Uluslararası İleri Doğa Bilimleri ve Mühendislik Araştırmaları Dergisi
Sayı 7, S. 119-124,4, 2023

Araştırma Makalesi

IJANSER
https://as-proceeding.com/index.php/ijanser
ISSN: 2980-0811

International Journal of Advanced Natural Sciences and Engineering Researches
Volume 7, pp. 119-124, 4, 2023

Research Article

ns-3 Ağ Simülatörü ile OpenSSL Kütüphanesinin Entegre Edilmesi ve Bir Kimliği Doğrulanmış Şifrelemenin Uygulanması

Samet Tonyalı[1*]

[1]*Yazılım Mühendisliği Bölümü, Gümüşhane Üniversitesi, Türkiye*

[*]*samet.tonyali@gumushane.edu.tr Başlıca yazarın mail adresi*

(Geliş Tarihi: 23 Nisan 2023, Kabul Tarihi: 04 Mayıs 2023)

(DOI: 10.59287/ijanser.2023.7.4.566)

ATIF/REFERENCE: Tonyalı, S. (2023). ns-3 Ağ Simülatörü ile OpenSSL Kütüphanesinin Entegre Edilmesi ve Bir Kimliği Doğrulanmış Şifrelemenin Uygulanması. *International Journal of Advanced Natural Sciences and Engineering Researches*, 7(4), 119-124.

Özet – Güvenlik, günümüz iletişim teknolojilerinin olmazsa olmazlarındandır. İletişim halindeki taraflar arasında akan ağ trafiğinin gizliliğinden ve bütünlüğünden emin olabilmek son derece önemlidir. Bunları sağlamak için geliştirilmiş algoritmalar ve protokoller güvenliğin sağlanmasının yanı sıra sanal özel ağlar gibi ağ erişilebilirliğinin kontrol edilebildiği özel ağların oluşturulmasında da kullanılabilir. Anahtar değişiminden şifrelemeye/deşifrelemeye, ordan dijital imzalamaya kadar birçok uygulama alanına sahip bu algoritmalar ve protokollerin öğretimi için her ne kadar fiziksel ağ ortamı oluşturup kullanmak öğretimin kalitesini artıracak olsa da böyle bir ortamın kurulması ve devamlılığının sağlanabilmesi her açıdan maliyetlidir. Bu yüzden bu tür eğitimler sırasında genellikle ağ simülatörleri kullanılır. Geliştirilen kriptografik algoritmalar ve protokoller simülatör ortamına entegre edilerek nasıl çalıştıkları uygulamalı olarak gösterilir ve ağ protokollerinin çalışma performansına etkisi bu çalışmalar sırasında gözlemlenebilir. Bu çalışmada biz ücretsiz, açık kaynak ve kaliteli destek ekibine sahip olması nedeniyle akademik çalışmalarda sıklıkla kullanılan ns-3 ağ simülatörünü ve uluslararası standartların belirlemiş olduğu kriptografik algoritma ve protokolleri içeren OpenSSL kütüphanesini kullandık. Çalışmamızın amacı OpenSSL kütüphanesini ns-3 ağ simülatörüne entegre etmektir. Özellikle ifade etmek gerekirse, biz bu çalışmada OpenSSL kütüphanesinin ve ns-3 ağ simülatörünün kurulumunun nasıl yapılması gerektiğini ve OpenSSL kütüphanesini ns-3 ağ simülatörüne entegre edebilmek için ns-3 simülatörünün yapılandırma dosyalarında ne tür düzenlemeler yapılması gerektiğini gösterdik. Son olarak, 256 bitlik anahtar kullanan ve Galois/Counter modunda çalışan AES algoritmasının ns-3 ağ simülatöründe var olan bir örnek simülasyon programına nasıl eklenmesi gerektiğini açıkladık. Çalışmamızın ağ/bilgisayar/IoT vb. güvenliği dersleri veren akademisyenler için uygulamalı gösterim konusunda faydalı olacağı kanaatindeyiz.

Anahtar Kelimeler – ns-3, OpenSSL, CMake, Ağ Simülatörü, Kimliği Doğrulanmış Şifreleme

I. GİRİŞ

Bilgisayım, İnternet ve iletişim teknolojilerinin gelişmesiyle birlikte el ile yürütülen birçok süreç dijital platformlara taşınmıştır. Bu sayede pandemi dönemlerinde veya doğal afetlerden sonra birçok çalışan işlerini evinden yürütebilmektedir. Çünkü kâğıt üzerinde yürütülen birçok süreç web uygulamaları şeklinde kullanıcılara sanal ortam üzerinden sunulmakta, kurumlar arası yazışmalar bu uygulamalar üzerinden yapılabilmekte, toplantılar

ise İnternet üzerinden erişilebilen bir sunucu üzerinden sesli ve görüntülü bir şekilde yapılabilmektedir. Hatta çalışmalarını yürütebilmesi için bilgisayarının, çalıştığı şirketin ağına bağlı olması gereken çalışanlar sanal özel ağ (Virtual Private Network - VPN) istemcileri getirmektedir. Özellikle kullandığımız İnternet altyapısını yazılım ve donanım açısından incelediğimizde İnternet üzerinden aktarılan verinin büyük tehditlerle karşı karşıya olduğunu görebiliriz. İnternet'in aktif bir şekilde kullanılabilmesi için geliştirilen ve bir kısmını hâlâ kullandığımız protokoller geliştirilirken iletişimin güvenliği göz önünde bulundurulmamıştı. Çünkü İnternet üzerinde iletişimin güvenlik açısından riskli olabileceği, o dönem öngörülememişti. Öte yandan İnternet altyapısında kullanılan ve bugün yaşı on yılları bulan yönlendirici (router) gibi bazı bileşenlerin güvenlik açıklıkları bugün herkes tarafından bilinmekte ve halka açık veritabanlarında listelenmektedir [1]. Dolayısıyla böyle bir altyapı üzerinden herhangi bir önlem almadan sürdürülen iletişimin güvenliği ve iletişimin içeriğine bağlı olarak bireylerin mahremiyeti dahi risk altında olacaktır.

Güvenli iletişim için sağlanması gereken üç şart vardır: Gizlilik, bütünlük ve erişilebilirlik. Özellikle gizlilik ve bütünlüğün sağlanabilmesi adına geliştirilmiş ve bugün de kullanılmakta olan birçok kriptografik algoritma ve protokol bulunmaktadır. Kişisel ağlardan (Personal Area Network - PAN) İnternet'e kadar her ölçekteki ağda gerçekleşen iletişimin güvenliğini sağlamak için bu algoritma ve protokollerden faydalanılmaktadır. Bu duruma eğitim kurumlarının kayıtsız kalması düşünülemez. Dolayısıyla ülkemizde ve dünyadaki hemen her ülkede Bilgi/Bilgisayar/Ağ Güvenliği isimleriyle dersler verilmektedir. Özellikle Ağ Güvenliği derslerinin işlenişi sırasında bu algoritma ve protokollerin ağ trafiğinin güvenliğini sağlamak üzere nasıl kullanılabileceği ve bunların ağ performansı üzerindeki etkileri uygulamalı olarak gösterilmelidir.

Büyüklüğüne ve içerdiği donanıma bağlı olmakla birlikte gerçek bir ağ kurmanın maliyetli olduğu su götürmez bir gerçektir. Ayrıca araştırmacıların ve ilgili sektördeki çalışanların geliştirmiş oldukları yeni algoritmaları/protokolleri gerçek ağlar üzerinde test etmeleri hâlihazırda çalışmakta olan bir sistemi bozma riskini taşımaktadır. Bu nedenle eğitim ve araştırma faaliyetlerinde ağ simülatörleri kullanarak evindeki bilgisayarı sanki o ağın içindeymiş gibi gösterebilmekte ve şirket ağında sunulan kaynaklara bu sayede erişebilmektedir.

Teknolojinin bu denli ilerlemesi her ne kadar bize en azından çalışma mekânı açısından esneklik sağlasa da bazı güvenlik risklerini de beraberinde sıklıkla kullanılmaktadır. Yalnız duruma göre protokol yığınının herhangi bir katmanına müdahale edilebilmesi için, kullanılan ağ simülatörünün tamamen açık kaynak olması gerekmektedir. İşte bu çalışmada, C++ dilinde geliştirilmiş ve özellikle araştırmacılar tarafından sıkça kullanılan *ns-3* (Network Simulator 3) [2] isimli açık kaynak ağ simülatörüne *OpenSSL* [3] kütüphanesini nasıl entegre edebileceğimizi ve kimliği doğrulanmış şifreleme (authenticated encryption) algoritmalarından birini nasıl kullanabileceğimizi aşama aşama göstereceğiz.

Çalışmanın geri kalan kısmı şu şekilde organize edilmiştir: Bölüm II'de çalışmamıza benzer, çalışmamızla ilgili daha önce yapılmış çalışmalar özetlenmiştir. Bölüm III'te ns-3 ağ simülatörü, OpenSSL kütüphanesi ve kimliği doğrulanmış şifreleme tanıtılmıştır.

II. İLGİLİ ÇALIŞMALAR

OpenSSL kütüphanesi araştırmacılar tarafından ns-3 ağ simülatöründe sıklıkla kullanılmaktadır [4] - [8]. Yalnız, her ne kadar bu kütüphanenin ns-3 simülatörüne entegrasyonu simülatörde yapılandırma gerektirse de bu çalışmaların hiçbirinde bu bilgiye yer verilmemiştir.

ns-3 kullanıcılarının Google grubunda bu tür kütüphane entegrasyon problemleri dile getirilse de [9] maalesef doğrudan OpenSSL kütüphanesinin entegrasyonuyla ilgili girdi bulunmamaktadır. Ayrıca ns-3 simülatörünün 36 numaralı versiyonundan itibaren Waf derleme ve kurulum sisteminden [10] CMake sistemine [11] göç etmiş olması nedeniyle OpenSSL kütüphanesinin entegrasyonu için gerekli yapılandırma ayarları tamamen değişmiştir.

Çalışmamız, akademik çalışmalarda raporlanması göz ardı edilmiş bir sürecin ve ns-3 ağ simülatöründe yakın zamanda gerçekleşmiş değişikliklerin meydana getirdiği bu boşluğu gidermeyi amaçlamaktadır.

III. GENEL BİLGİLER

Bu bölümde ns-3 ağ simülatörü, OpenSSL kütüphanesi ve kimliği doğrulanmış şifreleme,

okuyucunun çalışmanın geri kalan kısımlarını daha iyi anlamasını sağlayacak biçimde tanıtılmıştır.

A. ns-3 Ağ Simülatörü

ns-3 genelde komut satırından çalıştırılan bir kesikli olay simülatörüdür. Yüksek seviye bir modelleme dilinde değil doğrudan C++'ta geliştirilmiştir. Bu sayede simülasyonda gerçekleşen olaylar basitçe bir zamanlayıcı tarafından organize edilen C++ fonksiyon çağrılarıdır.

ns-3 simülatörü ücretsiz ve açık kaynak kodlu olduğu için herhangi bir kullanıcı kaynak kod dosyalarını indirip paylaşımlı (veya statik) kütüphaneler olarak derleyebilir ve bu kütüphaneleri kendi yazmış olduğu **`main()`** programına bağlayabilir. **`main()`** programı belli bir simülasyon senaryosunun yapılandırıldığı, çalıştırıldığı ve durdurulduğu yerdir.

ns-3 simülatörünün kaynak kodlarıyla birlikte sunulan örnek programlar sayesinde kullanıcılar hızlı bir şekilde kendi simülasyon senaryolarını geliştirmeye başlayabilir. Biz de bu çalışmada, verilen çalışmaya hazır programlardan birini kendi amacımız doğrultusunda düzenleyerek kullanacağız.

B. OpenSSL Kütüphanesi

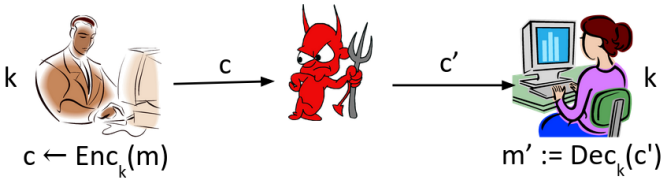
OpenSSL daha önce güvenli soket katmanı (Secure Sockets Layer – SSL) olarak bilinen taşıma katmanı güvenliği (Transport Layer Security – TLS) protokolü için güçlü, ticari olarak kullanıma hazır, çok özellikli ve açık kaynak kodlu bir alet kitidir. Bu protokol, kendi başına da kullanılabilen genel amaçlı bir kriptografik kütüphane kullanılarak gerçeklenir. Bu çalışmada OpenSSL kütüphanesinde tanımlı bir kimliği doğrulanmış şifreleme algoritması kullanılacaktır.

C. Kimliği Doğrulanmış Şifreleme

Genel anlamıyla şifreleme ağ trafiğinin gizliliğinin korunmasına odaklanır. Yalnız, bu yönde bir koruma sadece pasif saldırganlara, yani sadece iletişim ortamını dinleyen saldırganlara karşı etkilidir. Bu tür şifreleme şemaları, iletişim halindeki iki tarafın arasına Şekil 1'de gösterilen şekilde girip şifreli mesajı alacak tarafın bu mesajı deşifre ettiğinde elde edeceği açık mesajı kendi amacına uygun şekilde manipüle edebilen, hatta gönderici tarafı taklit ederek sanki onun tarafından üretilmiş gibi şifreli mesajlar üretip alıcı tarafa gönderebilen aktif saldırganlara karşı güvenli değildir. Taraflar arasında gidip gelen mesajların bütünlüğünün korunup korunmadığı veya alınan bir mesajın gerçekten iddia edilen kişi tarafından gönderilip gönderilmediği her iki tarafın daha önceden paylaşmış olduğu bir anahtar kullanılarak üretilen mesaj doğrulama kodları kullanılarak tespit edilebilir [12]. Bu çalışmada özellikle kimliği doğrulanmış şifreleme algoritmalarından 256 bitlik anahtar kullanan ve Galois/Counter modunda (GCM) çalışan AES şifrelemeyi simülatörün OpenSSL kütüphanesiyle entegre şekilde çalışabildiğini göstermek üzere kullanacağız.

IV. NS-3 AĞ SİMÜLATÖRÜ İLE OPENSSL KÜTÜPHANESİNİN ENTEGRASYONU

Bu bölümde sırasıyla ns-3 ağ simülatörünün ve OpenSSL kütüphanesinin indirilip uygun şekilde yapılandırıldıktan sonra nasıl kurulması gerektiği ve kullanıcıların simülasyon programlarında OpenSSL kütüphanesine referans verebilmesi için yapılandırma dosyalarında hangi değişikliklerin yapılması gerektiği açıklanmıştır. Son olarak kavramın ispatı niteliğinde var olan örnek simülasyon programlarından birinin AES-GCM kimliği doğrulanmış şifreleme algoritmasıyla nasıl çalıştırılabileceği gösterilmiştir.



Şekil 1. Ortadaki adam saldırısı

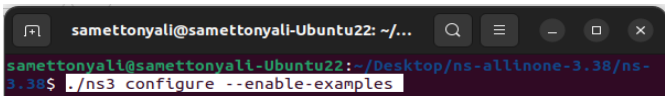
A. ns-3 Ağ Simülatörünün Kurulumu

ns-3 simülatörünün geliştirilmesi sırasında birçok harici kütüphaneden yararlanılmıştır. Bu yüzden simülatörün kurulumundan önce bu kütüphanelerin kullanacağınız makinede kurulu olması gerekir. Bu kütüphaneler ve nasıl yüklenebilecekleri bilgisi ns-3 simülatörünün resmi web sayfasında [13] yer almaktadır. Kurulum süreçlerini kolaylaştırması için biz bu çalışmada Linux tabanlı bir işletim sistemi olan ve bu çalışmanın yazıldığı sırada son versiyon olan Ubuntu 22.04'ü kullandık.

Tüm gerekli kütüphanelerin kurulumu yapıldıktan sonra ns-3 simülatörünün kaynak kodlarının indirilmesi ve gerekli yapılandırmalardan sonra

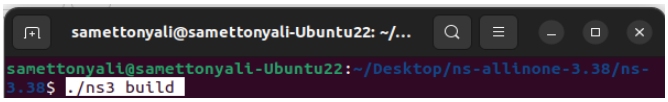
kurulumunun gerçekleştirilmesi gerekmektedir. Her ne kadar Git versiyon kontrol sistemi kullanılarak simülatörün geliştirilme aşamasındaki son sürüm elde edilebiliyor olsa da biz okuyucuya ns-3 simülatörünün resmi web sitesindeki *Sürümler (Releases)* sayfasında [14] yer alan son versiyonu indirmesini tavsiye ediyoruz. Biz bu çalışmada, bu çalışmanın yazıldığı sırada son versiyon olan ns-3.38'i kullandık.

İndirilen sıkıştırılmış dosya açıldıktan sonra komut satırı aracılığıyla doğrultumuzu ns-3.38 isimli klasörün içini gösterecek şekilde ayarladıktan sonra yapılandırma komutumuzu çalıştırırız (Şekil 2). Bu çalışmada kullanacağımız örnek simülasyon programı ns-3 simülatörüyle birlikte gelen örnek programlar içinde yer aldığından yapılandırma sırasında örnekleri aktif hale getirecek bayrak da (**`--enable-examples`**) yapılandırıcıya sunulmuştur.



Şekil 2. Komut satırındaki çalışma doğrultusu ve ns-3 simülatörünü yapılandırma komutu

Yapılandırma komutu başarılı bir şekilde sonlandıktan sonra simülasyon programlarının paylaşımlı kütüphanelerinin oluşturulması için kurulum komutu (Şekil 3) çalıştırılır ve ns-3 simülatörünün kurulumu sona ermiş olur.



Şekil 3. ns-3 simülatörünün kurulum komutu

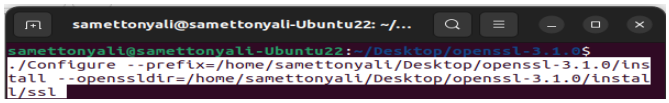
B. OpenSSL Kütüphanesinin Kurulumu

OpenSSL kütüphanesi ve program geliştirmede kullanılan başlık dosyaları (.h uzantılı dosyalar) Ubuntu 22.04 dağıtımı ile varsayılan olarak gelmektedir. Kütüphane üzerinde yapılacak değişikliklerin sistem tarafından kullanılan programın faaliyetlerini etkilememesi için OpenSSL kütüphanesinin son sürümlerinden birinin kaynak kodlarını indirip yerelde kurulumunun nasıl yapılacağını göstereceğiz.

OpenSSL kütüphanesinin kaynak kodlarını yazılımın resmi web sitesindeki *İndirmeler (Downloads)* sayfasından [15] veya GitHub deposundaki *Sürümler* sayfasından [16] indirebilirsiniz. Biz bu çalışmada, bu çalışmanın yazıldığı sırada son versiyon olan OpenSSL 3.1.0'ı kullandık.

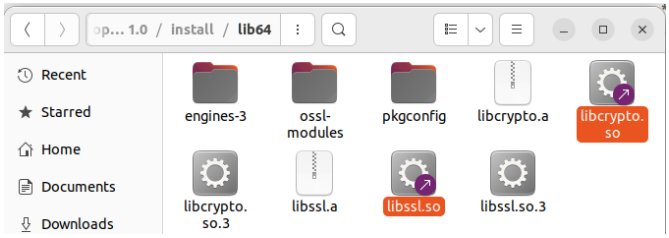
İndirilen sıkıştırılmış dosya açıldıktan sonra komut satırından doğrultumuzu OpenSSL 3.1.0 klasörünün içini gösterecek şekilde ayarlarız. Yerel bir kurulum yapacağımız için kurulumdan sonra OpenSSL yapılandırma dosyalarının ve varsayılan sertifika ve anahtar deposunun saklanacağı doğrultunun bilgisi (**`--openssldir`**) yapılandırma sırasında belirtilmelidir. Ayrıca kurulumun yapılacağı doğrultu ağacının en üst noktasını gösteren doğrultu (**`--prefix`**) yine yapılandırma sırasında belirtilmelidir.

İçinde bulunduğumuz klasörün içinde kurulumun yapılacağı ayrı bir klasör (*install*) oluştururuz. Yeni oluşturulan klasörün içinde yapılandırma dosyalarını, sertifika ve anahtar deposunu saklamak üzere bir başka klasör (*install/ssl*) daha oluştururuz. Son olarak yapılandırma komutumuzu çalıştırırız (Şekil 4).



Şekil 4. OpenSSL kütüphanesinin kurulum öncesi yapılandırılması

Yapılandırma tamamlandıktan sonra çalıştırılabilir dosyaların ve paylaşımlı kütüphanelerin oluşturulması, test edilmesi ve kurulması için sırasıyla **`make`**, **`make test`** ve **`make install`** komutlarının çalıştırılması gerekir. Kurulum komutunun tamamlanmasından sonra paylaşımlı kütüphanelerin (*libssl.so* ve *libcrypto.so*) *install/lib64* doğrultusunda olduğundan emin olmalıyız (Şekil 5).



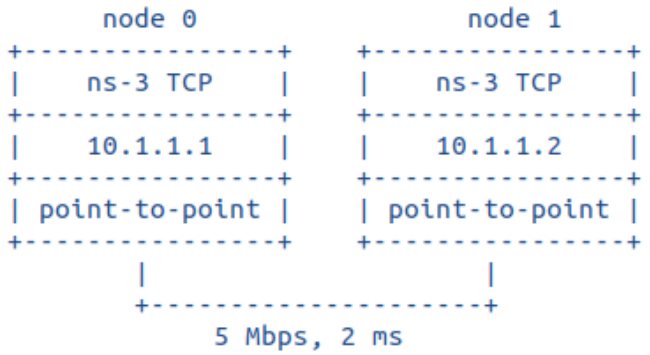
Şekil 5. OpenSSL kütüphanesinin ns-3 simülatörüne entegre edilmesinde kullanılacak paylaşımlı kütüphaneler

C. ns-3 - OpenSSL Entegrasyonu için Yapılandırma Ayarları

ns-3 simülasyon programlarından OpenSSL kütüphanesindeki kodlara yapılacak çağrıların doğru çalışabilmesi için simülasyon programlarının derlenmesi sırasında bu çağrılara karşılık gelen kod

parçalarının nerde bulunacağının derleyiciye (aslında bağlayıcıya – *linker*) bildirilmesi gerekir ki simülasyon kodlarıyla harici kütüphaneden yapılan çağrılar arasında bağlantı kurulabilsin.

Daha önce de belirttiğimiz üzere ns-3 simülatörünün kurulumu CMake aracılığıyla yapılmaktadır. CMake, yapılandırmalar için *CMakeLists* isimli dosyaları kullanır. Biz de bu çalışmada düzenleyeceğimiz örnek programın yapılandırma dosyasını Şekil 5'te bulduğumuz paylaşımlı kütüphanelerin doğrultularını içerecek şekilde düzenleyerek OpenSSL kütüphanesiyle çalıştıracağımız ns-3 simülasyonu arasında bağlantı kuracağız.



Şekil 6. Kullanılan simülasyon programındaki ağın yapısı

Bu çalışmada kullandığımız örnek simülasyon programı ns-3.38 klasöründeki *examples/tutorial* doğrultusunda yer alan *seventh.cc* isimli programdır. Şekil 6'da da görüldüğü üzere bu programda biri gönderici diğeri alıcı olan iki düğüm noktası vardır. Bu iki nokta, veri hızı (*data rate*) 5 Mbps ve gecikmesi (*delay*) 2 ms olan bir noktadan noktaya protokolle birbirine bağlıdır. Veri bağlama katmanının üstünde ise sırasıyla IP ve TCP protokolleri çalışmaktadır. Bu program her ne kadar ağ trafiğindeki yoğunluğa bağlı olarak TCP'nin ağ sıkışıklık penceresinde (*congestion window*) meydana gelen değişiklikleri gözlemlemek üzere yazılmış olsa da biz onu değiştirerek basit bir mesajın kimliği doğrulanmış şifreleme kullanarak şifrelenmesini ve alıcı tarafta hem şifreli metnin doğrulanmasını hem de deşifre edilmesini sağlayacağız.

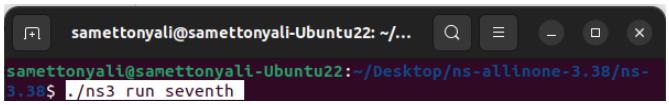
Düzenleyeceğimiz simülasyon programının kaynak koduyla aynı doğrultuda yer alan *CMakeLists.txt* isimli dosyayı açarak *seventh* isimli program için oluşturulmuş yapılandırma bölmesini buluruz. Bu bölmede *LIBRARIES_TO_LINK* başlığı altında yer alan kütüphaneler *SOURCE_FILES* başlığı altında yer alan kaynak dosyaların derlenip kurulabilmesi için gerekli olan kütüphanelerin doğrultularını bildirir. OpenSSL de artık bu programın çalışabilmesi için gerekli bir kütüphane olduğu için ilgili paylaşımlı kütüphanelerin doğrultularının Şekil 7'de gösterilen şekilde bu başlık altında yer alması gerekmektedir. Böylece simülasyon programımızda yer alan OpenSSL kütüphanesine yapılan çağrılar sorunsuz bir şekilde çalışabilecektir.

```
build_example(
  NAME seventh
  SOURCE_FILES seventh.cc
               tutorial-app.cc
  LIBRARIES_TO_LINK
    ${libcore}
    ${libstats}
    ${libpoint-to-point}
    ${libinternet}
    ${libapplications}
    /home/samettonyali/Desktop/openssl-3.1.0/install/lib64/libssl.so
    /home/samettonyali/Desktop/openssl-3.1.0/install/lib64/libcrypto.so
```

Şekil 7. *seventh* simülasyon programı için *CMakeLists* dosyasında yapılması gereken değişiklikler

D. Kimliği Doğrulanmış Şifreleme Kodlarının Simülasyon Programına Eklenmesi

Bu çalışmada kullanılan kimliği doğrulanmış şifreleme kodları OpenSSL kütüphanesinin GitHub sayfasında verilen *aesgcm.c* demo dosyasından [17] alınmıştır. Bu dosyada yer alan kodun yarısı şifreleme ve etiket oluşturmak için diğer yarısı ise etiketin doğrulanması ve şifreli metnin deşifre edilmesi için yazılmıştır. Dolayısıyla şifreleme ve etiket oluşturma ile ilgili kod simülasyon programındaki paket gönderiminin yapıldığı kaynak dosyaya (*examples/tutorial/tutorial-app.cc*), etiketin doğrulanması ve şifreli metnin deşifre edilmesi ile ilgili kod ise paket alımının yapıldığı kaynak dosyaya (*src/applications/model/packet-sink.cc*) eklenmelidir. Tüm bu düzenlemeler yapıldıktan sonra simülasyon programımız Şekil 'de gösterilen şekilde başlatılabilir.



Şekil 8. Düzenlemiş olduğumuz simülasyon programının çalıştırılması

V. SONUÇLAR

Bu çalışmada OpenSSL kütüphanesinin ns-3 ağ simülatörüne entegrasyonunu inceledik. Öncelikle simülatörün ve OpenSSL kütüphanesinin nasıl kurulması gerektiğini açıkladık. Simülatör ortamına nazaran harici bir kütüphane olması nedeniyle

simülasyon programlarından OpenSSL kütüphanesine yapılacak çağrıların sorunsuz çalışabilmesi için ns-3 yapılandırma dosyalarında yapılması gereken değişiklikleri listeledik. Yaptığımız değişikliklerin işe yaradığını göstermek için OpenSSL'in GitHub sayfasında bulunan kimliği doğrulanmış şifrelemeyle ilgili örnek bir kodun ns-3 simülasyon ortamına nasıl eklenmesi gerektiğini anlattık.

Bu çalışmada bahsedilen şifreleme/deşifreleme işlemleri hangi kaynak kodun içinde gerçekleştiyse ilgili kod oraya kopyalanıp üstünde gerekli düzenlemeler yapıldı. Benzer bir iş farklı bir kaynak dosyada gerçekleştirileceği zaman hemen hemen aynı kod parçası oraya da kopyalanıp üzerinde program özelinde değişiklikler yapmak gerekecektir. Bu da gereksiz kod tekrarına sebep olacaktır. Bu durumdan kurtulmak amacıyla OpenSSL kütüphanesi için bir ara katman tasarlamayı planlamaktayız.

KAYNAKLAR

[1] Özkan, S. (2017). CVE Details. Retrieved, 16, 2017.

[2] Henderson, T. R., Lacage, M., Riley, G. F., Dowell, C., & Kopena, J. (2008). Network simulations with the ns-3 simulator. SIGCOMM demonstration, 14(14), 527.

[3] Viega, J., Messier, M., & Chandra, P. (2002). Network security with OpenSSL: cryptography for secure communications. " O'Reilly Media, Inc.".

[4] Ozgur, U., Tonyali, S., & Akkaya, K. (2016, November). Testbed and simulation-based evaluation of privacy-preserving algorithms for smart grid ami networks. In 2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops) (pp. 181-186). IEEE.

[5] Al Mazroa, A., & Arozullah, M. (2015). Securing the user equipment (UE) in LTE networks by detecting fake base stations.

[6] Kishiyama, B., Guerrero, J., & Alsmadi, I. (2023). Security Policies Automation in Software Defined Networking. Available at SSRN 4384690.

[7] Benin, J., Poumailloux, J., Owen, H., & Bouabdallah, A. (2012, June). Impact of pseudonym subsequent pre-computation and forwarding in hybrid vehicular networks. In Proceedings of the ninth ACM international workshop on Vehicular inter-networking, systems, and applications (pp. 123-126).

[8] Cunsolo, V. D., Distefano, S., Puliafito, A., & Scarpa, M. (2009, December). Achieving information security in network computing systems. In 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing (pp. 71-77). IEEE.

[9] How I managed to run libcoap in DCE 1.9. (n.d.). How I Managed to Run Libcoap in DCE 1.9. https://groups.google.com/g/ns-3-users/c/t4dH9teYiIQ/m/bnyfwTYdDwAJ

[10] The Waf Book. (n.d.). The Waf Book. https://waf.io/book/

[11] Nguyen, K., Nguyen, T., & Phan, Q. S. (2022, May). Analyzing the CMake build system. In Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice (pp. 27-28).

[12] Katz, J., & Lindell, Y. (2020). Introduction to modern cryptography. CRC press.

[13] *Installation - Nsnam*. (n.d.). Installation - Nsnam. https://www.nsnam.org/wiki/Installation#Prerequisites

[14] *Releases - Nsnam*. (n.d.). Releases – Nsnam. | Ns-3. https://www.nsnam.org/releases/

[15] *Downloads – OpenSSL*. Downloads – OpenSSL. https://www.openssl.org/source/

[16] *Releases – OpenSSL GitHub*. Releases – OpenSSL GitHub. https://github.com/openssl/openssl/releases

[17] *AES-GCM Cipher Demo – OpenSSL GitHub*. AES-GCM Cipher Demo – OpenSSL GitHub. https://github.com/openssl/openssl/blob/openssl-3.1.0/demos/cipher/aesgcm.c