенията.

Последната важна промяна, която ще бъде разгледана подробно в следващата точка на тази глава, е свързано с удостоверяването при OSPFv3. Понеже OSPFv3 ползва IPv6 адресното пространство и тъй като IPsec е част от протокола IPv6, аутентикацията е била премаханта от OSPFv3. Полетата *Authentication Type* и *Authentication* са премахнати от хедъра на OSPF пакета и всички останали полета свързани с аутентикацията също са премахнати от OSPF областта и структурите от данни на интерфейсите. Сега OSPF разчита на *IP Authentication Header* и *IP Encapsulating Security Payload* за осигуряване на целостта на данните, аутентикация (удостоверяване) и конфиденциалност на обменяните съобщения [48].

Протоколът OSPF ползва пет вида протоколни пакета: *Hello* (открива съседи и установява съседство между тях), *Database description* (DBD – проверява синхронизацията на базите данни между маршрутизаторите), *Link-state request* (LSR – прави заявка за определен *link-state* запис от маршрутизатор към друг маршрутизатор), *Link-state update* (LSU – изпраща заявен вече *link-state* запис, съдържа един или повече LSA) и *Link-state acknowledgment* (LSA – изпраща се за да потвърди другите видове пакети (примерно получаването на LSU съобщението), съдържа маршрутизираща информация за съседите) [52]. Те споделят един общ протоколен хедър (16 байта), съдържащ *Router ID*, *Area ID*, *Type* (кода, отговарящ на вида на пакета, който следва) и др. Протоколът OSPF работи директно над IPv6 мрежов слой. Целият OSPF пакет е енкапсулиран в IP хедър (съдържащ адресите на източника и получателя), който от своя страна е в *Data Link Frame* хедър (съдържащ MAC адреси на източника и получателя).

Преди да се пристъпи към изследването на форматите на протокола и LSA типовете, ще се обърне внимание на едно поле, общо за *Hello* и *Database Description* пакетите, както и за *router-LSAs*, *network-LSAs*, *inter-area-router-LSAs* и *link-LSAs*. То позволява на маршрутизаторите да поддържат или не, определени опции и да споделят тези свои възможности с други OSPF маршрутизатори. На фигура 4.1 е даден формата на това поле.

```
                     1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8  9 0 1  2  3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+--+-+-+-+--+--+
| | | | | | | | | | | | | | | | | |*|*|DC|R|N|x| E|V6|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+--+-+-+-+--+--+
```

Фиг. 4.1 Формат на *Options* полето [48]

Ако маршрутиатор не разпознава някой бит, той игнорира стойността му и препредава пакета или LSA нормално.

Ако бит V6 е 0, то маршрутизатора (или конкретната връзка), ще бъдат изключени от IPv6 маршрутизирането. Бит E, определя как се разпространяват *AS-External LSA*. Бит x не се ползва в OSPFv3 и се игнорира. Бит N показва дали маршритизатора е закачен към NSSA (*Not-So-Stubby Area* – подобно на конфигурационната опция за OSPF *stub area*, но има допълнителни способности за добавяне на външни за автономната система пътища при ограничен модел [59, 60]). Бит R показва дали изпращача е активен маршрутизатор. Ако е 0, тогава пътищата, които предлага, не се обработват. Бит DC определя как се обработват *Demand circuits*. Това са мрежи, чиято цена се променя в зависимост от използването; промените могат да зависят и от времето на свързаност и от предаденото количество байтове или пакети. Пример за това е *dial-up* линиите [57]. Последните битове означени с * са запазени за OSPFv2, когато е необходимо.

В началото ще се представи общият протоколен хедър на OSPF пакетите. Той заедно с енкапсулиращият IPv6 хедър, съдържат цялата необходима информация, за да се определи дали пакетът е приет за по-нататъшно обработване от маршрутизатора.

На Фигура 4.2 е показан формата на хедъра на OSPF пакета. За новата версия на протокола в полето *Version* е записано 3.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Version #   |     Type      |         Packet length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Router ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Area ID                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Checksum             |  Instance ID  |      0        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Фиг. 4.2 Заглавна част (*Header*) на OSPF пакета [48]

В полето *Type* може да бъде записан един от следните пет вида пакети:

Type	Description
1	*Hello*
2	*Database Description*
3	*Link State Request*
4	*Link State Update*
5	*Link State Acknowledgment*

Дължината на целия пакет в байтове се съдържа в *Packet length*. Тя включва и стандартния OSPF хедър. Полето *Router ID* е на източника на пакета. Полето *Area ID* е 32-битово число, определящо областта, на която принадлежи пакета. Всички пакети са асоциирани с една област. Повечето изминават пътя само от един *hop*. Тези пакети, които са минали по витруален канал (*virtual link*), са означени с *backbone Area ID* равна на 0 (по-нататък в главата подробно е обяснено значението на *Backbone Area*). Полето *Checksum* съдържа контролната сума. Протоколът OSPF прави стандартно изчисление на контролната сума за IPv6, като използва *pseudo-header* с поле *Next Header*, съдържащо стойността 89. Полето *Instance ID* позволява много инстанции на OSPF да бъдат пуснати на една единствена връзка. Всяка протоколна инстанция трябва да бъде съпоставена с отделно *Instance ID*. Тази стойност има само локално значение за съответната връзка. Пакетите, чието *Instance ID* е различно то това на интерфейса на получателя, се отхвърлят. Последното поле е запазено, има стойност 0 и се игнорира.

Първият вид OSPF пакети е *Hello Packet*. Те се изпращат периодично по всички интерфейси (включително и виртуалните), за да установяват и управляват връзките със съседните маршрутизатори. Всички свързани помежду си маршутизатори трябва да се договорят и съгласят за параметрите (*HelloInterval* и *RouterDeadInterval*). На Фигура 4.3 е показан форматът на OSPF *Hello* пакета.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      3        |       1       |         Packet Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Router ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Area ID                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Checksum             | Instance ID   |     0         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Interface ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Rtr Priority  |             Options                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        HelloInterval          |       RouterDeadInterval      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Designated Router ID                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Backup Designated Router ID                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Neighbor ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          ...                                  |
```

Фиг. 4.3 Формат на OSPF *Hello* пакета [48]

След хедъра на *Hello* пакета, първото поле е *Interface ID*, 32-битов уникален номер, разграничаващ еднозначно интерфейса от останалите интерфейси на маршрутизатора. Полето *Rtr Priority* указва приоритета на маршрутизатора при избора на (*Backup*) *Designated Router*. Ако е 0, не може да бъде избран. Полето *Options* вече беше разгледано. Полето *HelloInterval* съдържа броя секунди, между които маршрутизатора изпраща *Hello* пакети. Полето *RouterDeadInterval* е времето, след изтичането на което, ако маршрутизатор не е отговарял, се счита за отпаднал от комуникацията. Полето *Designated Router ID*, идентифицира DR за тази мрежа. Стойността на DR е равна неговото си *Router ID*. Ако няма DR, тогава стойността на полето е 0.0.0.0. Полето *Backup Designated Router ID* е напълно аналогично на предишното. Полето *Neighbor ID* съдържа ID на маршрутизаторите, които са му съседи.

Следващият OSPF пакет *Database Description* е от тип 2. Форматът е показан на Фигура 4.4. Тези пакети се разменят, когато се инициализира установяването на съседството (*adjacency*). Те описват съдържанието на *link-state* БД (база данни). Може и няколко пакета да бъдат използвани за описване на базата данни. За тази цел се ползва така наречената процедура „*poll-response*“. При нея единият маршрутизатор е назначен за „главен“, другият

за „подчинен“. Главният изпраща DD пакети (*polls*), които се потвърждават от DD пакети (*responses*) изпратени от подчинения. Отговорите (*responses*) се свързват с проучванията (*polls*) чрез поредния номер на DD пакетите (*DD sequence number*).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      3        |       2       |         Packet Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Router ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Area ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |  Instance ID  |      0        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       0       |               Options                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Interface MTU          |      0        |0|0|0|0|0|I|M|MS|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     DD sequence number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-                                                             -+
|                                                               |
+-                     An LSA Header                           -+
|                                                               |
+-                                                             -+
|                                                               |
+-                                                             -+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          ...                                  |
```

Фиг. 4.4 Формат на OSPF *Database Description* пакета [48]

Формата на *Database Description* пакетите е много подобен на *Link State Request* пакетите и на *Link State Acknowledgment* пакетите. Главната част от всички тях е списък от елементи, всеки от които описват част от базата от данни. Пакетите *Link State Request* са от тип 3. Форматът им е показан на Фигура 4.5. След като съседните маршрутизатори си разменят DD пакети, единият може да види, че част от неговата *link-state* база данни е с по-стари данни. Пакетът LSR се ползва, за да изиска от съседа си част от базата данни, която да е по-нова. Това налага обикновено размеряването на множество LSR пакети. Маршрутизаторът, който изпраща LSR пакети, знае точно коя част от БД е изискал, като всяка част (инстанция) се определя от нейн LS пореден номер, LS контролна сума и LS възраст. Но тези полета не са специфицирани в самия LSR пакет. Затова маршрутизаторът може да получи като отговор дори по-нова инстанция.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      3        |       3       |        Packet Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Router ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Area ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Checksum             |  Instance ID  |      0        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0                  |        LS Type                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Link State ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Advertising Router                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
```

Фиг. 4.5 Формат на OSPF *Link State Request* [48]

Всяка LSA заявка се определя от нейния: LS тип, *Link State ID* и *Advertising Router*. Това еднозначно определя LSA, без да специфицира неговата инстанция. Пакетите *Link State Request* по подразбиране изискват най-последната инстанция от уточнените LSA.

Пакетите *Link State Update* са тип 4. Те имплементират механизма с разпространението (*flooding*) на LSA. Форматът им е показан на Фигура 4.6.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      3        |       4       |        Packet Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Router ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Area ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Checksum             |  Instance ID  |      0        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             # LSAs                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-                                                             -+
|                              LSAs                             |
+-                                                             -+
|                               ...                             |
```

Фиг. 4.6 Формат на *Link State Update* пакет [48]

Всеки *Link State Update* пакет пренася колекция от LSA съобщения от изходящия маршрутизатор, един маршрутизатор напред. Няколко LSA могат да бъдат включени в един пакет. *Link State Update* пакетите са *Multicast* на тези физически мрежи, които поддържат *Multicast/Broadcast*. За да се направи по-надежден механизмът по разпространение, LSA се потвърждават чрез *Link*

State Acknowledgment пакети. Ако препредаването на определени LSA е необходимо, тогава препратените LSA са винаги пренасяни от *Unicast Link State Update* пакети. Както се вижда и от Фигура 4.6, в тялото на пакета се съдържа списък от LSA, всеки от тях започва с общ 20-байтов хедър, който вече беше разгледан.

Пакетите *Link State Acknowledgment* са от пети тип. За да се направи разпространението на LSA пакетите надеждно, те потвърждават експлицитно или имплицитно. Експлицитното се извършва при получаването и изпращането на *Link State Acknowledgment* пакети. Множество LSA могат да бъдат потвърдени в един *Link State Acknowledgment* пакет. В зависимост от състоянието на интерфейса и изпращача на съответния *Link State Update* пакет, *Link State Acknowledgment* пакета се изпраща до *Multicast* адрес – *AllSPFRouters*, *Multicast* адрес – *AllDRouters* или до *Unicast* адреса на съседа. Формата на пакета е показан на фигура 4.7.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      3        |       5       |        Packet Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Router ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Area ID                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Checksum             |  Instance ID  |      0        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-                                                             -+
|                                                               |
+-                        An LSA Header                        -+
|                                                               |
+-                                                             -+
|                                                               |
+-                                                             -+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             ...                               |
```

Фиг. 4.7 Формат на *Link State Acknowledgment* пакет [48]

Той е подобен на пакета *Data Description*, понеже тялото и на двата е списък от LSA хедъри. Всеки LSA, който ще бъде потвърден, се описва с неговия LSA хедър. LSA форматите ще бъдат разгледани малко по-надолу в изложението. Като обобщение, за последните два вида пакети само може да се допълни, че *Link State Update* (LSU) се използва за доставяне на един или повече *Link-State Advertisement*, а *Link State Advertisement* (LSA) съдържат

информация за съседните маршрутизатори и цената на пътя. Но често тези два акронима се използват взаимно заменяемо, понеже LSU съдържа един или повече LSA [48, 53, 54, 56].

Видовете LSA са осем типа, като всеки започва със стандартен 20-битов LSA хедър. Сравнени с този на OSPFv2 LSA хедър, двата са почти идентични, като разликата е, че тук няма поле *Options* и полето *Link State Type* е по-голямо от това на OSPFv2 (16-бита). Причината, поради която това поле е по-дълго е, че има три водещи бита, кодиращи общите типове свойства. Самото поле е дадено на Фигура 4.8. Бит U определя как маршрутизаторът ще обработва LSA, ако не разпознае *LSA Function Code*. Ако бита е 0, тогава неразпознатите LSA ще се третират като с област на разпространение *link-local*.

```
 0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|U |S2|S1|           LSA Function Code           |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Фиг. 4.8 Формат на *LSA Type* полето [48]

Ако е 1, ще бъдат записани и разпространявани, както ако бяха разпознати. Битовете S1 и S2 определят областта на разпространение (*flooding scope*), както е показано в Таблица 4.1.

S2	**S1**	**Областта на разпространение – *Flooding Scope***
0	0	*Link-Local* – разпространяват се само по изходящата връзка
0	1	*Area Scoping* – разпространяват се само в съответната област
1	0	*AS Scoping* – разпрстраняват се в AS (*Routing Domain*)
1	1	*Reserved* – запазени

Таблица 4.1 Комбинации на битовете S1 и S2, определящи областта на разпространение

Останалите битове от полето *LSA Type* съответстват на OSPFv2 *Type field* поле. Това е онагледено в Таблица 4.2.

OSPFv3 LSAs		**OSPFv2 LSAs**	
LS Type	**Name**	**Function Code**	**Name**
0x2001	Router LSA	1	Router LSA
0x2002	Network LSA	2	Network LSA
0x2003	Inter-Area Prefix LSA	3	Network Summary LSA
0x2004	Inter-Area Router LSA	4	ASBR Summary LSA
0x4005	AS-External LSA	5	AS-External LSA
0x2006	Group Membership LSA	6	Group Membership LSA
0x2007	NSSA LSA	7	NSSA External LSA
0x0008	Link LSA		*Няма съответно LSA*
0x2009	Intra-Area Prefix LSA		*Няма съответно LSA*

Таблица 4.2 Съпоставка на OSPFv3 и OSPFv2 LSAs

Стойностите на *LS Type*, които са записани в шестнайсетичен код, като се декодират в двоичен се вижда, че бит U за всички е по подразбиране 0. За S битовете при всички, без два, областта за разпространение е *Area Scoping*. За останалите два: за *AS-External*, LSA областта на разпространение е *AS Scoping*, а за *Link LSA* е *Link-Local*.

Преди да бъдат разгледани по-подробно различните LSA, трябва да се обърне внимание на представянето на IPv6 *prefix* в OSPF. Адресите IPv6 са с дължина 128 бита. При маршрутизирането им (в частност с OSPF) се предават IPv6 адресни префикси. Тези префикси са поредица от битове, чиято дължина е в интервала от 0 до 128 бита (включително). При OSPF, IPv6 адресните префикси са винаги представени от комбинация от три полета *PrefixLength*, *PrefixOptions* и *Address Prefix*. *PrefixLength* е дължината на префикса в битове. *PrefixOptions* е 8-битово поле, описващо различни възможности свързани с префикса, като: неучастие в Unicast изчисленията; дали адреса е IPv6 адрес на интерфейс; дали да бъде препредаван пакета от NSSA граничен маршрутизатор. Също така има и контролен бит за *inter-area-prefix-LSA* или *AS-external-LSA*, свързан с VPN среда. *Address Prefix* се кодират като кратни на 32-битови думи, отместени с нулеви битове, когато е необходимо. Пътят по подразбиране (*default route*) се представя с префикс с дължина 0.

Както се вижда и от съпоставените имена в таблица 4.2, за двете версии, *Router* и *Network LSA* са със запазени имена, но това не означава, че няма разлика в реализацията им. По-съществената разлика е, че при OSPFv3 тези два LSA вече не разпространяват IPv6 *prefix*-и. Това е значително подобрение от гледна точка на скалируемостта на протокола. Тези два LSA се използват за представянето на маршрутизатора като връзка в SPF дървото. Когато се разпространяват, значи е настъпила някаква промяна в топологията и тя засяга маршрутизаторите в съответната област, които щом получат тези пакети, стартират изчислението на SPF алгоритъма. Но тъй като в OSPFv2 тези два LSA се ползват и за съобщаване на информация за свързаните подмрежи, то като настъпи промяна и тя бъде оповестена всички други маршрутизатори отново стартират SFP алгоритмите, независимо дали топологията се е променила и наистина се налага. Всичко това може да окаже негативен ефект за крайните (*edge* или *access*) маршрутизатори, които имат много стъб (*stub*) връзки, променящи се често. Затова OSPFv3 премества разпространението на информацията за префиксите в нов *Intra-Area Prefix LSA*. По този начин *Router LSA* и *Network LSA* разпространява информация, която засяга промените за SPF алгоритъма. При промяна на префикс или състояние на stub връзка, тази информация се разпростряна от *Intra-Area Prefix LSA* и не предизвиква стартирането на SPF алгоритъма [48, 53, 62].

Друго подобрение свързано с *Router LSA* и *Network LSA* е, че информацията която е важна само за директно свързаните съседи, вече не се разпространява в цялата област. С тази информация, при OSPFv3 се занимава новият *Link LSA*, който както се каза има *link-local* област на разпространение.

Inter-Area Prefix LSA, *Inter-Area Router LSA* и *Type-7 LSA*, променят имената си, но функционалността им се запазва същата като на *Network Summary LSA*, *ASBR Summary LSA* и *NSSA LSA*. А що се отнася за *AS-External LSA* и *Group Membership LSA* (за *Multicast OSPF* – MOSPF), те запазват и имената и функционалността си от OSPFv2 [48, 49]. Сега ще се разгледа всеки от типовете LSA каква задача изпълнява.

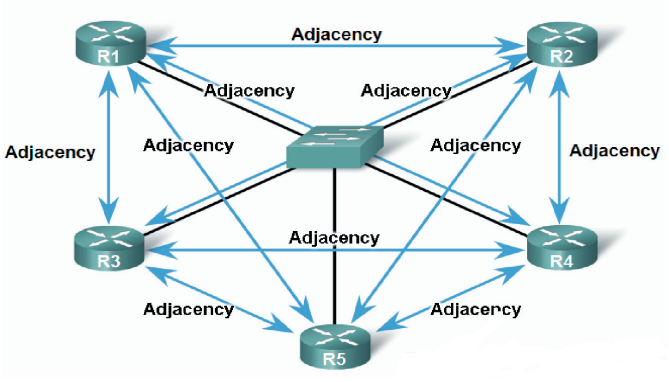
- Първият тип е *Router LSA*. Всеки маршрутизатор в дадена област създава един или повече *Router LSA*. Пълната колекция от тях, създадена от маршрутизаторът, определя състоянието, вида и цената на съответните интерфейси на маршрутизатора в областта.
- Следващият тип е *Network LSA*. Той се създава за всяка *Broadcast* и NBMA връзка в областта, която включва два или повече съседни маршрутизатора. Създава се от връзка на Назначения (*Designated Rauter – DR*) маршрутизатор и съдържа описание на всички закачени към връзката маршрутизатори, включително и DR. В полето *Link State ID* е записано това *Interface ID*, което DR е бил изпратил в *Hello* пакетите по връзката. В този LSA не се указва метрика, защото дистанцията за всички закачени маршрутизатори е нула.
- Третият тип е *Inter-Area Prefix LSA*. Тези LSA се изпращат от граничния за областта маршрутизатор и описват IPv6 префиксите на адресите, които принадлежат на други области. За всеки IPv6 префикс се създава отделен *Inter-Area Prefix LSA*. За крайните области с една връзка (*stub areas*), *Inter-Area Prefix LSA* могат също да бъдат използвани за описанието на пътя по подразбиране. Обобщените пътища по подразбиране (*Default summary routes*) се използват вместо разпространението на пълното множество от външни пътища. За да се окаже, че е един път е *default summary route*, *Inter-Area Prefix LSA* има *PrefixLength* равно на 0.
- Четвъртият тип е *Inter-Area Router LSA*. Те произхождат от граничния маршрутизатор на областта и описват граничните за другите области маршрутизатори (*AS boundary router – ASBR*), като за всеки се генерира отделно LSA.
- Петият тип е *AS-External LSA*. Отново произхождат от граничния за областта маршрутизатор. Те описват крайните дестинации, намиращи се извън областта. Могат да бъдат използвани за описание на път по подразбиране, ако не съществува специфичен път до целта. Тогава *AS-External LSA* има *PrefixLength* равно на 0.

– *Group Membership LSA* – те се специфицират за една OSPF област, тоест разпространението им е само в една област. Маршрутизатор закачен към няколко области (като например *area border router* – ABR), може да окаже на изходящите си *Group Membership LSA* по една *Multicast* посока за всяка от закачените области. Съдържанието на *Group Membership LSA* обаче може да е различно в зависимост от техните асоциирани области [58].
– Седмият тип е NSSA-LSA. При областите NSSA (*Not-so-stubby area*), могат да се импортват пътища от външни автономни системи и да се изпращат към други автономни системи, обаче не е възможно от други маршрутизатори да се получават *AS-External* пътища. NSSA е разширение на *Stub* областите, което позволява добавянето на външни пътища при ограничен модел към *stub* областта [32, 61]. За да се препращат адреси с *NSSA-LSA* с NSSA граничния маршрутизатор, трябва да бъде избран глобален IPv6 адрес. Избора трябва да прави допълнителна проверка да подсигури, че не се ползва IPv6 *link-local* адрес.
– Осмият тип е *Link-LSA*. Маршрутизаторът създава отделен *Link-LSA* за всяка закачена физическа връзка. Областта им на разпространение е *link-local* и никога не излизат извън съответната връзка. Тези LSA имат три задачи, които: 1. доставят *link-local* адреса на маршрутизатора на всички останали маршрутизатори, закачени за връзката; 2. информират другите маршрутизатори, закачени за връзката, за списъка от IPv6 префикси, асоциран с тази връзка; 3. позволяват на маршрутизаторът да предостави множество от *Options* битове в *Network LSA*, който произлиза от *Designated Router* на *Broadcast* или NBMA връзка. За *Network LSA*, *Link State ID* е равно на *Interface ID* на връзката на маршрутизаторът, от който произлиза.
– Последният тип е *Intra-Area Prefix LSA*. Маршрутизаторът използва тези LSA, за да рекламира един или повече IPv6 адресни префикси, които са асоциирани с локалните адреси на маршрутизатора, със закачения *stub* мрежов сегмент

или със закачен транзитен мрежов сегмент. При OSPFv3 с IPv6, всичката адресна информация, разпространявана при OSPFv2 с *Router LSA* и *Network LSA*, се разпространява с *Intra-Area Prefix LSA* [48, 53].

След като маршрутизаторите, ползващи OSPF, са разменили необходимата им информация и са създали базите данни с LSA съобщенията, получени от другите маршрутизатори, тази информация се подава и се изпълнява от алгоритъма *Shortest Path First* (SPF) на Dijkstra. Този алгоритъм създава SPF дърво (структура от данни). А SPF дървото се използва за попълване на маршрутизиращата таблица. Съобщенията LSA се обновяват на всеки 30 минути, а иначе обновления се пращат само при наличие на някаква промяна. OSPF ползва *cost* (цена) като метрика, за определянето на най-добрият път. Най-добрият има най-ниска сумарна цена от източника до получателя. Всяка връзка има своя цена, която е базирана на *bandwidth* (пропускателната способност). Тази стойност може да бъде настройвана при конфигурирането на маршрутизатора, защото ако *bandwidth* е различен от този, който е зададен по подразбиране, това ще повлияе и на намирането на най-добрия път. Видовете интерфейси (мрежови пътища) са *Point-to-point*, *Broadcast Multiaccess*, *Nonbroadcast Multiaccess* (NBMA), *Point-to-multipoint* и *Virtual links*. При *Multiaccess* мрежите има две предизвикателства. Първото е, че трябва да има множество съседства, установени между маршрутизаторите (*adjacency*). Съседството (*adjacency*) при OSPFv3 се формира при IPv6 *link-local* комуникация. Адресите *link-local*, са тези които винаги започват с FE80::/10 и не се маршрутизират от маршрутизатора, тоест ползват се локално за връзката, принадлежаща на интерфейсите, които са вързани за нея. Второто е, че има интензивно разменяне на LSA (така нареченото *flooding* или „наводняване“).

Както се вижда и на примера от Фигура 4.9, е необходимо създаването на голямо множество връзки, дори само при пет маршрутизатора.



Фиг. 4.9 Пример, онагледяващ броя на необходимите установени съседства (*adjacency*) при пет маршрутизатора [54]

С несложни изчисления може да се пресметне при n маршрутизатора, колко съседства ще се създадат. Формулата за пресмятането се извежда лесно. Взимаме сбора на естествените числа от 1 до n и ги събираме със същите числа, но записани под първите в обратен ред, тоест:

$$T(n) + T(n) = 1 + 2 + 3 + \ldots + (n-1) + n + n + (n-1) + (n-2) + \ldots + 2 + 1$$

което е еквивалентно на:

$$T(n) + T(n) = \sum_{i=1}^{n} i + \sum_{i=1}^{n} (n+1-i)$$

$$2T(n) = \sum_{i=1}^{n} (i+n+1-i) = \sum_{i=1}^{n} (n+1)$$

Тъй като сумираме числата (n+1) n пъти то:

$$2T(n) = n(n+1) \Rightarrow \mathbf{T(n) = n(n+1)/2}$$

Както се вижда и на Фигура 4.9, при два маршрутизатора има една връзка помежду им, ако се добави още един, за него ще трябва да се добавят по още две връзки към всеки от предходните два. Така за всеки добавен нов маршрутизатор с номер n към топологията, трябва да се добавят още n-1 връзки към всички предишни, тоест за да разберем колко са всички съседства общо, трябва да намерим техния сбор.

По формулата, която беше изведена по-горе, се замества търсеният брой n-1 на връзките и се получава:

$$T(n-1) = (n-1)((n-1)+1)/2 \Rightarrow \mathbf{S(n) = n(n-1)/2}$$

И от тук лесно се вижда вече, че за примерно 5 маршрутизатора, ще се установят S(5) = 5(5-1)/2 = 10 *adjacency*, между които ще се предават OSPF съобщения. Но при 100 маршрутизатора, S(100) = 100(100-1)/2 = 4950, което е значително количество излишен служебен трафик по мрежовата топология, при който освен изпратените LSA, се генерират и потвърждения връщани обратно към изпращащият маршрутизатор.

За решението на този *flooding* проблем, което е предложено при OSPF, се въвеждат така наречените *Designated router* (DR – назначен маршрутизатор) и *Backup designated router* (BDR – резервен). Тези два избрани маршрутизатора отговарят за изпращането и получаването на LSA. При версия 2 на OSPF всички други маршрутизатори изпращат LSA съобщенията до DR и BDR чрез мултикаст (Multicast) адреса 224.0.0.6, а за версия 3, този адрес е FF02::6 (всички OSPFv3 „назначени“ маршрутизатори – *designated*). От своя страна *Designated router* изпраща LSA съобщенията до всички други маршрутизатори при OSPFv2 чрез мултикаст адреса 224.0.0.5, а при OSPFv3 с мултикаст адрес FF02::5 (всички OSPFv3 маршрутизатори). Това значително намалява излишния трафик. При *Point-to-point* мрежите, каквито примерно са серийните връзки, DR/BDR не се назначават. При тях има само двама участника в комуникацията и тази техника не е необходима. Единствено при *Multiaccess* мрежите (примерно *LAN Ethernet* интерфейси) се назначават DR/BDR и маршрутизатори.

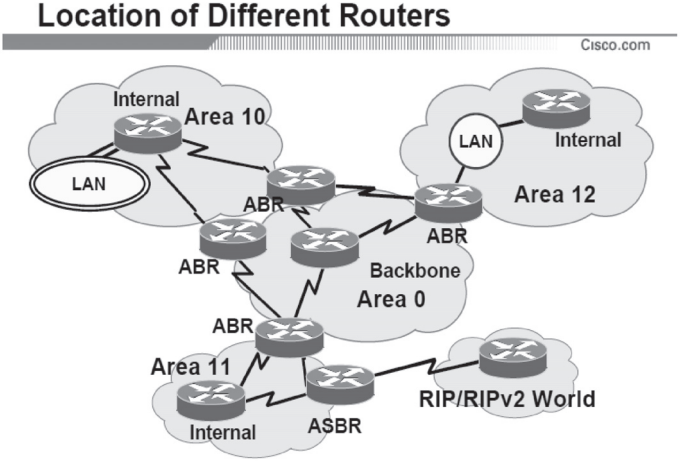
Критериите за избор на DR/BDR при двете версии на OSPF са относително еднакви. При версия 2, *Router ID* е можело да бъде зададено по един от следните начини: или чрез конфигурационна команда, или ако такава не е била въведена се избира най-високата стойност на адрес от някой от *loopback* интерфейсите. Ако няма конфигуриран *loopback* адрес, се избира най-високата стойност на някой активен интерфейс. Причината това да се прави е, че *Router ID* е 32-битово число. При OSPFv3, *Router ID* все още е такова число, но то задължително се конфигурира ръчно чрез команда, понеже IPv6 адресите за 128-битови.

В процеса на назначаване на DR/BDR влияе и OSPF приоритета, зададен на маршрутизатора. По подразбиране той е 1. Колкото е по-висок, толкова по-голяма е възможността да бъде избран за *Designated router*. Ако е зададен на 0, това означава, че този маршрутизатор не може да участва в надпреварата за DR/BDR.

Затова последователността на критериите по назначаване на *Designated router* са първо той да има най-висок приоритет, ако е така се назначава за DR. За *Backup Designated Router* се избира следващият маршрутизатор с приоритет по-нисък от този на избрания DR. В случай, че приоритетите са равни, се избира маршрутизаторът с най-голямо *Router ID* за *Designated router*. Когато един маршрутизатор е назначен за DR, той остава такъв независимо, дали по-добър е бил добавен към топологията. Сменя се при един от трите случая: ако поради някаква причина *Designated router* отпадне от мрежата, ако се случи проблем с OSPF процеса му и той спре да работи или ако самият *multiaccess* интерфейс на *Designated router* прекъсне. За да се предизвика избиране на нов DR може да се изключи сегашния, да се изчака да се избере нов и старият да се включи отново. Или да се рестартира OSPF процеса на участниците в OSPF областта [48, 53, 54, 55].

Както беше отбелязано в началото, и при двете версии на протокола е запазена идеята за областите (*OSPF Areas*). Тъй като OSPF е протокол за вътрешна маршрутизация (IGP) и предава информация между маршрутизатори под един административен контрол (в една автономна система / *Autonomous System* – AS/), е предоставена възможност AS да се дели на отделни области. И при OSPFv3 маршрутизацията в AS е на две нива, в зависимост от това дали източника и получателя са в една област (*area*) или са в отделни. В първият случай се ползва *intra-area routing* (информацията не се маршрутизира извън *OSPF area*), а във втория има *inter-area routing* (маршрутизация между областите). Има една специална запазена *Area* 0 (или записана като 32-битово число 0.0.0.0), която е OSPF *backbone area* (основна, област ядро), към която трябва да бъдат свързани всички останали. Тя разпределя информацията към останалите не-*backbone* области [49, 55]. Маршрутите между две не-*backbone* области преминават през *backbone* областта. На Фигура 4.10 е дадена илюстрация на

тази организация на областите. Пътят, който изминава пакета, е разделен на три части. От източника до граничния на областта маршрутизатор; в самия *backbone* – пътят между двете области; и накрая още един *intra-area* път във втората област до получателя. И тук се гледа сумарната цена за избор на най-добър път. Всеки граничен маршрутизатор в дадена област обобщава за нея цената до всички мрежи извън областта. След изчислението на SPF дървото за дадената област, маршрутите до всички *inter-area* (външни) цели, се изчисляват чрез използването на информацията от граничните маршрутизатори. Образно казано, AS се представя като топология тип звезда, при която *backbone* областта е центърът, разпределител на комуникацията между останалите области. Топологията с *backbone* може да бъде разширена и с добавянето на виртуални връзки (*virtual links*).



Фиг. 4.10 Организация на различните типове области (*Area*) и маршрутизатори при OSPF [56]

Класификацията на маршрутизаторите е според предназначението им, както се вижда и на Фигура 4.10. Има четири вида [49, 55, 56]:

– *Internal routers* (вътрешните за областта) – Това е маршрутизатор, на който всички директно свързани мрежи

принадлежат на същата област. Те изпълняват само едно копие на SPF маршрутизиращия алгоритъм;

– *Area border routers* (ABR) – Граничните маршрутизатори, закачени към няколко области (*OSPF areas*). Изпълняват се няколко копия на основния алгоритъм, по едно копие за всяка прилежаща област. Всеки ABR събира информацията за топологията на неговите области и я предава към *backbone*, който от своя страна я предава към останалите области;
– *Backbone routers* (вътрешните за *Area* 0) – Това са маршрутизатори, на които всички интерфейси са прилежащи на *backbone area* и имат интерфейси към повече от една други области, тоест към интерфейси на ASB;
– *AS boundary routers* (ASBR) – Това са маршрутизатори, които обменят маршрутизираща информация с такива, които принадлежат на други AS или работят с други протоколи, различни от OSPF. Пътя до ASBR се знае от всеки маршрутизатор в AS.

4.2 Използвани методи и средства за защита, чрез приложение на IPsec в процеса на маршрутизация

В оставащата част от тази глава, детайлно ще бъдат описани използваните методи и средства за защита в процеса на маршрутизация – установяване на съседство, обмен на маршрути и т.н. и по какъв начин IPsec намира приложение.

Методите и средствата за защита в процеса на маршрутизация (дейностите свързани с предаване и приемане на пакети) се различават в зависимост от версията на маршрутизиращия протокол. В предишната версия OSPFv2 се е поддържала аутентикация при процеса на маршрутизация с помощта на 64-битова парола (текстова) или за по-добра сигурност криптографска аутентикация с *Message Digest* 5 (MD5). В една област всички маршрутизатори трябва да бъдат конфигурирани с един и същ вид аутентикация. Паролата не е необходимо да бъде еднаква за всички маршрутизатори, но между съседни е задължително [7, 69].

За разлика от версия две, OSPFv3 няма вграден метод за аутентикация и разчита на IPsec за тази функционалност. IPsec може да осигури достоверността на произхода на пакетите, целостта на данните, конфиденциалността и защита от *replay* атака (т.е. когато някой записва легитимираща информация чрез следене на дадена комуникация и след това я повтаря, за да получи нерегламентиран достъп) [16].

При избора на средство за защита и удостоверяване на процесът на маршрутизация са проучени различни решения. Понеже единственият механизъм за удостоверяване предвиден за OSPFv3 е реализиран с IPSec, в Cisco е добавен още един механизъм за удостоверяване, според препоръките RFC6506. Тази опция се нарича „*OSPFv3 Authentication Trailer*“ и дефинира алтернативен механизъм за удостоверяване на OSPF пакетите. С нея се дава и защита от *replay* атаки, чрез използването на пореден номер. Друго положително качество е, че е архитектурно независима. Така, за да се използва не-IPSec криптографско удостоверяване устройствата с OSPFv3 добавят специален блок наречен *Authentication Trailer* към края на OSPFv3 пакета. Неговата дължина не се включва към дължината на OSPFv3 пакета, но е част от IPv6 *payload*.

От *Tata Consultancy Services* (TCS), които са софтуерна компания, са разработили патч, които да реализира препоръките от RFC6506, така че „*OSPFv3 Authentication Trailer*“ да бъде използваем и за „open-source“ маршрутизиращия софтуер Quagga. Причината за това е освен да се предостави допълнителен втори вариант за удостоверяване на OSPFv3, но и да се реши проблемът, при който в някои среди IPSec е трудно да се конфигурира и поддържа, с което става неизползваем механизъм. [162, 163]

Тъй като това изглеждаше едно добро решение беше експериментирано с него. Първоначално избраната версията на Quagga беше променена от 0.99.22 на 0.99.21, за която е създаден пачът. Пачът *RFC6506_patch* беше успешно интегриран с кода на Quagga 0.99.21, като крайния резултат е качен в Git хранилище [164]. Кода се компилира и инсталира успешно. За да може да се използва от операционната система TinyCore Linux, трябваше да се направи инсталационен *quagga_rfc6506.tcz* пакет, който да включва

компилираните изпълними файлове и необходимите библиотеки. Освен него за инсталацията са необходими още два файла. Единия е *quagga_rfc6506.tcz.dep*, който включва списък със свързани програми, а вторият е *quagga_rfc6506.tcz.md5.txt*, в който е записан изчисленият MD5 хеш за *quagga_rfc6506.tcz*. Този инсталационен пакет също е качен в публичното Git хранилище [165]. Всички те се слагат в папка `/mnt/sda1/tce/optional`, а *quagga_rfc6506.tcz* се добавя към списъка `/mnt/sda1/tce/onboot.lst` с програми за стартиране при *boot*. За стартирането и на *quagga* процесите е необходимо в `/opt/bootlocal.sh` да се добавят следните команди:

```
/usr/local/sbin/zebra -u root -d -A 127.0.0.1
   -i /usr/local/var/quagga/zebra.pid
   -f /usr/local/etc/quagga/zebra.conf
/usr/local/sbin/ospf6d -u root -d -A 127.0.0.1
   -i /usr/local/var/quagga/ospf6d.pid
   -f /usr/local/etc/quagga/ospf6d.conf
/usr/local/sbin/bgpd -u root -d -A 127.0.0.1
   -i /usr/local/var/quagga/bgpd.pid
   -f /usr/local/etc/quagga/bgpd.conf
```

След това Quagga 0.99.21 с RFC6506 патча е готова да се използва. Стартира се и се конфигурира нормално, новите настройки, добавени от патча се виждат и се конфигурират без видими проблеми. Но при конфигурирането на машрутизиращите протоколи се вижда проблем с предаването на пакетите, при което не се устовява съседство и маршрутизираща информация не се обменя. След изследване на възможните причини се установи, че в смият патч са открити доста проблеми, които потенциално могат да доведат до неправилно функциониране на Quagga. От TCS са казали, преди година и половина, че продължават да работят, но от тогава няма нови обновявания на патча. Затова тази реализация, на този етап не е възможен вариант за удостоверяване и защита при решения ползващи Quagga.

По този начин предимствата на IPSec като средство за защита и удостоверяване взимат превес и изглаждат единствено възможно средство за реализацията на задачата.

Чрез IPsec могат да се защитят както определени OSPFv3 интерфейси, така и OSPFv3 виртуални връзки (*virtual links*). Необходимо е отделно да се конфигурира маршрутизиращия протокол (OSPFv3) и IPsec. За разлика то Cisco където след конфигуриране на IPSec той трябва да се приложи към OSPFv3 интерфейсите или OSPFv3 виртуалните връзки, тук при Quagga и Linux конфигурациите са напълно разделени. С други думи, използването на IPSec комуникация в рамките на мултикасните групи предназначени за OSPF, не налага допълнително конфигуриране на протокола през Quagga интерфейса.

За реализацията на процесът на подсигуряване на трафика с маршрутизираща информация ще бъде използван инструмента IPsec-tools версия 0.8.0. Той е IPSec имплементация за Linux, като следствие от *KAME-project*, който е бил обединение на различни организации в Япония до 2006г., целящи да създадат свободен IPv6 и IPSec протоколен стек за BSD Unix операционни системи. Компонентите, които включва IPsec-tools са: *libipsec*, *setkey*, *racoon* и *racoonctl*. [159, 160, 161] С *racoonctl* се администрира *racoon*, поддържащ различни опции като презареждане на конфигурацията, показване и изтриване на IPsec SA (*IPSec Security Associations*) и други. За автоматичното управление и договаряне на IPsec връзките, отговаря *racoon Internet Key Exchange (IKE)* „*daemon*“. Това е много удобно и полезно свойство от гледна точка на преносимостта, използваемостта и скалируемостта. Но на базата на направено проучване беше установено, че OSPFv3 освен, че разчита само на IPSec за удостоверяване, използва и муилтикаст *link-local* групите FF02::5 и FF02::6, които от своя страна не се поддържат от *racoon* „*daemon*“, защото не може динамично да запише идентификатора на областа (*scope-id*) в SPD и SAD таблиците. Поради тази причина не може да се използва IKE/ ISAKMP за обмен на ключове и се налага ръчното конфигуриране на статични ключове върху всички маршрутизатори, които ще обменят маршрутизираща информация. По друг начин казано, ключовете зададени по този начин, все едно имат безкрайно време на живот. Поради тази причина, с цел ръчното конфигуриране на машините да е по-лесно и възможно за управление, особено от

гледна точка на преносимостта и скалируемостта, е избрано при създаването на записите в SPD и SAD, с инструмента *setkey*, да се ползват мултикаст адресите. Този инструмент казва на ядрото точно кой трафик да бъде обработван в IPSec и как. Освен директивите с мултикаст адресите, има още няколко директиви, които трябва да се добавят. Те са правила, които се задават на база *Link-local* адреса на интерфейсите на маршрутизаторите. Причината за това е, че при добавяне наново на маршрутизатор към мрежата, примерно при рестартиране, се активира *Decision* процесът, но при него все още не се използват OSPFv3 мултикаст адресите. При този случай маршрутизиращата информация остава незащитена. Що се отнася до комуникацията по протокола BGP, ако се налага конфигуриране на защитена комуникация между маршрутизатори под различен административен контрол, е препоръчително да се използва удостоверяване със сертификати. Но в случая това не е необходимо и IPSec защитеното тунелиране може да се конфигурира отново с ръчно обменени ключове по подобие на мултикаст записите за OSPFv3.

Процеса на конфигуриране на IPsec, включва конфигурирането на политиките за сигурност (*security policy*), които са комбинация от SPI (*security policy index*) и ключ, който създава и валидира MD5/SHA стойността. Първият параметър SPI е произволно число, но трябва да е между 256 и 4,294,967,295 (32-бита). След това трябва да се избере типът на аутентикация и ключовия низ, които ще се ползват. AH осигурява аутентикация с MD5 или SHA1, с низ с дължина от 128 или 160 бита съответно. По-слабият алгоритъм е MD5, но и за двата трябва да се избере произволен низ със съответната дължина като ключ. Тоест за SHA1 дължината е точно 40 hex цифри, а за MD5 32. Поради причини, изложени в секция 7 на „*Authentication/Confidentiality for OSPFv3*“ (RFC4552), едно и също кодиране се ползва за SA във всяка посока между два OSPFv3 съседа, което е различно от динамичната реализация на IPsec, при която за всяка посока, SA се създава от свой собствено уникално кодиране чрез IKE [32, 33].

Понеже конфигурирането може да бъде за отделен интерфейс или за цяла OSPF област, трябва да се има в предвид желаната

сигурност. При необходимост от по-голяма сигурност трябва за всеки интерфейс да се конфигурира отделна политика. Когато е за цяла област, политиката се прилага към всички интерфейси в тази област, с изключение на интерфейсите, които си имат конфигуриран IPsec специално за тях. Веднъж конфигуриран IPsec за OSPFv3, той става невидим за потребителите [67].

Съществуват и други варианти, с които може да се реализира IPSec тунелирането в Linux среда. Всеки от тях има своите предимства и недостатъци. Едни от най-широко използваните са *racoon/ipsec-tools* и *Strongswan/Openswan*. И *Openswan* и *StrongSWAN* са произлезли от *FreeSWAN*, тоест функционалността *racoon–tool* е на 99% покрита от тях [166]. Положително при тях е, че поддържат IKEv1 и IKEv2, докато raccon само IKEv1 за договаряне на SA за IPSec протокола. Друго положително е, че не изискват администраторски права (*root* права), за да се изпълнява. Що се отнася до *racoon/ipsec-tools*, неговите положителни характеристики са свързани с това, че при излизане на нова версия се запазва обратна съвместимост с предишните, което не нарушава съществуващите конфигурации, освен това racoon/setkey са исторически наложилите се инструменти, поддържани от FreeBSD, NetBSD, Mac OSX и Linux.

В тази глава беше разгледан протокола OSPF и беше обяснено функционирането му в контекста на особеностите на версия 3, пригодена за работа в IPv6 среда. Разгледани бяха различни възможни варианти за реализация на поставената задача. Изследвани са и детайлно са описни използваните методи и средства за защита в процеса на маршрутизация – установяване на съседство, обмен на маршрути и други, както и по какъв начин ще се приложи IPSec. В следващата глава ще се представи решение, онагледяващо процеса на защита на IPv6 маршрутизация с протоколи BGP4/4+ и OSPFv3, чрез изграждане на IPSec тунели на база на избрани технологии със свободен софтуер.

5. Пета глава – Проектиране и реализиране на модел на топология, приложима за онагледяване на процеса на защита на IPv6 маршрутизация с протоколи BGP4/4+ и OSPFv3 чрез изграждане на IPSec тунели

В тази глава ще бъде представен модела на топологията, онагледяващ процеса на маршрутизация. Ще се имплементира защита, гарантираща изграждането на стабилна и защитена комуникация между маршрутизаторите. Ще се представят използваните технологии и техните особености.

5.1. Технологии ползвани за реализирането на задачата

За реализирането на поставената задача ще бъде изградена топология базирана на виртуални машини, ползващи операционна система Linux и *Open Source* (Linux) среда за маршрутизация – Zebra/Quagga. Всичко това ще бъде интегрирано с помощта на мрежовия симулатор GNS3.

Мрежовият симулатор GNS3 е софтуер, лицензиран под GPL (*General Public License*), тоест с общо право на обществено ползване. Използва се за симулация на сложни мрежови конфигурации, позволявайки максимално приближаване до начина, по който работят реални устройства в компютърните мрежи. Инсталира се и работи на традиционен хардуер за преносими компютри, като поддържа различни операционни системи като Windows, Linux и MacOS X. Най-популярното използване на GNS3 е за симулация на мрежи изградени с Cisco IOS емулатора *Dynamips*. Освен това, предоставя възможност и за изграждане на мрежи, ползващи маршрутизатори базирани на технологии с отворен код. Това се постига с интеграцията на GNS3 с VirtualBox, продукт позволяващ x86 и AMD64/Intel64 виртуализация. Благодарение на тези технологии, системни администратори и инженери, могат да тестват своите нови конфигурации и промени по мрежовата топология, преди те да

бъдат пренесени на реалните устройства. [134, 135] Използваната версия на VirtualBox е 4.3.12 и е съвместима с GNS3.

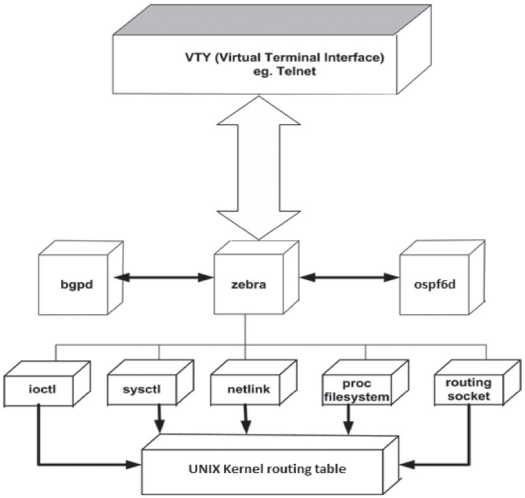
Изборът на операционна система, ползвана от маршрутизаторите инсталирани на VirtualBox е продиктуван от необходимостта успешно да се симулира топология с повече от пет виртуални машини (маршрутизатора), съобразно хардуерните възможности на хостовата машина. Такава Linux базирана операционна система е TinyCore Linux. За версия се ползва 4.7.7, от май 2013 година с версия на Linux ядро 3.0.21. Минималните изисквания за оперативна памет RAM е 64МБ. [136] За създадените виртуални машини са заделени по 128МБ, колкото са препоръчителните изисквания.
За да е възможно използването на тези виртуални машини като маршрутизатори се налага инсталирането на маршрутизиращ софтуер. За целта е избран Quagga. Версията, която се ползва е 0.99.21. По същество Quagga е софтуерен маршрутизиращ стек, чрез който се позволява имплементирането на протоколи като OSPFv3 и BGP4 за Unix платформи като FreeBSD, GNU/Linux и NetBSD.

Ако не се ползва такъв софтуер, един Unix базиран маршрутизатор може да бъде конфигуриран с помощта на стандартните команди `ifconfig` и `route`. [157] А статуса на маршрутизиращата таблица се вижда с командата `netstat`. Като за да се ползват тези команди обикновено се изискват администраторски права. При Quagga подходът е различен и много подобен както при Cisco IOS. Тук има два режима на работа. Единият е нормален, другият е т.нар. „*enable*“ режим. При първият се позволява само изпълнението на команди, при които се дава информация за системните конфигурации. А при вторият се позволява променянето и добавянето на конфигурации. Това позволява конфигуриране на Unix маршрутизаторите да става независимо от системните права на логнатия потребител.

Понеже маршрутизиращият софтуер GNU Zebra вече не се поддържа, разклонението му (*fork*), което е разработено от Kunihiro Ishiguro се явява негов заместител със свободен код.

За разлика от други маршрутизиращи софтуери като GateD, при които работи само един процес осигуряващ едновременно функционалността на всички поддържани маршрутизиращи протоколи, архитектурата на Quagga е била разработена модулно.

Използват се така наречените „*Unix daemons*“, това са процеси, които вървят на заден план без потребителска визуализация. Архитектурата на Zebra, т.е. и на Quagga позволява различните маршрутизиращи протоколи да функционират независимо като „*Unix daemons*“, свързани с главният *zebra*. Той е Zserv сървърът и се явява интерфейс към ядрото, тоест има свое така наречено Zserv API (*Application Programming Interface*). На Фигура 5.1 са показани само част от неговите имплементации: *zebra*, а *bgpd* и *ospf6d*, обслужващи на протоколите BGP4 и OSPFv3.



Фиг. 5.1 Системна архитектура на Quagga [125]

От друга страна, чрез различни системни API (множество от функции и протоколи) като *ioctl*, *sysctl*, *netlink*, *proc filesystem*, *routing socket* се осъществява връзката между Unix ядрото и процесите в потребителската област. Това позволява на *zebra* процеса да променя маршритизиращата таблица на Unix ядрото, както и да преразпределя пътища между различните маршрутизиращи протоколи. По този начин една Unix система успешно играе ролята на реален маршрутизатор, обменяйки информация за маршрути с други устройства, ползващи съответните протоколи за маршрутизация. [125, 137, 141]

Двата режима, за които беше вече казано и които ще бъдат използвани в целия процес на конфигурация са достъпни както при Zebra/Quagga, така и при Cisco IOS и се наричат „нормален“ („*normal*“ – *VTY View Mode*) и „разрешен“ („*enable*“ – VTY *Enable Mode*). Първият режим, „нормалният“ е не привилегирован и в него потребителя може да изпълнява част от командите, даващи информация за текущата конфигурация, като статистики и статус на системата. В този режим достъпа до CLI е само за четене (*read-only*). За да се излезе от него, потребителят може или да напусне цялата система или да влезне в другия „разширен“ режим. Този режим вече предоставя пълни привилегии и дава възможност за изпълняване на всички команди, включително и тези водещи до промяна на текущата конфигурация. При него достъпът до CLI е както за писане така и за четене (*read-write*). При напускане на режима, администраторът трябва или да излезе от системата или да влезне в нормалния режим. И при двата софтуерни продукта Quagga и Cisco IOS са възприети еднакви означения, отличаващи различните режима. За нормалния режим, промпта на конзолата е „>“ докато за разширения, той се променя на „#“. [137, 140] По този начин, човек лесно може да се ориентира какви права на достъп има и какви команди може да ползва. В разширения режим, администраторът може да влезне в множество конфигурационни режими, като общият се нарича глобален конфигурационен режим „*global configuration mode*“. От него може да се влезне и в други режими, конфигуриращи примерно различните маршрутизиращи протоколи като „*OSPFv3 router configuration mode*“ или „*BGP router configuration mode*“.

5.1.1. Особености при инсталацията на необходимия „open source“ маршрутизиращ софтуер

Маршрутизиращият софтуер Quagga е писан на езика за програмиране C и за инсталацията се налага сваляне на кода и компилирането му. Което включва три стъпки: конфигуриране (*configuration*) – с команда `./configure`, компилиране (*compilation* или *build*) – с команда `make` и инсталиране (*installation*) – с команда

`make install`. Що се отнася до първата стъпка Quagga има добър конфигурационен скрипт, който автоматично прави повечето конфигурации. Но при необходимост може да се променят някои, като например да се деактивира поддръжката на IPv6 или процеси, да се укаже местоположението на *log* файловете и др. При конфигурирането са използвани следните опции:

```
./configure --enable-user=root --enable-group=root
 --enable-vty-group=root --prefix=/usr/local
 --sysconfdir=/usr/local/etc/quagga --enable-vtysh
 --localstatedir=/usr/local/var/state/quagga
```

С първите три конфигурационни опции се указва поведението на Quagga „*daemons*“. С тях се конфигурират правата на потребителя и групата, с които се изпълняват *Quagga* „*daemons*“. При `--enable-vty-group` се създават *Unix Vty* сокети (използвани за *vtysh*) с определената в опцията група. Използването на *vtysh* се разрешава с опцията `--enable-vtysh`. По подразбиране изпълнимите файлове се намират в `/usr/local/sbin`, библиотеките, които ползва *Zebra* се намират в `/usr/local/lib`. [156] С опцията `--prefix=prefix` (`/usr/local`) се казва къде се инсталират архитектурно независимите файлове. Стойността на `--sysconfdir=dir`, показва директорията, в която са конфигурационните файлове. С `--localstatedir=dir` се конфигурира директорията, която zebra да използва за файловете със състоянието, като „*pid files*“ и „*unix sockets*“.

След като са направени конфигурациите, *Quagga* се компилира, за което е необходимо да има инсталиран компилатор като *gcc* и други модули. За операционната система, която е избрана за разработка на настоящата работа (TinyCore Linux), беше установено, че целия набор от необходими за компилатора инструменти може да се инсталира с `tce-load -wi compiletc` от командния ред. След компилирането, остава само да се инсталира в системата, чрез копиране на компилираните програми и съпътстващи файлове по зададените в конфигурацията места, които са `/usr/local/sbin` и `/usr/local/etc`, а *vtysh* е в `/usr/local/bin`.

Относно имплементацията на *Quagga*, разгледана в съпоставка със Cisco маршрутизатор, ще се установи една основна разлика. При Cisco маршрутизаторите системата върви на базата само на един системен „*image*“. Този единичен за маршрутизатора

конфигурационен файл се пази в NVRAM на системата. Достъпва се примерно през *telnet* или през конзолата на самото устройство. Най-основното предимство при един файл е това, че всички конфигурационни настройки са на едно място и може лесно да бъдат възстановени през TFTP, RCP или FTP. При „*open source*“ варианта, ползващ Zebra/Quagga, както вече се каза, се ползват множество процеси, взаимодействащи с UNIX ядрото, като всеки от тях има свой собствен конфигурационен файл, достъпен за потребител с администраторски права. [140] Тоест, тази модулност при Quagga, позволява по-лесно и независимо обновяване и надграждане в сравнение с Cisco IOS, където се налага при всяко такова действие да се рестартира цялата система. Още една особеност при Zebra/Quagga е, че не се предоставя и конзолен достъп. Вместо това, всеки Quagga „daemon“ процес има собствен терминален интерфейс (VTY – *Virtual TeletYpe interface*) и се достъпва чрез *telnet* настроен на специфичен порт. Това означава, че е възможно администраторът да се свърже към „daemon“ процес посредством протоколът „*telnet*“. За тази цел е необходимо на всеки VTY интерфейс да има зададена парола. В случая парола е „**quagga**“. След инсталацията на Quagga, номерата на портовете се записват в `/etc/services`, но ако порта се указва изрично при установяване на връзката с процеса, това не е задължително. Тези, които ще се ползват за общите настройки (zebra), BGPv4 и OSPFv3 са следните:

```
zebra     2601/tcp     # zebra vty
bgpd      2605/tcp     # BGPd vty
ospf6d    2606/tcp     # OSPF6d vty
```

Тази организация на процесите позволява всеки един протокол да бъде рестартиран или спрян независимо от другите.

Конфигурационните файлове на Quagga са в `/usr/local/etc/quagga/*.conf`. Такива са bgpd.conf, isisd.conf, ospf6d.conf, ospfd.conf и zebra.conf. В тях се записват VTY паролите, опции за дебъгване, конфигурации на маршрутизиращите протоколи и други. Цялата тази информация представлява инициализиращо множество от команди за маршрутизатора. Всеки един Quagga „daemon“ има свой собствен конфигурационен файл. По подразбиране името на процеса с добавено .conf образува името на конфигурационния

файл. Тоест за zebra е zebra.conf, за bgpd е bgpd.conf, а за ospf6d е ospf6d.conf. От една страна, това че всеки „daemon“ може да бъде конфигуриран по отделно през CLI (VTY) е добре, понеже при възникнал проблем при запис на една конфигурация, няма да се отрази на останалите, но от друга страна се явява и неудобство при администрирането на системата. Затова Quagga предоставя интегриран потребителски шел, наречен „vtysh“. [137, 138] Той свърза всички останали „daemon“ процеси и работи като прокси на потребителските входни данни, записвайки ги в съответните файлове. Има възможност да бъдат, също така в един общ Quagga.conf, което не се препоръчва като добра практика.

5.1.2. Конфигуриране на средата за симулация с виртуални машини

Конфигурирането на мрежовия симулатор GNS3, така че да се даде възможност за създаване на мрежова топология, ползваща „*open-source*“ маршрутизатори, е важен процес от гледна точка на практическата реализация на задачата. Тъй като целта е да се ползват продукти със свободен код хардуерния Cisco IOS емулатор Dynamips е неподходящ, поради което GNS3 трябва да се конфигурира с храдуерен емилатори и виртуализиращи технологии, които поддържат Linux машини. От възможните Qemu и VirtualBox, изборът е спрян на VirtualBox. И двете имат своите предимства и са широко използвани „*open-source*“ продукти в днешно време. Но при VirtualBox съществува едно предимство, а именно, позволява да се ползва x86 хардуерната виртуализация (на Intel VT-x или AMD-V) за различни операционни системи, като Windows, Mac OS или Linux, докато Qemu позволява само при Linux среда, чрез KVM (*Kernel-based Virtual Machine*), и то при условие, че хостовата и вируалната система ползват една и съща архитектура (примерно и двете трябва да са 64-битови ОС). Тази хардуерна виртуализация е важна, защото обезпечава производителност, сравнима с производителността на невиртулана машина. Освен това, от гледна точка на своя по-добър потребителски интерфейс (GUI) за управление на виртуалните машини, който не е необходимо

да бъде инсталиран допълнително, VirtualBox представлява едно удобно за използване и управление решение.

Проследяването на процесът на конфигурация на такъв мощен продукт, като GNS, за работа с Linux машини, представлява интерес и поради факта, че в глобалната мрежа има множество източници, показващи как да се настрои за Cisco или Juniper мрежови топологии, но много малко за „*open-source*“ решения. Причина за това е, че GNS, исторически е използван за симулация само на Cisco мрежови топологии. Той е широко използван, както в процеса на обучение, така и като инструмент в помощ мрежови администратори, искащи да експериметират с нови подобрения и промени в една изолирана и максимално реалистична среда. Тази обвързаност със Cisco е и причина GNS обикновено да бъде инсталиран и ползван под Windows ОС, където и поддръжката и самото инсталиране на продукта са по-добри и гарантират възникването на по-малко трудности.

Тъй като средата за симулация е базирана на виртуални машини, първото което трябва да се конфигурира са имано те. Самият процес на инсталацията на Linux ОС, не представлява интерес за настоящата работа. Както беше казано в началото на тази глава, изборът на операционна система е продиктуван от необходимостта успешно да се симулира топология с повече от пет виртуални машини, съобразно хардуерните възможности на хостовата машина. И тези изисквания са изпълнени от TinyCore Linux 4.7.7. При по-добри хардуерни параметри на хостовата машина и при използването на машината като истински работещ маршрутизатор в реална среда е възможно да се инсталира операционна система като CentOS или Debian, която да гарантира надеждност и сигурност на работата на „*open-source*“ маршрутизатора. Но от гледна точка на устройства като маршрутизаторите, където е необходимо високо ниво на сигурност TinyCore Linux също е много подходящ. Причината за това е, че всички промени направени докато е включена операционната система съществуват единствено в системната RAM памет и се премахват веднага щом машината се рестартира или изключи. По този начин вируси или грешки във файлове могат са бъдат премахнати просто чрез рестартиране на маршрутизатора.

Конфигурацията на VirtualBox машините, ще бъде с използването на предварително инсталиран на виртуален диск TinyCore Linux 4.7.7. Потребителското име за всички машини е „*tc*“. В него има инсталиран, по начина, по който беше описанието по-горе, Quagga 0.99.22 маршрутизиращ софтуер.

Процесът по настройка е много добре описан в документацията на официалните сайтове на VirtualBox и GNS, както и в *tutorial* на Brian Linkletter [134, 135, 139].

Първата стъпка е създаването на VirtualBox виртуална машина. Задават името на машината и типа и версията на операционната система. Размерът на заделената оперативна памет за машината е 128 МБ. От тях, по подразбиране, 90% са заделени за *rootfs*. За твърд диск се посочва, предварително създадения виртуален диск с инсталиран TinyCore Linux и Quagga. Щом се създаде, виртуалната машина се добавя в списъка със съществуващите. Виртуалните машини, които ще бъдат използвани за илюстрация на потребителски машини, се създават по аналогичен начин, но при тях се ползва виртуален твърд диск с операционна система TinyCore Linux, при който няма инсталиран Quagga маршрутизиращ софтуер.

Понеже, за реализацията на задачата, ще бъдат необходими пет маршрутизатора, останалите виртуални машини, които ще ги обслужват, се клонирани от първата, т.нар. „пълно клониране“, което прави техни собствени и независими твърди дискове, като и преинициализира всички MAC адреси на мрежовите карти.

След създаването на виртуалните машини, те трябва да се интегрират със симулатора GNS3. Това става по описания в отбелязаните източници начин. В зависимост от функциите, които ще изпълняват виртуалните машини, на маршрутизатор или хост, са зададени съответно осем мрежови интерфейса (NICs) или два за хост. Всички машини са настроени така, че да нямат достъп до външната мрежа. На машините е настроена възможност за конзолен достъп чрез *telnet* без да се ползва графична среда. Така виртуалните машини могат да бъдат достъпни единствено през конзола. По този начин виртуалните машини стават видими за използване и симулация в GNS.

За да се позволи следенето на мрежовия трафик (функцията „*Capture*“), чрез инструмента Wireshark, в конфигурацията на GNS3 трябва да се посочи точният път където е инсталиран той. Wireshark е безплатен и с отворен код анализатор на мрежови протоколи, който ще бъде използван за тестване на сигурността на трафика в следващата глава.

За да се изобразяват графично маршрутизаторите като такива, а не като хостови виртуални машини, в GNS3 може да се настроят символите, с които да се показват. Това което трябва да се направи е да се предефинира съществуващият символ за маршрутизатор, така че да бъде използван и за устройства от тип „*VirtualBox guest*“. За тези виртуални машини, които не са маршрутизатори, не е необходимо да се създава нов символ, защото такъв има по подразбиране, вече създаден „vbox_guest“ в GNS3 стандартната библиотеката за символи.

След конфигурирането на виртуалните машини и средата за симулация, вече може да се създаде топология, в която всяка една машина да бъде конфигурирана със съответните маршрутизиращи протоколи, адреси и връзки.

5.2. Представяне на модела на топология и създаване на адресна схема

В модела на топологията ще бъдат представени всички роли, които имат отношение по поставената задача за защита на процеса на IPv6 маршрутизацията с протоколи BGP4/4+ и OSPFv3 чрез изграждане на IPSec тунели. Ще бъде изградена тестова среда с минималния необходим брой маршрутизатори за симулация. Тя ще позволява конфигурирането и тестването на поставената задача и ще дава основа, така че постигнатият резултат да бъде референтен при по нататъшно разширяване на топологията в зависимост от конкретните нужди на системните администратори при изграждането по-големи мрежови топологии, ползващи тези протоколи.

5.2.1. Създаване на адресна схема

За да е възможно конфигурирането на маршрутизиращите протоколи BGP4/4+ и OSPFv3 с IPSec и тестването на тяхната работа е необходимо първо да бъде създадена адресна схема, като изчислените адреси след това да бъдат конфигурирани на съответните устройства.

Поради особеностите на IPv6 в сравнение с IPv4, процесът по създаване на адресния план също е различен. При IPv4, адресното пространство е ограничено и затова, когато организациите получат публичните си адреси, те трябва да направят своя адресен план по най-ефективния възможен начин, за да покрият нуждите както на вътрешната, така и на външната си мрежа. Докато при IPv6, клиентите получават от своя ISP (*Internet Service Provider*) обикновено адрес с префикс /48, което е 2^80 адреса. Това са толкова много адреси, че стремежа към най-ефективното разпределяне на адресите, както при IPv4 с максимално точно пресметнати необходими подмрежи и хостове в тях, би било по-скоро недостатък от колкото би донесло ползи. Освен този префикс могат да бъдат дадени и други. В RFC6177, което е в категорията най-добри практики [142], препоръчват да се раздават префикси, които отговарят по-точно на нуждите на клиентите. В действителност за домашен потребител /48 е твърде много, но /64 позволява създаването само на една подмрежа, което може да е недостатък, затова за такива потребители /56 или /60 са по-подходящи. Но за голяма организация /48 би било достатъчно ефективно.

При IPv6 адресния план може да се базира на местоположението на устройствата (*locations*), на видове според предназначението (*use types*) или на комбинация от двете. Така създадените адресни области се групират по логически най-ефективния начин. Има няколко преимущества от този подход [144]:

- по този начин се създават по-лесно политики по сигурността, като конфигурирането на *access lists* и *firewalls*
- когато се види даден адрес е по-лесно да се разбере какво е неговото местоположение и вид (*use type* или *location*)

– скалируемостта е голяма, понеже лесно могат да се добавят нови местоположения и типове

Това излишество от адреси, които не се оползотворяват напълно както при IPv4, не трябва да се считат за проблем, защото позволява, освен улеснение и по-добра ефективност. Това си проличава, примерно при маршрутизиращите таблици, където се избягва излишното увеличаване на таблицата, а адресите могат са се ползват такива каквито са.

При изчислението на префиксите, ще бъде използван адрес 2001:DB8:ABCD::/48. Този вид адрес е описан в RFC3849 „*IPv6 Address Prefix Reserved for Documentation*“. Резервирани са само за документални и тестови цели, затова администраторите трябва да ги добавят към списъка с немаршрутизируеми адреси и да се филтрират. [143] Те не се раздават от регионалните регистри (*Regional Internet Registries*). Освен този префикс, трябва да се блокират някои други префикси, дадени в Таблица 5.1.

Routes to Block	Prefixes
Default route	::/0
Unspecified address	::/128
Loopback address	::1/128
IPv4-compatible addresses	::/96
IPv4-mapped addresses	::ffff:0.0.0.0/96
Link-local addresses	fe80::/10 or longer
Site-local addresses (deprecated)	fec0::/10 or longer
Unique-local addresses	fc00::/7 or longer
Multicast addresses	ff00::/8 or longer
Documentation addresses	2001:db8::/32 or longer
6Bone addresses (deprecated)	3ffe::/16

Таблица 5.1 IPv6 префикси, които трябва да се филтрират от маршрутизатора [158]

Те не трябва да бъдат обявявани нито от локалния маршрутизатор, нито получавани от съседните. Всички тези видове префикси бяха разгледани във втора глава.

Ако адресът, вместо за документални цели, беше разпределен от съответния ISP, изчисленията биха били напълно аналогични.

Адресът, с който се започва е 2001:DB8:ABCD::/48. Както беше казано и преди, за *Host ID* са заделени 64 бита, то със лесни изчисления се извеждат броя битове, които остават свободни за използване като подмрежи. От 128-те бита на IPv6 адреса се изваждат битовете от общата мрежова част (48 бита), които не се променят никога, освен тях се изваждат и 64 бита за хост частта.

128 – 48 – 64 = 16 => остават 16 бита за подмрежи,
където от 2^16, следват 65536 подмрежи

Тези 16 бита определят интервала от IPv6 адреси:

[2001:DB8:ABCD:0000:: до 2001:DB8:ABCD:FFFF::]

Може да бъдат записани с префикса /64, като 2001:DB8:ABCD::/64, което обхваща същият интервал.

При създаването на конкретния адресен план е избрано той да се базира на видове според предназначението (*use types*). Ползите от този вид групиране на подмрежите, според вида им на използване е, че се улеснява създаването на политики за сигурността. Повечето политики при защитните стени (*firewall*) са базирани на типове потребителски устройства и тяхната употреба, а не физическото им местоположение в конкретна сграда, където е конфигурирана мрежата. Затова и *Type Based Subnet* е избраният подход.

Първото нещо, което трябва да се направи е да се определят колко и какви групи са необходими:

- боря групи, според предназначението (*Servers*, *Employees* и *Students*) – 3 групи
- групи предвидени за инфраструктурни нужди, като *backbone* и друга *infrastructure* – 1 група
- брой на групите предвидени за бъдещи нужди – 5 групи

Общо 9 групи. Трябва да се определи най-близката стойност на степените на 2, която е по-голяма от полученото число. За 9 групи, такава стойност е:

2^4 = 16 => 16 е най-близката степен на 2, която е по-голяма от 9

Това означава, че от всичките 65536 подмрежи, се взимат 16 подмрежи, които ще са заделени за типа на подмрежата. С други думи:

16 (бита заделени за подмрежа) – 4 (бита за тип „T“) = 12 (бита са свободни „S“)

Това е онагледено схематично с битове, по следния начин:

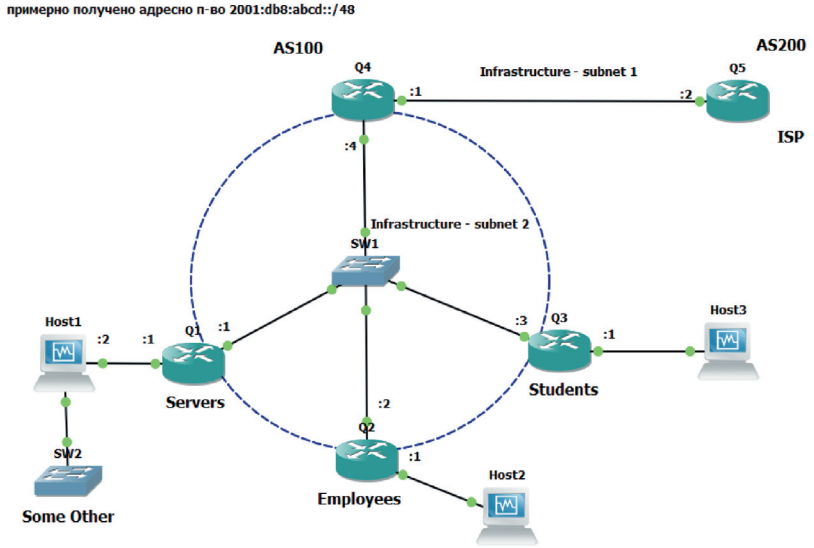
2001:DB8:ABCD:	T	T	T	T	S	S	S	S	S	S	S	S	S	S	S	S	::/64

или по-кратно записано с байтове:

2001:DB8:ABCD:	T	S	S	S	::/64

По този начин с 12 свободни бита се позволява, да има по 2^12 = 4096 подмрежи за всеки тип. След като тези изчисления са направени, адресите трябва да бъдат разпределени, за да е завършена адресната схема.

За всеки един от типовете, показани на Фигура 5.2 следва да се раздадат мрежови адреси.



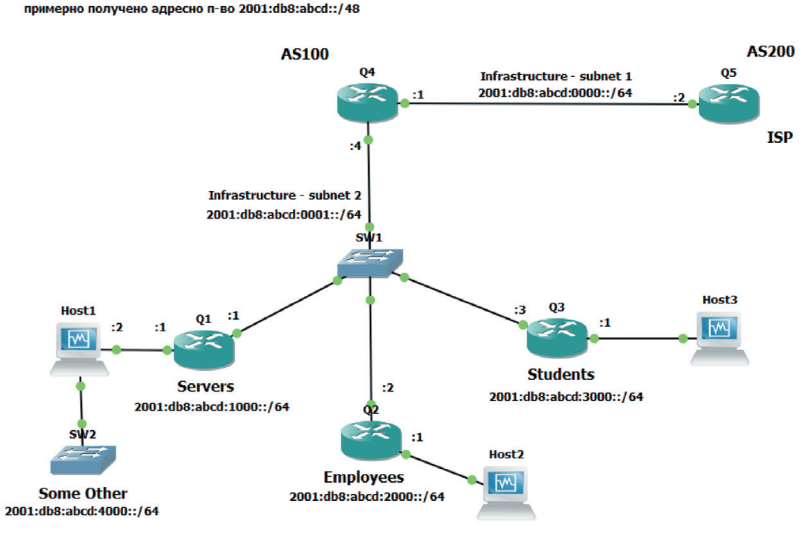
Фиг. 5.2 Мрежовата топология с изобразени типове на подмрежи

Типовете са разпределени по следния начин:

– за Infrastructure – тип 0 | 2001:DB8:ABCD: | **0** | S | S | S | ::/64
– за Servers – тип 1 | 2001:DB8:ABCD: | **1** | S | S | S | ::/64
– за Employees – тип 2 | 2001:DB8:ABCD: | **2** | S | S | S | ::/64
– за Students – тип 3 | 2001:DB8:ABCD: | **3** | S | S | S | ::/64

За нуждите на конфигурирането на машртутизиращите протоколи OSPFv3 и BGP4/4+ не са необходими допълнителни подмрежи за типовете *Servers*, *Employees* и *Students* и подмрежа „000“ е достатъчна, но за Infrastructure трябват две подмрежи. Затова окончателния вид на адресния план е:

– за Infrastructure – subnet 1 => | 2001:DB8:ABCD: | **0** | 0 | 0 | **0** | ::/64
– за Infrastructure – subnet 2 => | 2001:DB8:ABCD: | **0** | 0 | 0 | **1** | ::/64
– за Servers subnet => | 2001:DB8:ABCD: | **1** | 0 | 0 | 0 | ::/64
– за Employees subnet => | 2001:DB8:ABCD: | **2** | 0 | 0 | 0 | ::/64
– за Students subnet => | 2001:DB8:ABCD: | **3** | 0 | 0 | 0 | ::/64



Фиг. 5.3 Мрежовата топология с разпределените адреси по типове

На Фигура 5.3 е показана мрежовата топология с разпределените адреси. От допълнителните, заделени за бъдещи нужди, се разпределя адрес, от подмрежа 2001:db8:abcd:4000::/64 и на втория интерфейс на Host1. Това се прави с цел да се покажат, че с динамичните протоколи OSPFv3 и BGP4/4+, освен директно свързаните, се предават успешно и статични маршрутни пътища.

5.2.2. Представяне на видовете роли в моделната топология

На Фигура 5.3 са показани всички видове роли, участващи в процеса на маршрутизация и симулация. Маршрутизаторите Q1, Q2, Q3 и Q4 са предвидени за конфигурация и работа с маршрутизиращия протокол OSPFv3. Между тях ще се изпълни „*decision*“ процесът, ще се изберат *Designated Router* (DR) и *Backup Designated Router* (BDR). За всеки един от тях ще бъде настроено IPSec тунелиране, така че цялата мршрутизираща информация, която обменят да е криптирана и защитена от намеса на трети страни. Тези маршрутизатори ще предават както директно свързаните пътища от мрежи 2001:db8:abcd:1000::/64, 2001:db8:abcd:2000::/64 и 2001:db8:abcd:3000::/64, позволявайки „*ping*“-ване на машините Host1, Host2 и Host3 от останалите устройства, но и статичните към мрежа „*Some Other*“ 2001:db8:abcd:4000::/64.

Маршрутизаторът Q4 има двойна роля. Освен OSPFv3 протоколът, на него е конфигуриран да работи и BGP4/4+. Така той осъществява връзката между двете автономни системи AS100 и AS200 и предава пътищата научени от външния свят към устройствата от AS100 и обратно маршрутизира информация научена с OSPFv3 към маршрутизатора Q5. Политиките за сигурност настроени на Q4 са, от една страна обезпечаващи IPSec тунелирането с протокол OSPFv3 и от друга страна правят възможно тунелирането с IPSec на маршрутизираща информация получавана с протоколът за динамична маршрутизация BGP4/4+.

На маршрутизаторът Q5 е конфигуриран само протоколът BGP4/4+. Той предава информацията за пътищата, които знае (в случая ще е настроен само *default gateway* към *null0*), в посока към Q4, който от своя страна с OSPFv3 ги разпространява в AS100.

Политиките за сигурност, конфигурирани на Q5 са същите като тези на Q4, в частта отговаряща за BGP, за да е възможно изграждането на сигурен IPSec тунел между двата маршрутизатора.

5.3 Реализиране на метод за защита на процеса на маршрутизация, гарантиращ изграждането на стабилна и защитена комуникация между маршрутизаторите

Първата стъпка при конфигурирането на маршрутизаторите е добавянето на адресите на устройствата с вече създадената адресна схема. След което вече ще се конфигурират и маршрутизиращите протоколи BGP4/4+ и OSPFv3 с IPSec е необходимо да се конфигурират.

Както беше казано и по-рано, TinyCore Linux е подходящ като операционна система за устройства като маршрутизаторите, където е необходимо високо ниво на сигурност. И причината за това е, че всички промени направени докато е включена операционната система съществуват единствено в системната RAM памет и се премахват веднага щом машината се рестартира или изключи. За да се даде възможност, всички конфигурации на маршрутизаторите да се запазят, така че щом се пуснат наново, да бъдат в състояние да работят правилно, се налага да се обърне внимание на особеностите при записването на потребителски конфигурации в TinyCore Linux. Архитектурата на TinyCore Linux е показна на диаграмите „*TCZ Mount mode architecture*“, „*TCZ Copy mode architecture*“ и „*Full System TCZ/SCM architecture*“ [147, 148, 149]. При TinyCore има два основни файла `/boot/vmlinuz` и `/boot/core.gz`. В първият се намира Linux ядрото. Във вторият е файловата система на TinyCore. Когато системата се стартира, тези два файла се разархивират и се зарежат в системната памет. Конфигурационни файлове, които са създадени от потребителя се запазват в архив `/mnt/sda1/tce/mydata.tgz`. Скрипта `filetool.sh` архивира съдържанието на `/home` и `/opt` папките, както и на всички други папки описани в `/opt/.filetool.lst`, например Quagga директориите. В файла `/opt/.filetool.lst` са добавен всички

пътища до Quagga конфигурационни файлове, чрез добавянето им с вградения в TinyCore текстов редактор. Командата (скрипта), с която се запазват конфигурационните промени е:

```
$ filetool.sh -b
```

За да се възстанови записаното състояние, без да се рестартира системата, може да се ползва командата: `$ filetool.sh -r`

Всички файлове, които ще бъдат записани, могат да се видят с командата:

```
$ filetool.sh -d
```

При стартиране, TinyCore разархивира съдържанието на `/mnt/sda1/tce/mydata.tgz` в същите папки от `/opt/.filetool.lst` и в `/home` и `/opt` директориите. След като системата е стартирала се изпълнява *shell* скрипта `/opt/bootlocal.sh`. Потребителите могат да добавят команди, за да създадат инициализиращата конфигурация, която е необходима за даденото устройство, което е направено и за целите на настоящата работа. При мрежови настройки, които не са конфигурирани през Quagga, като линукс команди за IP адреси на хостовите машини, те трябва да се добавят в `/opt/bootlocal.sh` шел скрипта, за да се изпълнят при стартирането на системата, понеже `filetool.sh` не се грижи да ги архивира. [145, 146]

5.3.1 Конфигуриране на устройствата

В началото адресите, получени с изчисленията от предната подточка, са конфигурирани на устройствата. Адресният план с разпределените адреси по интерфейсите на устройствата е даден в Таблица 5.2. Адресите *Link-local* за образувани от физическия адрес на интерфейсите по метода EUI-64, описан в глава втора.

У-во	Интер-фейс	Адрес – Global IPv6	Адрес – Link-local IPv6	Мрежова маска / име
Q1	eth0	2001:db8:abcd:1::1	fe80::a00:27ff:fee6:9227	/64; Infrastructure – 2
	eth1	2001:db8:abcd:1000::1	fe80::a00:27ff:fecd:f928	/64; Servers
Q2	eth0	2001:db8:abcd:1::2	fe80::a00:27ff:fe92:bd78	/64; Infrastructure – 2
	eth1	2001:db8:abcd:2000::1	fe80::a00:27ff:fe16:942c	/64; Employees

Q3	eth0	2001:db8:abcd:1::3	fe80::a00:27ff:fe08:3b4d	/64; Infrastructure – 2
	eth1	2001:db8:abcd:3000::1	fe80::a00:27ff:fef9:2d24	/64; Students
Q4	eth0	2001:db8:abcd:1::4	fe80::a00:27ff:fea0:e5b2	/64; Infrastructure – 2
	eth1	2001:db8:abcd::1	fe80::a00:27ff:fe78:e0d1	/64; Infrastructure – 1
Q5	eth0	2001:db8:abcd::2	fe80::a00:27ff:fef4:8797	/64; Infrastructure – 1
Host1	eth0	2001:db8:abcd:1000::2	fe80::a00:27ff:fe90:7b86	/64; Servers
	eth1	2001:db8:abcd:4000::1	fe80::a00:27ff:fedc:3894	/64; Some Other
Host2	eth0	2001:db8:abcd:2000::2	fe80::a00:27ff:fea2:6d16	/64; Employees
Host3	eth0	2001:db8:abcd:3000::2	fe80::a00:27ff:fe95:e6ec	/64; Students

Таблица 5.2 Адресите присвоени на интерфейсите на устройствата

За маршрутизаторите базовите настройки са конфигурирани през Quagga. Те се записват в Quagga конфигурационния файл zebra.conf. Адресите се добавят със следните команди изпълнени в конфигурационен режим:

```
interface eth0
ipv6 address 2001:db8:abcd:1::1/64
no ipv6 nd suppress-ra
```

Първите две са аналогични за всички маршрутизатори, чиито интерфейси се конфигурират със адресите от таблицата. Последната от тях разрешава *Routing Advertisements*, да бъдат изпращани от интерфейса.

Освен това на всички маршрутизатори е разрешено препращането на пакети по протокола IPv6. Тоест разрешено е да изпълняват функциите на маршрутизатор с командата изпълнена в Quagga: `ipv6 forwarding`

Това е равносилно на задаването на стойност „1“ с командата `sysctl` за следните опции, които по подразбиране са със стойност „0“:

```
net.ipv6.conf.all.forwarding = 1
net.ipv6.conf.default.forwarding = 1
net.ipv6.conf.lo.forwarding = 1
net.ipv6.conf.dummy0.forwarding = 1
net.ipv6.conf.eth0.forwarding = 1
net.ipv6.conf.eth1.forwarding = 1
net.ipv6.conf.eth2.forwarding = 1
net.ipv6.conf.eth3.forwarding = 1
net.ipv6.conf.eth4.forwarding = 1
net.ipv6.conf.eth5.forwarding = 1
net.ipv6.conf.eth6.forwarding = 1
```

```
net.ipv6.conf.eth7.forwarding = 1
```

При хосовите машини настройките се правят с помощта на Linux командата „`ip`“ за версия 6 на протокола. Конфигурациите се добавят в файла `/opt/bootlocal.sh`, за да бъдат заредени наново и да не се изгубят след презареждане на операционната система. Тези команди се изпълняват в привилегирован режим, чрез командата sudo.

За Host1 те са следните:

```
sudo ip -6 addr add 2001:db8:abcd:1000::2/64 dev eth0
sudo ip -6 addr add 2001:db8:abcd:4000::1/64 dev eth1

sudo ip -6 route add default via 2001:db8: bcd:1000::1
```

С първите два реда се задават адресите на интерфейсите eth0 и eth1. С третата команда се задава път по подразбиране (*default gateway*), който да насочва към адреса (2001:db8:abcd:1000::1) на интерфейс eth1 на маршутизатора Q1. Пътищата по подразбиране са необходими, за да се насочват в правилата посока пакетите получени от хостовата машина. Примерно при командата *ping* изпълнена от Host3 към Host1 няма да се получи ICMP отговор, понеже Host1 не би могъл да знае накъде да предаде своето съобщение. Напълно аналогично е конфигурирането на другите два хоста, със съответните техни адреси, но без добавянето на адрес към eth1.

На всички машини е зададено хост име, чрез добавянето на реда `/usr/bin/sethostname Q1` (съответното за устройството име) във шелскрипт файла `/opt/bootsync.sh`.

С това маршрутизаторите са готови да бъдат конфигурирани с динамичните протоколи за маршрутизация OSPFv3 и BGP4/4+.

5.3.2. Конфигуриране на маршрутизаторите с OSPFv3

Всички настройки за OSPFv3 се извършват през Quagga с vtysh, а самите конфигурации се записват във файла ospf6d.conf. Всички маршрутизатори, на които ще върви OSPFv3 са настроени с минималните задължителни команди, така че да се задейства SPF алгоритъма, маршрутизаторите да обменят информация и

маршрутизиращите таблици да бъдат правилно попълнени. Затова и всички интерфейси, които ще участват в маршрутизацията са настроени към *backbone* областта 0.0.0.0. Следните редове са достатъчни, за задействането на OSPF на интерфейса eth0 на маршрутизатор Q1. Те са изпълнени в конфигурационен режим:

```
router ospf6
 router-id 10.10.10.10
 redistribute connected
 redistribute static
 interface eth0 area 0.0.0.0
```

С първият ред се влиза в режим за конфигурация на OSPFv3 протокола. Задава се идентификатор на маршрутизатора, които трябва да е различен от този на останалите. На база на него и на последователността на активиране на машините се определят DR и BDR. С следващите две директиви се казва OSPF да препредава към другите маршрутизатори директно свързаните си мрежи и тези, които са конфигурирани със статични пътища. За да се покаже това е конфигуриран един статичен път към мрежа 2001:db8:abcd:4000::/64 със следната команда изпълнена извън конфигурацията на OSPF, само в конфигурационен режим:

```
ipv6 route 2001:db8:abcd:4000::/64 2001:db8:abcd:1000::2
```

Така маршрутизаторът ще знае, че ако получи пакет предназначен за тази мрежа, ще трябва да се пренасочи към адреса на интерфейса eth0 на хостовата (тоест *next hop* интерфейса).

Настройките за Q2 и Q3 са напълно аналогични, като е сменено единствени *router-id* на машините. За Q2 то е 10.10.10.20, а за Q3 е 10.10.10.30. При Q4 е същото, но има една особеност. Понеже той ще участва, освен в OSPF комуникацията, но и в тази по протокола BGP, се налага да редитрибутира или препредава и маршрутите, които е научил от него. Затова в настройките за него е добавено следното:

```
redistribute bgp
```

По този начин в маршрутизиращите таблици на всички ще се запиват и пътищата научени от Q5.

Така направените конфигурации задействат OSPF процеса и резултата може да се провери с следните команди.

```
show ipv6  ospf6 neighbor
show ipv6  ospf6 database
show ipv6 ospf6   route
```

Резултата със снимки на информацията изведена с командите може да се види в Приложение 1.

5.3.3. Конфигуриране на маршрутизаторите с BGP4/4+

Също както при OSPF и тук всички настройки за BGP се извършват през Quagga с vtysh, но самите конфигурации се записват във файла bgpd.conf. Всички маршрутизатори, на които ще върви BGP са настроени с минималните задължителни команди, така че да се покаже, че работи маршрутизиращият процес и маршрутизиращите таблици се попълват правилно.

За маршрутизатор Q4, конфигурацията изглежда по следния начин:

```
router bgp 100
 bgp router-id 10.10.10.40
 no bgp default ipv4-unicast
 neighbor 2001:db8:abcd::2 remote-as 200

address-family ipv6
 redistribute connected
 redistribute static
 redistribute ospf6
 neighbor 2001:db8:abcd::2 activate
 neighbor 2001:db8:abcd::2 soft-reconfiguration inbound
 exit-address-family
```

Командите се изпълняват в режим за конфигурация на BGP маршрутизатор. С първата се влиза в този режим, като се оказва номерът на автономната система, за която се конфигурира, в случая AS100. Отново се задава идентификатор на маршрутизатора, като е избрано да съвпада с този конфигуриран за OSPF само за улеснение, но иначе няма изискване да съвпадат. Със следващата команда се изключва IPv4 маршрутизацията по BGP, защото иначе тя е включена по подразбиране. С последната команда от групата, се добавя адреса на съседния маршрутизатор и неговият

номер на AS200. С директивата `address-family ipv6` се влиза в IPv6 под режим за конфигуриране на BGP. Там, както и при OSPF се казва кои видове пътища известни на BGP да се препредават. В случая са конфигурирани директно свързаните, статичните и научените по протокола OSPFv3. Последното, което остава да се направи е да се активира адресът на съседа, за да започне комуникацията.

Винаги когато има промяна в политиките за маршрутизация, BGP сесията трябва да се презареди, за да започнат да бъдат използвани и маршрутизиращта таблица да се се конфигурира правилно. Има няколко начина, един от които е възможно да се направи това, като избраният тук е с командата `soft-reconfiguration inbound` зададена за съответния съсед. С нея маршрутизаторът записва копие на необработените обновления, получени то неговите съседи за бъдеща употреба. С командата `sh ipv6 bgp neighbors X:X::X:X received-routes` могат да се видят. Когато пътищата влезнат в Loc-RIB, тази временна таблица се изтрива.

При маршрутизатор Q5 е зададен в началото IPv6 път по подразбиране с командата:

```
ipv6 route ::/0 null0
```

С нея се указват, всички пакети получени от този маршрутизатор, за които не съществува друг маршрут, да се препращат към null0 (т.е. на никъде). В реална ситуация, това би бил адрес, който да извежда пакетите към глобалната мрежа, ако примерно става въпрос за доставчик на мрежови услуги – ISP (*Internet Service Provider*). В конфигурацията на BGP е казано този вид статичен маршрут да се предава към съседния маршрутизатор Q4.

```
router bgp 200
 bgp router-id 10.10.10.50
 no bgp default ipv4-unicast
 neighbor 2001:db8:abcd::1 remote-as 100

 address-family ipv6
 redistribute connected
```

```
 redistribute static
 redistribute ospf6
 neighbor 2001:db8:abcd::1 activate
 neighbor 2001:db8:abcd::1 default-originate
 neighbor 2001:db8:abcd::1 soft-reconfiguration inbound
 exit-address-family
```

Това е зададено с командата `neighbor 2001:db8:abcd::1 default-originate`. Останалите команди са аналогични на тези при Q4, като са променени съответно идентификатора на машрутизатора, автономната система и адреса на съседа.

С това комуникацията по портокала BGP между Q4 и Q5 е активирана. Това може да се провери с командите:

```
show ipv6 bgp summary
show ipv6 bgp neighbors 2001:db8:abcd::1 received-routes
show ipv6 bgp neighbors 2001:db8:abcd::1 advertised-routes
show ipv6 bgp neighbors 2001:db8:abcd::1 routes
```

Резултата със снимки на информацията, изведена с командите, може да се види в Приложение 1.

5.3.4. Конфигуриране на маршрутизаторите с IPSec тунелиране

В предишната глава, детайлно бяха описани използваните методи и средства за защита в процеса на маршрутизация – установяване на съседство, обмен на маршрути и т.н. и по какъв начин IPsec намира приложение.

Избраният за реализацията инструмент е ipsec-tools версия 0.8.0. Във връзка с описаните в предната глава съображения не е използвана функционалността за динамично договаряне на *Security Associations* с IKE, тоест компонента *racoon* не е използван. За конфигурирането е използван другият компонент на ipsec-tools, а именно инструмена *setkey*. С негова помощ са конфигурирани записите в *Security Policy Database* (SPD) и *Security Association Database* (SAD) в ядрото на операционната система, за да може то да определи кой точно трафик да бъде обработван в IPSec и по какъв начин.

Предварително е създаден файла setkey.conf в папката `/etc/racoon`. В него се конфигурират IPSec политиките, като посока на комуникация, участници и вид на комуникацията. Освен това са дефинирани и статичните SA с съответните генерирани ключове. Този файл се чете от инструмента *setkey*. За да не се изгуби съдържанието на файла след рестартиране, пътя до него `/etc/racoon` е добавен в списъка с пътища `/opt/.filetool.lst`, който трябва да влезнат в архива *mydata.tgz*.

За OSPFv3 маршрутизаторите Q1, Q2 и Q3, конфигурационния файл setkey.conf е следният:

```
flush;
spdflush;

spdadd ::/0 ff02::5 any -P in ipsec esp/transport//require;
spdadd ::/0 ff02::5 any -P out ipsec esp/transport//require;
spdadd ::/0 ff02::6 any -P in ipsec esp/transport//require;
spdadd ::/0 ff02::6 any -P out ipsec esp/transport//require;

add :: ff02::5 esp 175002505 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: ff02::6 esp 175002506 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
```

Първите две декларации премахват всички направени до момента и позволяват следващите да се създадат на чисто.

Следващите четири записа започващи с декларацията `spdadd` създават политики насочени към входящия (*in*) и изходящия (*out*) трафик насочен към мултикаст групите ff02::5 и ff02::6. Како беше казано, първата се използва за изпращане на LSA съобщенията до всички други маршрутизатори (OSPFv3), втората се използва от всички маршрутизатори, които не са „назначени“, да изпращат LSA съобщенията до DR и BDR. С защитата на трафика от тези адреси се защитава и комуникацията съдържаща маршрутизираща информация между OSPFv3 маршрутизаторите. С опцията *require* се казва, че правилата са със задължителен характер. В случая

е избрано IPSec да бъде конфигуриран в транспортен режим. Опцията *any* е, за да укаже, че не е конкретизиран протокол, а важи за всички пакети, дошли или изпратени от където и да е (мрежа ::/:).

От друга страна с директивата `add` се създават ръчно SA. За криптиране на трафика, понеже е настроено да се ползва IPSec с ESP, е избран каптиращ алгоритъм 3DES (192 бита). Удостоверяването на комуникацията се прави с алгоритъма HMAC-SHA1 (160 бита). Ключовете са създадени по начина описани в [32], с командата:

```
dd if=/dev/urandom count=1024 | sha1sum
```

Този ред генерира произволен SHA1 ключ. По същият начин, лесно с добавянето на още осем произволни шестнайсетични цифри се създава и ключа за 3DES.

Адрес на източника е зададен да е всеки ::, а адрес на получателя е съответно ff02::5 или ff02::6. Параметъра SPI (*security policy index*), както беше казано и в предишната глава е произволно число, в интервала 256 и 4,294,967,295 (2^32-1).

По този начин се изгражда *end-to-end* криптирана комуникация в транспортен режим, между OSPFv3 маршрутизаторите.

Освен показните и описани IPSec конфигурации има още няколко директиви, които трябва да се добавят. Те са правила, които се задават на база *Link-local* адреса на интерфейсите на маршрутизаторите. Причината за това е, че при добавяне наново на маршрутизатор към мрежата, примерно при рестартиране, се активира *Decision* процесът, но при него все още не се използват OSPFv3 мултикаст адресите. При този случай маршрутизиращата информация остава незащитена. Схематично показано, новите правила трябва да изглеждат по следния начин, а целите конфигурации могат да се видят в Приложение 2. Където има Q1-Q4 се замества с *Link-local* адреса на съответния и интерфейс.

```
Q1 -> fe80::a00:27ff:fee6:9227
Q2 -> fe80::a00:27ff:fe92:bd78
Q3 -> fe80::a00:27ff:fe08:3b4d
Q4 -> fe80::a00:27ff:fea0:e5b2
```

```
За Q1:
   spdadd Q1 Q2 any -P out ipsec esp/transport//require;
   spdadd Q2 Q1 any -P in ipsec esp/transport//require;
   spdadd Q1 Q3 any -P out ipsec esp/transport//require;
   spdadd Q3 Q1 any -P in ipsec esp/transport//require;
   spdadd Q1 Q4 any -P out ipsec esp/transport//require;
   spdadd Q4 Q1 any -P in ipsec esp/transport//require;
За Q2:
   spdadd Q2 Q1 any -P out ipsec esp/transport//require;
   spdadd Q1 Q2 any -P in ipsec esp/transport//require;
   spdadd Q2 Q3 any -P out ipsec esp/transport//require;
   spdadd Q3 Q2 any -P in ipsec esp/transport//require;
   spdadd Q2 Q4 any -P out ipsec esp/transport//require;
   spdadd Q4 Q2 any -P in ipsec esp/transport//require;
За Q3:
   spdadd Q3 Q2 any -P out ipsec esp/transport//require;
   spdadd Q2 Q3 any -P in ipsec esp/transport//require;
   spdadd Q3 Q1 any -P out ipsec esp/transport//require;
   spdadd Q1 Q3 any -P in ipsec esp/transport//require;
   spdadd Q3 Q4 any -P out ipsec esp/transport//require;
   spdadd Q4 Q3 any -P in ipsec esp/transport//require;
За Q4:
   spdadd Q4 Q2 any -P out ipsec esp/transport//require;
   spdadd Q2 Q4 any -P in ipsec esp/transport//require;
   spdadd Q4 Q3 any -P out ipsec esp/transport//require;
   spdadd Q3 Q4 any -P in ipsec esp/transport//require;
   spdadd Q4 Q1 any -P out ipsec esp/transport//require;
   spdadd Q1 Q4 any -P in ipsec esp/transport//require;
add :: Q1 esp 1751 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c3
   77a18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc
   2f;
add :: Q2 esp 1752 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c3
   77a18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: Q3 esp 1753 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c3
   77a18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc
   2f;
add :: Q4 esp 1754 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c3
   77a18f8f
```

```
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc
   2f;
```

За Q4 и Q5 трябва да се създадат и правила отговарящи за BGP комуникацията. Маршрутизаторът Q4 е конфигуриран и с двата, тоест описаните правила за Q1, Q2 и Q3 важат и за Q4, но е добавено и необходимото за защита на трафика предаван с протокола BGP.

Конфигурацията на IPSec за BGP е следната:

```
flush;
spdflush;
spdadd ::/0 2001:db8:abcd::/64 tcp -P in ipsec esp/
transport//require;
spdadd ::/0 2001:db8:abcd::/64 tcp -P out ipsec esp/
transport//require;

add :: 2001:db8:abcd::1 esp 17171 -m transport
   -E 3des-cbc 0xb4d1e75ca5c8f5552851534d7f6de467ca2e5fc1
   aaf4692f
   -A hmac-sha1 0x495fca248750e79937b4b6b9dadbb43e0e7be3
   82;
add :: 2001:db8:abcd::2 esp 17172 -m transport
   -E 3des-cbc 0xb4d1e75ca5c8f5552851534d7f6de467ca2e5fc1
   aaf4692f
   -A hmac-sha1 0x495fca248750e79937b4b6b9dadbb43e0e7be3
   82;
```

Политиките за сигурност зададени за BGP са описани с директивата `spdadd`. Както и при OSPF, интервалът включващ източниците е ::/, тоест „от кой и да е източник“. Интервалът на получателите, тук не е мутикст група, а самата мрежа към която принадлежат Q4 и Q5. В този случай е зададен и конкретен протокол, по който да се приложи IPSec, а именно TCP, използван от BGP като транспортен протокол. Останалите опции са същите и тук. Що се отнася до *Security Association Database* записите, както и при OSPF и тук са зададени същите алгоритми. Адреса на източника също е както и там. Разликата е в адресите за получател, които са адресите от интерфейсите на Q4 и Q5 участващи в мрежата 2001:db8:abcd::/64.

За да станат активни конфигурациите трябва да се изпълни командата:

```
sudo setkey -f /etc/racoon/setkey.conf
```

За да бъдат налични всеки път, когато се стартира маршрутизаторът, в TinyCore Linux операционната система, на всяка от машините трябва същата да се добави в шел файла `/opt/bootlocal.sh`.

Резултата от конфигурирането може да се види с командата:

```
sudo setkey -D
```

Ако тя бъде изпълнена на маршрутизатора ще могат да се видя всички записи в таблицата *Security Association Database*.

С това конфигурирането на протоколите OSPFv3 и BGP4/4+ с IPSec тунелиране завърши. Процесът на маршрутизация, установяване на съседство и обмена на маршрути е криптиран и удостоверен. В следващата глава ще бъдат проведени тестове с мрежовия анализатор Wireshark, за да се покаже, че в действителност потока от информация е защитен.

6. Шеста глава – Анализ и оценка на реализираната защитена среда за маршрутизация

В тази глава в мрежата от маршрутизатори с изградени IPSec тунели между тях ще се проведат тестове за сигурността в процеса на маршрутизация по BGP4/4+ и OSPFv3. За целта ще бъде използван мрежовият анализатор Wireshark. С негова помощ ще е възможно да се види потокът от информация, предаван по реализираната топология, което ще даде възможност за анализ и оценка на ефективността на защитата от *IP Spoofing* и други атаки, свързани с маршрутизацията.

6.1 Съпоставка между маршрутизациите в не защитена и защитена среда

В следващите подточки ще се направи съпоставка между двата режима на мршрутизация, защитен и не защитен. Ще се анализира предавания поток от информация, изходящ или входящ, през интерфейсите на маршрутизаторите, участващи в комуникацията по протоколи OSPFv3 и BGP4/4+.

6.1.1 Маршрутизация в незащитена с IPSec среда

Маршрутизацята в незащитена с IPSec среда може да бъде видима от всеки един, който има на разположение мрежов анализатор и е получил по някакъв начин достъп до преносната мрежа.

Както може да се види на Фигура 6.1, всички OSPF пакети на маршрутизатора Q1 – тези изпратени към мултикаст адрес и другите изпратени към съседния (в случая Q3) *Link-local* адрес по време на *Decision* процесът са свободно видими.

Filter: ospf | Expression... | Clear | Apply | Save

No.	Time	Source	Destination	Protocol	Length	Info
25	38.0756370	fe80::a00:27ff:fee6:9227	ff02::5	OSPF	94	Hello Packet
27	43.6813690	fe80::a00:27ff:fe08:3b4d	ff02::5	OSPF	94	Hello Packet
29	48.0787950	fe80::a00:27ff:fee6:9227	ff02::5	OSPF	94	Hello Packet
32	53.6849320	fe80::a00:27ff:fe08:3b4d	ff02::5	OSPF	94	Hello Packet
35	58.0798020	fe80::a00:27ff:fee6:9227	ff02::5	OSPF	94	Hello Packet
36	58.0805240	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fe08:3b4d	OSPF	82	DB Description
38	63.0773870	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fe08:3b4d	OSPF	82	DB Description
41	63.6859010	fe80::a00:27ff:fe08:3b4d	ff02::5	OSPF	94	Hello Packet
44	68.0827180	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fe08:3b4d	OSPF	82	DB Description
45	68.0836070	fe80::a00:27ff:fee6:9227	ff02::5	OSPF	94	Hello Packet
46	73.0883110	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fe08:3b4d	OSPF	82	DB Description
47	73.6771480	fe80::a00:27ff:fe08:3b4d	fe80::a00:27ff:fee6:9227	OSPF	82	DB Description
48	73.6805410	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fe08:3b4d	OSPF	182	DB Description
50	73.6842930	fe80::a00:27ff:fe08:3b4d	fe80::a00:27ff:fee6:9227	OSPF	130	LS Request
51	73.6860050	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fe08:3b4d	OSPF	282	LS Update
52	73.6865640	fe80::a00:27ff:fe08:3b4d	fe80::a00:27ff:fee6:9227	OSPF	162	DB Description
53	73.6867760	fe80::a00:27ff:fe08:3b4d	ff02::5	OSPF	94	Hello Packet
54	73.6878540	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fe08:3b4d	OSPF	118	LS Request
55	73.6888640	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fe08:3b4d	OSPF	82	DB Description
56	73.6926360	fe80::a00:27ff:fe08:3b4d	fe80::a00:27ff:fee6:9227	OSPF	246	LS Update
58	73.6983910	fe80::a00:27ff:fee6:9227	ff02::5	OSPF	158	LS Update
59	73.7003850	fe80::a00:27ff:fe08:3b4d	ff02::5	OSPF	234	LS Update
62	76.6942140	fe80::a00:27ff:fe08:3b4d	ff02::5	OSPF	190	LS Acknowledge
63	76.6967240	fe80::a00:27ff:fee6:9227	ff02::5	OSPF	210	LS Acknowledge
65	78.0862260	fe80::a00:27ff:fee6:9227	ff02::5	OSPF	94	Hello Packet
68	83.6947860	fe80::a00:27ff:fe08:3b4d	ff02::5	OSPF	94	Hello Packet
69	88.0911250	fe80::a00:27ff:fee6:9227	ff02::5	OSPF	94	Hello Packet

Фигура 6.1 Незащитени OSPFv3 пакети изпратени от Q1 и филтрирани чрез Wireshark

За всеки един от тях може да се види цялата пренасяна информация, както *Header* частта, така и в тялото на пакета. За илюстрация са избрани *OSPF Link State Request* и *LS Update Packet*. На Фигура 6.2 и Фигура 6.3 е показана пренасяната информация от тях. В чист текст са видими всички OSPF LSA съобщения, които бяха разгледани в четвърта глава.

```
⊟ Open Shortest Path First
  ⊟ OSPF Header
      Version: 3
      Message Type: LS Request (3)
      Packet Length: 28
      Source OSPF Router: 10.10.10.40 (10.10.10.40)
      Area ID: 0.0.0.0 (0.0.0.0) (Backbone)
      Checksum: 0xdda5 [correct]
      Instance ID: IPv6 unicast AF (0)
      Reserved: 00
  ⊟ Link State Request
      Reserved: 0
      LS Type: Intra-Area-Prefix-LSA (0x2009)
      Link State ID: 0.0.0.0
      Advertising Router: 10.10.10.30 (10.10.10.30)
```

Фигура 6.2 *OSPF Link State Request*, показан чрез Wireshark

```
⊟ Open Shortest Path First
  ⊟ OSPF Header
      Version: 3
      Message Type: LS Update (4)
      Packet Length: 228
      Source OSPF Router: 10.10.10.10 (10.10.10.10)
      Area ID: 0.0.0.0 (0.0.0.0) (Backbone)
      Checksum: 0xd623 [correct]
      Instance ID: IPv6 unicast AF (0)
      Reserved: 00
  ⊟ LS Update Packet
      Number of LSAs: 5
    ⊟ Link-LSA (Type: 0x0008)
        LS Age: 57 seconds
        Do Not Age: False
        LS Type: Link-LSA (0x0008)
        Link State ID: 0.0.0.3
        Advertising Router: 10.10.10.10 (10.10.10.10)
        LS Sequence Number: 0x80000002
        LS Checksum: 0xd634
        Length: 56
        Router Priority: 1
      ⊞ Options: 0x000013 (R, E, V6)
        Link-local Interface Address: fe80::a00:27ff:fee6:9227
        # prefixes: 1
        PrefixLength: 64
      ⊞ PrefixOptions: 0x00
        Reserved: 0
        Address Prefix: 2001:db8:abcd:1::
    ⊟ Intra-Area-Prefix-LSA (Type: 0x2009)
        LS Age: 57 seconds
        Do Not Age: False
        LS Type: Intra-Area-Prefix-LSA (0x2009)
        Link State ID: 0.0.0.0
        Advertising Router: 10.10.10.10 (10.10.10.10)
        LS Sequence Number: 0x80000001
        LS Checksum: 0xf61d
        Length: 44
        # prefixes: 1
        Referenced LS type 0x2001 (Router-LSA)
        Referenced Link State ID: 0.0.0.0
        Referenced Advertising Router: 10.10.10.10
        PrefixLength: 64
      ⊞ PrefixOptions: 0x00
        Metric: 1
        Address Prefix: 2001:db8:abcd:1::
    ⊟ AS-External-LSA (Type: 0x4005)
```

Както се вижда от примерите, предава се лесно видима информация за области, към които пролежат интерфейсите с OSPFv3, идентификатори на маршрутизаторите, адресните префикси, които са зададени. От друга страна *OSPF Hello Packet* пренася информация за това какви опции са зададени, кои са управляващи маршрутизатори (DR, BDR), кои са активни съседи и др.

Фигура 6.3 *LS Update Packet*, показан чрез Wireshark

На база на всичката тази информация може да се реконструира цялата топология от OSPF маршрутизатори и да се избере оптимален начин за атакуване на мрежата. Примерно, ако атакуващият научи кои са управляващите маршрутизатори (DR, BDR), може да приложи DoS атака към тях, да сложи на своята машина най-висок номер за *Router-ID* и при задействане на *Decision* процеса да бъде избран за управляващ маршрутизатор в топологията. Най-големият проблем в случая, обаче си остава липсата на механизъм за удостоверяване. Така без IPSec, когато няма налично и никакво удостоверяване за произхода на съобщенията, няма и защита от *Replay* и *IP Spoofing* атаки.

При BGP4/4+ потокът от информация също е достъпен в явен текст, ако не бъде приложен метода за защита с IPSec. Пакетите предавани по BGP между маршрутизаторите Q4 и Q5 са показани на Фигура 6.4.

Filter: bgp | Expression... | Clear | Apply | Save

No.	Time	Source	Destination	Protocol	Length	Info
34	25.0469240	2001:db8:abcd::2	2001:db8:abcd::1	BGP	139	OPEN Message
36	25.0496640	2001:db8:abcd::1	2001:db8:abcd::2	BGP	158	OPEN Message, KEEPALIVE Message
38	25.0514320	2001:db8:abcd::2	2001:db8:abcd::1	BGP	202	KEEPALIVE Message, KEEPALIVE Message, UPDATE Message
39	25.0526180	2001:db8:abcd::1	2001:db8:abcd::2	BGP	105	KEEPALIVE Message
41	26.0538030	2001:db8:abcd::2	2001:db8:abcd::1	BGP	179	UPDATE Message
42	26.0547860	2001:db8:abcd::1	2001:db8:abcd::2	BGP	272	UPDATE Message, UPDATE Message
52	56.0662180	2001:db8:abcd::1	2001:db8:abcd::2	BGP	179	UPDATE Message
60	85.0528670	2001:db8:abcd::2	2001:db8:abcd::1	BGP	105	KEEPALIVE Message
61	85.0582060	2001:db8:abcd::1	2001:db8:abcd::2	BGP	105	KEEPALIVE Message
81	145.059785	2001:db8:abcd::2	2001:db8:abcd::1	BGP	105	KEEPALIVE Message
82	145.064160	2001:db8:abcd::1	2001:db8:abcd::2	BGP	105	KEEPALIVE Message
102	205.068054	2001:db8:abcd::2	2001:db8:abcd::1	BGP	105	KEEPALIVE Message
103	205.072442	2001:db8:abcd::1	2001:db8:abcd::2	BGP	105	KEEPALIVE Message
125	265.075433	2001:db8:abcd::2	2001:db8:abcd::1	BGP	105	KEEPALIVE Message

Фигура 6.4 Незащитени BGP4/4+ пакети изпратени от Q4
и филтрирани чрез Wireshark

С помощта на Wireshark може, за всеки пакет на протокола BGP, да се види всичката предавана информация, което е видимо и от примера на Фигура 6.5. Показано е BGP UPDATE съобщение и транспортният TCP протокол, който служи за установяване на връзка и пренос.

```
⊞ Frame 41: 179 bytes on wire (1432 bits), 179 bytes captured (1432 bits) on interface 0
⊞ Ethernet II, Src: CadmusCo_f4:87:97 (08:00:27:f4:87:97), Dst: CadmusCo_78:e0:d1 (08:00:27:78:e0:d1)
⊞ Internet Protocol Version 6, Src: 2001:db8:abcd::2 (2001:db8:abcd::2), Dst: 2001:db8:abcd::1 (2001:db8:abcd::1)
⊟ Transmission Control Protocol, Src Port: 56297 (56297), Dst Port: 179 (179), Seq: 170, Ack: 92, Len: 93
    Source Port: 56297 (56297)
    Destination Port: 179 (179)
    [Stream index: 1]
    [TCP Segment Len: 93]
    Sequence number: 170    (relative sequence number)
    [Next sequence number: 263    (relative sequence number)]
    Acknowledgment number: 92    (relative ack number)
    Header Length: 32 bytes
  ⊞ .... 0000 0001 1000 = Flags: 0x018 (PSH, ACK)
    Window size value: 3600
    [Calculated window size: 14400]
    [Window size scaling factor: 4]
  ⊞ Checksum: 0xfba0 [validation disabled]
    Urgent pointer: 0
  ⊞ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ⊞ [SEQ/ACK analysis]
⊟ Border Gateway Protocol - UPDATE Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 93
    Type: UPDATE Message (2)
    Withdrawn Routes Length: 0
    Total Path Attribute Length: 70
  ⊟ Path attributes
    ⊞ Path Attribut - ORIGIN: INCOMPLETE
    ⊞ Path Attribut - AS_PATH: 200
    ⊞ Path Attribut - MULTI_EXIT_DISC: 1
    ⊟ Path Attribut - MP_REACH_NLRI
      ⊞ Flags: 0x80: Optional, Non-transitive, Complete
        Type Code: MP_REACH_NLRI (14)
        Length: 46
        Address family: IPv6 (2)
        Subsequent address family identifier: Unicast (1)
      ⊟ Next hop network address (32 bytes)
          Next hop: 2001:db8:abcd::2 (16)
          Next hop: fe80::a00:27ff:fef4:8797 (16)
        Subnetwork points of attachment: 0
      ⊟ Network layer reachability information (9 bytes)
        ⊟ 2001:db8:abcd::/64
            MP Reach NLRI prefix length: 64
            MP Reach NLRI prefix: 2001:db8:abcd::
```

Фигура 6.5 *Border Gateway Protocol – UPDATE Message*, показан чрез Wireshark

Вижда се информацията за това, кои мрежи са достъпни с техните адреси. Това дава потенциална възможност за неправомерно прочитане и дори промяна на стойностите на маршрутните атрибути (*Path Attribute*), чиито функции бяха обяснени в трета глава.

Но най-големият проблем и тук е липсата на защитено удостоверяване между комуникиращите маршрутизатори.

6.1.2 Маршрутизация в защитена с IPSec среда

За защитата на предавания OSPFv3 трафик, беше нужно създаването на два вида правила. Едното да защитава OSPFv3 трафика по *Link-local* адресите, а другото по мултикаст адресите. На Фигура 6.6 е даден примерен отрязък от предаваната информация по време на *Decision* прасецът, при който част от OSPF съобщенията остава незащитен.

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
75367	126690.800	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fea0:e5b2	OSPF	118	LS Update
75368	126690.801	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fea0:e5b2	OSPF	274	LS Request
75369	126690.801	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fea0:e5b2	OSPF	82	DB Description
75370	126690.802	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fe08:3b4d	OSPF	118	LS Update
75371	126690.802	fe80::a00:27ff:fea0:e5b2	ff02::5	ESP	154	ESP (SPI=0x0a6e5389)
75372	126690.806	fe80::a00:27ff:fea0:e5b2	fe80::a00:27ff:fee6:9227	OSPF	754	LS Update
75373	126690.806	fe80::a00:27ff:fe08:3b4d	fe80::a00:27ff:fee6:9227	OSPF	90	LS Acknowledge
75374	126690.806	fe80::a00:27ff:fea0:e5b2	ff02::5	ESP	146	ESP (SPI=0x0a6e5389)
75375	126690.809	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fe08:3b4d	OSPF	274	LS Request
75376	126690.809	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fe08:3b4d	OSPF	82	DB Description
75377	126690.810	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fea0:e5b2	OSPF	90	LS Acknowledge
75378	126690.816	fe80::a00:27ff:fe08:3b4d	fe80::a00:27ff:fee6:9227	OSPF	794	LS Update
75379	126690.819	fe80::a00:27ff:fee6:9227	ff02::6	ESP	314	ESP (SPI=0x0a6e538a)
75380	126690.821	fe80::a00:27ff:fee6:9227	ff02::6	ESP	194	ESP (SPI=0x0a6e538a)
75381	126690.827	fe80::a00:27ff:fea0:e5b2	ff02::5	ESP	314	ESP (SPI=0x0a6e5389)

Фигура 6.6 Защитени с IPSec, OSPFv3 пакети изпратени към мултикаст адресите и незащитени към *Link-local* адресите на съседните

На следващата Фигура 6.7 вече всичкият OSPFv3 трафик е защитен с IPSec, като зададения алгоритъма за аутентикация (удостоверяване) е SHA-1, а за криптиране 3DES. Така вече няма ефективен начин да бъде прочетена предаваната информация или да бъде подменена.

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
110644	174464.559	fe80::a00:27ff:fee6:9227	fe80::a00:27ff:fe92:bd78	ESP	122	ESP (SPI=0x000006d8)
110645	174464.564	fe80::a00:27ff:fe92:bd78	fe80::a00:27ff:fee6:9227	ESP	114	ESP (SPI=0x000006d7)
110646	174464.574	fe80::a00:27ff:fea0:e5b2	ff02::5	ESP	234	ESP (SPI=0x0a6e5389)
110647	174464.575	fe80::a00:27ff:fe92:bd78	ff02::6	ESP	146	ESP (SPI=0x0a6e538a)
110648	174464.579	fe80::a00:27ff:fea0:e5b2	ff02::5	ESP	186	ESP (SPI=0x0a6e5389)
110649	174464.585	fe80::a00:27ff:fe08:3b4d	fe80::a00:27ff:fee6:9227	ESP	282	ESP (SPI=0x000006d7)
110650	174464.617	fe80::a00:27ff:fe92:bd78	ff02::5	ESP	138	ESP (SPI=0x0a6e5389)

Фигура 6.7 Защитени с IPSec, OSPFv3 пакети изпратени към мултикаст адресите и към *Link-local* адресите на съседните

Тук, когато се отвори някой от защитените пакети, единствената информация която се вижда, независимо от типа, който е в оригинала, е SPI идекса и поредния номер, както е показано на Фигура 6.8.

```
⊞ Frame 69: 242 bytes on wire (1936 bits), 242 bytes captured (1936 bits) on interface 0
⊞ Ethernet II, Src: CadmusCo_e6:92:27 (08:00:27:e6:92:27), Dst: IPv6mcast_05 (33:33:00:00:00:05)
⊞ Internet Protocol Version 6, Src: fe80::a00:27ff:fee6:9227 (fe80::a00:27ff:fee6:9227), Dst: ff02::5 (ff02::5)
⊟ Encapsulating Security Payload
    ESP SPI: 0x0a6e5389 (175002505)
    ESP Sequence: 7
```

Фигура 6.8 Съдържание на тунелиран през IPSec пакет

При BGP4/4+ нещата изглеждат по-доста подобен начин. Разликата е, че адресите, които се използват са зададените глобални адреси на интерфейсите на Q4 и Q5 по които върви BGP трафика. Това е показано на Фигура 6.9.

```
Filter:                                   Expression...  Clear  Apply  Save
No.   Time        Source                  Destination             Protocol  Length  Info
2215  43621.07700 2001:db8:abcd::2        2001:db8:abcd::1        ESP       122     ESP (SPI=0x00004313)
2216  43622.09000 2001:db8:abcd::1        2001:db8:abcd::2        ESP       402     ESP (SPI=0x00004314)
2217  43622.09100 2001:db8:abcd::2        2001:db8:abcd::1        ESP       122     ESP (SPI=0x00004313)
```

Фигура 6.9 Защитени с IPSec, BGP4/4+ пакети изпратени към глобалните на Q4 и Q5

От всички тези извадки от маршрутизиращия трафик ясно се виждат разликите между защитената и незащитената машрутизираща комуникация. Като най-съществената от тях е, не само че трафика е криптиран, но по-важното е, че се дава възможност за удостоверяване на източника на предаваната информация. Другото важно следствие от тунелирането е, че комуникацията с приложен IPSec е защитата от *Replay* атаки, което се реализира с помощта на *ESP Sequence* полето, отчитащо поредността на предаваните пакети.

6.2 Оценка на реализираната защитена среда за маршрутизация

Изграждането на защитена среда за комуникация и маршрутизация и защитата на чувствителна информация в една автономна система е от съществена важност. По същество *IP Spoofing* означава източника на изпратеното съобщение да е

различен от този, за който се представя. Липсата на сигурен метод за удостоверяване на истинския източник води до DDoS атаки, неоторизиран достъп до информация или в случая с BGP до блокиране на глобалната мрежа с предаване на лъжливи обновления. Причината, поради която *IP Spoofing* е възможен, е защото маршрутизаторите, без допълнителни настройки, проверяват само адреса на получателя, за да разберат на къде да препратят съобщението. Адресът на източника не се взема в предвид и дори и да е подменен няма да попречи на съобщението да се предаде към съответния получател. Променен адрес на получателя (с *IP Spoofing*) е най-лесният начин за компроментиране на една система. Тази техника взима участие в различни видове атаки. Една известна такава атака е т. нар. *Smurf* атака [167]. Но DDoS атака може да се предизвика с наводняване с различни видове съобщения, каквото примерно са LSA съобщенията на OSPF. При такива атаки се разчита, че атакуващият има възможност да изпраща LSA съобщения към другите маршрутизатори и те от своя страна да обработят правилно тези съобщения. Но това не означава, че проблемът е в OSPF протокола, както и преди беше казано, функциите за които е бил създаден ги изпълнява напълно коректно [169]. С добавянето на IPSec защита на предавания трафик, вече не е така лесно външна машина да се включи в процесът на маршрутизация, понеже атакуващият, освен адресът на източника, трябва да знае и конфигурираните статични ключове, които се ползват за криптиране и удостоверяване. Само в такъв случай ще може да бъде част от маршрутизацията и ще може да изпраща LSA, които да бъдат обработвани от останалите маршрутизатори. В това отношение IPSec като част от протокола IPv6, използван от последната версия на OSPFv3, се явява едно сигурно решение за целите на маршрутизацията по този протокол.

Но като цяло защитата на маршрутизацията е много сложен процес. За протокола за външна маршрутизация BGP това важи с още по-голяма сила. Докато всеки, в глобалната мрежа, не се удостоверява пред всеки и докато всички невалидни пътища не бъдат филтрирани, което на практика е невъзможно, ще съществуват реални възможности за случайни или целенасочени

атаки като тези описани в първа глава. За този етап решения като реализираното в настоящата работа могат да бъдат ефективни, но не и абсолютно, на сто процента, щом има замесени различни зони на мрежова администрация под различен административен контрол. Най-ефективното решение на този етап е съществуването на компании като *BGPmon Network Solutions* и *Renesys* (от миналата година собственост на *Dyn* [168]), занимаващи се с кибер сигурност и специализирани в мониторинг и следене състоянието на интернет пътищата по света, които да сигнализират, всички заинтересовани, мрежови доставчици или държавни институции, когато засекат нередност.

Това, което могат да направят системните администратори е винаги преди да пуснат в действие ново решение за сигурността, да проведат тестове в среда максимално близа до реалната. Виртуалната среда за симулация (с Linux, Quagga и ipsec-tools), на която бяха проведени тестовете в тази глава, помогна да се види пропуск в защитата на OSPF протокола с IPSec. В първоначалния вариант политиките за сигурност разчитаха единствено на правилата обвързани с мултикаст адресите. Но с помощта на мрежовия анализатор, лесно се видя, че при рестартиране на някои от маршрутизаторите, OSPF съобщенията по време на *Decicion* процеса не се защитават. Това в една реална среда може да отнеме дори и години, за да се открие. В случая става въпрос за незначителен пропуск, но при по-сложни конфигурации с десетки маршрутизатори и национали или корпоративни правила за сигурност може да доведе не само до сериозни материални и финансови щети, но и до изтичане на секретна или класифицирана информация.

7. Седма глава – Заключение

7.1 Постигнати резултати

В настоящата работа беше предложено решение на задачата за защитено маршрутизиране, базирано на технологии с отворен код или лицензирани с право на обществено ползване. Беше направен анализ на новите версии на протоколите IPv6, OSPFv3 и BGP4/4+ и техните методи за защита на процеса на маршрутизация. Изградена беше мрежова среда за конкретната задача със възможности за симулация в реално време. В тази среда е изградена система за сигурна маршрутизация, позволяваща аутентиация между съседни маршрутизатори, които си обменят информация за рекламирани от тях мрежови префикси. Това решение ще може да послужи и помогне за решаването на проблеми, възникнали в различни реални ситуации. Както в процеса на обучение, така и на системни администратори и инженери, които искат да тестват своите нови конфигурации, като прилагането на IPSec при IPv6 базираните протоколи за маршрутизация, преди те да бъдат пренесени на реални устройства. Причината за това е, че симулираната мрежова топология, с избраните „*open-source*“ технологии, позволяват да се експериментира със сложни мрежови конфигурации, правейки възможно максимално да се приближат до начина, по който работят реалните устройства в компютърните мрежи.

В първа глава беше описан проблемът, неговата актуалност и причините налагащи прехода от IPv4 към IPv6. Разгледани бяха множество актуални и реални случаи довели до срив в глобалната мрежа, показвайки мащабите на щетите при евентуално предаване на невалидна или подменена маршрутизираща информация.

Във втора глава бяха разгледани особеностите и подобренията на новата версия на IP протокола. Бяха обобщени и представени новите типове IPv6 адреси, чрез които се предава както служебен, така и потребителски трафик и които могат да бъдат обект на атаки, ако не са защитени. Бяха изследвани свойствата на IPSec, приложими в процеса на аутентикация между съседни маршрутизатори, в контекста на протокола IPv6. Анализирани бяха

предимствата му пред други методи за удостоверяване в процеса на маршрутизация.

В трета глава беше разгледан и анализиран протоколът за външна маршрутизация BGP в частта установяване на съседство между маршрутизатори. Представен беше начинът му на работа и различните видове служебни съобщения, които се налага да бъдат защитени при предаване. Обяснено е защо механизъм за удостоверяване с пароли не е достатъчен за защита от намеса на „трети страни“ и защо се налага приложение на IPSec.

В четвърта глава беше разгледан протокола OSPF, като е обяснено функционирането му в контекста на особеностите на версия 3, пригодена за работа в IPv6 среда. Бяха разгледани различни възможни варианти за реализация на поставената задача. Бяха изследвани и детайлно описни използваните методи и средства за защита в процеса на маршрутизация – установяване на съседство, обмен на маршрути и други и по какъв начин се прилага IPSec.

В пета глава на базата на проучванията от предните глави беше предложен и реализиран метод за защита на процеса на маршрутизация, който да гарантира изграждането на стабилна и защитена комуникация между маршрутизаторите по протоколи BGP4/4+ и OSPFv3. За тази цел беше реализирана мрежа от виртуални TinyCore Linux маршрутизатори и изградени IPSec тунели между тях с инструмента ipsec-tools. За среда за маршрутизация беше използван маршрутизиращият софтуер с отворен код Zebra/Quagga. В шеста глава, като заключение, бяха проведени тестове за сигурността в процеса на маршрутизация по BGP4/4+ и OSPFv3 във виртуалната среда за симулация с помощта на мрежовия анализатор Wireshark. На база на проведените тестове е направен анализ и оценка на реализираната технология и по-специално ефективността на защитата от измами с IP адреси (*IP Spoofing*) и други атаки, свързани с маршрутизацията. Дадени бяха препоръки на базата на полученият опит по време на реализацията на настоящата работа.

7.2 Насоки за бъдещо изследване

В настоящата работа, за реализацията на IPSec тунелитане за защита на маршрутизацията беше използван софтуерният продукт ipsec-tools. Това е практически най-използваният в различни операционни системи пакет от IPSec инструменти, с отлична обратна съвместимост на версиите. Но има известни недостатъци в сравнение с по-нови решения като OpenSwan и FreeSwan. В бъдеща работа по тема е добре реализираната технология да се мигрира към тези нови решения и да се анализират техните предимства и недостатъци при решаването на конкретната задача поставена в настоящата разработка. Полезно би било да се изпробва и вторият метод за удостоверяване на OSPFv3, ползващ *Authentication Trailer* за аутентикация, когато работата по този *patch* е готова.

За реализацията бяха избрани и използвани виртуализиращия софтуер VirtualBox, средата за симулация GNS3, пакетния мрежов анализатор Wireshark и операционната система Tiny Core Linux. Но те не са единствените решения с отворен код и свободни за ползване, които могат да намерят приложение. В бъдещо развитие на темата могат да се използват други технологии и да се покажат предимствата на едните или другите в зависимост от конкретни нужди на хората, които ще ги ползват.

Използвана литература и ресурси

Литература:

2 Wendlandt, Dan, et al. “Don’t secure routing protocols, secure data delivery.” (2006).

6 Nordström, Ola, and Constantinos Dovrolis. “Beware of BGP attacks.” ACM SIGCOMM Computer Communication Review 34.2 (2004): 1-8.

7 Bruno, Anthony. “CCIE Routing and Switching Exam Certification Guide.”, Cisco Press. 2003.

11 Blanchet, Marc. Migrating to IPv6: a practical guide to implementing IPv6 in mobile and fixed networks. John Wiley and Sons, 2009.

19 Hermann, Penny. “Security Features in IPv6.”, SANS. 2002.

24 Brown, S., Brian Browne, Neal Chen. “Configuring IPv6 for Cisco IOS.”, Syngress Publishing, Inc.. 2002. ch. 2

30 Bruhadeshwar, Bezawada, et al. “Routing protocol security using symmetric key based techniques.” Availability, Reliability and Security, 2009. ARES’09. International Conference on. ieee, 2009.

34 Huang, Dijiang, Amit Sinha, and Deep Medhi. “A key distribution scheme for double authentication in link state routing protocol.” Performance, Computing, and Communications Conference, 2005. IPCCC 2005. 24th IEEE International. IEEE, 2005.

35 Hu, Yih-Chun, Adrian Perrig, and Marvin Sirbu. “SPV: Secure path vector routing for securing BGP.” ACM SIGCOMM Computer Communication Review 34.4 (2004): 179-192.

37 Khan, Naasir Kamaal, K. Gupta, and Z. A. Usmani. “Deployment Issues of sBGP, SoBGP and psBGP: A Comparative Analysis.” (2011).

38 Nicol, David M., Sean W. Smith, and Meiyuan Zhao. “Evaluation of efficient security for BGP route announcements using parallel simulation.” Simulation Modelling Practice and Theory 12.3 (2004): 187-216.

41 Kent, Stephen, Charles Lynn, and Karen Seo. “Secure border gateway protocol (S-BGP).” Selected Areas in Communications, IEEE Journal on 18.4 (2000): 582-592.

47 Labovitz, Craig, et al. “Delayed Internet routing convergence.” ACM SIGCOMM Computer Communication Review 30.4 (2000): 175-187.

82 Hines, Andrew. “Neighbour Discovery in IPv6.” (2004).

100 Scott, Hogg, Eric Vyncke. “IPv6 Security.”, Cisco Press. 2008.

102 Tracy, Yvonne. “Border Gateway Protocol -The Language of the Internet.”, SANS. 2002.

103 Nicholes, Martin O., and Biswanath Mukherjee. “A survey of security techniques for the border gateway protocol (BGP).” Communications Surveys & Tutorials, IEEE 11.1 (2009): 52-65.

104 Huston, Geoff, Mattia Rossi, and Grenville Armitage. “Securing BGP—A literature survey.” Communications Surveys & Tutorials, IEEE 13.2 (2011): 199-222.

113 Aiello, William, John Ioannidis, and Patrick McDaniel. “Origin authentication in interdomain routing.” Proceedings of the 10th ACM conference on Computer and communications security. ACM, 2003.

116 Butler, Kevin, et al. “A survey of BGP security issues and solutions.” Proceedings of the IEEE 98.1 (2010): 100-122.

125 Ramanath, Avinash. “A Study of the interaction of BGP/OSPF in Zebra/ZebOS/Quagga.” (2004).

129 Dunmore, Martin. “An ipv6 deployment guide.” (2005). p.227, p.232.

133 Elias, Sargon. “Is The Border Gateway Protocol Safe?.”, SANS. 2003.

140 Fatoohi, Rod, and Rupinder Singh. “Performance of Zebra Routing Software.” IMSA. 2001.

141 Kurnikov, Arseny. “Linux Kernel Application Interface.”. Aalto University publication series (2013): 4-9.

151 Carr, Jeffrey. “Inside cyber warfare.”, O’Reilly Media (2010).

152 Shannon, McFarland, Muninder Sambi, Nikhil Sharma, and Sanjay Hooda. “IPv6 for Enterprise Networks.”, Cisco Press. 2012., ch.3

153 Feibel, Werner. The encyclopedia of networking. Vol. 20. Network Press, 1996.

155 Sam, Halabi. Internet routing architectures., 2nd Edition, Pearson Education India, 2008.

157 Кирх, Олаф и Тери Доусън. “Linux – ръководство на мрежов ия администратор”. Прев. от анг. Вълко Йотов, Христо Йонков, Ивайло Иванов. Изд. СофтПрес ООД. 2001. стр. 132

158 Hogg, Scott, Eric Vyncke. “IPv6 Security.”, Cisco Press. 2009. p. 88

168 Rayamajhi, Aman, and Abhishek Kumar. “Preventing Autonomous System against IP Source Address Spoofing:(PASIPS) A Novel

Approach." International Journal of Network Security (2152-5064) 2.3 (2011).

169 Malik, Saif Ur Rehman, et al. "A methodology for OSPF routing protocol verification." Proceedings of international conference on scalable computing and communications (ScalCom), Changzhou, China. 2012.

Ресурси:

1	Routing Security	–	https://www.cs.columbia.edu/~smb/talks/routesec.pdf
3	Routing Security – RIPE NCC	–	http://www.slideshare.net/ripencc/routing-security-10084854
4	Getting IPv6 & Securing your Routing	–	http://www.slideshare.net/ripencc/getting-ipv6-securing-your-routing
5	Six worst Internet routing attacks	–	http://www.networkworld.com/news/2009/011509-bgp-attacks.html
8	IPv6 – The Future of the Internet	–	http://ipv6.com/articles/general/IPv6-The-Future-of-the-Internet.htm
9	IPv4	–	http://www.ipv6actnow.org/info/ipv4-exhaustion/
10	IPv4 Exhaustion	–	http://www.ripe.net/internet-coordination/ipv4-exhaustion
12	The H Ratio for Address Assignment Efficiency	–	http://tools.ietf.org/html/rfc1715
13	IPSec & IPv6 – Securing the NextGen Internet	–	http://www.ipv6.com/articles/security/IPsec.htm
14	What is The Difference Between IPv6 and IPv4?	–	http://www.webopedia.com/DidYouKnow/Internet/ipv6_ipv4_difference.html
16	OSPF for IPv6	–	https://tools.ietf.org/html/rfc5340
17	Implementing IPsec in IPv6 Security	–	http://www.cisco.com/en/US/docs/ios/ipv6/configuration/guide/ip6-ipsec.html#wp1106263

18	Six things small networks need to know about IPv6	–	http://venturebeat.com/2013/08/17/ipv4-ipv6-apocalypse/
21	Chapter 3 – IP Addressing	–	http://technet.microsoft.com/en-us/library/bb726995.aspx
22	Internet Protocol Darpa Internet Program Protocol Specification	–	http://tools.ietf.org/html/rfc791#section-3.1
23	Internet Protocol, Version 6 (IPv6)	–	http://www.ietf.org/rfc/rfc2460.txt
25	Mobility Support in IPv6	–	http://www.ietf.org/rfc/rfc3775.txt
26	Mobile IPv6	–	http://ipv6.com/articles/mobile/Mobile-IPv6.htm
27	Security Architecture for the Internet Protocol	–	http://www.ietf.org/rfc/rfc2401.txt
28	IP Authentication Header	–	https://www.ietf.org/rfc/rfc2402.txt
29	IP Encapsulating Security Payload (ESP)	–	http://www.ietf.org/rfc/rfc2406.txt
31	IPv6 Is The New Normal	–	http://www.worldipv6launch.org/
32	OSPFv3 authentication	–	http://packetlife.net/blog/2008/sep/3/ospfv3-authentication/
33	Authentication/Confidentiality for OSPFv3	–	https://www.ietf.org/rfc/rfc4552.txt
36	Understanding IPv6 EUI-64 Bit Address	–	https://supportforums.cisco.com/document/100566/understanding-ipv6-eui-64-bit-address
39	BGP Security Vulnerabilities Analysis	–	http://www.ietf.org/rfc/rfc4272.txt
40	Authentication/Confidentiality for OSPFv3	–	https://www.ietf.org/rfc/rfc4552.txt
42	Secure BGP Project (S-BGP)	–	http://www.ir.bbn.com/sbgp
43	Cisco's Visual Networking Index Forecast Projects Nearly Half the World's Population Will Be Connected to the Internet by 2017	–	http://newsroom.cisco.com/release/1197391/

44	Crash Course in IPv6	–	http://www.windowsnetworking.com/articles-tutorials/network-protocols/Crash-Course-IPv6-Part1.html
45	IP Version 6 Addressing Architecture	–	http://www.ietf.org/rfc/rfc4291.txt
46	CCNA Exploration 2 (2007)		
48	OSPF for IPv6	–	http://tools.ietf.org/html/rfc5340
49	OSPFv3 for IPv4 and IPv6 Cisco IOS support for OSPFv3 for multiple address families	–	http://www.networkworld.com/community/blog/ospfv3-ipv4-and-ipv6
50	Support of Address Families in OSPFv3	–	http://tools.ietf.org/html/rfc5838
51	OSPFv3 Instance ID Registry Update	–	http://tools.ietf.org/html/rfc6969
52	Notes on OSPF Database Exchanges	–	https://lockienotlucky.wordpress.com/2011/10/14/notes-on-ospf-database-exchanges/
53	Operation of OSPFv3	–	http://www.networkworld.com/subnets/cisco/050107-ch9-ospfv3.html
54	CCNA Exploration: Routing Protocols and Concepts (2007)		
55	OSPF Version 2	–	https://www.ietf.org/rfc/rfc2328.txt
57	Extending OSPF to Support Demand Circuits	–	http://tools.ietf.org/html/rfc1793
58	Multicast Extensions to OSPF	–	http://tools.ietf.org/html/rfc1584.html#page-19
59	The OSPF NSSA Option	–	http://tools.ietf.org/html/rfc1587
60	The OSPF Not-So-Stubby Area (NSSA) Option	–	http://tools.ietf.org/html/rfc3101
61	Open Shortest Path First	–	http://en.wikipedia.org/wiki/Open_Shortest_Path_First
62	OSPFv2 versus OSPFv3	–	http://packetlife.net/blog/2010/mar/2/ospfv2-versus-ospfv3/

63	IPSec Modes: Transport and Tunnel	–	http://www.tcpipguide.com/free/t_IPSecModesTransportandTunnel.htm
64	Understanding OSPFv3 Authentication	–	http://www.juniper.net/techpubs/en_US/junos14.2/topics/concept/ospfv3-authentication-overview.html
65	IPsec for OSPFv3	–	http://www.brocade.com/downloads/documents/html_product_manuals/FI_08010b_L3/GUID-0B129B59-4648-46F0-8499-8E1CA68A69B6.html
66	IPSec Security Associations and the Security Association Database (SAD); Security Policies and the Security Policy Database (SPD); Selectors; the Security Parameter Index (SPI)	–	http://www.tcpipguide.com/free/t_IPSecSecurityAssociationsandtheSecurityAssociation.htm
67	Implementing IPSec in IPv6 Security	–	http://www.cu.ipv6tf.org/pdf/v6_ipsec.pdf
68	The NULL Encryption Algorithm and Its Use With IPsec	–	http://tools.ietf.org/html/rfc2410
69	Sample Configuration for Authentication in OSPF	–	http://www.cisco.com/c/en/us/support/docs/ip/open-shortest-path-first-ospf/13697-25.html
70	Implementing IPsec in IPv6 Security	–	http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/configuration/15-2mt/ipv6-15-2mt-book/ip6-ipsec.html
71	IPv6 multicast over IPv6 IPSec VTI	–	http://cciethebeginning.wordpress.com/tag/ospfv3/
72	One Year after World IPv6 Launch, Number of IPv6-Connected Internet Users Doubles	–	http://www.worldipv6launch.org/press/one-year-after-world-ipv6-launch/

73	Someone's Been Siphoning Data Through a Huge Security Hole in the Internet	–	http://www.wired.com/threatlevel/2013/12/bgp-hijacking-belarus-iceland/
74	Cyber-security puzzle: Who is sending Internet traffic on long, strange trips?	–	http://www.csmonitor.com/World/Security-Watch/2013/1203/Cyber-security-puzzle-Who-is-sending-Internet-traffic-on-long-strange-trips
75	Internet Protocol Version 6 Address Space	–	http://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml
76	IPv6 Global Unicast Address Assignments	–	http://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml
77	Internet Numbers Registries	–	http://tools.ietf.org/html/rfc7249
78	IP Version 6 Addressing Architecture	–	http://tools.ietf.org/html/rfc4291
79	INE tutorials – IPv6 – (cisco video tutorial)		
80	Cisco IOS IPv6 Feature Mapping	–	http://docwiki.cisco.com/wiki/Cisco_IOS_IPv6_Feature_Mapping
81	Neighbor Discovery for IP version 6 (IPv6)	–	http://tools.ietf.org/html/rfc4861
83	IPv6 neighbor discovery	–	http://packetlife.net/blog/2008/aug/28/ipv6-neighbor-discovery/
84	EUI-64 in IPv6	–	http://packetlife.net/blog/2008/aug/04/eui-64-ipv6/
85	Unique Local IPv6 Unicast Addresses	–	http://www.rfc-base.org/txt/rfc-4193.txt
86	IPv6 Link-Local Addresses	–	http://packetlife.net/blog/2011/apr/28/ipv6-link-local-addresses/
87	IPv6 Stateless Address Autoconfiguration	–	http://tools.ietf.org/html/rfc4862
88	Логическо адресиране – Част 2	–	https://sdkteachers.files.wordpress.com/2014/03/3_1_ipaddressing_ipv6.pdf

89	Brazil Leak: If a tree falls in the rainforest….	–	http://www.renesys.com/2008/11/brazil-leak-if-a-tree-falls-in/#more
90	Prefix hijack by as16735	–	http://www.bgpmon.net/prefix-hijack-by-as16735/
91	Six worst Internet routing attacks How YouTube, Yahoo and others fell prey to router incidents and accidents	–	http://www.networkworld.com/article/2272520/lan-wan/six-worst-internet-routing-attacks.html
92	Hacker Redirects Traffic From 19 Internet Providers to Steal Bitcoins	–	http://www.wired.com/2014/08/isp-bitcoin-theft/
93	The Internet's Vulnerable Backbone How cybercriminals hijacked the Web's architecture to mine bitcoins.	–	http://www.slate.com/articles/technology/future_tense/2014/08/bgp_hijacking_cybercriminals_used_internet_architecture_to_mine_bitcoins.html
94	Protect BGP sessions with the TCP MD5 option	–	http://bgphints.ruud.org/articles/bgp-md5.html
95	Cisco – BGP Peer/Neighbor/Adjacency States	–	http://www.ciscoconsole.com/routing/bgp/cisco-bgp-peer-neighbor-adjacency-states.html/
96	BGP Adjacency States	–	http://archive.networknewz.com/networknewz-10-20060403BGPAdjacencyStates.html
97	Securing the Border Gateway Protocol	–	http://meetings.ripe.net/ripe-45/presentations/ripe45-eof-stephen.pdf
98	BGP MD5: Good, Bad, Ugly?	–	https://www.nanog.org/meetings/nanog39/presentations/Scholl.pdf
99	BGP Adjacency Security Tools	–	http://packetlife.net/blog/2009/dec/3/bgp-adjacency-security-tools/
101	Protecting the BGP Session with MD5 Authentication	–	http://www.costiser.ro/2013/03/31/bgp-md5-authentication/
105	Protection of BGP Sessions via the TCP MD5 Signature Option	–	http://tools.ietf.org/html/rfc2385
106	Multiprotocol Extensions for BGP-4	–	http://tools.ietf.org/html/rfc4760

107	Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing	–	http://tools.ietf.org/html/rfc2545
108	A Border Gateway Protocol 4 (BGP-4)	–	http://tools.ietf.org/html/rfc4271
109	BGP Security Vulnerabilities Analysis	–	http://www.ietf.org/rfc/rfc4272.txt
110	Analysis of BGP, LDP, PCEP, and MSDP Issues According to the Keying and Authentication for Routing Protocols (KARP) Design Guide	–	http://tools.ietf.org/html/rfc6952
111	How to configure MD5 authentication for BGP	–	http://www.networkworld.com/article/2345208/cisco-subnet/how-to-configure-md5-authentication-for-bgp.html
112	TCP Congestion Control	–	http://tools.ietf.org/html/rfc5681
114	BGP (Border Gateway Protocol).	–	http://www.orbit-computer-solutions.com/BGP.php
115	How To Authenticate MD5 for BGP Peers.	–	http://www.orbit-computer-solutions.com/How-To-Authenticate-MD5-for-BGP-Peers.php
117	Performance Evaluation of BGP-4+in IPv4/IPv6	–	http://users.utcluj.ro/~dobrota/pdf/Performances_Evaluation_of_BGP4+_in_IPv4_IPv6_ppt.pdf
118	The TCP Authentication Option	–	http://tools.ietf.org/html/rfc5925
119	Networking 101: Understanding BGP Routing	–	http://www.enterprisenetworkingplanet.com/netsp/article.php/3615896/Networking-101-Understanding-BGP-Routing.htm
121	BGP Remove-Private-AS	–	http://ccieblog.co.uk/bgp/bgp-remove-private-as
122	IP Routing: BGP Configuration Guide, Cisco	–	http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_bgp/configuration/15-mt/irg-15-mt-book/irg-bgp4.html

123	BGP Autonomous System Number (ASN)	–	http://www.inetdaemon.com/tutorials/internet/ip/routing/bgp/autonomous_system_number.shtml
124	BGP Messages	–	http://www.informit.com/library/content.aspx?b=CCIE_Practical_Studies_II&seqNum=77
126	BGP Protocol Overview	–	http://netcerts.net/bgp-protocol-overview/
127	Multiprotocol BGP (MBGP)	–	http://www.cisco.com/networkers/nw00/pres/3200/3200_c1_Mod5_rev1.pdf
128	Support for IPv6 BGP (BGP-4 Multiprotocol Extensions)	–	https://sc1.checkpoint.com/documents/R77/CP_R77_Gaia_Advanced_Routing_WebAdminGuide/87850.htm#o102934
130	Capabilities Advertisement with BGP-4	–	http://tools.ietf.org/html/rfc5492
131	hold-time (Protocols BGP)	–	http://www.juniper.net/documentation/en_US/junos12.3/topics/reference/configuration-statement/hold-time-edit-protocols-bgp.html
132	BGP Keepalive and Holddown Timer	–	http://rekrowten.wordpress.com/2013/05/31/bgp-keepalive-and-holddown-timer/
134	GNS3	–	http://www.gns3.net/
135	VirtualBox	–	https://www.virtualbox.org/
136	Welcome to The Core Project – Tiny Core Linux	–	http://tinycorelinux.net
137	Quagga Routing Suite	–	http://www.nongnu.org/quagga/
138	vtysh linux command	–	http://www.linuxcommand.org/man_pages/vtysh1.html
139	Set Up GNS3 with Open-Source Routers	–	http://www.brianlinkletter.com/set-up-gns3-with-open-source-routers/

142	IPv6 Address Assignment to End Sites	–	https://tools.ietf.org/html/rfc6177
143	IPv6 Address Prefix Reserved for Documentation	–	http://www.ietf.org/rfc/rfc3849.txt
144	PREPARING AN IPV6 ADDRESS PLAN MANUAL	–	http://www.ipv6forum.com/dl/presentations/IPv6-addressing-plan-howto.pdf
145	Persistent configuration changes in TinyCore Linux	–	http://www.brianlinkletter.com/persistent-configuration-changes-in-tinycore-linux/
146	Backup	–	http://wiki.tinycorelinux.net/wiki:backup
147	Tiny Core file architecture diagrams	–	http://wiki.tinycorelinux.net/wiki:tiny_core_file_architecture_diagrams
148	The Core Project File Architecture Diagram TCZ Copy Mode of Operation	–	http://distro.ibiblio.org/tinycorelinux/arch_copymode.html
149	The Core Project File Architecture Diagram	–	http://distro.ibiblio.org/tinycorelinux/arch_core.html
150	Number of IPv6-connected users doubles every year	–	http://cctld.ru/en/news/archive/index.php?ELEMENT_ID=5790
154	FCC: Yup, we’re going to stop “paid prioritization” on the ‘Net	–	http://arstechnica.com/tech-policy/2010/12/fcc-priority-access-deals-unlikely-to-get-past-new-open-internet-rules/
156	Installing and configuring quagga(BGP+OSPF)	–	http://www.cmtsinfo.net/?howto=quagga
159	PF_KEY Key Management API, Version 2	–	https://tools.ietf.org/html/rfc2367
160	Linux Kernel 2.6 using KAME-tools	–	http://www.ipsec-howto.org/x304.html
161	IPsec-Tools	–	http://ipsec-tools.sourceforge.net/
162	Supporting Authentication Trailer for OSPFv3	–	https://tools.ietf.org/html/rfc6506

163	RFC-6506(Supporting Authentication Trailer for OSPFv3) implementation in quagga-0.99.21 version	–	http://www.gossamer-threads.com/lists/quagga/dev/24965
164	Quagga-0.99.21 RFC6506-patch	–	https://github.com/venetay/Quagga-0.99.21-RFC6506-patch
165	Quagga-TinyCore-extention	–	https://github.com/venetay/Quagga-TinyCore-extention
166	Deprecating/removing racoon/ipsec-tools from Debian GNU/Linux and racoon from Debian/kfreebsd	–	https://lists.debian.org/debian-devel/2014/04/msg00075.html
167	Smurf IP denial-of-service attacks	–	http://www.cert.org/historical/advisories/ca-1998-01.cfm?

Приложение 1:

Резултат от изпълнението на OSPFv3 командите върху маршрутизатор Q1:

show ipv6 ospf6 neighbor:

```
Q1# show ipv6  ospf6 neighbor
Neighbor ID     Pri    DeadTime  State/IfState       Duration I/F[State]
10.10.10.20       1    00:00:39   Full/BDR           12:28:21 eth0[DROther]
10.10.10.30       1    00:00:39   Full/DR            12:28:26 eth0[DROther]
10.10.10.40       1    00:00:39 Twoway/DROther       12:28:22 eth0[DROther]
```

show ipv6 ospf6 database:

```
Q1# show ipv6  ospf6 database

        Area Scoped Link State Database (Area 0.0.0.0)

Type         LSId            AdvRouter        Age   SeqNum Cksm  Len Duration
Router       0.0.0.0         10.10.10.10       24 8000001a 5e3b   40 00:00:24
Router       0.0.0.0         10.10.10.20       25 8000001a 226d   40 00:00:23
Router       0.0.0.0         10.10.10.30       25 8000001a e59f   40 00:00:23
Router       0.0.0.0         10.10.10.40       20 8000001a a9d1   40 00:00:18
Network      0.0.0.3         10.10.10.30       20 8000001c 7376   40 00:00:19
Intra-Prefix 0.0.0.3         10.10.10.30       25 8000001a 5973   44 00:00:23

        I/F Scoped Link State Database (I/F eth0 in Area 0.0.0.0)

Type         LSId            AdvRouter        Age   SeqNum Cksm  Len Duration
Link         0.0.0.3         10.10.10.10       92 8000001b a44d   56 00:01:31
Link         0.0.0.3         10.10.10.20       80 8000001b 5966   56 00:01:19
Link         0.0.0.3         10.10.10.30       65 8000001b df0e   56 00:01:03
Link         0.0.0.3         10.10.10.40       31 8000001b 46f4   56 00:00:29

        AS Scoped Link State Database

Type         LSId            AdvRouter        Age   SeqNum Cksm  Len Duration
AS-External  0.0.0.0         10.10.10.10       92 8000001a 4be6   36 00:01:31
AS-External  0.0.0.1         10.10.10.10       92 8000001a 40e1   36 00:01:31
AS-External  0.0.0.2         10.10.10.10       92 8000001a 6987   36 00:01:31
AS-External  0.0.0.0         10.10.10.20       80 8000001a 0f19   36 00:01:19
AS-External  0.0.0.1         10.10.10.20       80 8000001a 15f2   36 00:01:19
AS-External  0.0.0.0         10.10.10.30       65 8000001a d24b   36 00:01:03
AS-External  0.0.0.1         10.10.10.30       65 8000001a e904   36 00:01:03
AS-External  0.0.0.0         10.10.10.40       31 8000001a 8490   36 00:00:29
AS-External  0.0.0.1         10.10.10.40       31 8000001a 8c86   36 00:00:29
AS-External  0.0.0.2         10.10.10.40       26 8000001a 813a   28 00:00:23
```

show ipv6 ospf6 route:

```
Q1# show ipv6 ospf6   route
*N E1 ::/0                          fe80::a00:27ff:fea0:e5b2    eth0 12:32:14
*N E1 2001:db8:abcd::/64            fe80::a00:27ff:fea0:e5b2    eth0 12:32:14
*N IA 2001:db8:abcd:1::/64          ::                          eth0 12:32:19
 N E1 2001:db8:abcd:1::/64          fe80::a00:27ff:fe08:3b4d    eth0 12:32:19
 N E1 2001:db8:abcd:1::/64          fe80::a00:27ff:fe92:bd78    eth0 12:32:14
 N E1 2001:db8:abcd:1::/64          fe80::a00:27ff:fea0:e5b2    eth0 12:32:14
*N E1 2001:db8:abcd:2000::/64       fe80::a00:27ff:fe92:bd78    eth0 12:32:14
*N E1 2001:db8:abcd:3000::/64       fe80::a00:27ff:fe08:3b4d    eth0 12:32:19
```

Резултат от изпълнението на BGP4/4+ командите върху маршрутизатор Q4:

show ipv6 bgp summary:

```
Q4# show ipv6 bgp  summary
BGP router identifier 10.10.10.40, local AS number 100
RIB entries 12, using 864 bytes of memory
Peers 1, using 2528 bytes of memory

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2001:db8:abcd::2
                4   200     760     765        0    0    0 12:37:02        2

Total number of neighbors 1
```

show ipv6 bgp neighbors 2001:db8:abcd::2 received-routes:

```
Q4# show  ipv6  bgp  neighbors  2001:db8:abcd::2 received-routes
BGP table version is 0, local router ID is 10.10.10.40
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> ::/0             2001:db8:abcd::2
                                                           0 200 i
*> 2001:db8:abcd::/64
                    2001:db8:abcd::2
                                             1             0 200 ?

Total number of prefixes 2
```

show ipv6 bgp neighbors 2001:db8:abcd::2 advertised-routes:

```
Q4# show ipv6 bgp neighbors 2001:db8:abcd::2 advertised-routes
BGP table version is 0, local router ID is 10.10.10.40
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 2001:db8:abcd::/64
                    2001:db8:abcd::1
                                                1         32768 ?
*> 2001:db8:abcd:1::/64
                    2001:db8:abcd::1
                                                1         32768 ?
*> 2001:db8:abcd:1000::/64
                    2001:db8:abcd::1
                                                1         32768 ?
*> 2001:db8:abcd:2000::/64
                    2001:db8:abcd::1
                                                1         32768 ?
*> 2001:db8:abcd:3000::/64
                    2001:db8:abcd::1
                                                1         32768 ?
*> 2001:db8:abcd:4000::/64
                    2001:db8:abcd::1
                                                1         32768 ?

Total number of prefixes 6
```

show ipv6 bgp neighbors 2001:db8:abcd::2 routes:

```
Q4# show ipv6 bgp neighbors 2001:db8:abcd::2 routes
BGP table version is 0, local router ID is 10.10.10.40
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> ::/0             2001:db8:abcd::2
                                                         0 200 i
*  2001:db8:abcd::/64
                    2001:db8:abcd::2
                                                1             0 200 ?

Total number of prefixes 2
```

Приложение 2:

Конфигурационни файлове на IPSec-tools, Setkey.conf за маршрутизаторите.

Файла Setkey.conf на маршрутизатор Q1:

```
flush;
spdflush;
spdadd ::/0 ff02::5 any -P in ipsec esp/transport//require;
spdadd ::/0 ff02::5 any -P out ipsec esp/transport//require;
spdadd ::/0 ff02::6 any -P in ipsec esp/transport//require;
spdadd ::/0 ff02::6 any -P out ipsec esp/transport//require;
add :: ff02::5 esp 175002505 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: ff02::6 esp 175002506 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;

spdadd fe80::a00:27ff:fee6:9227 fe80::a00:27ff:fe92:bd78 any
   -P out ipsec esp/transport//require;
spdadd fe80::a00:27ff:fe92:bd78 fe80::a00:27ff:fee6:9227 any
   -P in ipsec esp/transport//require;
spdadd fe80::a00:27ff:fee6:9227 fe80::a00:27ff:fe08:3b4d any
   -P out ipsec esp/transport//require;
spdadd fe80::a00:27ff:fe08:3b4d fe80::a00:27ff:fee6:9227 any
   -P in ipsec esp/transport//require;
spdadd fe80::a00:27ff:fee6:9227 fe80::a00:27ff:fea0:e5b2 any
   -P out ipsec esp/transport//require;
spdadd fe80::a00:27ff:fea0:e5b2 fe80::a00:27ff:fee6:9227 any
   -P in ipsec esp/transport//require;

add :: fe80::a00:27ff:fee6:9227 esp 1751 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: fe80::a00:27ff:fe92:bd78 esp 1752 -m transport
```

```
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: fe80::a00:27ff:fe08:3b4d esp 1753 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: fe80::a00:27ff:fea0:e5b2 esp 1754 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
```

Файла Setkey.conf на маршрутизатор Q2:

```
flush;
spdflush;
spdadd ::/0 ff02::5 any -P in ipsec esp/transport//require;
spdadd ::/0 ff02::5 any -P out ipsec esp/transport//require;
spdadd ::/0 ff02::6 any -P in ipsec esp/transport//require;
spdadd ::/0 ff02::6 any -P out ipsec esp/transport//require;
add :: ff02::5 esp 175002505 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: ff02::6 esp 175002506 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;

spdadd fe80::a00:27ff:fe92:bd78 fe80::a00:27ff:fee6:9227 any
   -P out ipsec esp/transport//require;
spdadd fe80::a00:27ff:fee6:9227 fe80::a00:27ff:fe92:bd78 any
   -P in ipsec esp/transport//require;
spdadd fe80::a00:27ff:fe92:bd78 fe80::a00:27ff:fe08:3b4d any
   -P out ipsec esp/transport//require;
spdadd fe80::a00:27ff:fe08:3b4d fe80::a00:27ff:fe92:bd78 any
   -P in ipsec esp/transport//require;
spdadd fe80::a00:27ff:fe92:bd78 fe80::a00:27ff:fea0:e5b2 any
   -P out ipsec esp/transport//require;
spdadd fe80::a00:27ff:fea0:e5b2 fe80::a00:27ff:fe92:bd78 any
   -P in ipsec esp/transport//require;
```

```
add :: fe80::a00:27ff:fee6:9227 esp 1751 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: fe80::a00:27ff:fe92:bd78 esp 1752 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: fe80::a00:27ff:fe08:3b4d esp 1753 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: fe80::a00:27ff:fea0:e5b2 esp 1754 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
```

Файла Setkey.conf на маршрутизатор Q3:

```
flush;
spdflush;
spdadd ::/0 ff02::5 any -P in ipsec esp/transport//require;
spdadd ::/0 ff02::5 any -P out ipsec esp/transport//require;
spdadd ::/0 ff02::6 any -P in ipsec esp/transport//require;
spdadd ::/0 ff02::6 any -P out ipsec esp/transport//require;
add :: ff02::5 esp 175002505 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: ff02::6 esp 175002506 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;

spdadd fe80::a00:27ff:fe08:3b4d fe80::a00:27ff:fe92:bd78 any
   -P out ipsec esp/transport//require;
spdadd fe80::a00:27ff:fe92:bd78 fe80::a00:27ff:fe08:3b4d any
   -P in ipsec esp/transport//require;
spdadd fe80::a00:27ff:fe08:3b4d fe80::a00:27ff:fee6:9227 any
   -P out ipsec esp/transport//require;
```

```
spdadd fe80::a00:27ff:fee6:9227 fe80::a00:27ff:fe08:3b4d any
   -P in ipsec esp/transport//require;
spdadd fe80::a00:27ff:fe08:3b4d fe80::a00:27ff:fea0:e5b2 any
   -P out ipsec esp/transport//require;
spdadd fe80::a00:27ff:fea0:e5b2 fe80::a00:27ff:fe08:3b4d any
   -P in ipsec esp/transport//require;

add :: fe80::a00:27ff:fee6:9227 esp 1751 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: fe80::a00:27ff:fe92:bd78 esp 1752 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: fe80::a00:27ff:fe08:3b4d esp 1753 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: fe80::a00:27ff:fea0:e5b2 esp 1754 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
```

Файла Setkey.conf на маршрутизатор Q4:

```
flush;
spdflush;

spdadd ::/0 ff02::5 any -P in ipsec esp/transport//require;
spdadd ::/0 ff02::5 any -P out ipsec esp/transport//require;
spdadd ::/0 ff02::6 any -P in ipsec esp/transport//require;
spdadd ::/0 ff02::6 any -P out ipsec esp/transport//require;
add :: ff02::5 esp 175002505 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: ff02::6 esp 175002506 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
```

```
spdadd fe80::a00:27ff:fea0:e5b2 fe80::a00:27ff:fe92:bd78 any
   -P out ipsec esp/transport//require;
spdadd fe80::a00:27ff:fe92:bd78 fe80::a00:27ff:fea0:e5b2 any
   -P in ipsec esp/transport//require;
spdadd fe80::a00:27ff:fea0:e5b2 fe80::a00:27ff:fe08:3b4d any
   -P out ipsec esp/transport//require;
spdadd fe80::a00:27ff:fe08:3b4d fe80::a00:27ff:fea0:e5b2 any
   -P in ipsec esp/transport//require;
spdadd fe80::a00:27ff:fea0:e5b2 fe80::a00:27ff:fee6:9227 any
   -P out ipsec esp/transport//require;
spdadd fe80::a00:27ff:fee6:9227 fe80::a00:27ff:fea0:e5b2 any
   -P in ipsec esp/transport//require;

add :: fe80::a00:27ff:fee6:9227 esp 1751 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: fe80::a00:27ff:fe92:bd78 esp 1752 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: fe80::a00:27ff:fe08:3b4d esp 1753 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;
add :: fe80::a00:27ff:fea0:e5b2 esp 1754 -m transport
   -E 3des-cbc 0x5b92c2046a5b4d1692e4687bf9ce0e36d2f7e4c377a
   18f8f
   -A hmac-sha1 0xab6945ddc07849994625df0c766ca8266019cc2f;

spdadd ::/0 2001:db8:abcd::/64 tcp -P in ipsec esp/
   transport//require;
spdadd ::/0 2001:db8:abcd::/64 tcp -P out ipsec esp/
   transport//require;

add :: 2001:db8:abcd::1 esp 17171 -m transport
   -E 3des-cbc 0xb4d1e75ca5c8f5552851534d7f6de467ca2e5fc1aaf
   4692f
   -A hmac-sha1 0x495fca248750e79937b4b6b9dadbb43e0e7be382;
add :: 2001:db8:abcd::2 esp 17172 -m transport
   -E 3des-cbc 0xb4d1e75ca5c8f5552851534d7f6de467ca2e5fc1aaf
   4692f
   -A hmac-sha1 0x495fca248750e79937b4b6b9dadbb43e0e7be382;
```

Файла Setkey.conf на маршрутизатор Q5:

```
flush;
spdflush;
spdadd ::/0 2001:db8:abcd::/64 tcp -P in ipsec esp/
   transport//require;
spdadd ::/0 2001:db8:abcd::/64 tcp -P out ipsec esp/
   transport//require;

add :: 2001:db8:abcd::1 esp 17171 -m transport
   -E 3des-cbc 0xb4d1e75ca5c8f5552851534d7f6de467ca2e5fc1aaf
   4692f
   -A hmac-sha1 0x495fca248750e79937b4b6b9dadbb43e0e7be382;
add :: 2001:db8:abcd::2 esp 17172 -m transport
   -E 3des-cbc 0xb4d1e75ca5c8f5552851534d7f6de467ca2e5fc1aaf
   4692f
   -A hmac-sha1 0x495fca248750e79937b4b6b9dadbb43e0e7be382;
```

РЕЦЕНЗИЯ
от доцент Стефан Станчев Димитров, ФМИ при СУ
на монографичен труд на тема:

„ПОДСИГУРЯВАНЕ НА ПРОЦЕСА НА IPV6 МАРШРУТИЗАЦИЯ С IPSEC ТУНЕЛИРАНЕ"

Автор: **ВЕНЕТА КАЛИНОВА ЙОСИФОВА**

Трудът включва увод, 7 глави, заключение, и 2 приложения. Тематиката е актуална поради бързото навлизане на IPv6 в TCP/IP архитектурата и необходимостта от осигуряване на сигурността в процесите на динамична маршрутизация.

В първата глава на работата е направен обзор на проблемите на сигурността на протоколите за динамична маршрутизация в Интернет с оценка на риска и подходите за намаляването на щети при обмен на информация между мрежовите устройства в Интернет с навлизането на IPv6.

Във втора глава е направен е обзор на IPv6 протокола с възможностите за разширяване на мрежата и услугите. Подробно са разгледани особеностите на IPv6. Подробно са обяснени вградените в IPv6 функции за IPSec криптиране за осигуряване на сигурност на транспорта на данни в мрежата с конфиденциалност, цялостност и достъпност на данните. Разгледани са режимите на IPSec транспорта в тунелен и транспортен режим, формат и употреба на служебните полета в дейтаграмата, осигуряващи IPSec.

В трета глава е направен подробен анализ на механизма и функциите на протокол за динамична маршрутизация BGP, с обяснение на понятието автономна система, диаграма на състоянията във вид на краен автомат, видове пакети и функции в процеса, механизми за удостоверяване, дефиниране на заплахи и атаки в различни фази на процеса и както и необходимостта от криптиране на TCP транспорта с IPSec.

В четвърта глава е направен анализ на протокола за динамична маршрутизация OSPFv3 – тип, алгоритъм, информация за маршрутизация, специфични особености в IPv6 среда, видове пакети, особености, улесняващи и намаляващи времето за конвергенция, видовете маршрутизатори и ролята им в зоните на мрежовата структура. За протокола OSPFv3 подробно са обяснени методите и средствата за защита в различни фази на процеса. Общото впечатление, което за всички глави, е ясното и целенасочено изложение на сложни процеси с препратки към литературни източници и препоръки (RFC) за архитектурата TCP/IP.

В пета и шеста глава се съдържат оригиналните приноси на авторката. Проектирана е и е изградена топология за практическа проверка на решения, осигуряващи сигурността на OSPFv3 и BGP в IPv6 среда. Като рецензент се запознах с работещата конфигурация - виртуални машини Linux върху VirtualBox, с инсталиран за маршрутизация пакет Quagga, с мрежова конфигурация на симулатор GNS3. Подробно са описани изискванията за симулация, обосновани са конфигурациите, мрежовата структура, адресен план и технологиите за осигуряване на сигурността на протоколите за динамична маршрутизация OSPFv3 и BGP с IPSec тунелиране, като за целта е избран пакет ipsectools. Направен е анализ и оценка на защитата на протоколите за динамична маршрутизация.

Имайки предвид горните изводи, оценявайки високо приносите на авторката, препоръчвам труда на Венета Йосифова да бъде отпечатан.

София, 02.04.2015 г. Рецензент: ……

/доц. Стефан Ддимитров/

Всички материали създадени за примерите на книгата
са свободно достъпни на адрес:

https://github.com/venetay/Securing-IPv6-routing-process-with-IPSec-tunnels

На корицата – изображение „Networking“ – достъпно на адрес
http://charterforcompassion.org/sites/default/files/images/NETWORKING.jpg
под лицнез Creative Commons Attribution 3.0. Пълните условия на лиценза
https://creativecommons.org/licenses/by/3.0/us/

Венета Йосифова

**Подсигуряване на процеса
на IPv6 маршрутизация
с IPSec тунелиране**

**Защита на протоколи OSPFv3 и BGP4/4+
базирана на технологии със свободен софтуер**

Българска
Първо издание

Рецензент доц. Стефан Димитров
Редактор инж. Мария Георгиева

Формат 60x84/16
Печатни коли 10

Издателство

Венета Йосифова притежава теоретичен и практически опит с компютърните мрежи, като първия си досег с тях е получила още в мрежовата академия на Технологично училище „Електронни системи" към Техническия университет в София. Завършила е висшето си образование във Факултета по математика и информатика на Софийския университет „Св. Климент Охридски". Придобила е образователно квалификационните степени бакалавър и магистър по информатика. Магистърската програма, която е изучавала, е „Защита на информацията в компютърните системи и мрежи". В този период води и упражнения по компютърни мрежи и комуникации, като хоноруван преподавател във ФМИ. Понастоящем работи като програмист, ползвайки основно „open-source" технологии в своята работа.

В книгата е направен анализ на новите версии на протоколи IPv6, OSPFv3 и BGP4/4+ и техните методи за защита на процеса на маршрутизация. Разгледани са редица реално случили се атаки, засегнали много участници в глобалната мрежа. Взета е предвид актуалността на проблема след световния старт на IPv6 и изчерпването на IPv4 адресно пространство. На базата на този анализ е предложено решение за защитено маршрутизиране, базирано на технологии с отворен код или лицензирани с право на обществено ползване. Създадена е мрежова топология за конкретна задача, разглеждаща прилагането на IPSec при IPv6 базираните протоколи за маршрутизация, с възможности за симулация в реално време. В тази мрежова среда е изградена система за сигурна маршрутизация, позволяваща защита и удостоверяване между съседни маршрутизатори, обменящи информация за рекламирани от тях мрежови префикси. Това решение може да послужи при решаването на проблеми, възникнали в различни реални ситуации. То може да е от помощ както в процеса на обучение на ученици и студенти, така и на системни администратори и инженери, давайки работещ пример как да тестват своите нови конфигурации преди те да бъдат пренесени на реални устройства. Симулираната мрежова топология, с избраните технологии, позволява да се експериментира със сложни мрежови конфигурации, правейки възможно максималното приближаване до начина, по който работят реалните устройства в компютърните мрежи. Показано е как с помощта на мрежов анализатор, могат да се проведат тестове, така че навреме да се открият пропуски в сигурността. Всички материали, създадени за примерите на книгата, са свободно достъпни.