Yazılım Güvenliği Üzerine Bir İnceleme

Seda YILMAZ ve Şeref SAĞIROĞLU

***Özet*— Artık birçok şirket, kurumsal işlemlerini, diğer iş ortaklarıyla iletişimlerini ve yürüttükleri işlemleri web uygulamaları üzerinden geliştirmektedir. Hatta ülkeler e-devlet projelerini hızla devreye sokmakta ve hizmetlerini bu platform üzerinden vermektedir. Web uygulamalarının bu kadar aktif bir şekilde kullanılması işlemlerin güvenliğinin daha çok sağlanması ihtiyacını da beraberinde getirmiştir. Bu çalışmada web uygulamalarında güvenli yazılım geliştirme kuralları, modelleri, güvenli bir yazılımda olması gereken ilkeler, web yazılımlarına yönelik tehditler ile kurumsal olarak güvenli yazılım geliştirme döngüleri incelenmiş ve bunu arttırmaya yönelik çözüm önerileri sunulmuştur.**

***Anahtar Terimler*— Yazılım, web, yazılım güvenliği, tehdit modelleme**

***Abstract*— Many companies run their businesses, operations, branches and businesses with other partners, communications and transactions conducted over web applications. Even the countries involved into e-government projects and services over this platform is rapidly activated. Rapid grow within web application usage bring with the needs to ensure the safety and security of operations and information. In this paper, secure software development, web applications and their rules, models and principles in developing secure are reviewed and discussed in terms of threats and secure enterprise software development cycles. Some suggesstions are given in terms of security and safety of operations.**

***Index Terms*— Software, web applications, software security life cycle, threat modeling**

I. GİRİŞ

Bilgisayarın icat edilmesinin ardında onun yönetilmesi için bir dizi komuta ihtiyaç duyulmuştur. Başlangıçta bazı parçaların yerine yeni kodlu parçaların yerleştirilmesiyle yani fiziksel müdahale ile bu ihtiyaç giderilemeye çalışılmıştır. Zamanla işlemlerin ve ihtiyaçların artmasıyla bu fiziksel müdahalenin dışında bir sistem geliştirilmesi zaruri hale gelmiş ve yazılım geliştirme süreci 1956 yılında IBM tarafından Fortran programlama dilinin geliştirilmesiyle başlamıştır [3]. İnternetin temelinin atıldığı 1960'lı yıllardan sonra yazılım sektörü için yeni bir çağ başlamıştır. Yıllar içinde yaygınlaşan internet ağı zamanla yazılım uygulamalarını yerel bilgisayara bağımlı olmaktan çıkarmıştır. Firmalar, kurumlar ve hatta kişisel kullanıcılar internetten faydalandıkça, uygulamalar web ortamına taşınmaya başlamış, bu durum beraberinde web yazılımların güvenliği kavramını ortaya çıkarmıştır.

Yazılım mühendisliğinde, yapılan saldırılar karşısında karşı yazılımlar/programlar ile güvenlik duvarı uygulamalarının yetersiz kalmaya başlamasıyla, kod ile işlevselliğin ve güvenliğin arttırılması ihtiyacını doğurmuştur [6]. Yazılım güvenliği, yazılımın geliştirme sürecinden itibaren başlayan ve birbirini takip eden süreçlerde bir bütünlük içinde devam etmesi gereken bir kavramdır.

Bu çalışmada ikinci bölümde yazılım geliştirrme modelleri, üçüncü bölümde güvenlik ve güvenli kodlama teknikleri, dördüncü bölümünde tehdit modelleme, beşinci bölümünde kod injection, altıncı bölümünde kurumlarda güvenli yazılım geliştirme döngüsüne geçiş ve yedinci bölümünde yazılım güvenliğini etkileyen diğer hususlar konuları tartışılmış ve sonuç ve öneriler sunulmuştur.

II. YAZILIM GELİŞTİRME MODELLERİ

Yazılım, donanım ve sistem güvenliği açısından standart belirleme çalışmaları 1983 yılında ABD'de başlamış ve bunu Avrupa'daki çalışmalar takip etmiştir.Ülkeler bilgi sistem ürünleri için standartlar belirlemiş ve ürünlerini bu standartları dikkate alarak üretmeye, ithal ve ihraç etmeye başlamışlardır Bu durum zaman içinde ihraç edilen BT (Bilgi Teknolojileri) ürünlerini için farklı ülkelerde standart farklılıklarından dolayı güvenlik testlerinin başarılı olamasına neden olmuştur. Süreç içerisinde çeşitli çalışmalarda, ortak bir standart oluşturulmaya çalışılmıştır. Günümüzde 1996 yılında Kanada, Fransa, Almanya, İngiltere, Avustralya, Yeni Zelanda ve Amerika Birleşik Devletleri tarafından oluşturulan "Ortak Kriterler (Common Criteria)" temel alınmaktadır [3].

Yazılım geliştirme modellerine genel olarak bakıldığında aslında hepsinin temelinde, "Güvenli Yazılım Geliştirme Yaşam Döngüsü"nün yer aldığı görülmektedir.

Şekil 1. Güvenli Yazılım Geliştirme Yaşam Döngüsü [1].

Şekil 1'den de takip edilebileceği gibi *eğitim ve farkındalığın* geliştirilmesi, döngünün başlangıcı olup bütün süreç boyunca devam eden en önemli unsurdur [1]. Bu aşamada [1], tehditlerin belirlenmesi, güvenli tasarım ve güvenlik testleri konusunda geliştiriciler eğitilmediler. *İhtiyaç analizleri* aşamasında yazılımın geliştirilmesi, güvenlik ve test için kontroller yapılmalı ve sonuçları analiz edilmelidir. *Tasarım ve risk* analizinde güvenlik açısından belirlenen kriterler uygunluk, tehdit modelleme, gizlilik risk analizi gibi kontroller yapılır. *Geliştirme* sırasında belirlenen güvenlik kriterlerine uyulmalı, kaynak kod analizlerinden faydalanılmalıdır. *Doğrulama* aşamasında oluşturulmak istenen ürün/yazılım bütünlük, gizlilik ve erişilebilirlik yönlerinden bir takım testlere tabi tutularak kodun güvenilirliği sağlanmasına ve kod hatalarından oluşabilecek açıkların önlenmesine çalışılır. *Devreye alım* artık nihai ürünün piyasaya sürüldüğü ve kullanıcının bilgilendirildiği aşamadır. Yazılım geliştirme sürecinin güvenli bir şekilde yapılıp yapılmadığı bu aşamadan sonra gözlenebilmektedir. *Açıklık* yönetimi ve geri dönüş aşaması ise kullanıcılardan yapılan geri dönüşlerin değerlendirildiği ve bu değerlendirmelerden sonuç çıkarımlarının yapıldığı aşamadır. Bu sonuçlar farkındalık eğitimi için kaynak görevi gördüğünden döngü en başa dönmüştür [1].

Bu önemli alanın güvenli olarak geliştirilebilmesi ve denetlenebilmesi için geliştirilen modeller aşağıda farklı başlıklar altında açıklanmıştır.

A. Yetenek Olgunluk Modeli (CMM - Capability Maturity Model

1986 yılında ABD'de geliştirilmiştir. Bu model daha çok var olan uygulamaların geliştirilmesini hedefler. Bu modelde amaç, ürünün güvenlik standartlarına göre geliştirilmiş olmasını sağlamak değil daha çok özel bir inceleme alanındaki zayıflıkların giderilmesidir [9].

B. Tümleşik Yetenek Olgunluk Modeli (CMMI Capability Maturity Model Integration)

CMM modelini temel alan bu modellemede amaç, organizasyonun iyileştirilmesi, durumunun gözden geçirilmesi ve yönetimi ile ilgili çözümler sunmaktır [3]. Bu model büyük firmalar tarafından yazılım ihalelerinde firma seçme işlemi ve şirket yöneticilerinin şirketin durumunu analiz etmesini sağlamak için etkin olarak kullanılmaktadır. CMMI modeli, ürün/hizmetin oluşturulması, idame ve satış süreçlerinin iyileştirilmesi aşamalarına yöneliktir. Model temel alınarak şirketler için CMMI sertifikası verilmektedir. Bu sertifika, şirketlerin "neleri nasıl yaptığı" sorusuna cevap bulmayı hedefleyen ve üç seviyesi olan bir sertifikadır. En alt seviye olan C seviyesinde kağıt üzerinden şirket iş akışı veya iyileştirme planları incelenir. Varolan her şirketin C seviyesinde olduğu kabul edilir. B ve A seviyesi inceleme sırasında ise gerçek projeler üzerinden üretim sonuçlarının incelenmesi ile yapılır. Model, şirketlerin mutlaka sahip olması gereken ve sertifikada özellikle aranan genel ve özel amaçları içeren **temel bileşenler**, kuruma özgü ve/veya kurumlar arasında işletim farklılıkları gösterebilen uygulamaları içeren **beklenen bileşenler** ve bunların dışında kalan herşeyi kapsayan **açıklamalar**dan oluşur. CMMI modelinde yapılan kurumsal inceleme esnasında genel olarak gereksinimlerin belirlenmesi/ yönetimi/ geliştirilmesi, süreçlerin tanımlanması, proje planlanması/izlenmesi, ölçme/değerlendirme faaliyetleri, risk yönetimi ve çözüm üretme konularında şirketlerin durumları incelenmektedir [15].

C. Federal Havacılık Yönetim Tümleşik Yetenek Olgunluk Modeli (FAA-iCMM Federal Aviation integrated Maturity Model)

Federal havacılık sisteminde daha çok tercih edilen FAA-ICMM modeli, geniş kapsamlı projelerde kaynak kullanımı ve yönetimini kapsamaktadır. [3].

D. Güvenli CMM/Güvenli Yazılım Metodolojisi (Trusted CMM/Trusted Software Methodology (T-CMM, TSM))

Güvenli yazılım metodolojisi 1990'lı yıllarda geliştirilmiştir. Bu model yazılım geliştirme sürecinde geliştirme hatalarını bulma yöntemlerini, ürünün kullanıma sunulmasının ardından da ataklara karşı koyma tekniklerini içerir [3].

E. Sistem Güvenlik Mühendisliği Yetenek Olgunluk Modeli (SSE-CMM -- Systems Security Engineering Capability Maturity Model)

Organizasyonların güvenlik mühendisliği uygulamalarının iyileştirilmesini hedefleyen bir modeldir. Bu model, güvenlik prensipleri içermesine rağmen uygulama performansının ölçülmesi ve iyileştirilmesi üzerine çalışmaktadır. ISO/IEC 21827 standardı olarak yayınlanmış ve halen The International Systems Security Engineering Association (ISSEA) tarafından uygulanmaya devam etmektedir [3].

F. Microsoft Security Development Lifecycle (SDL)

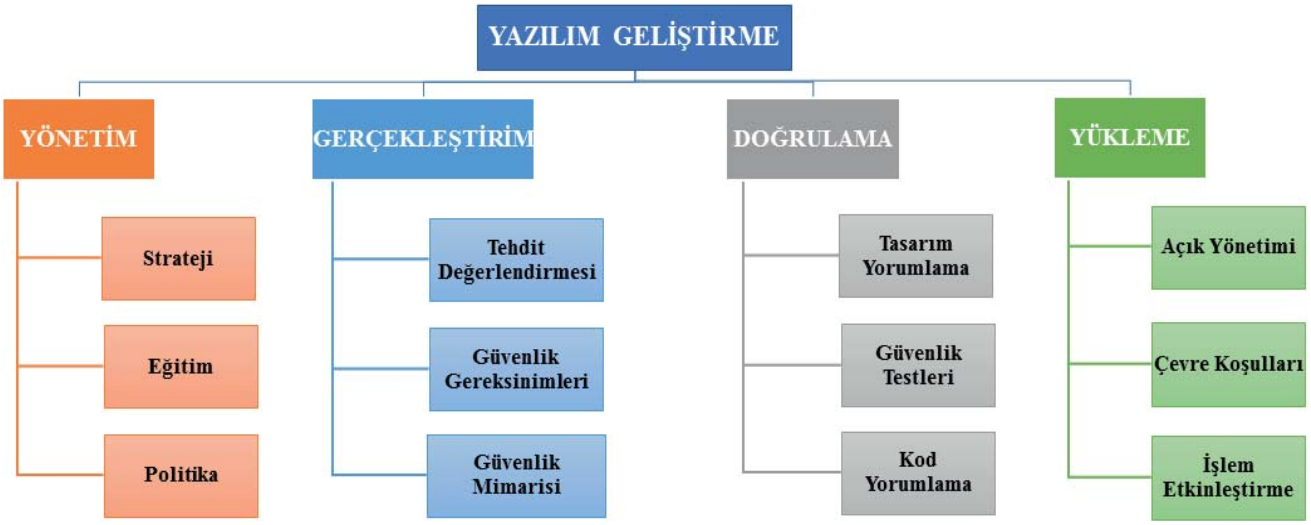
Microsoft yazılım geliştiricileri, karşılaşılan problemlerden dolayı 2004 yılında güvenlik ve gizlilik için bütünleştirici bir model geliştirmeyi hedeflemişler ve bir yazılım güvenliği süreci olan SDL'yi tasarlamışlardır (Şekil 2). Bu süreç, temel yazılım güvenliği döngüsünü rol model olarak alan ve doğrudan proje yöneticisine bağımlı bir modeldir [7].

Şekil 2. Microsoft SDL Döngüsü [7].

Şekil 2'den de görülebileceği gibi [7], *Eğitim* aşamasında personele yazılım geliştirme modeli ve yazılım ihtiyaçları doğrultusunda temel eğitim verilmesi hedeflenir. *Analiz* aşamasında programın hedef kitlesi, içereceği özellikler, test için kullanılacak saldırı modelleri ve program maliyetinin belirlenmesi gibi ön planlama işlemleri gerçekleştirilir. *Tasarım* aşamasında analiz aşamasınından elde edilen sonuçlara göre program planlanır. Yapılacak saldırı teknikleri, karşı koyma yöntemleri kararlaştırılır ve risk analizi yapılır. *Gerçekleştirim* aşamasında kod geliştirilirken ve sonrasında kullanıcının isteyebileceği güvenlik tedbirleri de dikkate alınarak esnek bir yapı oluşturulmaya çalışılır. White-box Security, Code Review gibi yöntemlerle kodun güvenliği *Doğrulama* aşamasında test edilir. *Yayınlama* aşamasında uygulama tamamlanmış ve kullanıcıya sunulmuştur. Fakat bu aşamaya gelmeden yani son kullanıcıya ürün sunulmadan önce güvenlik kontrollerinin sonuçlarından emin olmak gerekir. Son aşama olan *Bilgi Toplama* aşamasında ise kullanıcıdan gelen geri bildirimler dikkate alınarak açıklar tespit edilip ilgili yamalar üretilerek kullanıcıya sunulur [4].

G. SAMM (Yazılım Garanti Olgunluk Modeli)

Open Web Application Security Platform (OWASP) tarafından desteklenen SAMM modeli bilgi güvenliği aktivitelerini yazılım geliştirme sürecine dâhil etmeyi hedefleyen bir modeldir [12]. Bu model organizasyonun büyüklüğünden bağımsız olup büyük bir proje için kullanılabileceği gibi organizasyonun tamamı ya da küçük bir kısmı içinde son derece uygundur [12].



Şekil 3. SAMM Genel Yapısı [12].

SAMM'a gore yazılım geliştirme süreci modellemesi katmanlı bir mimariye sahiptir. İş fonksiyonları ve bunların her birinin güvenlik araştırmaları farklı katmanlarda yer alır.Şekil 3'de verilen iş fonksiyonları katmanında yönetim, gerçekleştirme, doğrulama ve yükleme ana başlıkları bulunur. *Yönetim* fonksiyonunda yazılım geliştirme sırasındaki güvenlik süreçlerinin yönetiminin gerçekleştirilir. *Gerçekleştirme* fonksiyonu, güvenli yazılım gereksinimleri ve tasarım aşamaları üzerinde yoğunlaşan, tehdit değerlendirme, güvenlik gereksinimleri ve güvenli mimari aktivitelerini içerir. *Doğrulama* fonksiyonu tasarım, kodlama ve yazılım testlerinin gözden geçirilmesini ve güvenlik testlerini kapsamaktaolup tasarım, kod denetimi ve güvenlik testlerinden oluşur. Yükleme (*Kurulum)* fonksiyonu ürünün son kullanıcıya iletilmesi ve bakım, idame işlemlerini içerir. Bu fonksiyonun aktiviteleri ise güvenlik açığı yönetimi, platform farklılaştırması ve işletim kurulumudur [12].

III. GÜVENLİK VE GÜVENLİ KODLAMA İLKELERİ

Bu husular aşağıda verilen alt başlıklarda açıklanmış ve tartışılmıştır.

A.Güvenlik

Sahip olduğumuz maddi ve manevi varlıklarımıza doğrudan ve/veya dolaylı yönelen her tehdidin derecesi ve vereceği bir zarar vardır. Bu zararın önceden tahmin edilerek uygun tedbirlerin alınması işlemi "güvenlik" olarak tanımlanabilir [8]. Bilgi varlıklarımızın güvenliğinin tam olarak karşılanması için belgeleme, yetkilendirme, denetim, gizlilik, bütünlük ve uygunluk esalarının yerine getirilmiş olması gerekmektedir.

Belgeleme, sistemdeki ya da sistemden yararlanan son kullanıcı, bilgisayar ya da diğer servislerin "kim olduğu" sorusuna verilen cevap, her bir işlemin belgelenmesidir. *Yetkilendirme,* belgelenen kullanıcıların erişme izinlerinin olduğu her türlü kaynağın kontrol edildiği aşamadır. *Denetim*, kullanıcının yaptığı her çalışmanın kayıt altına alınmasıdır. Böylece inkar etmenin önüne geçilmiş olur.

Gizlilik, sistemin yabancı kullanıcılarca izlenmesi ve bilgilerin güvenliğinin sağlanması için önlem alınması sürecidir. Tam bir gizlilik esasına göre, ağdan yapılacak dinlemelere, monitör takiplerine v.b uygulamalara karşı olarak şifreleme ya da erişim kontrol listeleri oluşturmak gibi tedbirler alınabilir. *Bütünlük,* verinin bilinmeden ya da kötü niyetli yapılan değiştirmelerden korunmasını sağlar. *Gizlilik*te olduğu bir bütünlükte de ağlar üzerinden iletilen verinin korunması önemli bir unsurdur. Ağ üzerinden iletilen veriler için bütünlük, karma teknikler ve ileti kimlik doğrulama ile sağlanmaya çalışılır. *Uygunluk,* mevcut durumun meşru kullanıcılar için sabit kalması, meşru olmayan kullanıcılar ve saldırı atakları için erişiminin olup olmadığının kontrol edildiği durumdur [8].

B.Güvenli Kodlama İlkeleri

Bilgi güvenliği, kullanıcıya sadece izin verilen kısma erişim hakkı veren, *güvenlik* yetkisi olmayan kullanıcıya veri değiştirme izinin verilmediği, *bütünlük* ve ihtiyaç halinde yetkili kullanıcıya müdahale hakkı veren *kullanılabilirlik* prensiplerine dayanan bir işlemdir. Bu prensipler temel alınarak geliştirilen sistemde öncelikle dikkat edilmesi gereken unsurgüvenli bir mimarinin oluşturulmasıdır. Güvenli olmayan bir mimari kod ne kadar güçlü olursa olsun saldırılara açık bir haldedir. Yazılım geliştiricinin mimariyi oluştururken kendine sorması gereken soruların başında "kötü niyetli biri olsam…" gelmektedir. Bu bakış açısı Andrew van der Stock tarafından ileri sürülen "kötü düşünce" sistemidir. Bu sistemde kişi kendi açıklarını dışardan bakmaya çalışarak bulmayı ve düzeltmeyi hedefler [2]. 2'nolu kaynakta verilen bu hususlar aşağıda özetlenmiştir.

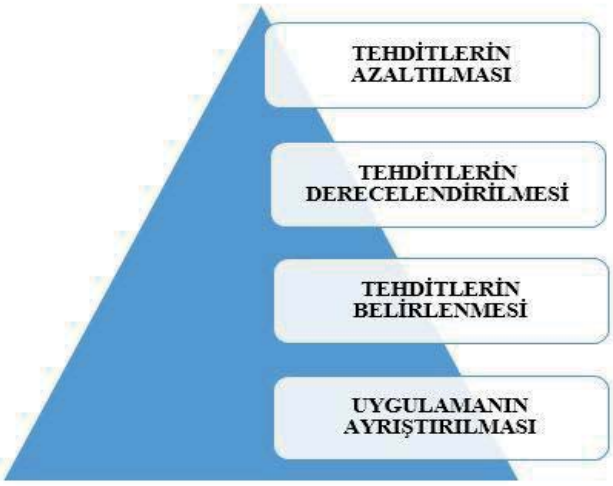
Saldırı yüzey alanını azaltmak kodlama işlemi sırasında dikkat edilecek en önemli ilkelerden biridir [2]. Böylelikle genel risk oranı azaltılmış olur. *Güvenli öntanımlar* yaparak kullanıcılar tarafından yapılacak bir çok işlem için önceden kurallar yaratma imkanı sağlanır. *En az öncelik* ilkesi ile kullanıcının hesap işlem süreçleri kontrol edilebilir. Sunuculara öncelik yetkisi verilmemelidir. *Kademeli savunma* ile koyulacak kontroller aşamalı olarak koyularak bir dizi kontrol elde edilmiş olur. *Güvenli aksama* ile kodda oluşacak aksamalar karşısında sistemin açık vermemesi sağlanmalıdır. Oluşan bu aksama kullanıcıya yetkisi olmayan hakların kapısını açmayacak şekilde kod düzenlenmelidir. *Dış sistemlerin güvenli olmadığı* akılda tutulması gereken bir ilkedir. Sistem sağlayıcılar her zaman kodunuz için gerekli güvenlik tedbirlerin tam olarak almamış olabilir. Farklı yetki sahiplerine farklı görevler vererek *görevlerin ayrımı* ile sistemde yapılan iş akışının kontrolü ve güvenliği sağlanış olur. *Belirsizlik yolu ile güvenliğe güvenmemek* gerektiği akıldan çıkarılmamalıdır. Güvenlik, tek bir sistemin kod içine gizlenmesiyle elde edilemez. Bunun yerine birbirini takip eden çok sayıda güvenlik kontrolü bulunmalıdır. *Basitlik*, kodun mümkün olan en basit şekilde yazılmasını öngörür zira kod karmaşıklaştıkça kontrolü de zorlaştıracaktır. *Güvenlik sorununun uygun şekilde ele alınması* ilkesinde ise yaşanabilecek sorunlar için test grupları oluşturulmalı ve yapılacak değişikliğin kodun ne kadarını etkileyeceğinin belirlenmesi için test süresi verilmelidir [2].

IV. TEHDİT MODELLEME

Yazılım tasarlarken karşılaşılabilecek tehditlere karşı model geliştirmek tehditlerin ve çeşitliğinin giderek arttığı bu dönemde daha da önemli olmaya başlamıştır. Tehdit modellemenin amacı, genel olarak sistemin güvenliğini tehlikeye atabilecek potansiyel durumları belirlemektir [8].

Tehdit modelleme, güvenlik tabanlı bir analiz sürecidir. Bu süreç uygulamanın anlaşılabilmesi, yeni gelen geliştiricilerin uygulama adaptasyonun hızlandırılması, tehditlerin belirlenmesi, kod hatalarının azaltılması gibi bir çok alanda faydalar sağlar [10,11]. Şekil 4'de tehdit modelleme aşamaları verilmiştir. Bu aşamalar aşağıda kısaca açıklanmıştır:

Uygulamanın Ayrıştırılması aşamasında uygulama alt alanlara ayrılır. Böylece hem uygulamanın anlaşılırlığı arttırılır hem de her bir alan için güvenlik profilleri oluşturulur. Her bir alt katmanın işlevleri bulunmaktadır. Bu işlevler aşağıda özetlenmiştir [10,11]:



Şekil 4. Tehdit Modelleme Aşamaları [10,11].

1) Güvenlik Çember: Sisteme giren ve sistemden çıkan bütün veriler kontrol altında tutulur.
2) Akış Diagramları: Sistemin genelinin modellemesi yapılır. Diagram hazırlanırken sisteme gelecek sorular ve bunların cevaplarının nasıl olacağı yüzeysel olarak ifade edilir.
3) Giriş Noktalarının Belirlenmesi: Her bir giriş noktası için yetkilendirme ve kimlik kontrolü yapılmalıdır.
4) Ayrıcalıklı kod Parçaları: Sistemde ayrıcalıklı ve korumalı bölgelere nüfuz edebildiği için bu alanlar özellikle dikkat edilerek geliştirilmelidir.
5) Güvenlik Profili Oluşturulması: Girdilere ne ölçüde izin verileceği, kimlik kontrolünün nasıl yapılacağı, kimlere yetki verileceği gibi sorular cevaplanmalı bu şekilde profil oluşturulmalıdır.

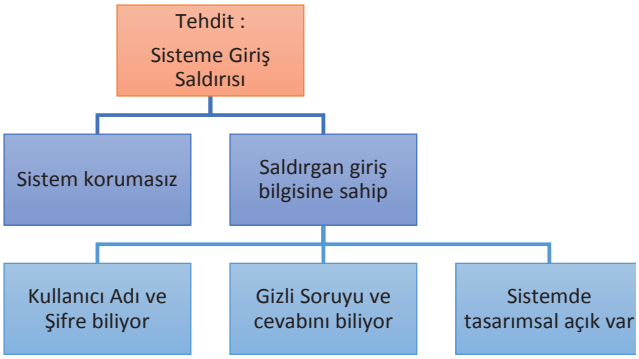
Tehditlerin Belirlenmesi aşamasında her bir kısım tehdit modeli olarak adlandırılmakta ve bunlardan oluşacak tehditler STRIDE modeline göre sınıflandırılmaktadır. Bu modeldeki sınıflar şunlardır [10,11]:

1) Kimlik Yanıltması (Spoofing Identity): Kayıtlı bir kullanıcı ya da geçerli bir sunucu gibi davranma yoluyla sistem için tehdit oluşturulmasıdır.
2) Verinin Değiştirilmesi (Tampering with Data): Verinin art niyet gözetilerek değiştirilmesidir.
3) Inkar (Repudiation): Yapılan işlemin ya da saldırının yapılmadığının iddaa edilmesidir.

4) Bilginin Açığa Çıkarılması (Information Disclosure): İzinsiz ve yetkisiz olarak sistemdeki veri ya da bilgilere ulaşılması ve bilmemesi gereken kişilere ifşa edilmesidir.

5) Hizmet Dışı Bırakma (Denial of Service): Sistemin hizmet veremez hale getirilmesidir.

6) **Yetki Kazanma (Elevation of Privilege):** Yetkisiz kullanıcının illegal yollarla sistemde yetki kazanmasıdır.



Şekil 5. Tehdit Ağacı Gösterimi [10,11].

Tehditlerin Derecelendirmesi aşamasında tehdit ağaçları oluşturulup durum değerlendirildikten sonra hangi tehdide öncelik verileceğini belirlenir. Her bir tehdit için tehdidin nasıl oluşacağı tehdit ağaçları ile gösterilir. Ağaçdaki kök tehdit kutusu, sonraki düğüm ve yapraklar ise tehdidin nasıl gerçekleşeceğini temsil eder Şekil 5'de tehdit ağacı sembolize edilmiştir [10,11]. Tehdit derecelendirilmesi işlemi için en çok tercih edilen yöntem DREAD yöntemidir. Yöntemin kriterleri aşağıda açıklanmıştır [10,11]:

1) Zarar Potansiyeli (Damage Potential): Tehdit gerçekleşirse oluşacak zararın büyüklüğü belirli aralıklar arasında derecelendirilir.

2) Uygulanabilirlik (Reproducibility): Saldırgan için bir tehdidin başarıya ulaşma ihtimali değerlendirilir.

3) Sömürülebilirlik (Exploitability): Tehdidi gerçekleştirmek için gereken bilgi ve süre değerlendirilir.

4) Etkilenen Kullanıcılar (Affected User): Tehdit gerçekleşirse nekadar kullanıcının zarar görebileceği değerlendirilir.

5) Keşfedilebilirlik (Discoverability): Açıkların hangi zorluk derecesinde bulunabileceği değerlendirilir.

Tehditlerin Azaltılması aşamasında bu işlem için dört farklı yaklaşım tercih edilebilir. Buna göre, tehdit konusunda tedbir alınmayabilir ki bu sonuçları ağır olabilecek bir seçenektir. Kullanıcılar tehdit konusunda bilgilendirilip kullanıcı bazlı tedbirler alınabilir (şifre derecesi zorlaştırma, şifreyi paylaşmama gibi). Tehdit konusu olan kısım sistemden çıkartılabilir ya da teknolojik önlemler alınarak tehdit için çözüm üretilebilir [10,11].

V. KOD ENJEKSİYONU

Kod enjeksiyonu, bilgisayar açıklarının (bugs) kötüye kullanımından kaynaklanan bir veri kaybı olarak nitelendirilebilir. Genel olarak enjeksiyon, kodlayıcının hatası sonucu oluşur. Kötü amaçlı enjeksiyon kullanımlarında, Sql enjeksiyon da olduğu gibi veri tabanındaki veriler değiştirilebilir, web uygulamasının kontrolü ele geçirilebilir, sunucuya erişilebilir, UNİX gibi işletim sistemlerinin çekirdekleri hedef alınabilir. İyi niyetli enjeksiyon kullanımlarında ise sayfanın düzenine geliştirme anlamında katkıda bulunulabilir, ihtiyaca uygun olarak verilerin filtrelenmesi için kullanılabilir ya da yazılımsal olarak bazı değişikliklerin daha az maliyetle yapılmasına olanak sağlayabilir [13,14].

Enjeksiyonları önlemek için girdilerin doğrulanması, girdi ve çıktıların kodlanması, program debuggerları için kilit olan kelimelerden kaçınılması, "store procedure" gibi ön derleme, değerlendirme mantığına dayalı kodlamaların tercih edilmesi gibi önlemler alınabilir [13,14].Bunlara ek olarak uygulamanın katmanlı mimariye sahip olması da önemli bir önlem olarak sayılabilir.

A. HTML Enjeksiyonu (XSS)

Bu açık, programcıların kodlama sırasında yaptığı hatalar sonucu oluşur. Web yazılımlarımda veri tabanına giren verilerin ya da çekilen verilerin bir kontrol mekanizmasından geçirilmemesi sebebidir. Bu açıktan faydalanılarak session ve cookie çalması yapılır. Korunmak için veriler sayfaya yüklenme aşamasında değil, veri tabanına gönderilirken ya da çekilirken bir filtrelemeden geçirilmelidirler [5].

B. SQL Enjeksiyonu (SQL Insertion Attacks)

SQL enjeksiyon, meta-karakter adı verilen ve derleyici için özel anlamları olan karakterlerin varlığına dayanan bir açıktır ve karakterlerin yanlış filtrelenmesinden kaynaklanır [14]. SQL enjeksiyon tipleri farklı şekillerde olabilir. Kullanıcıdan alınan kullanıcı adı ve şifre gibi giriş verileri bir filtrelemeye tabi tutulmadan SQL sorgusuna gönderildiğinde oluşan açıktan faydalanarak saldırı yapılabilir. Örneğin,bir web uygulamasının kullanıcı adı ve şifre ikilisi veri tabanına

"SELECT * FROM TABLE WHERE username = '" +Kullanıcı + "' AND password= '" +Şifre + "'"

şeklinde gönderildiğinde (" ") işaretleri içindeki veri bir filtrelemeye tabi tutulmazsa kullanıcının buraya yazacağı (OR "1=) şeklinde bir ifade sorguyu

"SELECT * FROM TABLE WHERE username = "OR"1=1" AND Password = " OR "1=1" "

haline getirir. Bu durum sorguda var olan bütün kayıtlar dönecektir. Böylece saldırgan bütün kullanıcı adı ve şifre bilgilerine sahip olacaktır [13,14].

Bir açık türü de tip kontrolü yapılmadığı durumlarda oluşur. Örneğin nümerik alana dair bir sorgu cümleciği gönderildiğini varsayalım.

"SELECT * FROM TABLE WHERE ID = '" +U_ID";'"

sorgusunu ele alırsak; burada programcı aslında sayısal bir alana ulaşmak istiyor ama gönderilen verinin(U_ID) string olarak gelip gelmediğini kontrol etmiyor. Bu durumda saldırgan tarafından bu sorguya "1;DROP TABLE `DBusers`" şeklinde bir ifade gönderilirse komut

"SELECT * FROM TABLE WHERE ID = '" 1;DROP TABLE `DBusers`';'"

olarak algılanacağından tablodaki kullanıcı bilgileri silinecektir [13,14].

Bir diğer durum ise Kör SQL Enjeksiyon denilen durumdur. Bu tür açık da, saldırgan tarafından sonuçlar veya veriler direk veritabanından görülmez. Bunun yerine gelen veriler farklı mantıksal ifadelerle elde edilir. Elde edilen bu sonuçlar sayfadaki var olan adımlarla elde edilmiş olur [13,14].

SQL enjeksiyonunu önlemek için uygulanabilecek yöntemlerde biri parametreli "store procedure" tekniğini kullanmaktır. Bu teknikte girdiler öncelikle derlendiği için açıklardan faydalanmak neredeyse imkânsızdır. Ayrıca uygulamalarda veri tabanından okuma için salt okunur yetkisi verilmesi de bir çözümdür. Oluşan hatalarda kullanıcıya dönen hata mesajlarında mümkün olduğunca az bilgi verilmesi de saldırganlara veri tabanı hakkında bilgi vermemek açısından önemlidir [13,14].

C. *Dinamik Değerlendirme Açıkları*

Bu açıklar daha çok PHP kodlarda karşılaşılan açıklardır. Saldırgan string değere karşılık sonuç gönderen fonksiyonları kullanır. Bu fonksiyonlara girdi olarak uygulamayı etkileyecek string komutlar göndererek istediği sonuçlara ulaşmayı hedefler [13,14].

D. *SHELL Enjeksiyon*

Bu enjeksiyon adını Unix'den almış olsa da komut satırı kullanan her uygulama için tehdit arz etmektedir. Alınan verinin komut satırı için özel komutlardan biri olup olmadığının kontrol edilmediği uygulamalarda saldırgan komut satırına ulaşarak amacını gerçekleştirmektedir [13,14].

VI. KURUMLARDA GÜVENLİ YAZILIM GELİŞTİRME DÖNGÜSÜNE GEÇİŞ

Kurumlarda karşılaşılabilecek güvenlik zafiyetlerini azaltmak için, [1] nolu kaynakta belirtilen ve aşağıdaki adımlarda özetlenen adımların hayata geçirilmeleri gereklidir.

1) *Kurumun Yazılım Geliştirme Yöntem ve Prosedürlerinin Araştırılması*

Yazılım geliştiren her kurum öncelikli olarak yazılım geliştirme konusunda mutlaka bir prosedürü veya yöntemi belirlemek için bir araştırma yapılmalı ve bir prosedür geliştirilmelidir. Bu prosedür projenin yönetim yaklaşımı, kuruma özel yöntemler ve yazılımlarla ilgili bağlayıcı hükümler içermelidir [1].

2) *Mevcut Modellerin İncelenmesi ve İhtiyaca Yönelik Eğitimler*

Güvenli yazılım geliştirmede kullanılan modeller incelenerek kurum, kurum içi ilişkiler ve etkileşimler, kurum kültürü ve modellerin her birinin kuruma uygun/uygun olmayan yönleri belirlenerek bir n-model seçilmeli ve personel için eğitimler yapılmalıdır [1].

3) *Kurum Şartlarına Uygun Güvenli Modelin Belirlenmesi*

Belirlenen model.kurumun ihtiyaçlarına göre revize edilerek güvenli ve başarılı yazılım geliştirme sürecine dahil edilmelidir. Gerekirse bir dönem belirlenen yapı test edilmeli, gerekli görüldüğünde modele ek tedbirler eklenebilmelidir [1].

4) *Politika ve Prosedür Dokümanlarının Hazırlanması*

Belirlenen modele ait politika ve prosedürlerin dökumante edilmesi, hem kuralların daha kolay anlaşılıp adapte edilmesini kolaylaştırır ve yanlış anlamaları önler hem de ortak bir çalışma yöntemi oluşturularak bu işin el kitabının hazırlanmasını sağlar. Bu yöntem, ekibe yeni katılan kişilerin de sisteme kolayca dahil olmasını sağlar ve sürecini hızlandırır [1].

5) *Eğitimlerin Hazırlanması ve İlgili Personele Eğitim Verilmesi*

Kullanılan Güvenli yazılım geliştirme döngüsü için kapsamlı bir eğitim programı hazırlanarak personelin eğitim alması sağlanmalıdır. Bu eğitimlerin gelişen teknolojiye paralel olarak düzenli olarak yapılması, hem teknolojiyi takip hem de karşıla-şılabilecek risklerin minimize edilmesini sağlayacaktır [1].

VII. YAZILIM GÜVENLİĞİNİ ETKİLEYEN DİĞER UNSURLAR

Yazılımların geliştirilmeleri kadar bu yazılımların işletildiği, barındırıldığı veya çalıştırıldığı ortamların da güvenliğinin sağlanması önemli bir husustur. "Ağdaki bir güvenlik açığı kötü niyetli bir kullanıcının host ya da uygulamayı istismar etmesine, hostdaki bir güvenlik açığı kötü niyetli bir kullanıcının ağ ya da uygulamayı istismar etmesine, uygulamadaki bir güvenlik açığı kötü niyetli bir kullanıcının host ya da ağı istismar etmesine izin verebilir" [7]. Belirtilen bu hususlar aşağıda başlıklarda kısaca açıklanmıştır.

A. *Ağ Güvenliği*

Güvenli bir yazılım ancak güvenli bir ağ yapısı üzerine çalıştırılabilir. Ağın korunması, TCP-IP ataklarına karşı korunmanın yanında güçlü şifreleme, güvenli erişim ve admin yapısıyla da desteklenmelidir. Güvenli bir ağ hem ağ trafiğini aksatmamalı hem de yönelen tehditleri geldiği katmanda kontrol altına almalıdır ki yazılım güvenliği yüksek oranda sağlanabilsin [8].

B. *Host Güvenliği*

Host güvenliğinde dikkat edilmesi gereken en önemli husus, web servisi, uygulama servisi ve veritabanı hizmetlerinin her birinin kendine has güvenlik protokollerinin olduğu ve her yeni gelen sistem için bu ayarların yapılması gerektiğidir [8].

C. *Uygulama Güvenliği*

Güvenlik konuları ile web uygulamaları karşılaştırılmaları analiz edildiğinde, güvenlik problemleri güvenlik açıkları olarak gruplanabilir. Bir uygulamanın güvenlik direncini ölçmek için güvenlik açıklarından elde edilen erişim doğrulama, belgelendirme, konfigürasyon yönetimi, oturum yönetimi, hassas veri şifreleme, parametre manipülasyonu, hata yönetimi, denetim ve log gibi güvenlik profilleri kullanılır [8]. *Erişim doğrulama,* uygulamanın giriş filtrelerini ifade eden bir profildir. *Bilgilendirme,* kullanıcının giriş verilerini ifade eder. *Yetkilendirme,* kullanıcının izinlerini ifade eder. *Konfigürasyon yönetimi,* uygulamanın nasıl çalışacağı hangi veritabanını kullanacağı bir operasyonel kavramları ifade eder. *Hassas veri,* uygulama verilerinin hafızada ya da korumalı diskte nerede tutulduğunu ifade eder. *Oturum yönetimi,* oturumda bir kullanıcı ve Web uygulaması arasında ilgili etkileşimlerin

nasıl olduğunu ifade eder. *Şifreleme,* gizli verilerin nasıl tutulduğu, şifrelemenin nasıl olduğunu ifade eder. *Parametre manipülasyonu,* parametrelerin nasıl işleyeceğini ifade eder. *Hata yönetimi,* uygulamada bir işlem başarısız olduğunda neler yapıldığını, nasıl bir hata döndürüldüğünü ifade eder. *Denetim ve log,* yapılan işlemlerin loglarını tutar. Yazılım geliştirilirken yukarıda belirtilen adımlara dikkat edilmiş ise uygulama katmanında güvenliğin büyük ölçüde sağlandığı anlamına gelmektedir [8].

VIII. SONUÇ

Web yazılım güvenliği, web uygulamalarının yaygınlaşmasıyla önem kazanmış bir alandır. Teknolojideki hızlı gelişmeye bağlı olarak uygulama alanı gelişmiş ve artık birçok işlem web üzerinden yapılabilir hale gelmiştir. Bankaların, kurumların ve hatta devletin hizmetlerini web uygulamaları şeklinde sunmaya başlamış olmaları bu alandaki saldırıları arttırmıştır. Yazılım geliştirilme aşamasında güvenli yazılım geliştirme modelleri referans alınarak güvenli kodlar yazılması, uygulamaların temellerinin sağlam oluşturulması, oluşabilecek tehditlerin analizi yapılarak gerekli tedbirlerin alınması, bunlara ek olarak ağ ve host üzerinde de ilave tedbirlerin alınması verilerin güvenliği açısından önem arz etmektedir. Web yazılım güvenliği, güvenlik standartlarına uygun ve risklerin göz önüne alınarak uygun yaklaşımlarla geliştirilen kodların ve sonrasında uygulandığı sistemdeki güvenlik tedbirlerinin bütünlüğü ile sağlanabilmektedir.

Yapılan inceleme çalışmasında, güvenli yazılım geliştirme modelleri, güvenli bir yazılımda olması gereken unsurlar, tehdit analizi ve web yazılım güvenliğinin sağlanabilmesi için ilave tedbirlerin neler olduğu araştırılmıştır. Ayrıca kurumsal olarak güvenli kod geliştirilmesi standardizasyonun nasıl yapılabileceği nelere dikkat edilmesi gerektiği üzerinde durulmuştur. Elde edilen veriler değerlendirilerek daha güvenli kod geliştirme uygulamaları gerçekleştirileceği düşünülmektedir.

Kurumsal bilgi güvenliği, web uygulama güvenliği, açık ve tehdit tespit yöntemleri üzerinde inceleme yapılabilecek kapsamlı konular olduğu, son dönemde meydana gelen açıkların/açıklıkların/saldırıların bu zafiyetlerden faydalanılarak yapıldığı tespit edildiğinden ülkemizde; bu alanda yapılan çalışmalara daha çok dikkat edilmesi, açık veya açıklıkların ana gerekçelerinin yazılımlarda meydana gelen güvenlik açıkları olduğu, gerekli tedbirlerin alınarak web yazılımlarının hizmete alınması gerektiği, mutlaka standartlara uygun yazılım geliştirilmesi gerektiği, ve en önemlisi de testlerin de prosedürlere uygun olarak yapılması gerektiği bir kez daha ortaya koyulmuştur.

Ülkemizde bu alanda eğitim-öğretim veren bölümlerin kurulmuş olması sevindirici bir unsur olsa da bu alanda çalışma yapan üniversitelerin müfredatlarında güvenlikle ilgili olarak derslerin ya hiç olmadığı veya çok üzerinde durulmadığı belirlenmiştir. Bu konuda eğitim veren üniversitelerin müfredatlarını yenilemeleri, ve bu bildiri kapsamında sunulan hususları dersler içerisinde mutlaka yer vermeleri, test ve standartları bilip uygulamaları gerekmektedir. Bunlara ilaveten, diğer önemli husus ise yazılım güvenliği gibi bölümlerin ülkemizde açılması bu alandaki uzman eksikliğinin kısa sürede tamamlanmasını da sağlayacaktır.

KAYNAKLAR

[1] ALPARSLAN Erdem, "Güvenli Yazılım Geliştirme Modelleri", http://www.bilgiguvenligi.gov.tr/teknik-yazilar-kategorisi/guvenli-yazilim-gelistirme-modelleri.html, 2009

[2] AKANYILDIR Çiğdem , "Güvenli Kodlama İlkeleri", http://www.webguvenligi.org/wp-content/uploads/2007/08/Secure%20Coding%20 PrinciplesTRK.pdf ,2010

[3] BEYDAĞLI Erkut, KARA Mehmet, BAHŞİ Hayretdin, ALPARSLAN Erdem," Güvenli Yazılım Geliştirme Modelleri ve Ortak Kriterler Standardı", http://www.bilgiguvenligi.gov.tr/bt-guv.-standartlari/guvenli-yazilim-gelistirme-modelleri-ve-ortak-kriterler-standardi.html, 2010

[4] BURAK Efe, "Microsoft'un Güvenli Geliştirme Süreci (SDL)", http://www.bilgiguvenligi.gov.tr/teknik-yazilar-kategorisi/microsoftun-guvenlik-gelistirme-sureci-sdl.html, 2009

[5] ÇİTİL Ferhat, "HTML Injection Tehlikesi", http://www.cyber-security.org.tr/Madde/220/HTML-Injection-Tehlikesi- ,2009

[6] GÜMÜŞ Emine, "Yazılım Güvenliğine Genel Bakış", http://www.cyber-security.org.tr/Madde/182/Yazılım-Güvenliğine-Genel-Bakış,2009

[7] MİCROSOFT, "Microsoft Security Development Lifecycle(SDL)", http://msdn.microsoft.com, 2011

[8] MEİER J.D., MACKMAN Alex, DUNNER Michael, VASİREDDY Srinath, ESCAMİLLA Ray and MURUKAN Anandha, "Improving Web Application Security: Threats and Countermeasures", http://msdn.microsoft.com/en-us/library/ff648636.aspx, 2003

[9] REDWİNE, S. T. ve DAVİS, N., "Processes to Produce Secure Software.", http://www.cyberpartnership.org/init-soft.html ,2004.

[10] POŞUL Abdulkadir," Yazılım Geliştirmede Tehdit Modelleme I", http://www.bilgiguvenligi.gov.tr/yazilim-guvenligi/yazilim-gelistirmede-tehdit-modelleme-i.html, 2011

[11] POŞUL Abdulkadir," Yazılım Geliştirmede Tehdit Modelleme 2", http://www.bilgiguvenligi.gov.tr/yazilim-guvenligi/yazilim-gelistirmede-tehdit-modelleme-ii.html, 2011

[12] Dr. TATLI Emin İslam, "SAMM ile Güvenli Yazılım Geliştirme", http://dergi.webguvenligi.org/websec/57-samm-ile-guvenli-yazilim-gelistirme.wgt,2010

[13] Wikipedia, "Code injection", http://en.wikipedia.org/wiki/Code_injection,2010

[14] Wikipedia, "SQL Injection", http://en.wikipedia.org/wiki/SQL_injection, 2011

[15] KALAYCI Orhan,"CMMI-Yöneticiler İçin Doğru Sorular", Shamrock Süreç İyileştirme ve Yenilikçilik,2007

Seda YILMAZ Gazi Üniversitesi Bilgisayar Bilimleri Bölümünde yüksek lisans eğitimine devam etmektedir. Araştırma konuları arasında kablosuz sensor ağlarda güvenlik, web yazılım güvenliği, siber güvenlik, bilgi ve bilgisayar güvenliği yer almaktadır.

Şeref SAĞIROĞLU Gazi Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü öğretim üyesidir. Zeki sistem kimliklendirme, tanıma ve modelleme, ve kontrol; yapay sinir ağları ve yapay zeka uygulamaları; sezgisel algoritmalar; endüstriyel robotlar; zeki anten analizi ve tasarımı; internet, web ve bilişim sistemleri ve uygulamaları; yazılım mühendisliği; bilgi ve bilgisayar güvenliği; biometri, elektronik ve mobil elektronik imza ve açık anahtar altyapısı; kötücül ve casus yazılımlar gibi konularda çalışmaktadır. 50'nin üzerinde SCI tarafından taranan uluslar arası dergilerde yayınlanmış makalesi, 50' nin üzerinde ulusal dergilerde yayınlanmış makalesi, 100'ün üzerinde uluslar arası konferans ve sempozyum bildirisi ile ulusal sempozyum, konferans ve çalıştaylarda sunulmuş100'e yakın bildirisi bulunmaktadır. 3 adet alınmış patenti vardır. 5 adet yayınlanmış kitabı, 2 adet kitapta bölüm yazarlığı bulunmaktadır. 4 kitabın da editörlüğünü yapmıştır.
Ulusal ve uluslar arası pek çok proje yürütmüş ve konferanslar düzenlemiş olup akademik çalışmalarına devam etmektedir.