Uluslararası Teknolojik Bilimler Dergisi
International Journal of Technological Sciences
UTBD IJTS

Orijinal Makale/Research Article

Yeni bir güvenli yazılım geliştirme uygulama modeli: GYG-MOD

Güncel SARIMAN[1], **Habibe ÇELİKTEN**[2]

[1]Muğla Sıtkı Koçman Üniversitesi, Teknoloji Fakültesi, Bilişim Sistemleri Mühendisliği Bölümü, 48000, Muğla, Türkiye
[2]Muğla Sıtkı Koçman Üniversitesi, Fen Bilimleri Enstitüsü, Bilişim Sistemleri Mühendisliği Anabilim Dalı, 48000, Muğla, Türkiye

Anahtar Kelimeler
Güvenli yazılım geliştirme
Yazılım Güvenliği
Yazılım Yaşam döngüsü

Makale geçmişi:
Geliş Tarihi: 27.02.2021
Kabul Tarihi: 26.04.2021

Öz: Teknolojik gelişmelerin ivmelenen hızıyla birlikte, internet ve mobilite insanların günlük yaşamına daha fazla nüfuz etmeye başlamıştır. Yaşamımızın internet ile bütünleşik hale gelmesi, kişisel bilgilerimizi ve üretilen verilerimizi daha güvenli hale getirmemizi zorunlu hale getirmektedir. Yaygın internet kullanımıyla beraber günümüzde artan güvenlik açıklıkları, sistemlerin altyapı olarak daha korunaklı ve güçlü olmasını gerektirmektedir. Bu altyapılar bilişim sistemlerinde ağ, sistem ve yazılım temelli olmaktadır. Ağ ve sistem mimarisi temelli güvenlik önlemleri daha çok bilişim ürünlerinin, son kullanıcıların kullanımına sunulduktan sonra alınabilecek güvenlik önlemlerine yönlendirmektedir. Yazılım ürünlerinde ise piyasaya sürülmeden henüz yapım aşamasındayken güvenlik önlemlerinin alınmasıyla bilişim maliyetleri düşmektedir. Bu çalışmada, yazılımların güvenli hale getirilmesi için, projelerin kullanıma başlamadan önce henüz geliştirme aşamasında iken güvenlik prensibinin yazılım geliştirme sürecine dâhil edilmesi üzerine bir uygulama modeli geliştirilmiştir. TÜBİTAK tarafından yayınlanan Güvenli Yazılım Geliştirme Kılavuzu temel alınarak güvenlik test yöntemleri, kuralların tanımı, geliştirilen kodun hangi zafiyetlere yol açabileceği bilgilerinin projelerin geliştirme aşamasında nasıl uygulanacağına dair tasarlanan model açıklanmıştır. Geliştirilen kodun parçalara ayrılarak uygulama güvenlik kurallarından geçirilmesi ve bu kuralların hangi kod bloklarında uygulandığı bilgisinin çalışma kapsamında hazırlanan rapora girilmesi modellenmiştir. Böylece son kullanıcılar ve ticari firmalar yazılımları kullanırken ortaya çıkan güvenlik raporu ile şüphe duymadan güvenli yapıda geliştirilen yazılımları kullanabilecektir. Tasarlanan modelin kullanılabilir olduğunu gösterebilmek adına, model 5 proje ekibine kullandırılmış ve geliştiricilere bir anket uygulanmıştır.

Atıf için/To Cite: Sarıman G. Çelikten H. Yeni bir güvenli yazılım geliştirme uygulama modeli: GYG-MOD. Uluslararası Teknolojik Bilimler Dergisi, 13(1), 39-49, 2021.

A novel secure software development application model: SSD-MOD

Keywords
Secure software development
Security rules
Software life cycle

Article history:
Received: 27.02.2021
Accepted: 26.04.2021

Abstract: With the accelerated speed of technological developments, the internet and mobility have started to effect more into people's daily life. The integration of our life with the internet makes it compulsory to make our personal information and our produced data safer. With the widespread use of the Internet, increasing security vulnerabilities today require systems to be more protected and stronger as infrastructure. These infrastructures are network, system and software based in information systems. Network and system architecture-based security measures redirect to the security rules after the products are made available for end users. In software products, costs are reduced by taking security measures while it is still under construction. In this study, an application model has been developed about the inclusion of the security principle in the software development process while the projects are still in development phase to make the software more secure. Based on the Safe Software Development Guide published by TÜBİTAK, the model designed for how to apply security testing methods, definition of rules, and what developed code may lead weaknesses during the development phase of the projects is explained. It is modeled that the application security rules are modeled by dividing the developed code into the parts and entered the rules into the report prepared within the scope of the study, in which lines and blocks of code are applied. Thus, end users and commercial companies will be able to use the software without any doubt which are

* İlgili yazar/Corresponding author: guncelsariman@mu.edu.tr

developed in a secure structure with the prepared security report. In order to show that the designed model is usable, the model was used by a small software team and a questionnaire was applied to the developers.

1. Giriş

Teknolojinin artan hızıyla birlikte günlük hayatımızın büyük bir bölümünü teknolojik cihazlarda geçirmekteyiz. Devlet daireleri, özel sektör firmaları, sosyal platformlar ve çeşitli kuruluşlar da sundukları hizmetleri artık sanal ortamlara taşımaktadırlar. Bununla birlikte, hizmetlerin dijitalleşmesi beraberinde çeşitli kullanım zorluklarını getirmektedir. Sanal ortamlarda kullanılan, dolaşan ve saklanan veriler, kullanıcılar için önemli mahrem bilgilere sahip olabilmektedir. Bilgilerin ele geçirilmesi ise kişiler üzerinde güvenlik zafiyetleri oluşturabilmektedir. Sosyal medya, elektronik posta, cep telefonları, web siteleri ve online platformlar üzerinden kişi veya kişilerce yapılan, onlara ait olmayan verileri çalmaya, dağıtmaya yönelik eylemler yaparak kişisel haklara saldırı işlemi siber saldırı olarak adlandırılmaktadır. Siber saldırılar, tek bir bireye yapıldığı gibi aynı zamanda kurumlara, bankalara, telefon şirketlerine veya devletlere de yapılabilmektedir. Bu saldırılar büyük veri kayıplarına, kimlik hırsızlığına, uluslararası sırların ortaya çıkmasına, internet erişiminin sağlanamamasına ve daha birçok soruna yol açabilmektedir. Bu sorunların üstesinden gelebilmek için siber güvenlik ön plana çıkmaktadır.

Bilgisayar sistemlerinin, sunucuların, ağların, elektronik sistemlerin insanlar veya kurumlar arası ilişkilerin bütünlüğünün, gizliliğin ve güvenliğinin korunması işlemi siber güvenlik olarak adlandırılmaktadır. Siber güvenlik, bir saldırı ve bunun sonunda verilecek bir zarar veya elde edilecek haksız kazanç, casusluk ve hırsızlığı önlemenin yanı sıra siber saldırganlar tarafından iletişim araçları ile yayılan asılsız haberler, bilgi çarpıtma, insanları galeyana getirme ve propaganda aracı olarak kullanılması gibi durumları da tespit etmektedir [1]. Askeri, finansal ve sağlık gibi kuruluşların barındırdıkları veriler oldukça büyük, önemli ve gizli tutulması gereken verilerdir. Bu veriler, kişilerin kimlik bilgilerini, kurumların özel dosyalarını, gizli tutulması gereken birçok bilgiyi barındırdığından verilere ulaşımın sınırlandırılması siber güvenliğin bir alanıdır. Bu gibi önemli ve büyük verilerin kötü niyetli kişiler tarafından ele geçirilmesi oldukça risklidir.

Günümüzde artan güvenlik açıklıkları, sistemlerin altyapı olarak daha korunaklı ve güçlü olmasını gerektirmektedir. Bu altyapılar bilişim sistemlerinde ağ, sistem ve yazılım temelli olmaktadır. Ağ ve sistem mimarisi temelli güvenlik önlemleri daha çok bilişim ürünlerinin, son kullanıcıların kullanımına sunulduktan sonra alınabilecek güvenlik önlemlerine yönlendirmektedir. Fakat ürünlerin piyasaya sürülmeden henüz yapım aşamasındayken güvenlik önlemlerinin alınmasıyla bilişim maliyetlerinin daha aza indirildiği yapılan çalışmalarda ifade edilmiştir [2]. Bu araştırmalar da güvenli yazılımın son zamanlarda daha çok önem kazanmasına yol açmıştır. Bu durum mevcut olan yazılımların güvenliğinin sağlanmasını, hesaba katılmayan maliyetlerin ortaya çıkmasını da beraberinde getirmiştir. Yazılım sürecinin erken safhalarında alınan güvenlik önlemlerinin önemi hem maliyet hem de yazılımın güvenliği açısından oldukça fazladır.

Kişisel verilerin yetkisiz kişiler tarafından kazara kopyalanması, değiştirilmesi, kötü amaçlar için kullanılması veri ihlali olarak tanımlanmaktadır. Ponemon'un 2020 yılında yayınladığı rapora göre Dünya genelinde 500'ü aşkın şirket üzerinde yapmış olduğu analizde bir veri ihlalinin maliyeti son 5 yıl içerisinde %12 artarak 3,9 milyon dolara ulaşmıştır. Türkiye'den 22 şirketin de katıldığı araştırmanın sonuçlarına göre ise, Türkiye'de veri ihlalinin ortalama toplam maliyeti 2019 yılına kıyasla yüzde 18,5 artarak 11,15 milyon TL'ye yükselmiştir [3]. Bu rapordan yola çıkılarak maliyetler gün geçtikçe artmaya devam etmektedir. Veri ihlali maliyetlerinin en aza indirgenmesi için yazılımın tüm aşamalarının güvenliği oldukça önem arz etmektedir.

Güvenli bir yazılım, zararlı yazılımlara karşı koruma yöntemleri barındırarak geliştirilir. Yazılım zafiyetleri güncel takibi oldukça zordur. Denim Grup tarafından 600 geliştiriciye yapılan anket sonucunda geliştiricilerin yalnızca
%56'sının yazılım güvenliği ile ilgili sorulara doğru cevap verdiği görülmüştür. Anketten çıkarılan bir diğer sonuç ise geliştiricilerin teoride bildikleri güvenlik önlemlerini kendi projelerinde nasıl uygulayacaklarını bilmemeleridir [4].

Aspect Security tarafından 1425 geliştiriciye yapılan ikinci bir ankette ise hassas verilerin korunması konusunda %80'i, web hizmetleri güvenliği konusunda %64'ü, tehdit modelleme ve güvenlik mimarisi inceleme konusunda geliştiricilerin %74'ünün yanlış cevap verdiği tespit edilmiştir [5].

Yukarıdaki araştırma sonuçlarına dayanarak, geliştiricilerin büyük çoğunluğunun güvenli yazılım konusunda yeterli düzeyde bilgi sahibi olmadığı, edindikleri bilgileri ise uygulayamadıkları

görülmektedir. Son yıllarda yazılıma artan ilginin sonucunda birçok yazılım projesinin çok hızlı bir şekilde piyasaya sürülme çabası, proje ekiplerinin çok az sayıda çalışandan oluşuyor olması projelerin güvenlik prosedürlerinden geçirilmemesine neden olmaktadır. Böylece, veri ihlallerinin de uygulama seviyesinde ortaya çıkma oranı artmaktadır.

Raj vd. [6] çalışmalarında, yazılım geliştirme yaşam döngüsü öneminin yanı sıra güvenliğin önemini incelemiş, güvenli web uygulaması geliştirme için süreç tasarlamıştır. Güvenlik sorunları üzerine yapılan tartışma ve analizlerden sonra, güvenliği, bulut bilgi işlem hizmeti geliştirme senaryoları gibi farklı sanallaştırılmış ortamlarda uygulamak için Software Development Life Cycle'yi (SDLC) yükseltmeyi planlamışlardır.

Karim vd. [7] çalışmalarında, Suudi Arabistan'daki yazılım geliştirme metodolojileri araştırarak güvenliği yazılım geliştirme yaşam döngüsüne entegre etmek için bir model oluşturmuştur. Çalışmada önerilen model SDLC'nin her aşamasında gerçekleştirilmesi uygun olan faaliyetler için öneriler ve destekleyici doğrulamalar içermektedir.

Westner [8] çalışmasında, bulut bilişimde bilgi güvenliği riskleri ve bu riskleri azaltma önlemlerini araştırmıştır. Bulut kaynaklı risklerin sınıflandırılması ve bilgi güvenliği üzerindeki etkileri vurgulayıp bu risklere ilişkin çözümleri araştırıp eksik kalan yerleri listelemiştir.

Wijesiriwardana ve Wimalaratne [9] çalışmalarında, çok çeşitli yazılım analizlerini gerçekleştirmek için birden fazla statik analiz araç çıktısını tek bir yere entegre eden kavramsal bir çerçeve sunmayı amaç edinmiştir. Yazılım sistemi, güvenlik açığı algılama işlemine tabii tutulup güvenlik açığı listesi ile basit sınıflandırma modeli sunulmuştur.

Kim ve Ryou [10] çalışmalarında, var olan problem tespit tekniklerinin dışına çıkarak kaynak kod analizi ile dinamik bellek kullanımı için statik bir tahmin tekniği önermiştir. Güvenlik açıklarının tespiti için Kod Klon tespiti, karakteristik kriterlere göre tampon taşması statik analiz yöntemleri kullanmışlardır.

Anis vd. [11] çalışmalarında, Open Web Application Security Project (OWASP) 'den yararlanarak Sql injection, XSS ve kaynak değiştirme saldırılarını önlemiş, geliştiricilere kılavuzlar sağlayarak web uygulamalarını güvenli hale getiren bir yaklaşım sunmuştur. Anis, bu çalışma ile saldırı önleme oranını %33 artırmış, bütünlük doğrulama modülü, kod okuma saldırılarını da uygun okuyucuyla yaklaşık 2,5 saniye kadar kısa bir sürede raporladığı gözlemlenmiştir.

Sarıman ve Küçüksille [12] çalışmalarında, web servislerin güvenliğini test etmek için hibrit bir model geliştirmişlerdir. Statik ve dinamik analiz yöntemleri uygulanmıştır. Geliştirilen modelle bazı zafiyetlerin ve güvensiz kodların kesin olarak tespit edilebildiğini kanıtlamışlardır.

Koç [2] yüksek lisans çalışmasında, güvenli yazılım geliştirmek için uygulamaların scrum çerçevesi içerisinde uygulamasını amaçlayan bir güvenli yazılım modeli önermiştir. Önerilen modelin uygulanabilirliğini sahada çalışan yazılımcılar ile test etmiş ve olumlu sonuçlar almıştır.

Kaplan [13] yapmış olduğu yüksek lisans çalışmasında, mimari, tasarım ve uygulama seviyesinde 30'un üzerinde güvenli tasarım deseni incelemiştir. İncelenen bazı güvenli tasarım desenleri örnek bir uygulamada kullanılmıştır. Bu çalışma ile oluşturulan güvenli tasarım desenleri, yazılım güvenlik açıklarına karşı iyi bir koleksiyon oluşturmuştur.

Bulut [14] yüksek lisans çalışmasında, makine öğrenmesi, metin madenciliği ve doğal dil işleme yöntemleri kullanarak yazılım güvenlik zafiyetlerinin tahmin edilmesini amaçlamıştır. Güvenlik zafiyetleri ile ilgili yapılan çalışmalarda açık kaynak sistemler kullanılmıştır. Zafiyetlerin tespiti için sınıflandırma algoritması ve regresyonu içine alan denetimli öğrenme algoritmaları kullanılmıştır. Oluşturulan modelle, yazılım zafiyetlerini tahmin etmede başarılı sonuçlar elde edilmiştir.

Kaynak taramaları sonucunda, son zamanlarda dikkat çeken siber güvenlik konusu ile yapılan tez, makale ve bildiri çalışmalarında siber güvenliğe ve yazılıma ait çeşitli konularda çalışmalar yapıldığı tespit edilmiştir. Bununla birlikte yazılım güvenliği konusunda oldukça az çalışma olduğu gözlemlenmiş, yapılan çalışmaların çoğunun öneri niteliğinde olduğu ve uygulama aşamasında yeterli tespit ve analizlerin yapılamadığı sonucuna varılmıştır.
Bu çalışmada yazılımların, güvenli yazılım geliştirme sürecinin kurallarına uygun bir şekilde geliştirilmesine katkı sağlayacak bir uygulama modeli geliştirilmiştir. Uygulama modelinin işleyişi ve yazılım yaşam döngüsündeki rolü modelde detaylı bir şekilde açıklanmaktadır. Çalışmanın materyal ve metot bölümünde yazılım yaşam döngüsüne güvenlik prensibinin eklenmesi, güvenli yazılım geliştirme olgunluk modelleri, statik kod analizi ve uygulama güvenlik kuralları anlatılmaktadır. Bulgular bölümünde ise geliştirilen model anlatılmaktadır. Sonuç bölümünde ise geliştirilen modelin uygulanması

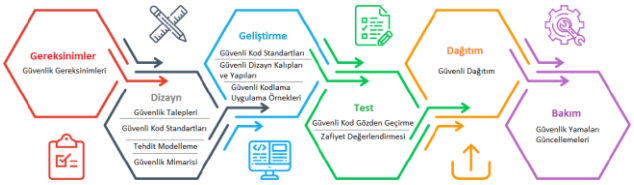
durumunda etki edeceği durumlar tartışılmış, bir proje üzerindeki etkisi ölçülmüş ve farklı alanlardaki etkileri tartışılarak sonraki çalışmalar için yapılabilecekler anlatılmıştır.

2. Materyal ve Metot

2.1. Güvenli yazılım geliştirme yaşam döngüsü

Yeni doğan bir fikrin üretilmesi, fark yaratması ve sektörde tutunması için bir dizi işlemden geçmesi gerekmektedir. Hedeflenen bir işin gerçekleştirilmesi için uyulması gereken yaşam döngüsü öne sürülen fikirlerin işlemlerden geçerek piyasaya sürülmesi, kullanılması ve sonunda yok olması bu döngünün bir gereğidir. Yazılım alanında da bu işlem basamaklarına, döngülerine ihtiyaç duyulmaktadır. Hedeflenen bir yazılımın gerçekleştirilmesi, sistemli bir şekilde uygulanması ve daha sonra doğacak sorunlara bulunacak önlemlerin alınması gibi işlemler yürütülen projenin az zamanda veriminin ve talebin artmasını sağlayabilmektedir.

Yazılım birçok alanda yaygınlaşmakla birlikte, bu alanlarda depo edilen veriler de çoğalmakta ve karmaşıklaşmaktadır. Farklı alanlar için geliştirilen yazılım projelerinde, karmaşıklıkların sistemli bir şekilde sürdürülmesi, doğru ve kaliteli bir şekilde üretimin yapılması için Yazılım Geliştirme Yaşam Döngüsünden (SDLC) yararlanılır. Şekil 1 'de Güvenli Yazılım Geliştirme Yaşam Döngüsü verilmiştir.



Şekil 1. Yazılım geliştirme yaşam döngüsü

2.1.1. Gereksinimler

Gereksinim aşaması SDLC'nin ilk aşamasıdır. Bu aşamada yazılım projesi hakkında müşteriden gerekli bilgiler toplanır. Toplanan bilgilerden yola çıkılarak gereksinimler planlanır ve uygun şekilde kategorize edilir. Üretilecek olan yazılımın, risk analizinin yapılması da gereklidir. Potansiyel risklerin tanımlanması, bu risklere karşı çözüm önerilerinin sunulması, üretilecek olan yazılımın daha sağlam temellere inşa edilmesini sağlayacaktır. Gereksinim aşaması, yeni üretilecek olan bir yazılım için önem arz etmektedir. Yazılımın son kullanıcıya kadar olan bütün işlemleri hiçbir soru işareti kalmayana kadar bu aşamada planlanır.

2.1.2. Dizayn

Bu aşamada, kategorize edilen gereksinimler tüm sistem bileşenleri, kullanıcı etkileşimleri, veritabanı işlemleri geliştirici ve kullanıcılara göre uygun şekilde tasarlanır. Tasarım yapılırken dikkat edilmesi gereken bir diğer konu güvenlik taleplerinin gözden geçirilmesidir. Günümüzde kullanılan sistemlerin barındırdığı bilgilere verilen her türlü zarar ve güvenlik ihlalleri tehdit olarak adlandırmaktadır. Tehdit Modelleme, verilerin sınıflandırılması, güvenlik ihlallerinin en aza indirilmesi için sisteme yönelik her türlü zarara karşı korunma önlemlerinin alınmasıdır. Oluşturulan tasarımın güvenliği, sonradan doğacak zafiyetlere karşı direncin yüksek olmasını sağlayabilecektir.

2.1.3. Geliştirme

Geliştirme aşaması, tasarımın kaynak koda dönüştürüldüğü kısımdır. Geliştirici, yazılım aşamasında müşteri ile etkileşim halindedir. Uygun görülen kısımlar tekrar düzenlenip değiştirilir. Bu aşama sonunda güvenlik önlemleri dikkate alınarak geliştirilen yazılım, test edilebilir ve uygulanabilir hale gelmiş olur. Statik kod analizi de bu bölümde gerçekleştirilir.

2.1.4. Test

Yazılımın kalitesini artırmak için uygulanacak en önemli aşamadır. Geliştirilen yazılımın her yönüyle testi yapılır. Karşılaşılan sorunlar üzerinde tekrar çalışılıp çözülünceye kadar bu evre devam eder. Başarılı bir şekilde tamamlanan test aşaması sonunda yazılım, son kullanıcıya hazır hale gelmiş olur.

2.1.5. Dağıtım

Tamamlanan yazılımın yayınlanma ve izleme aşamasıdır.

2.1.6. Bakım

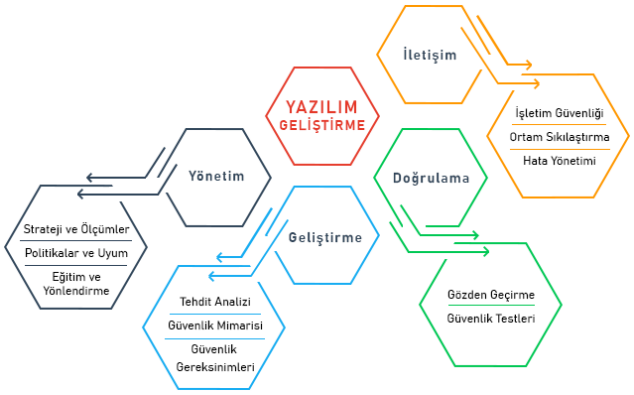
Son kullanıcı tarafından kullanılan yazılımın güncellenmesi, yeni oluşan zafiyetlere karşı önlem alınması gibi tekrarlı eylemler bakım aşamasında gerçekleştirilir.

2.2. Güvenli yazılım geliştirme olgunluk modelleri

Güvenli yazılım geliştirme olgunluk modelleri, bir yazılımın güvenliğinin ne derece uygulandığını derecelendiren ölçüm modelidir. Aşağıdaki alt başlıklarda Açık Web Uygulaması Güvenlik Projesi (OWASP) tarafından belirtilmiş 2 adet model incelenmiştir.

2.2.1. Yazılım güvencesi olgunluk modeli (SAMM)

Software Assurance Maturity Model (SAMM), kuruluşların karşılaştıkları risklere göre uyarlanmış bir yazılım güvenliği stratejisi oluşturmasına ve uygulanmasına yardımcı olan çerçevedir [15]. SAMM bileşenleri Şekil 2'de verilmiştir.



Şekil 2. SAMM bileşenleri [15]

SAMM modelinde en temel risklere karşı alınması gereken önlemler seviye 1, temel seviyenin ötesindeki tehditleri önlemek için alınan önlemler seviye 2, tüm güvenlik risklerine karşı alınması gereken önlemler seviye 3 şeklinde derecelendirilmiştir.

2.2.2. Olgunluk modelinde inşa güvenliği (BSIMM)

Building Security In Maturity Model (BSIMM), dünyanın en gelişmiş güvenlik ekipleri de dâhil olmak üzere tüm şekil ve büyüklükteki kuruluşların yazılım güvenlik stratejilerini nasıl yürüttüklerini anlamak için etkili bir araç olarak hizmet vermektedir [16]. BSIMM modeli etkinlik alanı ve faaliyetleri Şekil 3'te verilmiştir.

Şekil 3. BSIMM Etkinlik Alanları Ve Faaliyetleri [16]

2.3. Statik kod analizi

Bir yazılımın, çalıştırılmadan önce kaynak kodunun incelenerek güvenlik zafiyetlerine neden olabilecek hataların ayıklanma yöntemidir. Geliştirilmekte olan yazılımın kod kalitesini artırmak amacıyla birçok yöntem kullanılır. Bu yöntemler aşağıdaki başlıklar altında incelenmiştir.

2.3.1. Kaynak kod ve tasarım gözden geçirme

Geliştirilen yazılımın, kod kalitesini artırmak, geliştiricilerin kod becerilerini geliştirmek, olası hatalı yazımların düzenlenmesi, proje hakkında ekipteki diğer yazılımcıların da bilgi sahibi olması amacıyla kaynak kod gözden geçirme işlemi yapılır. Ekipçe yapılan kod gözden geçirmesi, yazılım hakkında beyin fırtınalarının oluşmasına, fikir alışverişinin sağlanmasına yol açacaktır. Böylelikle üretilecek olan yazılımın daha sonra yaşanacak zafiyetlerin, kod hatalarının önüne geçilmesi ve en aza indirgenmesi sağlanabilecektir.

2.3.2. Web uygulama açıklık tarayıcısı

Gelişen teknolojik olayların en büyük sonucu olan internet kullanımı ile birlikte masaüstü uygulamaların yerini çoğunlukla web ve mobil uygulamalar almıştır. Web uygulamaları, günlük yaşantımızda ihtiyacımız olan her türlü alanda kullanılmaktadır. Bu eğilimden dolayı son yıllarda, web uygulamaları saldırganların hedefi haline gelmiştir. Saldırıların önüne geçilebilmesi ve güvenli kod geliştirme farkındalığının oluşturulması adına bir dizi güvenlik işlemleri uygulanmalıdır. Buna yardımcı olabilecek uluslararası en aktif proje ise OWASP'tır. OWASP aracılığıyla güncel zafiyet taraması ve gerekli önlemlerin nasıl uygulanacağı konusunda en güncel bilgiler elde edilebilmektedir.

2.4. Uygulama güvenlik kuralları

Tablo 1. Uygulama güvenliği kuralları [17]

Değerlendirme Listesi	Uygulama Güvenlik Kuralları
Lahika-01	Mimari, tasarım ve tehdit modelleme
Lahika-02	Kimlik doğrulama
Lahika-03	Dosyalar ve kaynakların güvenliği
Lahika-04	Oturum yönetimi
Lahika-05	Erişim denetimi
Lahika-06	Güvenli kurulum ve yapılandırma
Lahika-07	Güçlü kriptografik mekanizmaların kullanımı
Lahika-08	Veri koruma
Lahika-09	Hata ele alma ve kayıt
Lahika-10	İletişim güvenliği
Lahika-11	İş mantığı
Lahika-12	Kötücül işlemleri engelleme
Lahika-13	Mobil uygulama güvenliği
Lahika-14	Girdi ve çıktı süzme
Lahika-15	Web servisleri güvenliği
Lahika-16	Kişisel verilerin korunması

Uygulama geliştirme aşamasında yazılımın güvenli bir yapıda geliştirilmesi için dikkat edilmesi gereken aşamaların belirlenmesi ve buna göre proje geliştirme

evresinin sürdürülmesi uygulama güvenliği için önemli bir aşamadır. Yazılım projeleri geliştirilirken, kullanıcı doğrulama, yetkilendirme, izinsiz dosya erişimi, oturum (session ve cookie) yönetimleri, web ve mobil projelerdeki girdi geçerleme, web servisleri ve projeler arasındaki iletişimin sağlanması vb. yöntemler güvenlik perspektifinde dikkat edilmesi gereken en temel güvenli yazılım geliştirme aşamalarındandır. TÜBİTAK Bilgem Güvenli Yazılım Geliştirme Kılavuzunda da güvenlik kuralları 16 temel başlık altında toplanmıştır. Kılavuzda, ISO 27034 Uygulama Güvenliği standardı dikkate alınarak Uygulama Güvenliği Kuralları (UGK) tanımlanmıştır [17]. Çalışmamızda bu kuralların uygulama modeli çıkarılmıştır. Tablo 1'deki listede uygulama güvenliği kuralları verilmiştir.

2.5. Zafiyetlerin raporlandığı uluslararası veritabanları

Zafiyetlerin raporlandığı uluslararası veritabanları sayesinde en güncel zafiyetler hakkında geniş kapsamlı bilgi alınabilmektedir. Bu çalışmada ise TÜBİTAK'ın da referans aldığı OWASP ve Common Weakness Enumeration (CWE) 'deki zafiyet listeleri kullanılmıştır.

2.5.1. Açık web uygulaması güvenlik projesi (OWASP)

OWASP, zararlı yazılımlara karşı mücadele etmek için kurulan bir topluluktur. Geliştirilen yazılımın zafiyetlerinin test edilmesi, tanıtılması ve sızma testi(pentest) alanında deneyim kazanmak isteyen kişilere test alanı sağlar. Çalışmanın uygulama modelinde de OWASP tarafından ortaya çıkarılan zafiyetler ve kod örnekleri referans olarak alınmıştır.

2.5.2. Ortak zayıflık numaralandırması (CWE)

CWE, bilgisayar yazılımlarında karşılaşılabilecek sorunlar için geliştirilmiş zayıflık sözlüğüdür. Hazırlanmış olan yazılımın, son kullanıcıya dağıtıma çıkmadan yazılım hatalarının, güvenlik açıklarının tanımlanmasını, bulunmasını ve nasıl çözülebileceğini kullanıcılarına sunar [18].
CWE, TÜBİTAK Bilgem Güvenli Yazılım Geliştirme Kılavuzunda zayıflık referansı olarak verilmiştir. Verilen referans kodu, (Örneğin; CWE-640) kodun hangi zafiyet için olduğunun açıklamasını, diğer zafiyetlerle olan ilişkilerini, bu zafiyetin nasıl, ne zaman ve hangi platformda ortaya çıkacağını, zafiyet sonucunda ortaya çıkabilecek sorunları ve zafiyete karşı alınması gereken önlemlerin örnek kodlarını içermektedir. Böylelikle geliştirilen yazılımın, yeni çıkan veya hâlihazırda olan zafiyetlere karşı güvenlik önleminin alınması işlemi CWE ile daha ulaşılabilir ve uygulanabilir olmaktadır.

3. Araştırma ve Bulguları

3.1. Güvenli Yazılım Geliştirme Uygulama Modeli: GYG-MOD

Bu çalışmada, yazılımların güvenli hale getirilmesi için, projelerin kullanıma başlamadan önce henüz geliştirme aşamasında iken güvenlik prensibinin yazılım geliştirme sürecine dâhil edilmesi üzerine bir uygulama modeli geliştirilmiştir.

Güvenlik önlemleri, geleneksel yaklaşımda geliştirme aşamasındaki tüm süreçler tamamlandıktan sonra devreye alınmaya çalışılsa da olması gereken, sürecin yazılım geliştirme sürecine dâhil edilmesidir. Bu sebeple yazılım geliştirme aşamasında güvenlik prensiplerinin devreye alınması, sürecin daha sağlıklı ve emin adımlarla ilerlemesini sağlayacaktır.

Geliştiriciler, zaman kısıtları ve proje yaşam döngüsünde güvenliğin planlanmamasından dolayı sadece projeyi çalıştırma ve yürütme üzerine yoğunlaşmaktadırlar. TÜBİTAK tarafından güvenli yazılım geliştirme sürecinin nasıl ilerlemesi gerektiğiyle ilgili bir kılavuz yayınlanmıştır. Fakat bu ve uluslararası seviyede geliştirilen benzeri kılavuzların tavsiye ve uyulması gereken kurallar listesi niteliğinde olup, geliştiriciler ve yazılım ekipleri tarafından genellikle gündeme alınmadığı yapılan araştırmalarda ortaya çıkarılmıştır [4,5].

Bu çalışmada ise kılavuzda [17] belirtilen her modüle ait güvenlik test yöntemleri, kuralın tanımı, geliştirilen kodun hangi zafiyetlere yol açabileceği bilgilerinin, projelerin geliştirme aşamasında nasıl uygulanacağına dair tasarlanan model açıklanmaktadır. Modele göre proje ekipleri geliştirecekleri projenin planlamasını yaptıktan sonra proje yönetim süreçlerini ortaya koymalıdırlar. Uygun olan proje yönetim modeli seçildikten sonra aşama aşama hangi yazılım modüllerinin geliştirileceği ve görev dağılımları yapılmalıdır. Modüllerdeki sınıf ve fonksiyonların türüne göre güvenli kodlamayı destekleyen uygulama güvenliği kural listesi geliştirici tarafından her aşamada kontrol edilmelidir. Her modüldeki kontrol sonrasında projenin tamamlanma aşamasında hangi kontrollerin yapıldığına ait bir rapor ortaya çıkarılmalıdır. Tasarlanan rapor örneği Tablo 2'de gösterilmektedir. Böylece yazılımın yayınlanma veya piyasaya sürme aşamasında güvenlik kontrollerinden hangi kuralları uygulayarak geçtiği kanıtlanabilmektedir.

Yazılım geliştirme aşamasında kontrolü sağlanan güvenlik kuralları, Tablo 2'deki kontrol listesinde proje iş paketleri seviyesinde ve modül bazlı ayrılmaktadır. Her modül veya sınıf temelinde bu kontroller yapılabilmelidir. Yapılan kontroller "Tam Kontrol", "Gerekli Kontrol" ve "Genel Kontrol" olarak detaylandırılmıştır.

Tablo 2. Tasarlanan yazılım güvenlik kontrolü onay raporu

Yazılım Güvenlik Kontrolü Onay Raporu				
Proje		İş Paketi:	Tarih:	
Değerlendirme Listesi-1: Mimari, Tasarım ve Tehdit Modelleme				
	Uygulama Güvenlik Kural Adı	Modül veya Sınıf Adı	Uygulanan Güvenlik Seviyesi	Kontrol Durumu
MT-01	Uygulamanın mimarisi Güvenli Yazılım Geliştirme Kılavuzu'nda belirtilmiş olan güvenli yazılım ilkelerine uygun olmalıdır.			☐ Tam Kontrol ☐ Gerekli Kontrol ☐ Genel Kontrol
MT-02	Uygulamadaki bileşenler hata durumlarında varsayılan olarak güvenli durumlara geçmelidir.			☐ Tam Kontrol ☐Gerekli Kontrol ☐Genel Kontrol
MT-07	Geçersiz olmuş, potansiyel olarak tehlikeli işlevlerin bulunduğu kütüphane ve modüller tasarımda ve uygulamada yer almamalıdır.			☐ Tam Kontrol ☐ Gerekli Kontrol ☐ Genel Kontrol
..	.....			
Kontrol Edilen Kural Sayısı:			Kontrolü Yapılmayan Kural Sayısı:	
İş Paketinden Sorumlu Yazılım Personel (ler)i:		İmza:	Proje Ekip Lideri:	İmza:

Tam Kontrol aşaması, ilgili kod satırlarının tamamında UGK kuralının uygulandığını, Gerekli Kontrol, sadece önemli zafiyet oluşturabilecek UGK' nın uygulandığını, Genel Kontrol ise sadece hızlı gözden geçirme yöntemiyle kontrollerin yapıldığını ifade eder. Bu sayede hizmeti satın alan müşteri, servis sağlayıcı ve geliştirici firma, yazılım seviyesinde gerekli güvenlik kontrollerinin yapıldığını onaylı bir şekilde ispatlayabilecektir. Aynı zamanda ilgili iş paketi ve proje geneli için tehdit modelleme yapısının çıkarılması veya siber saldırı sırasında projenin nerelerde daha zayıf olabileceğine dair öngörü bu rapor sayesinde daha hızlı tespit edilebilecektir.

3.2. Yazılım geliştirme sürecinde GYG-MOD

Her uygulama aynı düzeyde güvenlik kontrollerine sahip olmamaktadır. Bununla birlikte tanımlanan her bir Uygulama Güvenlik Kuralı (UGK) birden fazla uygulama güvenlik seviyesinde geçerli olabilmektedir. UGK'ların hangi seviyelerde geçerli olduğu kurallar tanımlanırken belirtilmiştir. Seviye 1'de uygulanan UGK'lar, sadece en temel risklere karşı alınması gereken önlemi tarif eder. Bu UGK'lar tüm uygulamalar için geçerlidir. Seviye 2'de bulunan UGK'lar, farklı endüstrilerde temel seviyenin ötesindeki tehditleri karşılamak ve e-devlet uygulamalarında kullanılmak üzere geçerlidir. Seviye 3'teki UGK'lar ise uygulamanın güvenli olması için bilinen tüm yazılım güvenliği risklerine karşı alınması gereken önlemleri tarif etmektedir. Bu seviyede bulunan UGK'lar insan sağlığı, kurumsal varlığı tehdit eden tehditlerin var olduğu bir bağlamda çalışan veya milli güvenlikle ilgili uygulamalar ve kritik altyapılar için geçerli olmaktadır [17]. Bu güvenlik seviyeleri, geliştirilecek uygulamanın risk teşkil etme seviyesine göre lahikalardaki UGK'lar içerisinden seçilerek kontrol edilir.

3.2.1. Gereksinimler

Yeni işe alınan geliştiricilerin işe alım süreci ve devamındaki süreçte kurum içi güvenli yazılım geliştirme eğitimleri alması gerekmektedir.

3.2.2. Dizayn

Güvenli Yazılım Geliştirme Kılavuzu'nda yer alan güvenli yazılım geliştirme denetim listesi incelemesi yapılarak projeye başlanması istenmelidir. Bu safhada denetim listesindeki her bir maddenin güvenlik seviyesi kontrol edilmelidir. Böylece projenin ilgili modüllerindeki kontrol önem seviyesi belirlenmiş olacak ve test yöntemi için ön hazırlık yapılabilecektir. Güvenli Yazılım Geliştirme Kılavuzuna göre örnek bir denetim listesi Tablo 3'te verilmiştir.

Tablo 3. Güvenli yazılım geliştirme denetim listesi [17]

Denetim Maddesi	Güvenlik Seviyeleri (1-2-3)	Tespit/Test Yöntemi
Uygulamanın üst seviye mimarisi tanımlanmalıdır.	1,2,3	Gözden geçirme
Güvenli Yazılım Geliştirme Kılavuzu'nda belirtilmiş olan güvenli yazılım ilkelerine uygun olarak uygulama mimari geliştirme çalışmaları yürütülmelidir.	1,2,3	Gözden geçirme
Uygulama gizlilik dereceli bilgi işliyor veya içeriyorsa, veri öğeleri ve onların güvenlik sınıflandırmalarını tanımlayan bir Güvenlik Sınıflandırma Kılavuzu hazırlanmalıdır.	3	Gözden geçirme

3.2.3. Geliştirme

Projenin geliştirildiği modüle göre uygulama güvenliği kuralları ve her bir kuraldaki alt denetimlerin uygulanacağı güvenlik seviyelerine göre ilgili kod satırları (sınıf, fonksiyon vb.) kontrol edilmelidir. UGK' ya göre örnek bir kural şablonu Tablo 4'te verilmiştir.

Tablo 4. Uygulama güvenliği kural şablonu [17]

Sıra No	UGK No	UGK Açıklaması	Uygulama Güvenlik Seviyeleri		
			1	2	3
1	MT-01	Uygulamanın mimarisi Güvenli Yazılım Geliştirme Kılavuzu'nda belirtilmiş olan güvenli yazılım ilkelerine uygun olmalıdır.	X	X	
2	MT-02	Uygulamadaki bileşenler hata durumlarında varsayılan olarak güvenli durumlara geçmelidir.	X	X	X
3	MT-07	Geçersiz olmuş, potansiyel olarak tehlikeli işlevlerin bulunduğu kütüphane ve modüller tasarımda ve uygulamada yer almamalıdır.		X	X

Kılavuzdaki UGK' ya ait lahikalardan (Değerlendirme Listesi) birincisi Tablo 5'te verilmiştir. Lahika-01, Mimari, Tasarım ve Tehdit Modelleme, ilgili yazılımın uyması gereken kuralı vermektedir. Örneğin Tablo 4'ün 3. maddesinde güvenlik seviyesi (MT-07) 2. ve 3. seviyedeki uygulamalar için uygulanmalıdır. Tasarlanan yazılım güvenlik kontrolü onay raporu, her bir değerlendirme listesindeki maddeler için uygulanacağı güvenlik seviyesine göre ayrı ayrı doldurulmalıdır. Değerlendirme listesindeki her bir güvenlik kuralına ait detaylar için Tablo-5'e bakılmalıdır. Tüm kuralların listesi [17]' de verilmiştir.

Tablo 5. Uygulama Güvenliği Kuralı. [17]

Kural no: MT-01				
Kural tanımı: Uygulamanın mimarisi Güvenli Yazılım Geliştirme Kılavuzu'nda belirtilmiş olan güvenli yazılım ilkelerine uygun olmalıdır.				
Referans: GYGK/CWE	Güvenlik Seviyesi:	1	2	3
Karşı koyduğu zayıflıklar:Güvenli Tasarım İlkelerinin İhlali	Zayıflık referansı: CWE-657			
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme				

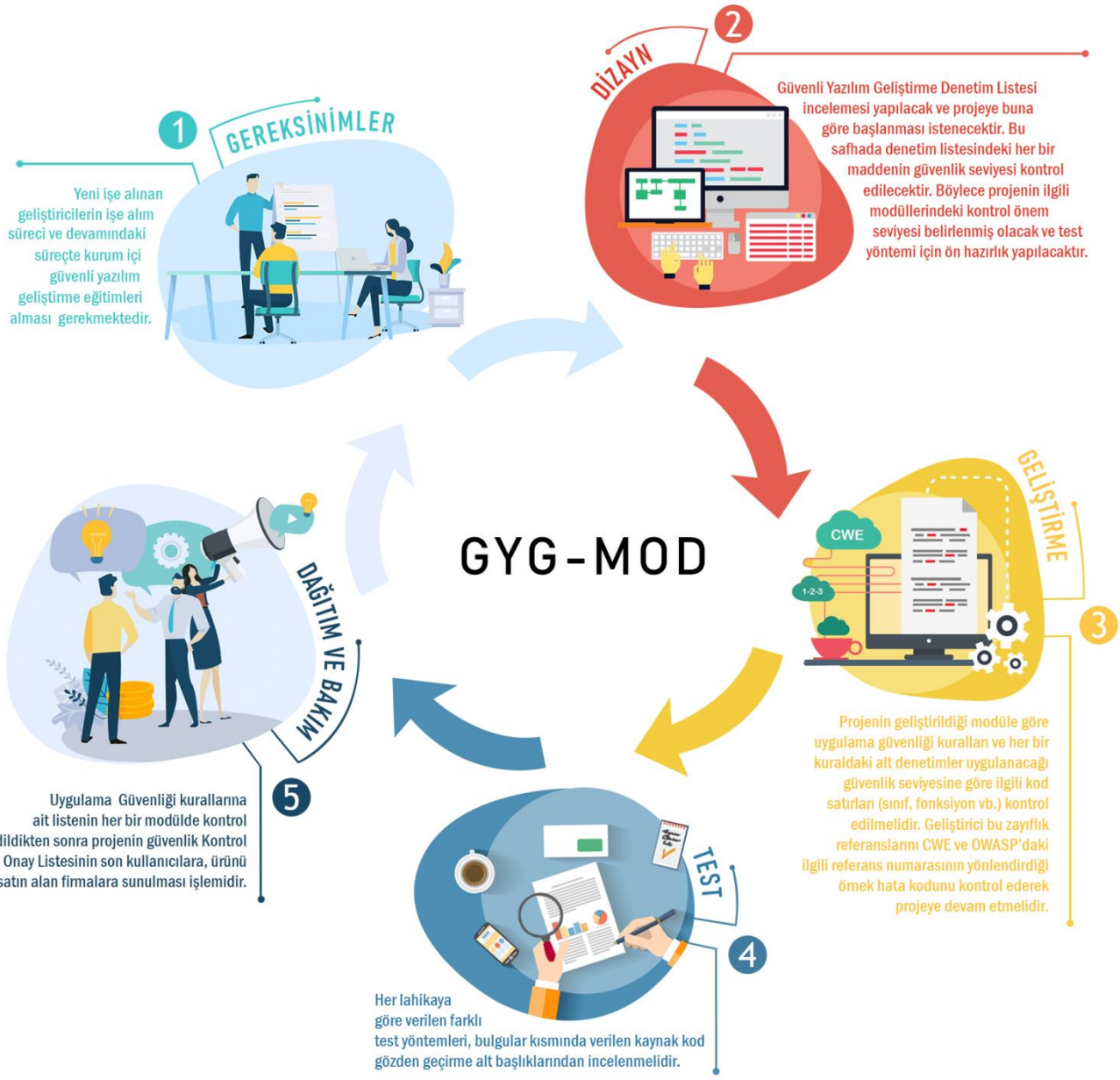
Bu kontroller yapılırken Güvenli Yazılım Geliştirme Kılavuzu'ndaki lahikalar altındaki zayıflık referansları kontrol edilmelidir. Geliştirici bu zayıflık referanslarını CWE ve OWASP'taki ilgili referans numarasının yönlendirdiği örnek hata kodunu kontrol ederek projeye devam etmelidir. Verilen zayıflık referansı https://cwe.mitre.org/ adresinden aratılmalıdır.

3.2.4. Test

Test yöntemi, Tablo 3'te verilmiştir. Verilen test yöntemi her bir proje modülünde uygulanmalıdır. Her lahikaya göre verilen farklı test yöntemleri, bulgular kısmında verilen kaynak kod gözden geçirme alt başlıkları dikkate alınarak incelenmelidir.

3.2.5. Dağıtım & bakım

Bu aşamada, UGK'ya ait listenin her bir yazılım modülünde kontrol edildikten sonra, kontrolü sağlanmış güvenlik kontrol onay raporunun son kullanıcılara ve ürünü satın alan firmalara verilmesi planlanmaktadır.
Şekil 4'te GYG-MOD için tasarlanan yaşam döngüsü ve uygulama modeli diyagramı verilmiştir.



Şekil 4. Tasarlanan güvenli yazılım geliştirme uygulama modeli: GYG-MOD

Çalışmada, TÜBİTAK'ın yayınlamış olduğu Güvenli Yazılım Geliştirme Kılavuzu temel alınarak, bir yazılımcının dikkat etmesi gereken adımlar tasarlanan uygulama raporunda listelenmiştir. Geliştirici, projenin her modülünde veya aşamasında, ilgili kod satırlarında oluşabilecek zafiyet türlerine göre güvenlik denetim listesini kontrol ederek önerilen güvenlik kurallarını koda uygular ve kod bloklarının bulunduğu sınıfa veya fonksiyona uyguladığı UGK'yı ve denetim seviyesini modelde tasarlanan rapora yazarak tamamlar.

Bu şekilde ilgili modül kontrolü sağlandıktan sonra raporun en altında geliştirici ve proje sorumlu adı imza ile birlikte doldurulur. Bu sayede proje döngüsü tamamlandığında ilgili kod satırlarına uygulanan UGK'ların raporu elde edilmiş olacaktır. Rapor, geliştirilen yazılım projesinin TÜBİTAK'ın önerdiği uygulama güvenliği kurallarına uyarak tamamlandığını ve projeden yazılımsal boyutta herhangi bir güvenlik zafiyeti beklenmemesi gerektiğini sağlayabilecektir.

Tasarlanan uygulama modeli 5 proje ekibine test amacıyla uygulanmıştır. Projelerin tümü web uygulaması olduğu için 16 alt başlıkta toplanan uygulama güvenliği kurallarının hepsi kontrol edilmemiştir. Lahika 13 dışında tüm UGK lar denetlenmiştir. 157 adet güvenlik listesi tüm projelerin farklı modüllerinde ihtiyaca göre uygulanmıştır. Projelerden 4'ü tamamlanan 1 tanesi ise yeni başlayan projedir. Tamamlanan projelere uygulanan kurallar ve onay raporu hazırlama için yazılım geliştirme ekipleri ortalama 30 iş günü fazladan zaman harcamıştır. Yeni

başlanan projede ise onay raporları, geliştirme aşamasındayken ilgili geliştirici tarafından yazılan ve kontrolleri sağlanan UGK lar için projenin tamamlanma süresinin sadece 10 günü aştığı geliştiriciler tarafından ifade edilmiştir.

Ayrıca daha geliştirme aşamasındayken kodu yazan kişinin güvenlik kontrollerini sağlaması, yazılım test sürecinin de kısalmasını sağladığı ifade edilmiştir. 5 Proje ekibinden 10 kişiye yapılan ankette ise tasarlanan GYG-MOD raporunun etkinliği ölçülmüştür. Tablo 6 da anket soru ve cevapları verilmiştir.

Tablo 6. GYG-MOD değerlendirme anketi

Soru	Evet	Hayır
Yazılım projesinin teslim tarihini işveren mi planlıyor?	9	1
Yazılım Güvenliği Prensipleri hakkında Bilginiz Var mı?	4	6
Yazılım Projesini teslim etmeden önce güvenlik kontrollerini sağlıyor musunuz?	3	7
Projenize uyguladığınız GYG-MOD, projenizi geliştirme aşamasında denetleyen bir yapıya sahip mi?	10	0
GYG-MOD sebebiyle projenizin teslim tarihi uzadı mı?	6	4
Hazırlanan raporlar daha sonraki süreçte size yardımcı olur mu?	10	0

Yukarıdaki cevaplar incelediğinde GYG-MOD ile yapılan kontrollerin ve hazırlanan raporların, geliştirme sürecinde güvenli yazılım geliştirme sürecini olumlu etkilediği fakat bu denetim listesinin hazırlanması için iş verenin de projeyi tamamlattırmak için baskı halinde olmaması gerektiği anlaşılmaktadır. Ayrıca projenin teslim tarihinin de ilk bakışta uzamış olduğu görünse de herhangi bir zafiyet halinde geriye dönük düzeltmelerin maliyeti daha da arttırdığı 4 geliştirici tarafından ifade edilmiştir. Hazırlanan raporların da ekibe katılacak yeni kişilere yol gösterici olabileceği, satın alacak firmalara da doğrulama ve sertifikalandırma amacıyla verilebileceği ifade edilmiştir.

4. Sonuçlar

Siber güvenlik çerçevesinde bilişim ve teknoloji servislerinin korunaklı ve herhangi bir zafiyete yol açmaması için, kullanıcıların dikkati kadar bu servislerin de olabildiğince güvenli olması gerekmektedir. Bu güvenlik perspektifi ağ, sistem ve yazılım gibi ana unsurlar çerçevesinde değerlendirilebilmektedir. Bilişim servislerinin internet ağı üzerindeki veri transfer süreçleri, sunucuların güvenlik kurallarına göre ayarlanması, geliştirilen yazılımların da güvenli yazılım perspektifi ile geliştirilmesi bu çerçevede siber güvenliğin temel taşlarını oluşturmaktadırlar.

Bu çalışmada, geliştirilen yazılımların genellikle güvenlik kurallarına dikkat edilmeden geliştirildiğini dikkate alarak [4], sağlıklı bir geliştirme sürecinin daha iyi tasarlanabilmesine katkı sağlamak adına bir uygulama modeli kurulmuştur. Uygulama modeline göre güvenli yazılım geliştirme kurallarının, yazılım geliştirme yaşam döngüsünün her aşamasında uygulanması gerekliliği ortaya koyulmaktadır. Anket çalışmasına verilen cevaplar da bu modelin proje ekiplerince kullanılabileceğini göstermektedir.

Bu çalışmanın, yazılım projelerinin piyasaya sürülmesi ve satış aşamasında "güvenli uygulama prensipleri dikkate alınarak geliştirilmiştir" etiketi almasına aracı olması planlanmaktadır. Böylece son kullanıcılar ve ticari firmalar yazılımları kullanırken şüphe duymadan güvenli yapıda geliştirilen yazılımları kullanabilecektir. Ayrıca bu model temel alınarak geliştirilen yazılımların çoğalması durumunda, piyasada daha ucuz ve kalitesiz yazılımların ortadan kalkması, bu sayede ulusal ve uluslararası siber güvenliğin yazılımsal boyutta üst seviyede olması sağlanabilecektir.

Sistemlere yapılan herhangi bir siber saldırı anında, bu raporlar sayesinde yazılımların daha çok dikkat edilmesi gereken bölüm veya kod bloklarının tespit edilebilmesi daha hızlı olabilecektir. Tehdit modelleme yapısı da daha hızlı ve kolay çıkarılabilecektir.

Yazılım geliştirme son yıllarda popülerliğini ve yaygınlığını arttırarak daha çok ürün çıkartma tarafına odaklanmıştır. Bu da güvenlik bakışını arka plana itmiştir. GYG-MOD yapısı yazılım firmalarına ve kod geliştiricilere, var olan yazılı kural kümelerinin yazılım geliştirme sürecinde hangi aşamalarında neler yapılması gerektiğini daha net ve uygulamaya dönük bir yapıda anlatmaktadır. Geliştiricilere, yazılım yaşam döngüsünün aşamalarında yol gösterici ve güvenlik raporu hazırlama imkânı sağlamaktadır. İlerleyen çalışmalarda ise GYG- MOD yapısı geniş çaplı yazılım firmalarına anlatılarak projelere bu modelin uygulanması, projelerine ait GYG-MOD raporlarının hazırlanması ve bu raporlara göre tehdit modellemelerinin çıkarılması planlanmaktadır.

Kaynaklar

[1] Kaya A., Öğün M. N., Siber Güvenliğin Milli Güvenlik Açısından Önemi ve Alınabilecek Tedbirler, *Güvenlik Stratejileri Dergisi*, 9(18), 145–181, 2013.

[2] Koç G., Yazılım geliştirme modellerinin güvenlik açısından analizi ve bir güvenli yazılım geliştirme modeli önerisi, Yüksek Lisans Tezi (MSc Thesis), Hacettepe Üniversitesi, Ankara, Türkiye, 2017.

[3] IBM News Room., IBM Araştırması: Türkiye'de her bir veri ihlalinin ortalama maliyeti 11,15 milyon TL, https://tr.newsroom.ibm.com/2020-03-06-ibm-

survey-average-cost-of-data-breach (Erişim Tarihi: 06.12.2019)

[4] Dickson B. J., AppSec Survey 2.0 Fine-Tuning an AppSec Training Program Based on Data, https://www.slideshare.net/denimgroup/appsec-survey-20-finetuning-an-appsec-training-program-based-on-data-final-91714 (Erişim Tarihi: 06.12.2019)

[5] Vonnegut S., 5 Steps That WILL Raise Your Developers Information Security Awareness, Checkmarx, https://www.checkmarx.com/2015/07/17/5-steps-that-will-raise-your-developers-information-security-awareness/ (Erişim Tarihi: 06.12.2019)

[6] Raj G., Singh Dr. D., Bansal Dr. A., Analysis for Security Implementation in SDLC, *2014 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence)*, Noida, Hindistan 221-226, 25-26 Eylül 2014.

[7] Abdul Karim N. S., Albuolayan A., Saba T., Rehman A., The Practice of Secure Software Development in SDLC: An Investigation Through Existing Model and a Case Study, *Wiley Online Library*, 9, 5333–5345, 2016.

[8] Wulf F., Strahringer S., Westner M., Information Security Risks, Benefits, and Mitigation Measures in Cloud Sourcing, *21st Conference on Business Informatics (CBI)*, Moscow, Rusya, 258-267, 15-17 Temmuz, 2019.

[9] Wijesiriwardana C., Wimalaratne P., On the Detection and Analysis of Software Security, *2017 International Conference on IoT and Application (ICIOT)*, Nagapattinam, Hindistan, 1-4, 19-20 Ekim, 2017.

[10] Kim S., Ryou J., Source Code Analysis for Static Prediction of Dynamic Memory Usage, *2019 International Conference on Platform Technology and Service (PlatCon)*, Jeju, Güney Kore, 1-4, 28-30 Ocak, 2019.

[11] Anis A., Zulkernine M., Iqbal S., Liem C. and Chambers C., Securing Web Applications with Secure Coding Practices and Integrity Verification, *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/ PiCom/ DataCom/ CyberSciTech)*, Athens, Yunanistan, 618-625, 12-15 Ağustos, 2018.

[12] Sarıman G., Küçüksille E. U., Web Servislerinin Yazılım Güvenlik Testleri İçin Önerilen Hibrit Yaklaşım, *SDU International Journal of Technological Science*, 8(2), 1–14, 2016.

[13] Kaplan F., *Yazılım güvenlik açıklarına karşı güvenli tasarım desenlerinin incelenmesi ve uygulanması*, Yüksek Lisans Tezi, KTO Karatay Üniversitesi, Konya, Türkiye, 2019.

[14] Bulut F. G., *Predicting Software Vulnerabilities Using Topic Modeling With Issues*, Yüksek Lisans Tezi, Istanbul Technical University, İstanbul, Türkiye, 2019.

[15] Chandra P., Software Assurance Maturity Model, https://www.owasp.org/images/6/6f/SAMM_Core_V1-5_FINAL.pdf (Erişim Tarihi: 10.12.2019)

[16] BSIMM, About the BSIMM, https://www.bsimm.com/about.html (Erişim Tarihi: 10.12.2019)

[17] TUBİTAK, Bilişim ve Bilgi Güvenliği İleri Teknolojiler Araştırma Merkezi, Siber Güvenlik Eğitim Portalı, May 28, 2018, URL: https://egitim.sge.gov.tr/pluginfile.php/6115/mod_page/content/26/SGE-KLV-GuvenliYazilimGelistirmeKilavuzu_R1.1.pdf (Erişim Tarihi: 25.10.2019)

[18] Common Weakness Enumeration, https://cwe.mitre.org/ (Erişim Tarihi: 26.12.2019)