Trustworthy Scrum: Scrum ile Güvenli Yazılım Geliştirme

Trustworthy Scrum: Development of Secure Software with Scrum

Güler Koç
Bilgisayar Mühendisliğii Bölümü
Hacettepe Üniversitesi
Ankara, Türkiye
Email: glrkoc@gmail.com

Murat Aydos
Bilgisayar Mühendisliği Bölümü
Hacettepe Üniversitesi
Ankara, Türkiye
Email: maydos@hacettepe.edu.tr

Özetçe —**Yazılım geliştirme süreçleri, yazılımların planlanan zamanda ve planlanan maliyetle bitirilmesi için aşamaların düzen ya da sıralarının organize edilmesine odaklanmaktadır. Ancak piyasaya sürülen yazılımlardaki zafiyetlerin yaygınlığı ve çokluğu, zaman ve maliyet kriterlerini sağlamakla beraber büyük ölçüde güvenliğe odaklanan daha sıkı bir yazılım geliştirme sürecine gereksinim olduğunu göstermektedir. Bununla beraber, güvenlik odaklı aktivitelerin eklendiği geleneksel yazılım geliştirme süreçlerinin, güvenliği sağlamada oldukça etkili olduğunu ve çevik geliştirme süreçlerinin geleneksel yazılım geliştirme süreçlerine göre zaman ve maliyet kriterlerini sağlama konusunda daha başarılı olduğunu gösteren bazı çalışmalar mevcuttur. Yapılan bu tespitten ve yukarıda bahsedilen ihtiyaçtan yola çıkarak bu çalışmada, güvenli yazılım geliştirmek için hem çevik yöntemlerin hızlı, değişime ayak uyduran ve aşamalı ilerleyen yapısından yararlanılarak, hem de yazılımda güvenliği sağlamak için gerekli olan ve büyük şirketler tarafından uygulanarak başarısı kanıtlanmış¸ olan güvenlik aktiviteleri kullanılarak güvenlik destekli Scrum modeli (Trustworthy Scrum) önerilmiştir. Güvenlik uygulamaları Scrum çerçevesine eklenirken, geleneksel metotlardaki güvenlik yaklaşımı ile çevik yazılım ilkelerinin çeliştiği durumu değerlendirilerek, yazılım güvenliği ve çevik yazılımın temel ilkelerinin korunması göz önünde tutulmuştur.**

Anahtar Sözcükler—Güvenli yazılım geliştirme; Çevik yöntemler; Scrum.

Abstract—Software development process models focus on ordering and combination of phases to develop the intended software product within time and cost estimates. However, commonness of software vulnerabilities in the fielded systems show that there is a need for more stringent software development process that focuses on improved security demands. Meanwhile, there are some reports that demonstrate the efficiency of existing security enhanced conventional processes and success of agile projects over conventional waterfall projects. Based on this finding and the demand for secure software, we propose a security enhanced Scrum model (Trustworthy Scrum) by taking advantages of both security activities and Scrum framework which has fast adaptation and iterative cycle. While enhancing Scrum with security activities, we try to retain agile and security disciplines by considering that conventional security approach conflicts with agile methodologies.

Keywords—Secure software development; Agile methodologies; Scrum.

978-1-5386-0930-9/17/$31.00 © 2017 IEEE

I. GİRİŞ

Geçmişten günümüze kadar etkin bir şekilde kaliteli ve olgun yazılım üretmek için farklı yazılım geliştirme süreç modelleri oluşturulmuştur. Bu modellerde daha çok planlanan zaman ve maliyet kriterlerini sağlamak için aşamaların hangi düzende uygulanacağına odaklanılmıştır. Ancak yazılım projelerindeki başarısızlık oranlarına bakıldığında istenen sonucun elde edilemediği görülmektedir. “The Standish Group” adlı şirketin 1995 yılında yayınladığı “Chaos” raporuna [1] göre, yazılım projelerinin gerçek başarı oranı, %9-28 arası başarıyla çok düşük seviyelerdedir. “Chaos” raporunda [1] başarı tanımı, “projeyi başlangıçta belirlenmiş tüm özellik ve işlevlerini yerine getirerek planlanan zamanda planlanan bütçeyle bitirmek” olarak verilmiştir. Yani projenin güvenlik, hatasızlık ve sağlamlık boyutları hesaba katılmamıştır. Sahaya sürülen yazılımların güvenlik açısından incelendiği raporlar, başarılı görülen projelerin aslında birçok güvenlik sorunları ve programlama hatalarına sahip olduğu gerçeğini ortaya çıkarmaktadır. “Coordination Center Computer Emergency Response Team (CERT/CC)” tarafından yayınlanan raporlara göre çok yaygın ve gittikçe artan bir güvenlik açığı sorunu mev-cuttur. CERT/CC 2002 yılında 4,129 adet rapor edilmis¸ sistem açığı tanımlamıştır, bu sayı 2001 yılındaki sayının % 70’i kadar artış ve 2000 yılındaki sayının ise 3 katı kadar bir artış olduğunu göstermektedir [4, 5, 6]. Bununla beraber yazılım-ların karmaşıklığı ve genişleyebilirliği ihtiyaçları karşılamak için gittikçe artmakta ve sonuç olarak hata olma olasılığı da oldukça yükselmektedir. CERT/CC tarafından yapılan başka bir analize göre güvenlik açıklarının %90’dan fazlası, kodlama, tasarım ve gereksinim hataları gibi birçok hata tipinin dâhil olduğu bilinen yazılım hatalarından kaynaklanmaktadır [26]. Bu analiz, bilinen bu hata tiplerine karşı yapılan küçük bir iyileştirmenin bile oldukça etkili sonuçlar vereceğini göster-mektedir.

Yazılım endüstrisinde günümüzün oldukça gereksinim duyulan güvenlik talebini karşılamanın anahtarı, güven-liği ortaya koyan tekrarlanabilir işlemlerin gerçekleştir-ildiği yazılım sürecidir [7]. Yazılım geliştiren bazı şirketler,

varolan geliştirme süreçlerine güvenlik uygulamaları ekleyerek, yazılım geliştirme yaşam döngüsü boyunca güven-lik açıklarını bulup minimize etmeyi amaçlamışlardır. Microsoft'un varolan standart yazılım geliştirme sürecine güvenlik aktivite ve elementlerini eklemesi sonucu oluşturduğu güvenli yazılım geliştirme süreci olan Microsoft Security Development Lifecycle (SDL) ile daha önceden 16 olan yıllık yayınlanan güvenlik bülten sayısını 3'e düşürdüğü görülmektedir [7]. Aynı şekilde Microsoft, SDL ile geliştirilen ve piyasaya sürülen yazılımlarda sistem açıklarının %50'den daha fazla oranda düştüğünü belirtmiştir [9]. Bununla beraber güvenli yazılım geliştirmek için yazılım mühendisliği konusunda disiplinli ve olgun ilkeler sunan Team Software Process (TSP) ile kalitenin %20 arttığı, öngörülen takvim planlamasından ve harcanan efordan %8 tasarruf sağlandığı rapor edilmiştir [10]. Yukarıda bahsedilen ihtiyaçtan hareketle bu çalışmada, güvenli yazılım geliştirmek için güvenliği hedefleyen uygu-lamaların Scrum çerçevesine entegrasyonu üzerine çalışılarak Scrum çerçevesi ile güvenli yazılım geliştirme üzerine bir model önerisi (Trustworthy Scrum) yapılmıştır. Ancak bu çalışmanın en büyük zorluğu yazılım güvenliğinin Scrumın çeviklik yönüyle bazı noktalarda uyuşmamasıdır. Küçük it-erasyonlar ile geleneksel yazılım geliştirme modellerindeki güvenlik yaklaşımı uyuşmamaktadır [2] ve küçük iterasyonlar gerekli testleri yapmak için yeterli değildir [11, 12]. Çevik modellerin, güvenlik olaylarını göz ardı etmesinin nedeni, güvenlik konusunda farkındalığın olmaması [11] veya güven-liğin yazılım geliştirmeyi aksattığı ve yavaşlattığı şeklindeki yanlış¸ kanı da [13] olabilir.

Güvenlik güvencesini çevik yöntemlere uyarlamaya çalışan Beznosov ve Kruchten [2], güvenlik güvencesi metotlarını çevik yöntemlerle çakışma derecelerine göre sınıflandırmış ve uyumsuzluğu azaltıcı önermeler yapmıştır. Güvenlik testi yarı uyarlanabilir kategorisinde verilmiş ve araçların otomatikleştirilmesi önerilmiştir. Ancak her ne kadar test otomasyonu faydalı olsa da, uygulamaya özgü test geliştirmesi, güvenlik uzmanı tarafından güvenlik yönelimli test anlayışı gerektirmektedir. Sistemin kapsamlı dokümantasyonu ve harici güvenlik veya resmi doğrulama uzmanlarının gerekliliği gibi çevik yöntemlerle uyuşmayan uygulamalar için iki tane olasılık sunulmuştur. Gerçekleştirmesi zor olan yöntem, çevik yöntemlere uygun yeni güvenlik metotlarının oluşturulması olarak verilmis¸ ancak bu yöntemlerin ne olacağı konusunda bir anlayış sunulmamıştır. Önerilen diğer yöntem ise güvenlik metotlarının, geliştirme yaşam döngüsü boyunca en az iki defa uygulanmasıdır.

Waryrnen çalışmasında [15], XP Planning Game aşamasına, kötüye kullanım senaryoları ve güvenlikle alakalı kullanıcı hikayelerinin oluşturulmasını dahil etmeyi önermektedir. Burada önerilen yöntemin, senaryolardaki tehditler ve bunları önlemeye yönelik olarak belirlenen kodlama ve tasarım standartlarını ilgili senaryolarla eşleşme yapmaya yardımcı olacağı öngörülmektedir.

Başka bir çalı¸smada [16], web uygulamalarını hedef alan güvenlik destekli Scrum versiyonu olan S-Scrum önerilmiştir. S-Scrum, paydaşlar tarafından olu¸sturulan gereksinimlerden güvenlik için hikâye oluşturup eğer kritikse devam etmekte olan Sprintin sonlandırılarak güvenlik gereksiniminin gerçekleştirimi ve testinin yapılmasını, kritik değilse sonraki Sprinte kadar beklemesini önermektedir. Ancak S-Scrum daha çok kötüye kullanım senaryolarıyla desteklenen güvenlik gereksinimlerini oluşturma sürecine odaklanmakta, tasarım, gerçekleştirim ve doğrulama aşamalarında ne tür uygulamalar yapılacağına dair bir yöntem sunmamaktadır.

Scrum üzerine başka bir çalışma olan Secure-Scrum [17] ise, güvenlik kaygılarının gösterimi için S-Tag ve bunları ilgili kaleme bağlayan S-Mark denilen işaretlerin kullanımına dayanmaktadır. S-Tag, kullanım senaryosu, kötüye kullanım senaryosu veya güvenlik gereksinimini anlatan herhangi bir gösterim olabilir. Güvenlik olaylarını tanımlama işlemi, paydaşlar tarafından öneme sahip parçalara odaklanarak, güvenlikle alakalı kullanıcı hikâyeleri tanımlama üzerine kuruludur.

Scrum üzerine olan bu iki çalışmada [16, 17], yazılımın sadece paydaşlarla belirlenen parçalarının tanımlanarak bu parçaların güvenliğinin sağlanması amaçlanmıştır ve içsel tehditlere yönelik uygulamalar sunulmamıştır. Ancak güvenlik tüm sistemi ilgilendiren gerekli bir nitekliktir. Örneğin, arabellek taşması, gizlilik, bütünlük, kullanılabilirlik gibi güvenlik özellikleri veya kritik bir arayüzde ortaya çıkmasından bağımsız bir güvenlik problemidir [18]. Yazılım güvenliği, riske dayalı güvenlik gereksinimlerinin belirlenip önleminin alınmasıyla beraber, yazılım içerisinde güvenlik sorunlarına yol açacak, programlama veya tasarım hatalarından oluşabile-cek içsel tehditlerin de irdelenmesini gerektirir [4, 18, 19, 20]. Güvenlik gereksinimlerini, paydaşlarla yazılımın kritik parçalarına odaklanarak belirlemek bunu sağlamada yetersiz kalabilir. Aynı zamanda güvenlik aktivitelerinden olan eğitim, kodlama standartları, güvenli tasarım ilkeleri, kod ve tasarım gözden geçirmeler, statik analiz araçlarının kullanımı gibi uygulamalara değinilmemiştir. Trustworthy Scrum güvenliğe, yazılımın sadece kritik parçalarını değil tüm yazılımı ilgilendiren bir nitelik olarak bakmakta ve geliştirme süreci boyunca tüm aşamalarda güvenliği hedefleyen uygulamalar içermektedir. Ayrıca sadece web uygulamaları değil tüm yazılımlar için kullanılabilir bir yöntem sunmaktadır.

Başka bir çalışmada [21], güvenlik gereksinimleri için ayrı bir güvenlik ürün iş listesi oluşturulması önerilmektedir. Ancak güvenlik geliştirmesi ve standart geliştirmenin ayrı yürütülmesi, yazılım güvenliğinin, "yazılım geliştirme süreci boyunca güvenlik odaklı düşünüp güvenliğin her geliştirme aşamasına dahil edilmesi" ilkesini karşılamamaktadır. Trustworthy Scrum, güvenlik gereksinimlerinin varolan Ürün ¨Iş Lis-tesine eklenmesini ve standart gereksinimlerden ayırt edilecek şekilde etiketlenmesini önermektedir. Bazı güvenlik gereksin-imlerinde alınan tasarım kararları diğer standart gereksinim-lerin geliştirmesini etkileyebileceği için aralarında bağlantı kurulmasına imkân vermektedir.

Aşağıda bu makaledeki çalışmanın dayandığı iki temel konu olan güvenli yazılım geliştirme ve Scrum çerçevesi hakkında ön bilgi verilmiştir:

A. Güvenli Yazılım Geliştirme İlkeleri

Yazılım güvenliği, yazılımı geliştirirken güvenli olacak şekilde inşa etmektir, tasarımı güvenli yapmak, güvenli kodlama ilkelerine uymak, yazılımın güvenli olduğundan emin olmak, geliştiricileri güvenlik konusunda eğitmek gibi birçok uygulamayı kapsar [4]. Güvenli yazılım geliştirme modellerinde bulunması gereken önemli uygulamalar aşağıda verilmiştir:

1) Güvenlik ekibi yazılım geliştirme organizasyonu bünyesinde oluşturulmalı, bu yüzden tüm proje ekibi üyelerine güvenlik eğitimi verilmelidir. Gerektiğinde sürece güvenlik uzmanı dahil edilmelidir.
2) Güvenlik gereksinimleri belirlenmeli ve kötüye kullanım senaryoları oluşturulmalıdır.
3) Risk analizi yapılmalıdır.
4) Tasarım aşamasında gerçekleştirilmesi gereken güvenlik eylemlerine değinilerek en iyi tasarıma karar verilmelidir.
5) Kodlama standartlarına ve güvenli kodlama ve test ilkelerine uyulmalıdır.
6) Statik analiz araçları kullanılmalıdır.
7) Kod ve tasarım gözden geçirme yapılmalıdır.
8) Güvenlik testi, nüfuz testi ve bulandırma testi yapılmalıdır.
9) Kod yeniden yapılandırma ile güvenlik eklemeleri ve düzeltmeleri yapılmalıdır.
10) Güvenlik dökümanları oluşturulmalıdır.

B. ***Scrum***

Scrum, Scrum kılavuzunda [22] aşağıdaki gibi tanımlanmaktadır: Scrum, 1990'ların başından beri karmaşık ürün geliştirme sürecini yönetmek için kullanılan bir süreç çerçevesidir. Scrum, bir ürün geliştirme tekniği veya süreci değildir; içerisinde çeşitli süreçleri ve teknikleri kullanabileceğiniz bir çerçevedir. Scrumın temelinde deneysel süreç kontrol teorisi yer alır. Scrum, öngörülebilirliği en iyi seviyeye çıkarmak ve riski kontrol etmek için iterasyonlu ve artımlı bir yaklaşım kullanır.

Bu çalışmada, günümüz yazılımlarındaki güvenlik açığı sorununa dikkat çekilerek, yazılımlardaki bu güvenlik gereksiniminin yazılım geliştirme süreçleriyle sağlanması amaçlanmıştır. Güvenlik konusunu irdeleyen az sayıda yazılım geliştirme modeli olmakla beraber, bu modeller çoğunlukla güvenlik uygulamalarının geleneksel süreç modeline eklenmesiyle oluşturulmuştur. Ancak güvenlik ile çevikliğin bazı noktalarda çakışmasından dolayı çevik yöntemlere hitap eden çalışma sayısı oldukça azdır. ˙Ilgili çalışmalarda, gelenek-sel yaklaşımla güvenlik gereksinimlerini belirleyip güvenli tasarımın yapılması ilkesinin, projenin baştan sona kapsamlı tasarımını gerektirdiği için ve güvenlik uygulamalarının her iterasyonda yapılması maliyetli olduğu için çevik yöntemlerin iterasyonlu ve artırımsal yapısıyla çeliştiği üzerine ortak görüş mevcuttur. Bu çalışmadaki temel amaç, yazılım güvenliğini sağlamaya yönelik uygulamaları anlamak ve bu uygulamaların Scrum çerçevesi içerisinde kullanımı için bir yöntem sunmaktır.

II. YÖNTEM

Scrum ile Güvenli Yazılım Geliştirme çalışması için uygulanan yöntem 4 adımdan oluşur.

İlk adımda, yazılım güvenliği tanımı ve kapsamını anlamak amaçlanmıştır. Yazılımda güvenliği hedefleyen uygulamalar üzerine literatür araştırması yapılmıştır..

İkinci adımda, yazılımlar için referans model niteliği taşıyan olgunluk modelleri ve güvenli yazılım geliştirme modelleri güvenlik bakış açısıyla incelenmiştir. Bu amaçla, yazılım için olgunluk modellerinden CMMI, FAA-ICMM, SSE-CMM ve SAMM modeli incelenmiştir. Bu modellerden sadece SSE-CMM ve SAMM modellerinin doğrudan güvenliği vurgu-layan uygulamalar seti sunduğu görülmüştür. Güvenli yazılım geliştirme modellerinden Microsoft SDL, TSP ve Correct-ness by Construction incelenmiştir. Bu modellerle geliştirilen projelerin başarılı sonuçlar verdiği görülmüştür. Bunu yapmaktaki amaç standartların yazılım güvenliğine bakışını, sunduğu uygulamaları analiz etmek ve güvenli yazılım geliştirme modellerinin süreç içerisinde bu uygulamaları nasıl faaliyete geçirdiğini anlamaktır.

Üçüncü adımda, Scrum çerçevesi ve bu çerçevenin benimsediği çevik yaklaşım üzerine araştırma yapılmıştır. Güvenlik uygulamalarının çevik yaklaşımın ilkeleriyle uymayan kısımları analiz edilmiştir.

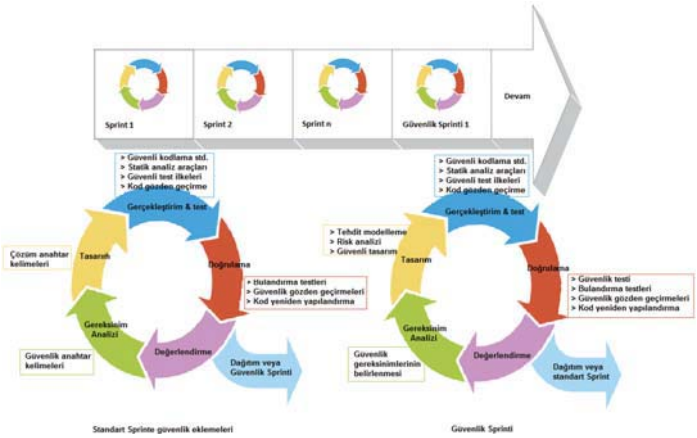
Son adımda Scrumın temelindeki ilkeler ve yazılım güvenliği ilkeleri korunarak, güvenliği hedefleyen uygulamaların, Scrum çerçevesine entegrasyonuna çalışılmıştır..

III. ÖNERİLEN GÜVENLİK EKLENTİİLİ SCRUM MODEL˙I: TRUSTWORTHY SCRUM

Bu çalışma kapsamında önerilen model, 3 temel bölüme ayrılmaktadır. Bunlar Scrum Sprintlerine her zaman eklenen uygulamalar, Güvenlik Sprinti etkinliği ve Sprintlere ihtiyaca göre eklenen uygulamalardır.

Her iterasyonda yapılması oldukça maliyetli ve zaman alıcı olduğu için Scrumın iteratif yapısına uymayan uygulamalar için bu çalışmada oluşturulan Güvenlik Sprinti etkinliğinin, her iterasyonda değil birkaç sayıdaki iterasyonlar arasında (dönüm noktaları) yapılması önerilmektedir.

Bu çalışma kapsamında önerilen modelin (Trustworthy Scrum) genel resmi şekil 1'de verilmiştir.



¸Sekil 1: Trustworthy Scrum

A. Her Sprint için Eklenen Güvenlik Uygulamaları: Güvenlik ve Çözüm Anahtar Kelimesi Yöntemi

İçsel tehditleri önlemeye yönelik olan bu uygulamaların Scrumın varolan standart Sprintlerine eklenmesi önerilmektedir. Bu iterasyonlarda güvenlik gereksinimleri için kötüye kullanım senaryoları yerine Güvenlik Anahtar Kelimeleri kullanılmaktadır. Güvenlik Anahtar Kelimeleri Sprint İş¸ Listesindeki bir kalemi geliştirirken güvenlik açısından dikkat edilmesi

ve uyulması gereken hususları belirtmektedir. Benzer şek-ilde kalemin gerçekleştirmesini yapan geliştirmeci bu anahtar kelimeleri odak noktasında tutarak bulduğu çözümü Çözüm Anahtar Kelimeleri olarak girmektedir. Bu bölüm Scrumın Geliştirme Takımı için uygulamalar sunmaktadır..

1) Güvenlik Anahtar Kelimeleri: Ürün İş Listesindeki kalemler için, olası gerçekleştirim hatalarını ve güvenlikle ilgili dikkat edilmesi gereken hususları ifade eden Güvenlik Anahtar Kelimeleri girilir. Bu kelimeler, tasarım, kodlama, test ve gözden geçirme aşamalarında dikkat edilecek güvenlik ilkelerini belirler. Örneğin; "Bellek taşması".

2) Çözüm Anahtar Kelimeleri: Güvenlik Anahtar Kelimesi ile ifade edilen açığın gerçekleşmemesi için bulunan yöntem veya alınacak önlemler Çözüm Anahtar Kelimeleri olarak girilir. Örneğin; "Bellek taşması" Güvenlik Anahtar Kelimesine sahip olan bir kalem için Çözüm Anahtar Kelimesi olarak "tüm dizin sınırlarının kontrolü" ve "tüm allocation (ayırma) işlemlerini ayırmayı kaldırma (deallocation) işlemleriyle eşleştirme" gibi güvenlik ilkelerini tarif eden çözümler girilir.

3) Güvenli Kodlama ve Test ˙Ilkeleri: Güvenli kodlama standartları ve güvenli tasarım ilkeleriyle ilgili kaynaklarda Güvenlik ve/veya Çözüm Anahtar Kelimelerine değinen ilkelere göre geliştirme yapılır ve testleri yazılır. Statik analiz araçları ile kaynak kod taranarak hatalar giderilir. Kod gözden geçirmeler, Güvenlik ve Çözüm Anahtar Kelimeleri odak noktasında tutularak yapılır. Doğrulama aşamasında bulandırma testleri ile yazılım test edilir.

4) Bitti Tanımı: Kalemi "Bitti" konumuna getirebilmek için, tüm aşamalarda Güvenlik ve/veya Çözüm Anahtar Kelimelerinin gerekliliklerinin yapıldığı onaylanmalı, uygulanabilir olmayanlar "Uygulanabilir Değil" şeklinde işaretlenmelidir. Bunun dışında zaman yetersizliği vb. nedenlerle geliştirmesi yapılmayan kalem "Ertelendi" işaretiyle tekrar Ürün ˙Iş Listesine tarihçesiyle birlikte gönderilir. Bu şekilde güvenlik gereksinimleri tamamlanmayan kalemlerin kapanmasına izin verilmez, sonraki Sprintlerde kaldığı aşamadan itibaren geliştirimine devam edilmesi sağlanır.

B. Güvenlik Sprinti

Güvenlik Sprinti, diğer standart Sprint gibi birbirini biter bitmez takip etmez. Projenin başlangıç aşamasına ve gerek duyuldukça belli sayıdaki iterasyonlar arasına Güvenlik Sprinti konabilir. şekil 2'de Güvenlik Sprinti akış diyagramı verilmiştir.

Güvenlik Sprinti, Scrumın standart Sprintinde yapılan Scrum Etkinliklerinin güvenlik bakış açısıyla yapılmasını ve Scrum Takımına güvenlik uzmanı dâhil edilmesini önerir. Bu bölüm tüm Scrum Takımına uygulamalar sunar. Güven-lik uzmanı, Scrum Master ve Ürün Sahibine ürünün güven-lik gereksinimlerini belirlemede yardımcı olur ve Geliştirme Takımına güvenlik geliştirmesi konusunda destek sağlar. Güvenlik Sprinti, standart Sprint gibi düzenli olarak biribirini takip etmediğinden ne zaman yapılacağına Scrum Master tarafından Sprint Değerlendirme sonuçlarına göre karar verilir.

1) Güvenlik Gereksinimlerinin Belirlenmesi: Burada riske dayalı güvenlik gereksinimleri odak noktasıdır. Dış tehditler ve saldırgan bakış açısıyla bakılarak olası tehditleri ve gereksinimleri belirlemeyi gerektirir. Aktör, tehdit, güvenlik gereksinimi, risk derecesi gibi gereksinimi belirleme ve risk analizine yardımcı olabilecek ayrıntılar kaleme eklenir ve Ürün İş Listesine atılır. Bu tanımlar gereksinimleri belirleme aşamasında beyin fırtınası yapmayı kolaylaştırır. Diğer gereksinimlerden ayırt etmek için kalemin sınıfı "Güvenlik Gereksinimi" yapılabilir veya etiketle işaretlenebilir. Gösterim olarak kötüye kullanım senaryolarından da yararlanılabilir.

2) Risk Analizi: Risk analizi planlı, isteğe bağlı veya olaya bağlı olarak gerçekleştirilebilir ve literatürde bunun için farklı yaklaşımlar mevcuttur. Önerilen modelde risk analizi temel olarak aşağıdaki adımlardan oluşur:

1) Risk analizi için gerekli kavramlar belirlenir ve talep yönetim sisteminde ilgili kaleme girdi sağlanır. Temel olarak tehdit, sistem açığı, tehditin olma olasılığı belirlenir.
2) Tehlikenin gerçekleşmesi durumunda ne tür etkileri neden olacağına dair analiz yapılır ve gereksinime riskin derecesine göre "Rank" değeri atanabilir.
3) Hafifletme stratejisi belirlenir.
4) Kaybın maliyeti ve önleyici aktivitenin maliyeti arasında kıyas yapılarak yani risk analizi yapılarak;
    a) Geliştirmenin bu Sprintte yapılacağına karar verilirse bu Güvenlik Sprintinin iş listesine atılır veya
    b) Riskin derecesi yüksek görünmüyorsa daha sonra yapılmak üzere veya yeni verilerle yeniden risk analizi yapılmak üzere Ürün İş Listesine gönderilebilir.
5) Yeniden yapılan her risk analizinde çıkan sonuca göre gereksinimin rankı artırılır/azaltılır veya riskin derecesi güncellenir.

3) Güvenli Tasarım: Değerler, tehditler ve olası sistem açıkları belirlendikten sonra yazılımın atağa açık alanlarını en aza indirgeyecek şekilde tasarım yapılır. Howard ve Lipner'e göre atak alanı kod parçası, arayüz, servis, protokol ve özellikle yetkisiz kullanıcılar olmak üzere tüm kullanıcılara açık olan uygulamalar olarak tanımlanmıştır [25]. Güvenli tasarım için yazılımda güvenliği sağlama adına gerçek deneyimlerden yola çıkılarak oluşturulan pratikleri sunan ilkeler, kılavuzlar ve kodlama kuralları gibi kaynaklardan ve güvenlik uzmanından yararlanılabilir. Ayrıca, talep yönetim sistemine girilmiş ve kapanmış olan daha önceki benzer güvenlik gereksinimlerinden yararlanılabilir.

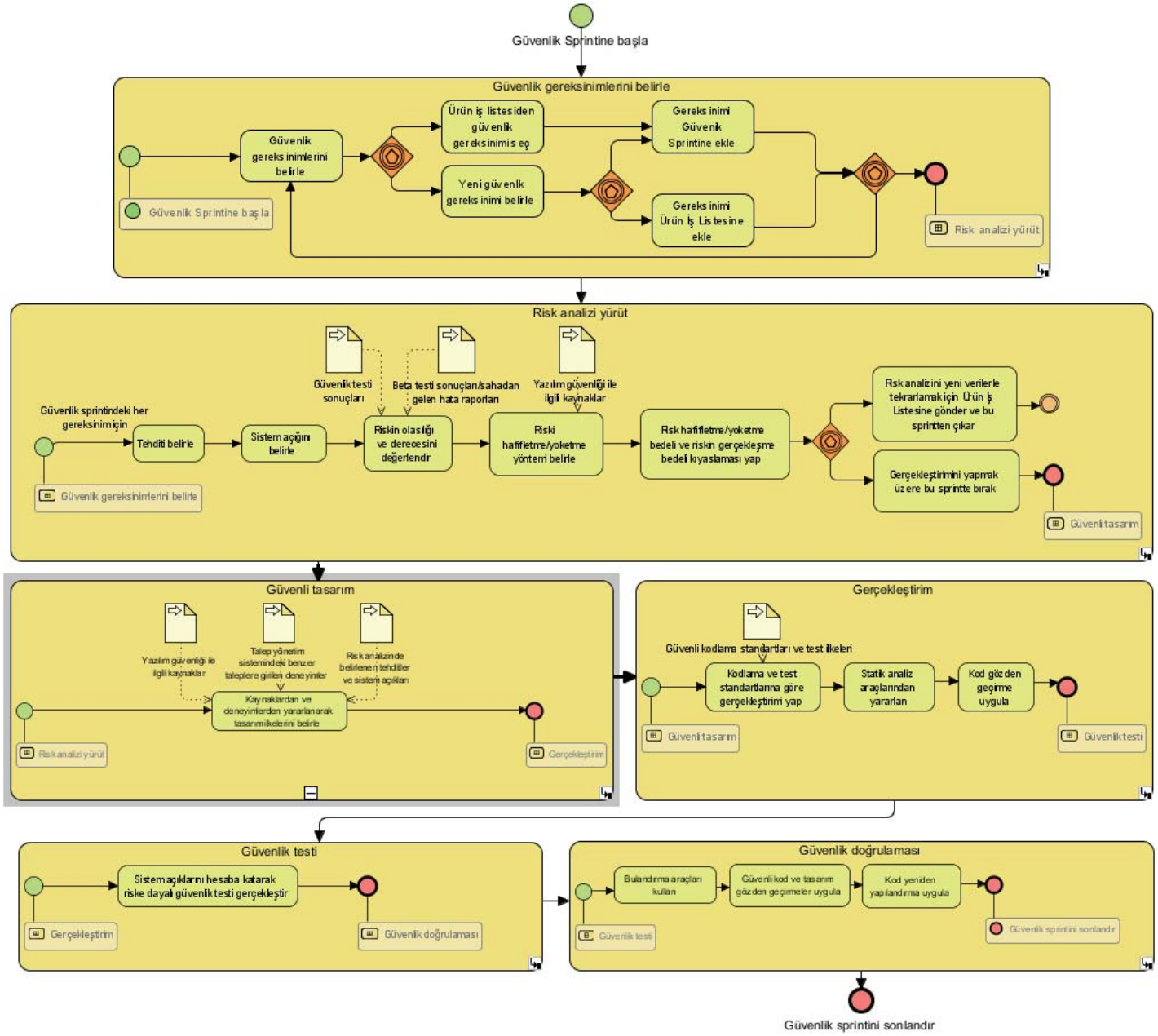
4) Güvenli Gerçekleştirim: Güvenli kodlama ve test ilkelerine göre gerçekleştirim yapılır. Statik analiz araçları ile kod taranır. Kod gözden geçirmelerle, araçların bulamadığı hatalar bulunur.

5) Güvenlik Testi: Güvenlik testi şu iki yaklaşımı içermelidir [18]: 1) Güvenlik mekanizması fonksiyonlarının doğru gerçekleştirildiğinin testi, 2) Saldırganın bakış açısıyla bakıp sistem açıklarını göz önünde bulundurarak riske dayalı güvenlik testi.

6) Güvenlik Doğrulaması: Bulandırma araçları ile yazılıma geçersiz girdiler verilip test edilir. Kod ve tasarım güvenlik gözden geçirmeleri gerçekleştirilir.

C. Kod Yeniden Yapılandırma

Güvenlik gereksinimi olarak, eski kodlar gözden geçirilir, güvenlik hataları aranır, eğer bulunursa düzeltilir. Kod



¸Sekil 2: Güvenlik Sprinti akı¸s diyagramı

yeniden yapılandırma işlemi, Scrum Master veya Geliştirme Takımı tarafından Güvenlik Sprintine veya standart Sprintlere gerek duyuldukça kalem olarak eklenebilir. Sahaya sürülmeden önce kodun güvenlik açısından gözden geçirilip yeniden yapılandırılması güvenliği sağlama konusunda oldukça fayda sağlar.

D. Kod ve Tasarım Güvenlik Gözden Geçirmeleri

Ürünün sahaya sürülmeye hazır olduğunun doğrulanması açısından, kodun ve tasarımın güvenlik açısından gözden geçirilmesi gerekir. Gerektiğinde güvenlik uzmanı dâhil edilebilir ve kod yeniden yapılandırma ile birlikte yürütülebilir. Sprint Değerlendirmesi sonucu gerek duyuldukça Scrum Master veya Geliştirme Takımı tarafından Sprint Planlamalarında Güvenlik Sprintine veya standart Sprintlere kalem olarak eklenebilir.

E. Güvenlik Eğitimi

Belli bir plana göre geliştirme takımına uzman tarafından güvenlik eğitimi ve/veya yazılım güvenliğiyle ilgili kitap, cd vb. materyal sağlanır. Scrum Retrospektifinde yapılan değerlendirme ve gözlemlerden yararlanılarak, Scrum Takımının ihtiyaç duyduğu konulara göre Scrum Master tarafından planlaması yapılabilir.

IV. SONUÇ

Yazılımlarda güvenlik açıklarının çok yaygın olmasına rağmen, yazılım geliştirme modelleri güvenliği sağlamak için gerekli olan uygulamaları içermemektedir. Güvenlik konusunu irdeleyen az sayıda yazılım geliştirme modeli olmakla beraber, bu modeller çoğunlukla güvenlik uygulamalamarının geleneksel süreç modeline eklenmesiyle oluşturulmuştur. Bu çalışmada çevik yazılım ilkeleri ve yazılım güvenliği ilkeleri göz önünde bulundurularak, güvenlik uygulamalarının Scrum çerçevesinde uygulanmasına dair bir yöntem önerilmiştir. Bu şekilde günümüz yazılımlarında oldukça ihtiyaç duyulan güvenlik gereksiniminin Scrum ile nasıl sağlanacağına dair bir çalışma olarak literatüre katkı sağlamıştır. Önerilen model, çevik yöntemlerin hızlı, değişime ayak uyduran ve aşamalı ilerleyen yapısından yararlanırken aynı zamanda güvenlik uygulamalarının da uygulanabilirliğini sağlamayı amaçlamaktadır. Bu çalışmanın temel iki katkısı:

- Dikkatsizlik vb. nedenlerle oluşan kodlama hatalarının da dahil olduğu içsel tehditlerin oluşmasını aza indirecek "Güvenlik ve Çözüm Anahtar Kelimeleri Yöntemi" ve
- Atağa açık ortamlardaki yazılımlarda, dış tehditlerden gelebilecek saldırıları veya etkisini önlemeye yönelik "Güvenlik Sprinti"dir.

Bu çalışmada önerilen Trustworthy Scrum'ın, güçlü ve zayıf yönlerinin analizi sonucu çıkarılan özellikleri aşağıda verilmiştir:

- Hem gerçekleştirim ve tasarım hataları gibi iç tehditlere karşı hem de internet ortamı gibi saldırıya açık alanlarda dış tehditlerden gelebilecek saldırılara karşı güvenlik önlemleri sunmaktadır.
- Sadece standart Sprintlere eklenen iç tehditleri önlemeye yönelik uygulamaların yapılması ile dahi saldırganların suistimal edebileceği sistem açıklarının azalmasına yardımcı olmaktadır.
- Güvenlik Sprinti etkinliği, her iterasyonda yapılması oldukça maliyetli ve zaman alıcı olan güvenlik uygulamalarının dönüm noktalarında yapılmasını sağlayarak maliyeti azaltacaktır.
- Çevik ilkelere uygun olarak minimal dökümantasyon için kısmen çözüm önerisi sunulmuştur ancak Güvenlik Sprintinde güvenlik ilkelerinden olan kapsamlı dökümantasyon gerekliliğine tam anlamıyla çözüm getirilememiştir.

Gelecek çalışma olarak, önerilen model, geliştirilen yazılımın kritiklik seviyesine göre içermesi gereken uygulamalar ve dereceleri bakımından seviyelere ayrılabilir veya belli bir sektör hedeflenerek daha ayrıntılı ve sektöre özgü uygulamalar sunulabilir.

KAYNAKÇA

[1] Standish Group, The Chaos Report, West Yarmouth, MA: The Standish Group, 1995.

[2] Beznosov, K., & Kruchten, P. (2004, September). Towards agile security assurance. In Proceedings of the 2004 workshop on New security paradigms (pp. 47-54). ACM.

[3] The Standish Group Report, CHAOS Report 2015, http://www.standishgroup.com/ (November, 2016).

[4] McGraw, G. (2004). Software security. IEEE Security & Privacy, 2(2), 80-83.

[5] J. Viega and G. McGraw, Building Secure Software, Addison-Wesley, 2001; www.buildingsecuresoftware.com.

[6] G. McGraw, "From the Ground Up: The DIMACS Software Security Workshop," IEEE Security & Privacy, vol. 1, no. 2, 2003, pp. 59–66.

[7] Microsoft MSDN, The Trustworthy Computing Security Development Lifecycle, https://msdn.microsoft.com/en-us/library/ms995349.aspx (November, 2016).

[8] Howard, M., & Lipner, S. The Security Development Lifecycle: A Process for Developing Demonstrably More Secure Software, Microsoft Press, 2006.

[9] Microsoft MSDN, https://msdn.microsoft.com/en-us/library/windows/desktop/ee790617.aspx. (November, 2016).

[10] Humphrey, W. S. (2000). The Team Software Process (TSP). Carnegie Mellon University, Software Engineering Institute.

[11] Bartsch, S. (2011, August). Practitioners' perspectives on security in agile development. In Availability, Reliability and Security (ARES), 2011 Sixth International Conference on (pp. 479-484). IEEE.

[12] Securosis, Secure Agile Development, https://securosis.com/assets/library/reports/SecureAgileDevelopment_Nov2014_FINAL.pdf (November, 2016).

[13] Zurko, M. E., & Simon, R. T. (1996, September). User-centered security. In Proceedings of the 1996 workshop on New security paradigms (pp. 27-33). ACM.

[14] Wäyrynen, J., Bodén, M., & Boström, G. (2004, August). Security engineering and eXtreme programming: An impossible marriage?. In Conference on Extreme Programming and Agile Methods (pp. 117-128). Springer Berlin Heidelberg.

[15] Boström, G., Wäyrynen, J., Bodén, M., Beznosov, K., & Kruchten, P. (2006, May). Extending XP practices to support security requirements engineering. In Proceedings of the 2006 international workshop on Software engineering for secure systems (pp. 11-18). ACM.

[16] Mougouei, D., Sani, N. F. M., & Almasi, M. M. (2013). S-scrum: a secure methodology for agile development of web services. World of Computer Science and Information Technology Journal, 3(1), 15-19.

[17] Pohl, C., & Hof, H. J. (2015). Secure Scrum: Development of Secure Software with Scrum. arXiv preprint arXiv:1507.02992.

[18] Potter, B., & McGraw, G. (2004). Software security testing. IEEE Security & Privacy, 2(5), 81-85.

[19] G. Hoglund and G. McGraw, Exploiting Software: How to Break Code, Addison-Wesley, 2004.

[20] US-CERT Build Security In, Architectural Risk Analysis, https://www.us-cert.gov/bsi/articles/best-practices/architectural-risk-analysis/architectural-risk-analysis (December, 2016).

[21] Azham, Z., Ghani, I., & Ithnin, N. (2011, December). Security backlog in Scrum security practices. In Software Engineering (MySEC), 2011 5th Malaysian Conference in (pp. 414-417). IEEE.

[22] Schwaber, K., & Sutherland, J., The Scrum Guide, The Definitive Guide to Scrum: The Rules of the Game, 2016 Scrum.Org and ScrumInc. http://www.scrumguides.org/

[23] Agile Alliance. Manifesto for Agile Software Development, http://agilemanifesto.org/ (2016)

[24] Breu, R., Burger, K., Hafner, M., Jürjens, J., Popp, G., Wimmel, G., & Lotz, V. (2003, December). Key issues of a formally based process model for security engineering. In International Conference on Software and Systems Engineering and their Applications.

[25] Howard, M. (2004). Attack surface: Mitigate security risks by minimizing the code you expose to untrusted users. MSDN Magazine (November 2004), https://msdn.microsoft.com/magazine/msdn-magazine-issues.

[26] Redwine, S. T., & Davis, N. (2004). Processes to Produce Secure Software: Towards More Secure Software, Volume II.