Makale Türü: Araştırma Makalesi
Makale Gönderim Tarihi: 07.03.2021
Makale Kabul Tarihi: 25.03.2022

Tersine Mühendislik Yöntemi ile Test Senaryo Üreten ve Yürüten Çerçeve: Finansal Bir Uygulamada Vaka Çalışması

Test Case Generation and Execution Framework with Reverse Engineering Method: Case Study for a Financial Application

Emine Dumlu Demircioğlu[1], *Oya Kalıpsız*[1]

[1]Bilgisayar Mühendisliği
Yıldız Teknik Üniversitesi
eminedumlu@gmail.com, oyakalipsiz@gmail.com

Özet

'u makalede, birbiriyle API mesajlarını kullanarak 'etişimde bulunan istemci-sunucu mimarisine dayalı istemler için, farklı veri formatlarını destekleyen bir mesaj üdümlü test senaryosu üreten ve yürüten çerçeve yaklaşımı nerilmektedir. Temel olarak, çerçeve ağ üzerinde istemci-unucu uygulamaları arasındaki iletişimden elde edilen ağ g dosyasından tersine mühendislik yöntemi ile API esajlarını elde ederek yürütülmesini sağlamaktadır.

'üyük ölçekli iş uygulamalarının fonksiyonel doğruluğunun krarlı bir şekilde manuel olarak test edilmesi zaman alan ve ataya açık bir süreçtir. Bu sürecin uçtan uca (test senaryosu retiminden yürütülmesine, doğrulanması ve hata aporlanması kadar geçen sürecin) otomatikleştirilmesi azılım testinde verimliliği artırmaktadır. Günümüzde API estinde kullanılan mevcut test araçları, tüm iş alanlarına ntegre edilememektedir. Dahası, var olan test otomasyon raçları daha çok HTTP protokolü üzerinden erişilebilen VEB-API'leri içindir. Bu çalışmanın motivasyonu da, 'etişim olarak API mesajlarını kullanan istemci-sunucu imarisine dayalı uygulamalara yönelik regresyon testi tomasyonu çerçevesi azlığından kaynaklanmaktadır.

'u makalede önerdiğimiz yaklaşımımızın etkinliğini eğerlendirmek için finansal bir sisteme uyguladık.

nahtar kelimeler: Uygulama Programı Arayüzü Testi, Otomasyon Testi, İstemci-Sunucu Uygulmaları, Vaka 'alışması

Abstract

n this paper, a framework that generates and executes a essage driven test scenario that supports different data ormats is proposed for the systems based on client-server rchitecture that communicate with each other by using ifferent data formats. Basically, the framework provides the eneration of API messages by using reverse engineering ethod from the network log file obtained from the ommunication between client-server applications on the etwork. Repetitive manual testing of the functional correctness of large-scale business applications is a time-consuming and error-prone process. Automating this process by ensuring end-to-end increases software testing efficiency. The studying is motivated due to the fact that there is a lack of regression test automation framework in a specific domain: client-server apps which uses API messages for the communication, such as financial applications. We have applied the proposed testing framework in this paper into a financial systems in order to evaluate the effectiveness of the framework.

Keywords: API Testing, Automation Testing,Client-Server applications, Case Study

1. Giriş

Yazılım testi, yazılım ürünündeki hataları tespit etmek, ürünün kalitesinden emin olmak amacıyla manuel veya otomatik olarak yürütülebilen yazılım geliştirme yaşam döngüsünde büyük öneme sahip olan bir süreçtir. Yazılım geliştirme süreç modeli olarak artırımlı süreç modelini kullanan şirketlerde artırımlı olarak geliştirilen yazılım, her artırımda gelen yeni gereksinimler veya yazılım değişikliği talepleri mevcut yazılımda değişiklik yapılmasını gerektirir. Yapılan değişiklikler ve geliştirilen yeni fonksiyonlar devreye alım öncesi doğrulanma ve etkili bir şekilde regresyon testlerinin gerçeklenmesi ihtiyacını ortaya çıkarmaktadır. Regresyon testi, yazılıma yeni eklenen özelliklerin veya yazılımdaki değişikliklerin mevcut sistemde herhangi bir yan etkiye sebep olup olmadığını doğrulamak amacıyla gerçekleştirilen testtir [1] [2]. Günden güne artırımlı süreç modeli kullanımındaki artış, etkili bir regresyon testi stratejisi ihtiyacını da ortaya çıkarmıştır. Bu testleri özellikle büyük ölçekli iş uygulamalarında manuel şekilde gerçekleştirmek epey zordur [3] [4] [5]. Binlerce farklı test senaryosunun manuel oluşturulması, yürütülmesi ve doğrulanması insan kaynağına büyük yatırım yapmayı gerektirmektedir [26]. Yeni bir yazılım versiyonunun devreye alım öncesi yapılan bu testler, elle yazılmış veya öndecen kaydedilmiş istekleri yeni sürüme göndermek ve belirli yanıtları almak şeklinde gerçekleştirilebilmektedir. Bu süreç,

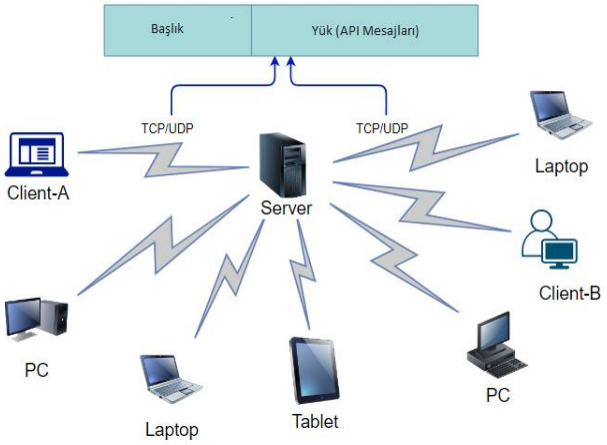
şitli test araçları ile desteklense bile zahmetli bir iştir. Bu
bir test tipinde, yeni sürümün kontrol edilmesi bazı iş
ınlarında tamamen otomatik olacak şekilde
rçekleşememektedir [21]. Bunun yanında, manuel testin
şarısının, manuel testi yürüten kişinin o günkü ruh haline
ğlı olduğu da bilinmektedir. Dikkat eksikliği, odak azalması
gecikme gibi faktörler yazılım testinin verimliliğini
cilemekte ve tüm projeye zarar verebilirmektedir [8][9].
st otomasyonu, test senaryolarının (test cases)
ışturulmasından yürütülmesine, doğrulanması ve hata
orlaması da dahil olmak üzere uçtan uca otomasyon olarak
şünülmelidir [9]. Otomatik olarak test senaryolarının nasıl
ışturulacağına ilişkin yapılan çalışmalar, otomasyon
stinde önemli bir yere sahiptir ve hala günümüzde
ıştırma konusudur [11].
ı çalışmanın amacı da, istemci-sunucu mimarisine dayalı
stemler için, sunucu üzerindeki API versiyonu
ncellemeleri sonrası bir regresyonun oluşup oluşmadığını
spit etmek için bir regresyon test otomasyonu uygulaması
klaşımı önermektedir. Motivasyonumuz, iletişim olarak
PI mesajlarını kullanan sistemlere yönelik literatürde
gresyon testi otomasyon çerçevesi azlığından
ynaklanmaktadır.
ılışmamız üç aşamadan oluşmaktadır:

1. Test senaryolarının üretilmesi,
2. Üretilen test senaryolarının otomatik olarak yürütülmesi,
3. Testlerin doğrulanması ve hata raporlama

ıklaşımımız test senaryolarının yeniden kullanımını
steklemekle birlikte API mesajları ile iletişimde bulunulan
TCP sunucusunun regresyon testlerinde %100
omasyonu sağlamaktadır.
ı makalede ise, test senaryolarının otomatik olarak
etilmesi, yürütülmesi ve doğrulanmasını sağlayan çerçeve
klaşamımız anlatılacaktır.
ı makalenin yazım organizasyonu şu şekildedir; Bölüm
de; çalışmaya ilişkin temel kavramlar, Bölüm 3'te API
stine yönelik literatür taraması yapılmıştır. Bölüm 4'te;
erdiğimiz çerçeveye ilişkin detaylar, bölüm 5'te çerçevenin
gulanması ve vaka analizine ilişkin sonuçlar
ylaşılacaktır. Bölüm 6'da ise sonuçlar ve gelecek
lışmalarımız hakkında bilgi verilecektir.

2. Temel Kavramlar

z konusu hedef sistem mimarisini Şekil 1'de görebilirsiniz.
emci-Sunucu arasındaki iletişim TCP veya UDP protokolü
erinden kurulmaktadır. İstemci uygulaması, sunucu IP'si ve
nucu üzerinde önceden tanımlanan port bilgisini kullanarak
ğlantı gerçekleştirmektedir. TCP (Transmission Control
otocol), internet üzerinde veri aktarımında kullanılan
venilir, bağlantı temelli bir protokoldür. UDP (User
atagram Protocol) ise, IP üzerinde çalışan bağlantısız
nelli bir protokoldür. İstemci ve sunucu arasındaki iletişim,
CP veya UDP veri paketleri kullanılarak sağlanmaktadır.
r veri paketi başlık ve veri alanı (payload) bölümünden
ışmaktadır. Başlık bölümü, kaynak/hedef port numaralarını
erir. TCP başlığı ayrıca ne kadar verinin gönderildiğini
cip etmek için sıra numaraları bilgisini de
lundurmaktadır. Veri bölümü ise, iletişimde kullanılan
verileri (API mesajlarını) taşır [30] .



Şekil 1 Hedef Sistem Mimarisi

Bu makalede önerilen yaklaşım, veriye paket seviyesinde ulaşmak ve test senaryolarını, tersine mühendislik yöntemi ile elde etmektir. Bu amaçla Wireshark aracı [12] kullanılarak istemci-sunucu arasındaki iletişimde kullanılan veri paketleri yakalanmıştır. Wireshark bir paket analizi aracıdır. Veri paketleri, bu araç kullanılarak bir PCAPs (Packet Captures) dosyasında saklanır. PCAPs dosyası, istemci-sunucu arasındaki iletişimi gösteren bir ağ log dosyasıdır. Örnek bir PCAPs dosyası içeriği Şekil 2'de gösterilmiştir. Dosyadaki her veri paketi, kullanılan taşıma katmanı protokolüne bağlı olarak bir TCP veya UDP veri paketidir ve bu paketler iletişimde kullanılan API mesajlarını içerir.

3. İlgili Çalışmalar

API testi, uygulama programı arayüzünün test edilmesini sağlayan ve iş mantığının işlevselliğini, güvenilirliğini, güvenliğini sağlamak için büyük önem taşıyan bir test türüdür. API'lerin, bulut teknolojilerinin ve birbirine bağlı uygulamaların sayısının artmasıyla birlikte API testi, yazılım testinin kritik bir parçası haline gelmiştir. Bu test, kullanıcı arayüzü olmadığı için mesaj seviyesinde gerçekleşir. İş mantığı katmanının içeriği, uygulamaların başarılı çalışması için çok önemlidir. Bir API mesajındaki hata büyük ölçekli yazılım hatalarına sebep olabilir. Bundan kaynaklı, API mesajlarının kalitesini sağlamak için API testi zorunlu hale gelmiştir. API testi ve bu amaç için var olan araç desteği konusunda araştırma topluluğunda daha fazla çalışmaya ihtiyaç bulunmakta ve endüstriyel uygulamada test otomasyonunun gerçek uygulamalarına ilişkin deneyim raporlarının nadir olduğuna inanılmaktadır [31].

Literatürde var olan API testi araçları çoğunlukla bir uygulama katmanı protokolü olan HTTP protokolü ile erişilebilien Web API'leri içindir; özellikle Rest-API en yaygın kullanılan API'dir [5][13][14]. REST tabanlı web uygulamaları temel olarak REST API yanıtları vermek için JSON veya XML formatlarını kullanır [13]. Bu tür test

raçları, kullanılabilirlik açısından tüm iş alanlarına entegre dilememektedir.

.iteratürde, SoapUI [25], Postman [24], jMeter ve diğerleri ibi regresyon testlerinde kullanılabilecek pek çok test aracı ulunmaktadır. Bu tür test araçları ile sabit bir test havuzu luşturulup, regresyon testleri yürütülebilir. Ancak test enaryolarının otomatik olarak üretilmesi hala araştırma onusudur. Bu tür test araçları daha çok HTTP protokolü ile rişilebilen Web tabanlı sunucu testlerinde kullanılmaktadır e daha uygun olduğu bilinmektedir [21]. TCP sunucu estlerine entegre edilmesinde bir takım kısıtlamalar ulunmaktadır. Ayrıca var olan otomasyon araçlarını ullanmak için test script kodunun yazılması ve bunun eğişen gereksinimlerde bakımının yapılması gerekmektedir.

ayrıca mevcut API testi araçlarının yüksek ek yük, uzun test erisi hazırlama ve yeniden kullanım zorluğu gibi bazı orlukları bulunmaktadır [15][19][20]. Günümüzde yaygın larak kullanılan Jmeter test otomasyon uygulamasını ele ldığımızda da bir takım zorluklar gözlemlemekteyiz. Bu ygulama TCP sunucu testlerini desteklese bile söz konusu edef sistemimize uyarlama aşamasında başarısız olmuştur. Manuel olarak oluşturulan test senayrolarının onaylanma şamasında bir takım kısıtlamalar ile karşılaşılmıştır.

ithub'da "goreplay" [27], "pollyjs" [28] ve "node-replay" 29] gibi ağ paketlerini yakalıp, test sistemine karşı oynatan est araçları da bulunmaktadır. Bu araçlar, daha çok yük estleri için kullanılmakta ve HTTP trafiğini tekrar ynatmaktadır. Uygulamanın fonksiyonel işlevselliği test dilmemektedir. Önerdiğimiz yaklaşım ile ağ paketleri test istemine karşı tekrar oynatılmakta ve test sisteminin onksiyonel işlevselliği test edilmektedir. Literatür aramasına göre, ağ paketlerinden testlerin oluşturulması, egresyon testinde uygulanması ve kullanılmasına yönelik ir çalışmaya rastlayamadık. Ağ log dosyası standart bir osya olup, çalışmamıza özel değildir. Ağ log dosyası API stek ve yanıt mesaj çiftlerini içermektedir.

Bu makalede, yukarıda tanımı yapılan sistemler için iteratürde mevcut API test aracı azlığından ötürü ağ üzerinde aket seviyesinde mesajları yakalayan ve regresyon testinde ullanılacak test senaryoları üreten bir tersine mühendislik aklaşımı önermektedir. Kullanılacak verinin paket eviyesinde yakalanması fikri, hem daha uygulama bağımsız lmayı hedeflememiz hem de farklı istemcilerden gelen istek nesajlarını en kolay şekilde elde etmemizi sağlamasından aynaklanmaktadır.

4. Test Senaryo Üreten Çerçeve

Bu bölümde, ağ log dosyasından API mesajların elde dilmesine ilişkin uyguladığımız yöntemin ana iş akışına yer ermekteyiz. Elde ettiğimiz API mesajları, test senaryolarına arşılık gelmektedir. Önerdiğimiz çerçeveyi Java'da eliştirdik. Ağ log dosyasını okuyabilmek için "pkts.io" açık aynak kodlu kütüphaneyi kullandık [16]. Temel olarak, nerdiğimiz çerçeve istemci-sunucu arasındaki iletişim aketlerinin yakalanıp saklandığı bir Pcaps dosyasını girdi osyası olarak alır. Bu dosya, çerçeve tarafından analiz edilir ve istemci-sunucu arasındaki haberleşme mesajlarını elde etmek için tersine mühendislik yaklaşımını uygulanır. Böylece test edilecek sisteme doğru bu dosyayı tekrar oynatmak için (replay) test senaryolarından oluşan test grubu (test suite) üretilmiş olmaktadır. Bir PCAPs dosyasında, farklı protokollere ait API mesajları olabilir. Bu durumda çerçeve her farklı protokol için farklı test grupları oluşturur. Test grubu içerisinde yer alan her test senaryosu istemciden sunucuya gönderilen istek mesajını ve sunucudan istemciye dönen yanıt mesajlarını içerir. Çerçeve bu eşleştirmeyi Kaynak IP:Port ve Hedef Ip:Port bilgisine bakarak yapmaktadır.

PCAPs dosyasında yer alan bir veri paketi okunduğunda, çerçeve bu veri paketini analiz etmeye başlar. İlk olarak, iletişimde hangi taşıma katmanı protokolünün (TCP/UDP) kullanıldığını belirler. Eğer TCP kullanılmış ise, çerçeve bu veri paketinin yeniden iletilen (retransmitted packet) bir paket olup olmadığına karar verir. Bir paketin yeniden iletilen paket olup olmadığı bilgisi, paketin üzerinden yer alan sıra numaralarından ve verinin boyutu bilgisinden hesaplanmaktadır. Eğer bir paketin yeniden iletilen bir paket olduğu tespit edilirse, ilgili paketin önceden test grubuna eklendiği bilindiği için o pakette yer alan mesajlar yeni bir test senaryosu olarak test grubuna eklenmez. Bu durum aynı test senaryosunun üretilmesinin önüne geçer. UDP bağlantısız temelli bir protokol olduğundan, yeniden iletim söz konusu değildir.

Bu aşamadan sonra, hedef sistem için iletişimde hangi uygulama katmanı protokollerinin kullanıldığı tespit edilmektedir. Bu aşamada sezgisel arama yöntemi (heuristic search methods) uygulanmıştır. Belirlenen mesaj desenleri, paketin veri bölümünde aranır. Örneğin, finansal verinin iletimi için kullanılan FIX protokolüne göre, her FIX protokolü mesajı "8 = FIXT" ile başlar ve "10 = XYZ" ile biter [17]. Bir veri paketinin payload bölümünde, FIX protokolüne ait bir API mesajının olup olmadığı aşağıdaki desen uygulanarak arama yapılmıştır:

```
Pattern.compile("8=FIXT.*?\\x0110=.{3}\\x01")
```

Sezgisel arama yöntemi, sırayla ağ log dosyasından elde edilen veriden ilgili mesaj desenini üretmeye çalışır. Farklı uygulama alanları için, farklı desenler belirlenerek çerçeveye entegre edilebilir.
Kullanılan uygulama katmanı protokolü tespit edildikten sonra, paket içerisinde yer alan veri çıkarılır ve söz konusu sistem için belirlenen mesaj formatına göre mesajlar yeniden oluşturulur. Bir mesaj oluşturulurken, paketin içerisindeki veri bölümü parçalanmış olabilir. Çünkü paket üzerindeki veri bölümünün taşıyabileceği maksimum veri boyutu bulunmaktadır. Eğer parçalanma tespit edilirse, veri bölümü hafızada saklanır ve anlamlı mesaj oluşabilmesi için bir sonraki paketin veri bölümü ile birleştirilir.
Son olarak, paketin varış zamanı, Kaynak IP:Port, Hedef IP:Port bilgileri de paket üzerinden elde edilir. Bu bilgiler doğrultusunda mesajların istemci tarafından üretilen bir istek mesajı veya sunucudan alınan bir yanıt mesajı olup olmadığına karar verilir. Yukarıda verdiğimiz bilgiler

doğrultusunda çerçeveyi geliştirmek için kullandığımız algoritma aşağıda verilmiştir.

orithm 1: createTestSuite
ıt: pcapFile
eat
1. packet ← getNextPacket();
2. extractPacketDetail (packet);
3. messages ← constructMessagesInPacket(packet);
4. addToTestSuite(messages);
l end of pcapFile

Algoritma 1, girdi olarak PCAPs doyasını alan ve dosyada yer alan her paketi okuyarak dosya sonuna kadar döngü kuran createTestSuite rutinini gösterir. Her iterasyonda bir sonraki paketi almak için getNextPacket() fonksiyonu çağrılır. İkinci satırda paketin varış zamanı, Kaynak/Hedef IP/Port ve kullanılan taşıma katmanı protokolü (TCP/UDP) bilgilerini elde etmek için extractPacketDetail(packet) fonksiyonu çağrılmaktadır. 3. satırda ise paketin veri bölümünde yer alan API mesajları, hedef sistem için belirlenen mesaj desenleri kullanılarak sezgisel arama yöntemi ile uygun mesaj formatlarına dönüştürelerek test senaryoları oluşturur. 3. adımda elde edilen API mesajları test grubuna eklenir.

Bu yaklaşım farklı uygulama alanlarına kolay bir şekilde adapte edilebilir. Bunun için constructMessagesInThePacket metodu söz konusu hedef sistemin mesaj yapısına uygun olarak uygulanması gerekmektedir. Bu yaklaşımı kullanarak test senaryoları paket düzeyinde otomatik olarak oluşturulabilir. Her test senaryosu, istemciden sunucuya gönderilen istek mesajları ve sunucudan gelen beklenen yanıt mesajlarını içerir.
Bazı uygulama katmanı protokolleri, ortam ile ilgili çeşitli veriler içerir ve bu nedenle bu veriler tekrar kullanılamaz. Örneğin, HTTP yanıtı, hedef sayfanın son değiştirilme zamanını, yanıt tarihini veya oturum bilgilerini içerir. Tüm paketleri herhangi bir analiz yapmadan kullanmak, bu değişken değerler nedeniyle onaylanma hatasına (doğrulama hatasına) neden olabilir. Bu yaklaşım kullanılarak geliştirilecek test otomasyonunda, bu tür çevre ile ilgili değerlerin, değişken verilerin çıkarılması gerekir. Bu nedenden ötürü, tüm TCP/UDP paketlerini kaydetmek ve bunları test ortamına karşı yürütmek yerine, yukarıda bahsedilen uygulama katmanı mesajlarının çıkarılması adımları gerçekleştirilir.

5. Vaka Çalışması

Önerilen yaklaşımın etkinliğini ölçmek amacıyla, borsalarda iletişim protokolleri olarak kullanılan FIX ve OUCH API'yi geliştirdiğimiz çerçeveye uyarladık. İletişim protokolleri olarak kullanılan FIX-API ve OUCH-API protokolleri tamamen birbirinden farklı yapıda mesaj formatlarına sahiptir. FIX (Financial Information Exchange), finansal verinin değişiminde kullanılan bir iletişim protokolüdür. FIX protokolü mesaj yapısına ait tüm bilgiler FIX spesikasyonu dökümanında yer almaktadır [17]. OUCH protokolü de iletişimde kullanılan bir emir iletimi protokolü olup, istemcinin emir girmesine, mevcut emirlerini iptal etmesine, güncellemesine ve bunlara ilişkin işlem bilgilerini almasına izin veren bir protokoldür. OUCH protokolü, istemci uygulaması ile OUCH sunucusu arasında geçen mantıksal mesajlardan oluşmaktadır [18]. FIX protokolü metin (text) tabanlı, OUCH protokolü ise binary tabanlı bir protokoldür.

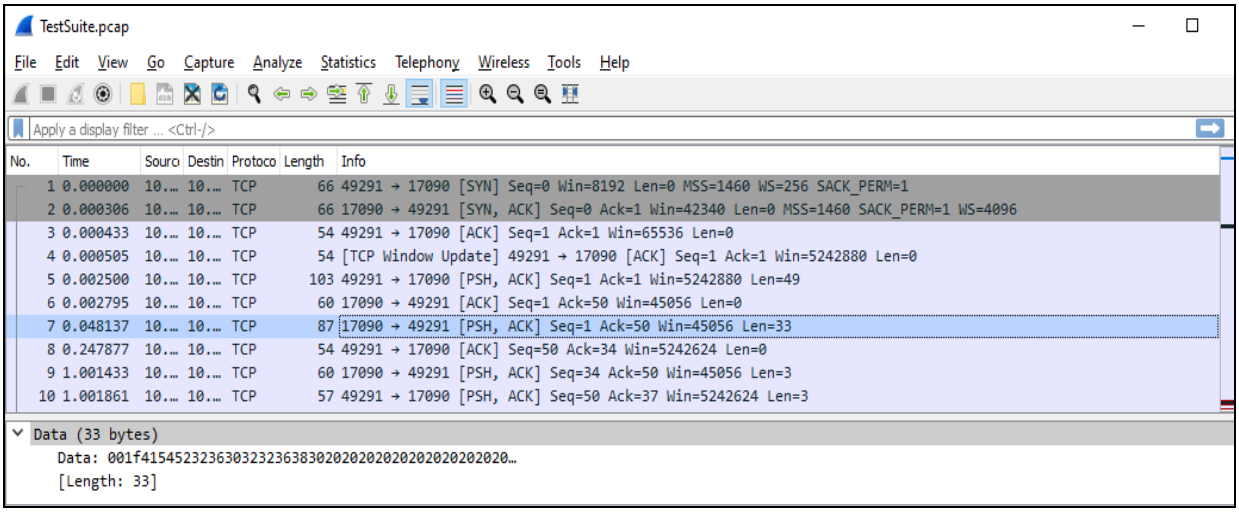
5.1. Verinin Hazırlanması

Çerçevede kullanılacak PCAPs dosyasının elde etmek için, keşif testi uygulanarak test senaryolarının bir listesi hazırlanmıştır. İnsan bilgisine dayalı olarak manuel olarak oluşturulan test senaryosu tasarımı, keşif testi olarak bilinir [9]. Testçiler hazırlanan test senaryolarını test edilen sisteme manuel olarak yürütürken, ağ paketleri Wireshark aracıyla yakalanır. Böylelikle PCAPs dosyası elde edilerek, geliştirilen çerçeveye girdi olarak verilir.

PCAPs dosyası oluşturmanın bir başka yolu ise, üretim/canlı ortamdaki istemci-sunucu arasındaki trafik dinlenerek gerçekleştirilebilir. Böylece gerçek veriler elde edilerek, gerçek verilerden test senaryoları oluşturulur. Bu yöntem ile bir istemcinin üretim ortamındaki sunucuya karşı olan davranışı da yakalanmış olur. Böylece gerçek ortamda bir istemcinin davranışı kaynaklı oluşabilecek bir hatanın hızlı bir şekilde tespiti için, ilgili istemcinin davranışını test ortamına karşı simüle edilmesi sağlanabilir.

5.2. Çerçevenin Uygulanması

Bu çalışmada, hem FIX protokolü hem de OUCH protokolüne ait API mesajlarını içeren TestSuite.pcap dosyası kullanılmıştır. Şekil 2, TestSuite.pcap dosyasındaki ilk 10 TCP paketini göstermektedir. Çerçeve bu dosyayı okuyarak 2 farklı test grubu oluşturur. Tablo 1'de OUCH protokülüne ait TestSuite.pcap dosyasından üretilen seçilmiş 5 adet test senaryosu gösterilmektedir. Tablodaki her satır, istemciden sunuya gönderilen istek mesajını ve sunucudan alınan yanıt mesajını gösterir. Örneğin 1. test senaryosuna göre, tablodaki verileri kullanarak "LoginRequestPacket" mesajı test sunucusuna tekrar gönderilirse, sunucunun da yanıt olarak "LoginAcceptedPacket" mesajı dönmesi beklenecektir. 2. test senaryosuna göre, "UnsequencedData-NewMsg" paketi tablodaki veriler ile sunucuya gönderilirse, sunucunun da "SequenceData-AcceptedMsg" paketi dönmesi beklenmektedir.
Tablo 2'de ise FIX protokolü için seçilmiş 5 adet test senaryosunu görebilirsiniz. Örneğin 1. test senaryosundaki veriler, yeni sürüm yüklenen test sistemine karşı yürütüldüğünde, sunucudan alınan mesajın, tabloda yer alan mesaj ile uyumlu olması beklenir. Beklenmedik bir mesajın gelmesi durumunda regresyon olduğu düşünülmektedir.
Tablodaki test senaryoları, test sistemine karşı otomatik yürütülmeden önce, bazı değişken verilerin (örneğin mesajın oluşturulma zamanı, checksum bilgisi gibi) olabileceği bilinmektedir. Bu değerler, API spesifikasyon dökümanından belirlenmiş ve doğrulama hatalarından kaçınmak için validasyon aşamasında çıkarılmıştır.



Şekil 2 TestSuite.pcap dosyası – Wireshark Ekran Görüntüsü

↓↓

Tablo 1. TestSuite.pcap dosyasından üretilen seçilmiş OUCH-API Test Senaryoları

Hedef IP:Port	İstek (Request) API Mesajı	Yanıt (Response) API Mesajı / Beklenen Mes
10.A.B.C: 1709	**PacketType: LoginRequestPacket** [USER1,ABC, TR18,1]	**PacketType: LoginAcceptedPacket** [TR18, 1]
10.A.B.C: 1709	**PacketType: Unsequenced Data Packet-NewMsg** [**O**, 21, 7798, Buy, 22, 600,,NoChange, XY, Pass, 6789, 0, Client, Normal_hours]	**PacketType: Sequence Data Packet-AcceptedMsg** [**A**, 21, 7098, Buy, 715305981147226, NoChange, XY OnBook, Pass, 6789, 22, 0, Client, Normal_hours]
10.A.B.C: 1709	**PacketType: Unsequenced Data Packet-UpdateMsg** [**U**,21,1b,99,0,NoChange,0,Empty]	**PacketType: Sequence Data Packet-ReplacedMsg** [**U**, 1b, 21, 7798, Buy, 715305981145226, 99, 600, NoChange, XY, OnBook, Pass, 6789, Client]
10.A.B.C: 1709	**PacketType: Unsequenced Data Packet-CancelMsg** [**X**, 21]	**PacketType: Sequence Data Packet-CanceledMsg** [**C**,21,7798,Buy,71530598114735226,Canceled]
10.A.B.C: 1709	**PacketType: LogoutRequestPacket**	**PacketType: LogoutResponsePacket**

Tablo 2 TestSuite.pcap dosyasından üretilen seçilmiş FIX-API Test Senaryoları

Hedef IP:Port	İstek (Request) API Mesajı	Yanıt (Response) API Mesajı / Beklenen Me
10.A.B.C: 1200	**NewMsg:** ***8=FIXT.1.1*** **35=D** 34=1 49=X 50=F6 52=20211019-10:32.06 56=TEST 1=XY **11=10** 38=22 40=2 44=5.000 54=1 55=A.E 59=0 60=20211019-10:32:06.843 70=ASD ***10=044***	**ExecutionReport: AcceptedMsg:** ***8=FIXT.1.1*** 35=8 34=1 49=TEST 56=X 57=F6 1= 6=0 **11=10** 14=0 17=152 22=M 37=**6363BF122222** 38=22.0000000 39=0 40=2 44=5.0000000 48=616 55=A.E 151=20.0000000 60=20211019-10:32:06.9 ***10=032***

10.A.B.C: 1200	**CancelMsg:** **8=FIXT.1.1 35=F** 34=2 49=X 50=F6 52=20211019-10:32:15 56=TEST **11=23** 37=**6363BF1222228** 38=22 54=1 55=A.E 60=20211019-10:32:15.380 **10=184**	**ExecutionReport: CanceledMsg** **8=FIXT.1.1** 35=8 34=2 49=TEST 56=X 57=F6 1=X 6=0 **11=23** 14=0 17=158 22=M 37=**6363BF122222** 38=22.0000000 39=4 41=10 48=616 54=1 55=A.E 70=ASD 150=4 151=0 60=20211019-10:32:15.450 **10=252**
10.A.B.C: 1200	**NewMsg:** **8=FIXT.1.1 35=D** 34=3 49=X 50=F6 52=20211019-10:32:09 56=TEST 1=XY **11=11** 38=77 40=2 44=11.000 54=1 55=A.E 59=0 60=20211019-10:32:16.200 70=ASD **10=022**	**ExecutionReport: AcceptedMsg** **8=FIXT.1.1** 35=8 34=3 49=TEST 56=X 57=F6 1=X 6=0 **11=11** 14=0 17=157 22=M 37=**6363AC111111** 38=77.0000000 39=0 40=2 44=11.0000000 48=616 55=A.E 59=0 70=ASD 119=220.0000000 150=0 151=20.0000000 60=20211019-10:32:16.500 **10=0**
10.A.B.C: 1200	**UpdateMsg:** **8=FIXT.1.1 35=G** 34=4 49=X 56=TEST 50=F6 52=20211019-10:32:16.43=N 37=**6363AC11111181 11=12** 55=A.E 60=20211019-10:32:16.800 38=27 59=0 54=1 40=2 44=11.000 **10=035**	**ExecutionReport: UpdatedMsg** **8=FIXT.1.1** 35=8 49=TEST 56=X 34=4 57=F6 37=**6363AC11111181 11=12** 17=89485 150=5 39=(1=XY 55=A.E 48=616 22=M 54=1 38=27.0000000 44=11.000 59=0 151=27.0000000 14=0 6=0 60=20211019-10:32:16.900 **10=166**
10.A.B.C: 1200	**CancelMsg:** **8=FIXT.1.1 35=F** 34=5 49=X 50=F6 52=20211019-10:32:15.492 56=TEST **11=13** 37=**6363AC11111181** 38=20 54=1 55=A.E 60=20211019-10:32:17.200 **10=184**	**ExecutionReport: CanceledMsg** **8=FIXT.1.1** 35=8 34=5 49=TEST 56=X 57=F6 1=X 6=0 **11=13** 14=0 17=158 22=M 37=**6363AC111111** 38=20.0000000 39=4 41=40360 48=616 54=1 55=A 70=ASD 150=4 151=0 60=20211019-10:32:17.250 **10=252**

Tüm API mesajlarını oluşturup eşleştirdikten sonra, çerçeve bu test senaryolarını test ortamında yeniden oynatmaya başlar. Her test senaryosunun yukarıdaki tabloda da gözükeceği gibi hedef IP adresi ve Port bilgisi vardır. Bu bilgileri kullanarak çerçeve hedef IP:Port'a bağlanır ve test sunucusuna, test senaryolarında yer alan istek API mesajlarını göndermeye başlar. PCAPs dosyasından çıkarılan her test senaryosu Tablo 1 ve Tablo 2'de de görüleceği gibi beklenen/cevap API mesajına sahiptir. Çerçeve test ortamından dönen gerçek yanıt mesajlarını aldıktan sonra, gerçek ve beklenen mesajları karşılaştırarak her test senaryosunu doğrular.

Hangi mesaj alanlarının doğrulanma sürecine dahil edileceğini, spesifikasyona dayalı doğrulama yaklaşımı kullanarak belirledik. Bu aşamada her API protkolü için XML dosyaları oluşturduk. OUCH API için hazırladığımız örnek bir XML dosyasının içeriğini Tablo 3'de görebilirsiniz. Oluşturulan OUCH_API.xsd ve FIX_API xsd dosyaları çerçeveye input olarak verilmiştir ve çerçevenin derlenme aşamasında XML dosyasından ilgili mesaj sınıfları otomatik olarak üretilmiştir. Bununla birlikte her protokolde yer alan mesajların hangi alanlarının doğrulama sürecine dahil edileceği bilgisi de use="required" özelliği ile tanımlanmıştır.

Tablo 3. OUCH protokolü için doğrulama aşamasında kullanılan XML dosyası

OUCH_API.xsd
:element name="CanceledMsg"> <xs:complexType> <xs:attribute name="msgType" fixed="C"/> <xs:attribute name="timeStamp" type="xs:long"/> <xs:attribute name="orderToken" type="xs:string" /> <xs:attribute name="orderBookId" type="xs:int" **"required"** /> <xs:attribute name="side" type="xs:string" **"required"** /> <xs:attribute name="orderId" type="xs:long" /> <xs:attribute name="canceledReason" type="xs:int" **"required"** /> </xs:complexType> **'xs:element>**
element name="AcceptedMsg"/>
element name="ReplacedMsg"/>
element name="RejectedMsg"/>

Doğrulanma süreci tamamlandıktan sonra, çerçeve test sonuçlarının özetini bir rapor olarak üretmekte ve ilgili kişilere bildirim göndermektedir.

6. Sonuç ve Gelecek Çalışmalar

Tekrarlı bir şekilde yapılan manuel testler özellikle büyük ölçekli iş uygulamalarında oldukça zordur. Bu tür uygulamalarda istemci davranışını simüle etmek için binlerce farklı test senaryosu olabilir ve bu senaryoların manuel olarak gerçekleştirilmesi zaman alıcı bir süreçtir. Bu tür uygulamaların test edilmesinde test otomayonu kritik bir öneme sahiptir. Literatüre göre HTTP protokolü dışında kendi uygulama katmanı protokollerine sahip TCP sunucular için var olan test otomasyon uygulamalarının tüm iş alanlarına entegre edilemediğini görmekteyiz ve entegrasyonunda bir takım kısıtlamalar ile karşılaştık. Bu nedenle, bu tür sistemlerin regresyon testleri için istemci ve sunucu arasındaki ağ paketlerini kullandığımız yeni bir yaklaşım önerdik.

Bu makalede istemci-sunucu mimarisine dayalı sistemler için test senaryosu üretimi yapan ve test sistemine karşı tekrar yürüten bir yaklaşım önerdik. Bilgilerin paket düzeyinde kullanılması, uygulama bağımsızlığını sağlamıştır. Bu çerçeveyle, test senaryolarını otomatik olarak paket seviyesinde oluşturarak, API testindeki otomasyonu artırıyoruz. Ayrıca yaklaşımımız, test senaryolarının yeniden kullanımını da sağlamaktadır.

Gelecek çalışmalar olarak iki çalışma yapmayı planlamaktayız. İlk olarak, farklı iş alanlarına yönelik yeni bir API protokolünü çerçeveye tanıtmak ve uyarlamaktır. İkinci olarak ise, üretim ortamından alınan gerçek veri paketlerini test sürecinde kullanmaktır. Bu sayede gerçek verilerden test senaryoları oluşturulabilecek ve böylece bir istemcinin gerçek ortamdaki davranışı, test ortamında otomatik olarak simüle edilebilecektir.

7. Kaynaklar

[1] A. K. Sultanía : Developing software product and test automation software using Agile methodology, Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT), Hooghly, pp. 1-4. (2015)

[2] S.Dalal, K.Solanki, : Challenges of Regression Testing: A Pragmatic Perspective in International Journal of Advanced Research in Computer Science, vol.9, no.1, February (2018)

[3] Z. Liu, Q. Chen and X. Jiang : A Maintainability Spreadsheet-Driven Regression Test Automation Framework, IEEE 16th International Conference on Computational Science and Engineering, Sydney, NSW, pp. 1181-1184. DOI= 10.1109/CSE.2013.175 (2013)

[4] Bangare, Sunil & Borse, Seema & Bangare, Pallavi & Nandedkar, Shital. (2012). AUTOMATED API TESTING APPROACH. International Journal of Engineering Science and Technology. 4.

[5] Isha, A. Sharma and M. Revathi, "Automated API Testing," 2018 3rd International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2018, pp. 788-791.

[6] R. M. Sharma, : Quantitative Analysis of Automation and Manual Testing, International Journal of Engineering and Innovative Technology (IJEIT) Volume 4, Issue 1, (2014)

[7] X. Han, N. Zhang, W. He, K. Zhang and L. Tang, "Automated Warship Software Testing System Based on LoadRunner Automation API," 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Lisbon, 2018, pp. 51-55.

[8] Gonçalves, Wellington & Barreto de Almeida, Carlos & Araújo, Ladyanny & Ferraz, Mateus & Xandú, Rogerio & Junior, Ivaldir. (2017). The Impact of Human Factors on the Software Testing Process: The Importance of These Factors in a Software Testing Environment. Journal of Information Systems Engineering & Management. 2. 10.20897/jisem.201724.

[9] V. Garousi and F. Elberzhager, : Test Automation: Not Just for Test Execution, in IEEE Software, vol. 34, no. 2, pp. 90-96, Mar.-Apr. DOI= 10.1109/MS.2017.34. (2017)

[10] J. Itkonen and M.V. Mantyla,: Are test cases needed? Replicated comparison between exploratory and test-case based software testing, Emprical Software Engineering (2014)

[11] Fernandez-Sanz, L., & Misra, S. (2012). Practical application of UML activity diagrams for the generation of test cases. Proceedings of the Romanian academy, Series A, 13(3), 251-260.

[12] Wireshark Aracı: https://www.wireshark.org/ (son erişim 09 Şubat 2022)

[13] K. Sneha and G. M. Malle, : Research on software testing techniques and software automation testing tools, International Conference on Energy, Communication, Data Analytics and Soft Computing, Chennai, pp. 77-81. DOI= 10.1109/ICECDS.2017.8389562 (2017)

14] N. Bhateja, : A Study on Various Software Automation Testing Tools 2015 International Journal of Advanced Research in Computer Science and Software Engineering Volume 5, Issue 6, June (2015)
15] D. Xu, W. Xu, M. Kent, L. Thomas and L. Wang, "An Automated Test Generation Technique for Software Quality Assurance," in IEEE Transactions on Reliability, vol. 64, no. 1, pp. 247-268, March 2015, doi: 10.1109/TR.2014.2354172.
16] Java Library for reading and writing PCAPs., https://github.com/aboutsip/pkts (son erişim 08 Şubat 2022)
17] FIXAPI-Protocol: https://www.borsaistanbul.com/files/genium-inet-fix-protocol-specification.pdf
18] OUCHAPI-Protocol: https://www.borsaistanbul.com/files/OUCH_ProtSpec_BIST_va2414.pdf
19] Y. Chen, Y. Gao, Y. Zhou, M. Chen and X. Ma, "Design of an Automated Test Tool Based on Interface Protocol," 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Sofia, Bulgaria, 2019, pp. 57-61, doi: 10.1109/QRS-C.2019.00024
20] K. V. Aiya and H. Verma, "Keyword driven automated testing framework for web application," 2014 9th International Conference on Industrial and Information Systems (ICIIS), Gwalior, 2014, pp. 1-6, doi: 10.1109/ICIINFS.2014.7036478.
21] Patrice Godefroid, Daniel Lehmann, and Marina Polishchuk. 2020. Differential regression testing for REST APIs. In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2020). Association for Computing Machinery, New York, NY, USA, 312–323. DOI:https://doi.org/10.1145/3395363.3397374
22] E. Viglianisi, M. Dallago and M. Ceccato, "RESTTESTGEN: Automated Black-Box Testing of RESTful APIs," in 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), Porto, Portugal, 2020 pp. 142-152.
23] H. Ed-Douibi, J. L. C. Izquierdo, and J. Cabot. Automatic generation of test cases for REST APIs: A specificationbased approach. In 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC), pages 181–190. IEEE, 2018.
24] Postman-API Development Environment. https://www.getpostman.com/
25] SOAP: https://www.soapui.org/
26] E. Çelik, S. Eren, E. Çini and Ö. Keleş, "Software test automation and a sample practice for an enterprise business software," 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, 2017, pp. 141-144, doi: 10.1109/UBMK.2017.8093583.
27] GoReplay: https://github.com/buger/goreplay (son erişim 03 Ocak 2022)
28] Pollyjs: https://github.com/Netflix/pollyjs (son erişim 10 Ocak 2022)
29] NodeReplay: https://www.npmjs.com/package/replay (son erişim 7 Şubat 2022)
[30] TCP/UDP: https://www.cs.nmt.edu/~risk/TCP-UDP%20Pocket%20Guide.pdf (son erişim 7 Şubat 2022)
[31] V. Garousi et al., "Automated Testing of Simulation Software in the Aviation Industry: An Experience Report," in IEEE Software, vol. 36, no. 4, pp. 63-75, July-Aug. 2019, doi: 10.1109/MS.2018.227110307.

Özgeçmişler



Emine Dumlu Demircioğlu, lisans eğitimini Ege Üniversitesi Bilgisayar Mühendisliği, yüksek lisans eğitimini ise Sabancı Üniversitesi Bilgisayar Bilimleri ve Mühendisliği'nde tamamlamıştır. Şuanda Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği'nde doktora öğrencisidir ve Borsa İstanbul'da yazılım geliştirme uzmanı olarak çalışmaktadır.



Oya Kalıpsız, yüksek lisans eğitimini Istanbul Teknik Üniversitesi'nde tamamlamıştır. Doktora derecesini ise İstanbul Üniversitesi Sayısal Yöntemler alanında almıştır. Şuanda Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği bölümünde Profesör olarak çalışmalarına devam etmektedir. İlgi alanları: Yazılım Mühendisliği, Veritabanı Sistemleri, Veri Madenciliği, Sistem Analizi ve Yönetim Bilişim Sistemleri'dir.