Bulut Tabanlı Yazılımlarda Tasarım Modeli Üzerinden Güvenlik Tehditlerinin Tespiti

Engin DEVECI, M. Ufuk ÇAĞLAYAN
Boğaziçi Üniversitesi, Bilgisayar Mühendisliği Bölümü, İstanbul
engin.deveci@boun.edu.tr, caglayan@boun.edu.tr

Özet

Bulut servislerinin firmalar tarafından kabul görmesi ve kullanım oranı dünya çapında her geçen gün artmaktadır. Bulut bilişim, Türkiye'nin 2023 Bilgi Toplumu stratejisinin de en önemli parçalarından birisidir. Firmalar, bulut uygulamalarını kullanma kararı verirken ve servis sağlayıcı seçerken farklı kriterlere göre hareket ederler. Güvenlik, toplam sahip olma maliyeti ve işletme maliyetleri bunlardan en önemlileridir. Uygulama güvenliğini sağlama almak ve güvenilirliklerini arttırmak adına çeşitli sertifikasyonlara katılmaktadırlar. Bunlardan bazıları SOC 1 (SAS70), SOC 2, SOC 3, ISO/IEC 27001, BITS Shared Assessments, SysTrust, WebTrust, AT Section 101'dir. Bulut tabanlı servis sağlayıcılar, sertifika sürecini geçtikten sonra uygulama içinde ya da çevresel faktörlerde diledikleri gibi değişiklik yapabilmektedirler. Bu nedenle bulut tabanlı uygulamaların sertifikasyonunda tek seferlik güvenlik kontrolü yeterli olmamaktadır ve sürekli gözlemleme ve mümkünse sertifika sürecinin belirli aralıklarla tekrar edilmesi tavsiye edilmektedir. Bu bildiride, sürekli güvenlik sertifikasyonu sürecinde, bulut servis sağlayıcının ve sertifikasyon otoritesinin işlerini kolaylaştırmak amacıyla yeni bir metodoloji ve çeşitli araçlar sunuyoruz. Önerdiğimiz metodoloji ve araçlar, bulut yazılımlarının, tasarım modelleri üzerinden olası güvenlik tehditleri için yarı-otomatik olarak incelenmesini sağlayacak, bu sayede bulut servis sağlayıcıya ve sertifika otoritesine, yazılımda ve çevresel faktörlerde yapılan değişiklikler sonucunda ortaya çıkabilecek yeni güvenlik tehditlerinin bulunmasında yardımcı olacak ve sürekli sertifikasyon sürecinde harcanan zamanı azaltacaktır.

Anahtar Sözcükler: Bulut Yazılım Tasarımı, Güvenlik, Tasarım Modeli, Tehdit Modeli, Güvenlik Açıkları, Güvenlik Tehditleri, UML, OCL, Güvenlik Sertifikaları

Abstract:

Worldwide enterprise cloud adoption rate is climbing up every day. Also in Turkey, cloud is one of the main pillars of 2023 Information Society Strategy. Enterprises are looking at different angles before deciding to use cloud services. Security, total cost of ownership and operational expenses are some of the major decision criteria for cloud adoption. Cloud service providers are getting certifications in order to ensure and present the security of their solutions. There are several certifications, including but not limited to SOC 1 (SAS70), SOC 2, SOC 3, ISO/IEC 27001, BITS Shared Assessments, SysTrust, WebTrust, AT Section 101. However, one-time certification of the application and its environment is not sufficient for cloud ser-vices. Cloud service provider can make changes in the environment, integrations and functionalities of the cloud service after the certification is received. This is why the certified system has to be continu¬ously monitored or sometimes certification process has to be repeated during the lifecycle of the cloud application. In this paper, we propose a new framework to ease the preparations and also the certification process for the cloud applications. Our framework provides tools and methodology to identify possible security threats from the design models of the cloud application. Frame¬work will help the application provider to identify new security threats after any changes are introduced on the application or on the environment and help the certification authority to auto¬mate the threat checking process and hence reduce the time spend on checking the application de¬sign models during the continuous certification process.

1. Giriş

2015'te Türkiye bulut pazarının büyüklüğü 3 milyar TL'ye ulaşmıştır. 2018'de ise bu ra-kamın 6 milyar TL'ye ulaşması öngörülmek-tedir. Market araştırmalarına göre bu pazarın yaklaşık olarak %70'i bulut tabanlı yazılım-lardan oluşmaktadır. Bu verilerden yola çıkarak bulut yazılımların Türkiye'deki işletmeler için önemi anlaşılmakta ve işletmelerin büyük çoğunluğunun önümüzdeki 5-10 yıl içinde iş uygulamaları için bulut tabanlı yazılımları kullanacağı öngörülebilmektedir.

Bulut tabanlı yazılımların güvenliği ve güve-nilirliği firmaların en önemli seçim kriterleri arasındadır. Yazılımdaki güvenlik açıklarının, hizmeti kullanan firmalar için basit veri kaybından rekabet gücünün azalmasına ve hatta firmanın iflasına kadar götürebilecek çok ciddi sonuçları olacaktır. Bu nedenle firmalar alacakları yazılım hizmeti için çeşitli sertifikalar aramaktadırlar. Bu sertifikalardan bazıları SOC 1 (SAS70), SOC 2, SOC 3, ISO/IEC 27001, BITS Shared Assessments, SysTrust, WebTrust, AT Section 101'dir. Bulut yazılımlarının güvenlik sertifikasyonunda en önemli sorunlardan biri bulut yazılımının çalışma ortamının ve yazılım içeriğinin son kullanıcının kontrolünde olmamasıdır. Örneğin bulut yazılım hizmet sağlayıcı firma güvenlik sertifikasını aldıktan sonra yazılım mimarisinde, işlevlerinde veya çevresel fak-törlerde değişikliklere gidebilmekte, bu da yeni güvenlik sorunlarını ortaya çıkarabil-mektedir. Bu nedenle bulut yazılım sertifi-kasyonlarında aranması gereken en önemli özellik

sertifikasyonun tek seferlik yapılma-ması ve sürekli gözlemlemeye dayanmasıdır. Bu nedenle sertifikasyon süreçlerinin müm-kün olduğu kadar kolaylaştırılması ve otomatikleştirilmesi esas olmalıdır.

Bu çalışmada, güvenlik sertifikasyon süreçle-rini desteklemek adına, bulut yazılım mimarilerinin tasarım modelleri üzerinden olası güvenlik tehditleri için yarı-otomatik olarak incelenmesini sağlayacak bir yöntem ve bu yöntemi destekleyen araçları sunmaktayız. Sunduğumuz yöntem yazılım ekosisteminin yoğun bir şekilde kullandığı ve bilgi sahibi olduğu UML ve OCL dillerini kullanmaktadır. Yöntemi destekleyen araç ise Eclipse yazılım geliştirme ortamı üzerinde sunulmuştur. Bu araç, bulut yazılım tasarım modellerini (Sınıf Diyagramı, Kullanım Senaryoları ve Sıralama Diyagramları) girdi olarak almakta, bu modeller üzerinde önceden belirlenmiş OCL sorgularını koşturmakta ve bu sorguların sonuçlarını analiz ederek muhtemel güvenlik tehditlerini belirlemektedir. Bu girdiler için ön koşul ise diyagram unsurlarının yöntem ile birlikte sunduğumuz UML Profili kullanılarak anlamlandırılmış olmasıdır. Çalışmamızın sonraki safhalarında ise belirlenen muhtemel güvenlik tehditleri incelenmekte ve detaylı güvenlik gereksinimlerine dönüştürülmektedir. Bu gereksinimler bulut yazılım geliştirici firma için detaylı tasarım sırasında yol gösterici olarak kullanılabilmektedir. Sunduğumuz bir başka araç vasıtasıyla bulut yazılım geliştirici firmanın hazırladığı alt düzey tasarım modelleri, belirlenmiş olan güvenlik gereksinimleri kullanılarak, model denetlemesi yöntemi ile incelenirler. Bu ince-leme sonucunda, güvenlik gereksinimlerini karşılamak için alt seviye tasarımda ilgili bileşenlerin düzgün olarak eklenip eklenme-diği anlaşılabilmektedir.

Bu bildiride, çalışmamızın ilk aşaması olan bulut yazılım tasarım modelinin yarı otomatik olarak incelenmesi ve muhtemel güvenlik tehditlerinin belirlenmesi için geliştirdiğimiz yöntem ve araçları anlatmaktayız.

2. Temel Bilgiler

UML (Unified Modeling Language) ve UML Profilleri

UML [1], nesneye dayalı tasarımların model-lenmesi için kullanılan bir dildir. Sistemlerin mimarilerini, bileşenlerini, aktivitelerini, entegrasyon detaylarını, kullanım senaryola-rını, etkileşimlerini ve daha birçok statik ve dinamik özelliklerini modellemek için yazılım analistleri, tasarımcıları, geliştiricileri ve test ekipleri tarafından yoğun bir şekilde kullanılmaktadır.

Gerektiği durumlarda UML, güvenlik, güve-nilirlik, vb. farklı alanlar için genişletilebil-mekte ve özelleştirilebilmektedir. İki farklı genişletme mekanizması bulunmaktadır:

- UML profillerinin kullanılması [2]
- Yeni meta-sınıflar ve meta-ilişkiler kullanılarak yeni bir UML lehçesinin oluşturulması

Bu çalışmada, daha yaygın kullanımı ve araç desteğinin olması nedeniyle UML profillerini, bulut uygulamalarının güvenlik özelliklerinin modellenmesi için seçtik ve kullandık.

OCL (Object Constraint Language)

OCL [3], UML modelleri üzerinde koşturula-bilecek olan sorguları ve yine aynı modeller üzerinde uygulanabilecek kısıtlama ve kont-rolleri resmi olarak tanımlayabilmek için geliştirilmiş bir dildir. OCL tanımları komut ve ifadeler olarak yazılırlar. Örnek bir OCL ifadesi aşağıda gösterildiği gibidir:

```
self.allOwnedElements() ->
  select ( a | a.oclIsTypeOf( uml::Class ) ) ->
  forAll ( b | b.getAppliedStereotypes().name ->
    intersection( Set (
      'SoftwareComponent'
      'Database',
      'Asset' ) ) -> size() > 0 )
```

Şekil 1: OCL ifadesi

Bu çalışmada, güvenlik açıklarının otomatik olarak belirlenebilmesi için, UML tasarım modelleri üzerinde, hazırladığımız çeşitli OCL ifadelerinin koşturulması üzerine kurulu bir metodoloji sunmaktayız.

OWASP (Open Web Application Security Project)

OWASP [4] web uygulamalarının güvenliğine adanmış bir online topluluktur. Bu topluluk, tüm dünyadan çeşitli firmalar, eğitim kurumları ve bireylerden oluşmuştur. Toplu-luğun amacı web uygulama güvenliği hak-kında ücretsiz olarak ulaşılabilecek makale, metodoloji, dokuman, araç ve teknolojiler oluşturmak ve yazılım geliştiricileri, tasarım-cıları ve işletmeleri web uygulamalarında karşılaşılabilecek en yaygın güvenlik açıkları konusunda uyarmaktır.

Bu organizasyon oldukça popüler olan web uygulamaları için ilk 10 güvenlik açığı ve bu açıklara karşı savunma önerileri listesinin de yayıncısıdır.

Bu çalışmada, OWASP'ın yayınladığı yaygın güvenlik açıklarından bazılarını ele aldık ve önerdiğimiz araç ve metodolojileri kullanarak bu açıkların bulut uygulamalarının tasarım modelleri üzerinden nasıl tespit edilebileceğini gösterdik.

Tehdit Modelleme

Tehdit modelleme, uygulamaların güvenliği-nin incelenmesi için kullanılan bir yöntemdir. Bu yapısal yaklaşım, uygulamaların ilişkili olabileceği güvenlik

tehditlerinin bulunması, ölçümlenmesi ve irdelenmesi için yardımcı olur.

Tehdit modelleme, kod incelemesi ile ilgili bir yöntem değildir, ancak, kod inceleme sürecini tamamlayıcı bir yaklaşımdır. Tehdit modellemenin yazılım geliştirme sürecinde tasarım aşamasında kullanılması, uygulamadaki güvenlik tehditlerinin önceden belirlenmesini ve tedbirlerin en baştan alınmasını sağlamaktadır.

Üç farklı tehdit modelleme yaklaşımı vardır:

- Saldırgan – merkezli: Bu yaklaşım bir saldırgan ile başlar, saldırganın hedeflerini inceler ve bu hedeflere nasıl ulaşabileceğine bakar.
- Tasarım – merkezli: Bu yaklaşım sistem tasarımı ile başlar, sistem ta-sarım modeli üzerindeki her elemana karşı olabilecek saldırılara bakar.
- Varlık – merkezli: Bu yaklaşım sis-temin ilgili olduğu tüm varlıklar (sis-temde işlenen, barındırılan veya ak-tarılan veriler) ile başlar, varlıkların önem derecesine ve saldırganların bu varlıklara ulaşma isteklerine göre tehditler sınıflandırılır.

Bu çalışmadaki tehdit modelleme yaklaşımı-mız tasarım merkezli olup varlık merkezli yaklaşımın da çeşitli öğelerini içinde bulun-durmaktadır.

3. Önceki Çalışmalar

Yazılım tasarım modelleri üzerinde güvenlik tehditlerinin araştırılması günümüzde bazı yazılım geliştirme metodolojilerinin içinde yer almaktadır. Microsoft'un Güvenlik Geliş-tirme Yaşam Döngüsü (SDL) [5] buna örnek olarak gösterilebilir. Bu yazılım geliştirme süreci, daha güvenli yazılımların geliştirilmesi ve güvenlik kriterlerine uyumluluk konula-rında yardımcı olmakta ve geliştirme maliyetlerini azaltmaktadır. Microsoft Threat Modeling Tool [6] bu süreç içerisinde kullanmak amacıyla geliştirilmiş bir araçtır ve yazılım tasarım modellerini inceleyerek güvenlik tehditlerini önceden bulmayı amaçlamaktadır. Bu konuda geliştirilmiş diğer bir araç MyAppSecurity Threat Modeller'dır. Ancak bu araçlar kendilerine özgü bir modelleme gösterimi kullanmakta, UML modelleri üzerinde çalışmamaktadır ve bu nedenle sıklıkla kullanılan yazılım geliştirme yöntem ve araçları ile entegre olamamaktadırlar.

Johnstan da [8] bu konuda çalışma yapan araştırmacılardan birisidir ve güvenli yazılım geliştirme sürecinde güvenlik tehditlerinin araştırılması için DFD (Data Flow Diagram) şemaları yerine UML aktivite şemalarının kullanılmasını önermiştir. Ancak UML, mev-cut hali ile yazılımların güvenlik özelliklerinin ve ihtiyaçlarının modellenmesi için yeterli değildir. Bu nedenle araştırmacılar güvenlik odaklı yeni UML profilleri önermişlerdir ve hatta OMG bu konuda yeni bir RFP (UML Threat and Risk Model) [9] yayınlamıştır. Bu RFP'ye yanıt vermek için çeşitli organizas-yonlar ve çalışma grupları kurulmuştur. Aktif olarak çalışan gruplardan birisi de "Threat & Risk Community"dir [10]. Bu RFP süreci henüz tamamlanmadığı için bu çalışmada biz de yeni bir UML Profili önermekteyiz. Süreç tamamlandığında ve OMG standardı oluştu-ğunda çalışmamızı güncelleyip araçlarımızı yeni standarda göre adapte edeceğiz.

Wang ve arkadaşları [11] ve Abi-Antoun ve arkadaşları da [12] DFD şemaları kullanarak benzer bir çalışma yapmışlardır. DFD şema-larını çeşitli eklentiler ile güvenlik özellikleri açısından genişletmişler ve sistemdeki güven-lik tehditlerinin bulunması için kullanmışlar-dır. Bu çalışmada bizim amacımız DFD ya da diğer standart olmayan gösterim biçimlerinin yerine UML'i ve OCL'i kullanmak, ve genişlettiğimiz bu gösterim biçiminin mevcut yazılım geliştirme ve güvenlik sertifikasyon süreçlerinin bir parçası olmasına olanak sağlamaktır.

UML yazılım geliştirme süreçlerinde çok yoğun olarak kullanılan bir dildir. Son yıllar-da güvenli yazılım geliştirme süreçlerinin de bir parçası haline gelmiştir. UML'in yazılım-ların güvenlik ihtiyaçlarının ve özelliklerinin modellenmesinde kullanılması amacıyla çok çeşitli profil önerileri bulunmaktadır. Bu önerilerin en önde gideni Jurjens'in önerdiği UMLsec'tir [14]. Peralta ve arkadaşları da [15] UML'i çeşitli güvenlik ile alakalı şab-lonlar kullanarak genişletmişler ve bu yeni gösterimi UML modellerinden otomatik ola-rak test vakaları üretmek için kullanmışlardır.
Guadalo ve Seret [16] ise bilgi sistemleri güvenliği üzerinde çalışmış, UML'i gizlilik, güven ve rol tabanlı erişim kontrolü şablonla-rı ile genişletmişlerdir. Villarroel ve arkadaş-ları [17] veri ambarlarında veri gizliliği prob-lemleri üzerinde incelemeler yapmış ve "Se-cure Data Warehouses (SECDW)" adında yeni bir UML profili önermişlerdir. Ayrıca bu çalışmada OCL'i de yeni veri tipleri ile geniş-letmiş ve aktif olarak kullanmışlardır.

Sohr ve arkadaşları [18] rol bazlı erişim kont-rolü için UML/OCL tabanlı bir yetki motoru ve doğrulama çerçevesi geliştirmiş ve bunu web servislerinin güvenlik poliçeleri için kullanmışlardır. Pavlich ve arkadaşları [19] UML'in erişim kontrolü için biçimsel olarak yeterli olmadığını belirtmiş ve rol bazlı, isteğe bağlı ve zorunlu erişim kontrolleri için yeni şemalar önermişlerdir. Lodderstedt ve arkadaşları [20] de erişim kontrolü amacıyla yeni bir UML profili (secureUML) önermiş-lerdir.

Poniszewska Maranda [21] rol bazlı erişim kontrolü konusuna farklı bir açıdan yaklaş-mış, yeni bir UML profili kullanmadan gü-venlik kısıtlarının standart UML modeli üzerinde gösterimi üzerine çalışmıştır. Po-niszewska
Maranda bu çalışmada UML ile birlikte OCL de kullanmıştır. Fernandez Medina ve arka-daşları da

[22] OCL kullanan araştırmacılar-dandır. OCL'i genişleterek yeni bir dil (Object Security Constraint Language - OSCL) önermiş ve bunu güvenli veri tabanı tasarımı konusunda kullanmışlardır.

Mehr ve Schreirer [23] servis odaklı mimari-lerde güvenlik ihtiyaçları konusunda çalış-mışlar ve mevcut yaklaşımların tasarım sırasında mesaj güvenliği konusunun ele alınmasında yeterli olmadığını düşünmüşlerdir. Bu amaçla UML profillerini, UML şablonlarını ve OCL'i incelemişler ve sonuçta mesaj güvenlik ihtiyaçları için modelden ayırdıkları bir mesaj güvenlik poliçesi dosyası önermişlerdir. Bu poliçeyi de modele genişletilmiş OCL kullanarak entegre etmişlerdir.

Buchholtz ve arkadaşları [24], UML şemala-rını "process calculus"e çevirme amacıyla bir UML profili (For-LySa) yaratmışlardır. Daha sonra bu veriyi otomatik kimlik doğrulaması analizi için kullanmışlardır. Peterson ve arkadaşları [25] UML sınıf şemalarının içine güvenlik ihtiyaçlarını gömmek amacıyla yeni bir UML profili önermişlerdir. Breu ve arkadaşları [26] ve Alam ve arkadaşları da [27] servis odaklı mimarilerde güvenlik ihtiyaçlarının modellenmesi için SECTET ve SECTET-PL UML profillerini önermişlerdir.

Bu bölümde anlatılan önceki çalışmalardan da anlaşılacağı gibi günümüzde model tabanlı güvenlik tasarımı ve analizi amacı ile kullanılabilecek standart diller ve araçlar bütünü bulunmamaktadır. Bu nedenle, bu çalışmada yazılım tasarımında en çok kabul gören UML ve OCL üzerine odaklandık ve sunduğumuz araçları Eclipse gibi açık kaynak kodlu platformlar üzerinde geliştirdik.

4. Güvenlik Tehditlerinin Tespiti

Tehdit Tespit Aracı

Tehdit Tespit Aracı, Eclipse yazılım geliştir-me ortamına bir eklenti olarak geliştirilmiştir. Gerektiğinde farklı tasarım ortamlarında hazırlanan UML modelleri üzerinde de çalı-şabilmektedir. Özetle, Tehdit Tespit Aracı girdi olarak UML tasarım modellerini almak-ta ve çıktı olarak muhtemel güvenlik tehdit listesini vermektedir. Girdi olarak beklenen dört farklı UML şeması bulunmaktadır:

- Sınıf Şeması: Yazılımın analiz mo-delinde bulunan standart sınıf şemasıdır.
- Kritik Varlık Şeması: Sınıf şeması-nın önerilen UML profili ile zengin-leştirilmiş şeklidir. Kritik varlıklar "Asset" şablonu ve ilgili etiketli de-ğerler ile zenginleştirilmiştir.
- Kullanım Senaryosu Açıklamaları: Sıralama şemalarının UML profili ile zenginleştirilmiş şeklidir. Sistem bileşenleri arasındaki veri akışını göstermesi için "ConveyedInformation" şablonu ve ilgili etiketli değerler ile zenginleşti-rilmiştir.
- İletişim Şeması: Sınıf Şemasının ve-ri iletişim yolları ve kullanılan pro-tokolleri göstermesi amacıyla öneri-len UML profili şablonları (System-Component, Database, Communication, vb.) ve ilgili etiketli değerler ile zenginleştirilmiş şeklidir.

Yukarıdaki şemalarda kullanılan UML Profi-line ait şablonlar ve etiketli değerlerin bir alt kümesi gelecek bölümde (Örnek UML Profili) anlatılmıştır.

Bahsedilen dört farklı şema Tehdit Tespit Aracı'na girilir. Tehdit Tespit Aracı öncelikle girişen şemaların eksiksiz ve hatasız olup olmadığını kontrol eder. Bunun için modeller üzerinde önceden belirlenmiş olan OCL kısıt-larını koşturur. Eğer şemalardan herhangi birinde hata varsa, Tehdit Tespit Aracı gere-ken uyarıyı kullanıcıya iletir. Girilen şemalar eğer hatasız ve eksiksiz ise, Tehdit Tespit Aracı, şemaları kritik varlıklar, sistem mima-risi, veri akışı vb. özelliklerine göre inceler ve muhtemel güvenlik tehditlerini belirler.

Tehdit Tespit Aracı tehditleri belirlemek için tehdit kütüphanesine ihtiyaç duyar. Tehdit kütüphanesi muhtemel tehditlerin açıklamalarının, hangi durumlarda tehdidin ortaya çıkabileceğinin ve diğer bilgilerin bulundurulduğu bir kütüphanedir. Tehdit kütüphanesi gelecek bölümde (Tehdit Kütüphanesi) detaylı olarak anlatılmıştır.

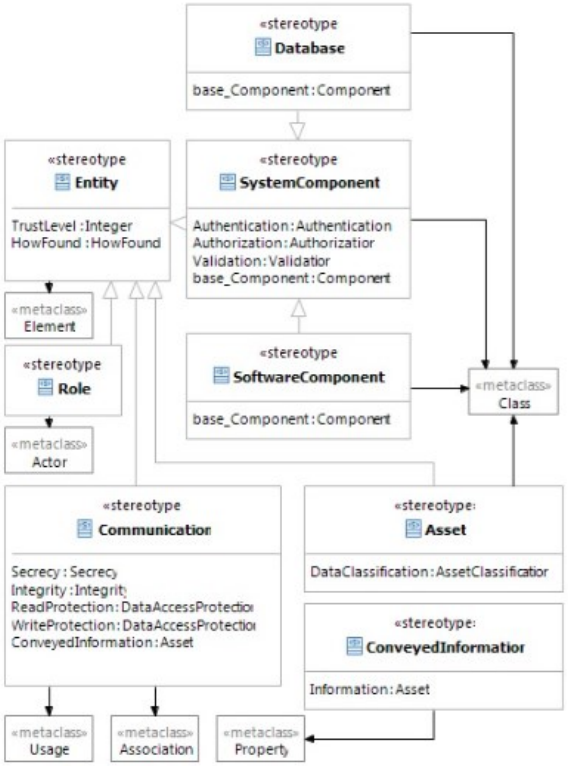
Tehdit kütüphanesinde bulunan her bir tehdit için var olma koşulları bulunmaktadır. Bu koşullar kütüphanede OCL sorguları olarak saklanmaktadır. Tehdit Tespit Aracı, kütüp-hanede bulunan tüm tehdit var olma koşulla-rını (OCL sorguları) girilen tüm UML şema-ları üzerinde çalıştırarak hangi tehditlerin var olma ihtimali olduğunu belirler. Kütüphanede bulunan OCL sorgularından bazıları gelecek bölümde (Örnek OCL Sorguları) detaylı olarak anlatılmıştır.

OCL sorguları koşturulduktan ve muhtemel tehditler bulunduktan sonra, Tehdit Tespit Aracı tehdit listesini analiste sunar. Bu tehdit-ler için karşı tedbirler otomatik olarak öneri-lebilir ve son olarak güvenlik ihtiyaçlarına kadar ayrıştırılabilir. Analiz ve tasarımcılar bu karşı tedbirleri ve ihtiyaçları alıp yazılım tasarımına, daha sonra da koda yansıtmalıdırlar.

Tehdit Kütüphanesi

Tehdit kütüphanesi yaklaşımı daha önce Saeki ve arkadaşları [28] tarafından kullanıl-mıştır. Bu çalışmada biz de aynı yaklaşımı kullanarak tehdit kütüphanesini E-COFC (Extended Commercially Oriented Functionality Class) ECMA-271'de [29] listelenen güvenlik tehditleri ile doldurduk ve OWASP'ın ilk 10 güvenlik açığı listesi ile hizaladık. Mevcut durumda tehdit kütüphanesinde sadece güvenlik tehditleri değil, o güvenlik tehditleri için karşı tedbirler ve güvenlik gereksinimleri de saklanabilmektedir.

Örnek UML Profili
Önerdiğimiz UML profilinin bir alt kümesi aşağıdaki şekilde gösterilmiştir.



Şekil 2: Örnek UML Profili

Bu UML profili, Wang ve arkadaşlarının [11] ve Abi-Antoun ve arkadaşlarının [12] DFD şemaları üzerinde yaptıkları çalışmalardan esinlenerek hazırlanmıştır. Araştırmacılar makalelerinde DFD şemalarını çeşitli özellik-ler ile genişletmiş ve bu genişletilmiş şemaları güvenlik tehditlerinin bulunmasında kul-lanmışlardır. Bu çalışmada biz DFD yerine standart UML kullanmakta ve UML'i yukarıdaki şekilde alt kümesi verilen UML profilini kullanarak genişletmekteyiz.

UML profilindeki şablonlar, çoğu zaman birçok etiketli değer içerirler. Bu etiketler, tasarımcıya şablonun yanı sıra ekstra anlamlar ve özellikler yükleme şansı verir.

UML profilindeki tüm şablonlar Entity şablo-nundan türetilmiştir. Entity şablonu UML Element sınıfından türetilmiştir. İki adet etiketli değer içerir:

- TrustLevel: Sistem açısından Entity'nin ne kadar güvenilir oldu-ğunu gösterir
- HowFound: Sistem açısından Entity'nin nasıl ulaşıldığını gösterir

"Asset" şablonu tehlikelere karşı korunması gereken önemli varlıkları işaretlemek için kullanılır. Bu şablon UML Class üst-sınıfından türetilmiştir. Bir adet etiketli değer içerir:

- DataClassification: Varlığın önem derecesini gösterir

"Role" şablonu kullanım senaryoları açısın-dan dış aktörleri gösterir. Bu şablon UML Actor üst-sınıfından türetilmiştir. Bu aktörler insan ya da sistem ile entegrasyonu olan diğer dış sistemler olabilir. Bu şablonun etiketli değerleri yoktur.

"Communication" şablonu aktörler ve sistem bileşenleri arasındaki ilişkileri gösterir. Bu şablon UML Association ve Usage üst-sınıflarından türetilmiştir. Dört adet etiketli değer içerir:

- Secrecy: Varlıkların sistem bileşen-leri ve aktörler arasında ne şekilde transfer edildiğini gösterir. Örneğin, açık metin, şifreli, vb.
- Integrity: Varlıkların transferi sıra-sında bütünlüğün korunması için bir yöntem olup olmadığını gösterir
- ReadProtection: Varlıkların transfer sırasında okunabilir olup olmadığını gösterir
- WriteProtection: Varlıkların transfer sırasında yazılabilir olup olmadığını gösterir

"SystemComponent" şablonu sistemin bile-şenlerini işaretlemek için kullanılır. Sistem bileşenleri sunucular, veri tabanları, uygula-malar, uygulama bileşenleri, servisler, vb. olabilir. Bu şablon UML Component üst-sınıfından türetilmiştir. Üç adet etiketli değer içerir:

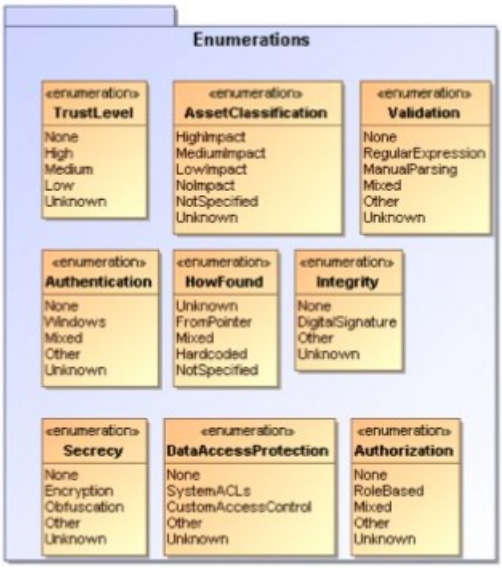
- Authentication: Sistem bileşeninin diğer sistem bileşenleri ve aktörler ile iletişimi sırasında hangi kimlik doğrulama mekanizmasını kullandığını gösterir
- Authorization: Sistem bileşeninin diğer sistem bileşenleri ve aktörler ile iletişimi sırasında hangi yetkilen-dirme mekanizmasını kullandığını gösterir.
- Validation: Sistem bileşeninin diğer sistem bileşenleri ve aktörler ile ile-tişimi sırasında aktarılan verilerin hangi doğrulama mekanizmasını kullandığını gösterir.

"SoftwareComponent" şablonu "System Component" şablonundan türetilmiştir. Bu şablon, bileşenin bir uygulama, web servisi, vb. yazılım bileşeni olduğunu gösterir.

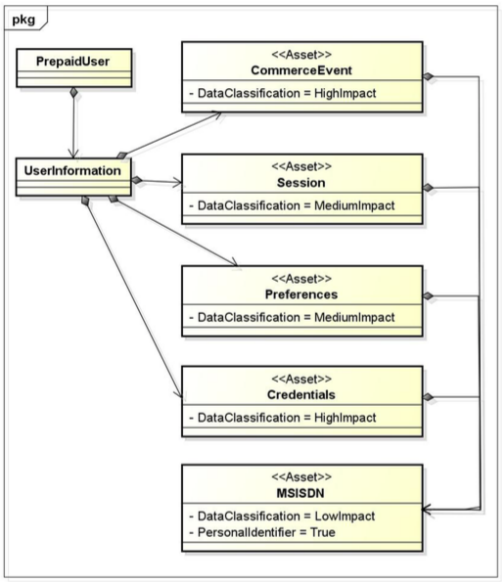
"Database" şablonu "SystemComponent" şablonundan türetilmiştir. Bu şablon, bileşe-nin, varlıkların saklandığı bir veri tabanı olduğunu gösterir. "Communication" şablo-nunda olduğu gibi dört adet etiketli değer içerir:

- Secrecy
- Integrity
- ReadProtection
- WriteProtection

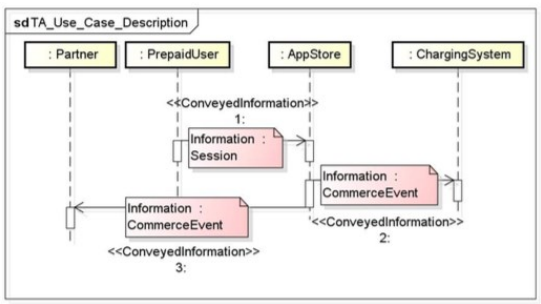
Etiketli değerlerin kullandığı ayrıntılı listeler de aşağıdaki şekilde gösterilmiştir.



Örnek bir kritik varlık şeması aşağıdaki şe-kilde sunulmuştur. Bu şemada "Asset" şablo-nu ve ilgili etiketli değerler kullanılmıştır.



Aşağıdaki şekilde ise örnek bir kullanım şeması açıklaması (sıralama şeması) sunul-muştur. Bu şemada ise "ConveyedInforma-tion" şablonu ve ilgili etiketli değerler kulla-nılmıştır.



Örnek OCL Sorguları

Tehdit Tespit Aracı'nın karar verme meka-nizması OCL sorgularına dayanmaktadır. OCL sorguları ayrıca Tehdit Tespit Aracı'na girdi olarak sunulan UML şemalarının doğru-lanmasında da kullanılmaktadır.

```
inv: self.allOwnedElements() ->
select (a | a.oclIsTypeOf(uml::Class)) ->
forAll (b | b.getAppliedStereotypes().name ->
   intersection (Set'SoftwareComponent','Database','Asset') ->
   size() > 0
)
```

Örneğin, yukarıdaki şekilde sunulan OCL sorgusu analiz modelindeki tüm sınıflar için doğru UML profil şablonlarının ("SoftwareComponent", "Database" ve "Asset" şablonları) kullanılıp kullanılmadığı-nı kontrol eder.

Aşağıdaki şekildeki OCL sorgusu ise analiz modelindeki ilişki "Communication" şablonu ile işaretlenmiş ise sadece bir ucunun "ConveyedInformation" şablonu ile işaret-lenmiş olup olmadığını ve bu şablonun "In-formation" etiketinin değerinde "Asset" sını-fından bir obje olup olmadığını kontrol eder.

```
inv: self.allOwnedElements()->
select(a | a.oclIsTypeOf(uml::Association)) ->
forAll (b | b.getAppliedStereotypes().name ->
   includes ('Communication') implies
   b.oclAsType (uml::Association).memberEnd ->
   select (p | p.getAppliedStereotypes().name ->
      includes ('ConveyedInformation')) ->
   size()=1and b.oclAsType(uml::Association).memberEnd ->
   select (p | p.getAppliedStereotypes().name ->
      includes ('ConveyedInformation')) ->
    first().oclAsType (Property).
   extension_ConveyedInformation.Information.
   oclIsTypeOf (ThreatAnalysis::Asset)
)
```

Tehditlerini tespit edilmesini sağlayan karar mekanizmasında kullanılan OCL sorgularına bir örnek aşağıdaki şekilde verilmiştir.

```
package uml context Package
inv: self.allOwnedElements() ->
  select (a | a.oclIsTypeOf(uml::Association)) ->
  select (a | a.getAppliedStereotypes().name->
    includes('Communication')) ->
  forAll (a | a.oclAsType(Association).memberEnd ->
    select (p | p.getAppliedStereotypes().name ->
      includes ('ConveyedInformation')) ->
    first().type.oclAsType (Class).
    extension_SystemComponent.TrustLevel >
    a.oclAsType (Association).memberEnd ->
    select (p | not p.getAppliedStereotypes().name ->
      includes ('ConveyedInformation')) ->
    first().type.oclAsType (Class).
    extension_SystemComponent.TrustLevel
    implies a.oclAsType (Association).memberEnd ->
    select (p | p.getAppliedStereotypes().name ->
      includes ('ConveyedInformation')) ->
    first().oclAsType (Property).
    extension_ConveyedInformation.
    Information.oclAsType (ThreatAnalysis::Asset).
    DataClassification =
    ThreatAnalysis::AssetClassification::NoImpact
  )
Endpackage
```

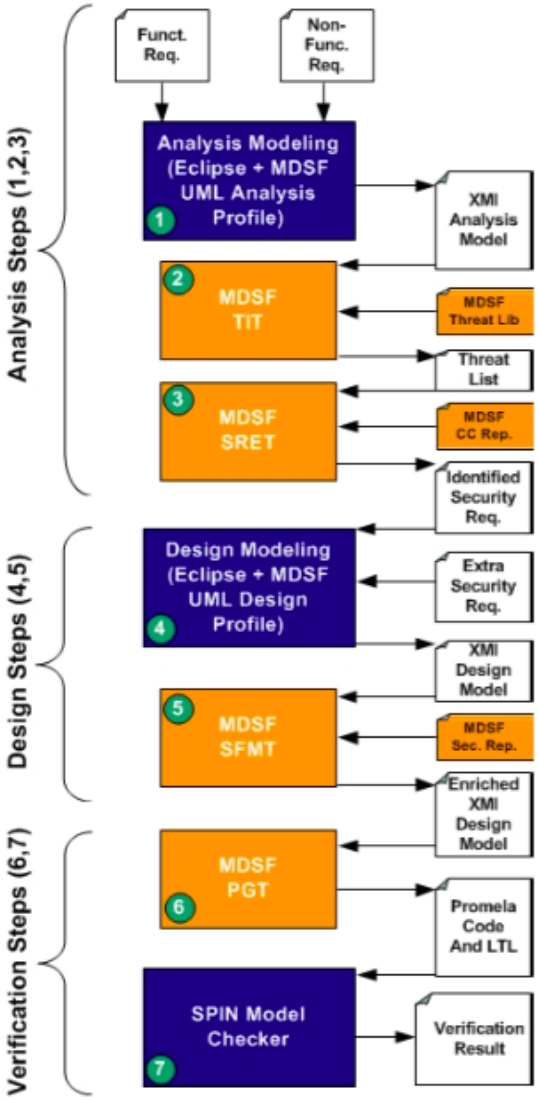
Yukarıdaki OCL sorgusu UML şemasındaki tüm ilişkiler üzerinde çalışır. Eğer ilişki sınıfı "Communication" şablonu ile işaretlenmiş ise ilişkinin iki bacağındaki sistem bileşenlerini ve aktörleri inceler. Eğer ilişkinin kaynak bacağındaki bileşen hedef bacağındaki bile-şenden daha yüksek güven seviyesine sahipse sorgu "evet" sonucunu döner. Diğer bir deyişle, eğer kritik bilginin alıcısı bilginin göndereninden daha az güvenilir ise bilginin ifşası sorunu ortaya çıkabilir.

```
package uml context Package
inv: self.allOwnedElements() ->
select (a | a.oclIsTypeOf (uml::Association)) ->
select (a | a.getAppliedStereotypes().name->
  includes ('Communication')) ->
forAll (a | a.oclAsType(Association).
  extension_Communication.TrustLevel >
  a.oclAsType (Association).memberEnd ->
  select (p | p.getAppliedStereotypes().name->
    includes ('ConveyedInformation')) ->
  first().type.oclAsType (Class).extension_SystemComponent.
  TrustLevel implies a.oclAsType (Association).memberEnd ->
  select (p | p.getAppliedStereotypes().name ->
    includes('ConveyedInformation')) ->
  first().oclAsType(Property).
  extension_ConveyedInformation.Information.oclAsType
  (ThreatAnalysis::Asset).DataClassification =
  ThreatAnalysis::AssetClassification::NoImpact
)
Endpackage
```

Yukarıdaki OCL sorgusu ise yanıltma (spoo-fing) tehdidinin bir göstergesidir. OCL sorgusu UML şemasındaki tüm ilişkiler için çalışır ve eğer ilişki "Communication" şablonuna sahipse, ilişkinin ve ilişki kaynağının güven seviyelerini karşılaştırır. Eğer ilişkinin güven seviyesi kaynaktan daha yüksek ise sorgu "evet" sonucunu döner. Diğer bir deyişle, eğer bilginin göndereni iletişimin sağlandığı hattan daha az güvenilir ise yanıltma tehdidi ortaya çıkabilir.

5. Yazılım Tasarımı ve Güvenlik İnceleme-si için Model Tabanlı Güvenlik Çerçevesi

Tehdit Tespit Aracı, güvenli yazılım geliştir-mek için tasarlanan daha büyük bir çerçeve-nin ilk parçasıdır. Aşağıdaki şekil bu çerçe-veyi ana hatları ile anlatmaktadır.



- Birinci Aşama: Önerilen UML profili kullanılarak yazılımın analiz modeli oluşturulur
- İkinci Aşama: Analiz modeli Tehdit Tespit Aracı'na girilir ve muhtemel güvenlik tehditleri belirlenir
- Üçüncü Aşama: Belirlenen tehditler Güvenlik Gereksinimleri Ayrıştırma Aracı'na girilir ve belirlenen tehdit-lere karşı tedbir oluşturmak için ge-rekli olan güvenlik gereksinimleri belirlenir
- Dördüncü Aşama: Yazılım tasarım-cıları önerilen UML profilini de kul-lanarak ve güvenlik ihtiyaçlarını göz önüne alarak tasarım modelini geliş-tirirler
- Beşinci Aşama: Güvenlik Fonksiyo-nu Birleştirme Aracı kullanılarak daha önceden test edilmiş olan gü-venlik özellikleri ve fonksiyonları tasarıma birleştirilebilir
- Altıncı Aşama: Tasarım modeli mo-del incelemesi yöntemi için PROMELA diline çevrilir
- Yedinci Aşama: PROMELA'ya çev-rilen tasarım modeli SPIN aracı ile kontrol edilir. Hatalı/eksik olarak ta-sarlanan güvenlik özellikleri elde bulunan güvenlik gereksinimleri kul-lanılarak ortaya çıkarılır

6. İleri Çalışmalar ve Öneriler

Bu bildiride, çalışmamızın ilk aşaması olan bulut yazılım tasarım modelinin yarı otomatik olarak incelenmesi ve muhtemel güvenlik tehditlerinin belirlenmesi için geliştirdiğimiz yöntem ve araçları anlattık. Çalışmamızın kaynağında güvenlik tehditlerinin ortaya çıkarılması için kullanılan OCL sorguları ve UML profili bulunmaktadır. Güncel durumda kullanılan UML profili ve OCL sorguları ancak bir kısım güvenlik tehditleri için hazırlanmıştır. Çalışmamızın sonraki aşamalarında UML profilini ve OCL sorgularını bulut yazı-lımlarda karşılaşılabilecek tüm güvenlik tehditlerini kapsayacak şekilde geliştirmeyi planlamaktayız.

7. Kaynaklar

[1] "OMG UML Specifications", http://www.omg.org/spec/UML/.
[2] "UML Profile Diagrams", http://www.uml-diagrams.org/profile-diagrams.html.
[3] "OMG OCL Specifications", http://www.omg.org/spec/OCL/.
[4] "OWASP: The Open Web Application Security Project", https://www.owasp.org.
[5] "Secure Development Lifecycle", http://www.microsoft.com/en-us/sdl/.
[6] "Microsoft SDL Threat Modelling Tool", https://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx.
[7] "MyAppSecurity Threat Modeller", http://myappsecurity.com.
[8] Johnston, M., N., "Threat Modelling with Stride and UML", Proceedings of the 8th Australian Information Security Management Conference, 2010.
[9] "UML Threat and Risk Model RFP", http://www.omg.org/hot-topics/threat-modeling.htm.
[10] "Threat and Risk Community", http://threatrisk.org.
[11] Wang, D. and P. Torr, "Checking Threat Modeling Data Flow Diagrams for Imple- mentation Conformance and Security", ASE '07: Proceedings of the twenty-second IEEE/ACM international conference on Au-tomated software engineering, pp. 393– 396, 2007.
[12] Abi-Antoun, M. and J. M. Barnes, STRIDE-based security model in Acme, Tech- nical Report CMU-ISR-10-106 , Tech. rep., Carnegie Mellon Univ., 2010.
[13] Jurjens, J., "UMLsec: Extending UML for secure systems development", UML 2002 - The Unified Modeling Language, Vol. 2460, pp. 412–425, Springer, 2002.
[14] Jurjens, J., Secure Systems Development with UML, Springer, 2003.
[15] Peralta, K. P., A. M. Orozco, A. F. Zorzo and F. M. Oliveira, "Specifying Security Aspects in UML Models", Proceedings of the Workshop on Modeling Security, 2008.
[16] Goudalo, W. and D. Seret, "Toward the Engineering of Security of Information Sys-tems (ESIS): UML and the IS Confidentiali-ty", The

Second International Con- ference on Emerging Security Information, Systems and Technologies, 2008.
[17] Villarroel, R., E. Fernandez-Medina, M. Piattini and J. Trujillo, "A UML 2.0/OCL Extension for Designing Secure Data Ware-houses", Journal of Research and Prac- tice in Information Technology, Vol. 38, No. 1, February 2006.
[18] Sohr, K., T. Mustafa, X. Bao and G. J. Ahn, "Enforcing Role-Based Access Con- trol Policies in Web Services with UML and OCL", Annual Computer Security Applica-tions Conference, 2008.

[19] Pavlich-Mariscal, J., L. Michel and S. Demurjian, "Enhancing UML to Model Cus-tom Security Aspects", Proceedings of Aspect Oriented Modeling, 2007.
[20] Lodderstedt, T., D. Basin and J. Doser, "SecureUML: A UML-Based Modeling Lan-guage for Model-Driven Security", Lecture notes in computer science, 2002.
[21] Poniszewska-Maranda, A., "Security Constraints in Access Control of Information System Using UML Language", Proceedings of the 15th IEEE International Work- shops on Enabling Technologies:Infrastructure for Collaborative Enterprises, 2006.
[22] Fernandez-Medina, E., M. Piattini and M. A. Serrano, "Specification of Security Constraint in UML", IEEE 35th International Carnahan Conference on Security Technolo-gy, pp. 163–171, 2001.
[23] Mehr, F. and U. Schreirer, "Modelling of Message Security Concerns with UML", 9th International Conference on Enterprise In-formation Systems, 2007.
[24] Buchholtz, M., C. Montangero, L. Per-rone and S. Semprini, "For-LySa: UML for Authentication Analysis", Global Computing: IST/FET International Workshop, 2004.
[25] Peterson, M. J., J. B. Bowles and J. M. Eastman, "UMLpac: An Approach for Inte-grating Security into UML Class Design", Proceedings of IEEE, pp. 267–272, March 2005.
[26] Alam, M., M. Hafner and R. Breu, "A constraint based role based access control in the SECTET a model-driven approach", Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Tech- nologies and Business Services, 2006.
[27] Breu, R., M. Hafner, F. Innerhofer-Oberperfler and F. Wozak, "Model-Driven Secu- rity Engineering of Service Oriented Systems", Information Systems and eBusi-ness Technologies, Vol. 5, pp. 59–71, 2008.
[28] Saeki, M. and H. Kaiya, "Using Com-mon Criteria as Reusable Knowledge in Se- curity Requirements Elicitation", Modeling Security Workshop, Models 08 , 2008.
[29] Association, E. C. M., Extended Com-mercially Oriented Functionality Class (E-COFC), Standard ECMA-271, ECMA Tech-nical Report TR/78, Tech. rep., ECMA, 1999.