T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM GÜVENLİĞİ METRİKLERİ VE BİR YAZILIM GÜVENLİĞİ METRİK ÇERÇEVESİ TASARIMI

ALİ GÖKŞEN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
SİBER GÜVENLİK PROGRAMI

GEBZE
2022

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM GÜVENLİĞİ METRİKLERİ VE BİR YAZILIM GÜVENLİĞİ METRİK ÇERÇEVESİ TASARIMI

ALİ GÖKŞEN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
SİBER GÜVENLİK PROGRAMI

DANIŞMANI
PROF. DR. İBRAHİM SOĞUKPINAR

GEBZE
2022

T.R.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

SOFTWARE SECURITY METRICS AND DESIGN OF SOFTWARE SECURITY METRIC FRAMEWORK

ALİ GÖKŞEN

A THESIS SUBMITTED FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER ENGINEERING

CYBER SECURITY PROGRAM

THESIS SUPERVISOR

PROF. DR. İBRAHİM SOĞUKPINAR

GEBZE

2022

GEBZE TEKNİK ÜNİVERSİTESİ	**YÜKSEK LİSANS JÜRİ ONAY FORMU**

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 07/02/2022 tarih ve 2022/08 sayılı kararıyla oluşturulan jüri tarafından 01/06/2022 tarihinde tez savunma sınavı yapılan Ali GÖKŞEN'in tez çalışması Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE (TEZ DANIŞMANI)	: Prof. Dr. İbrahim SOĞUKPINAR
ÜYE	: Prof. Dr. Ali Gökhan YAVUZ
ÜYE	: Doç. Dr. Mehmet GÖKTÜRK

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun

…..…/…..…/..…… tarih ve ……/…….. sayılı kararı.

İMZA/MÜHÜR

ÖZET

Bu çalışmada yazılım güvenliği metrikleri incelenmiş, incelenen metriklerin yazılım güvenliğini ölçme kriterleri ortaya konulmuştur. Büyük bir projede yazılım güvenliğini ölçmek için yazılım güvenliği metrikleri hesaplanmış ve sonuçları paylaşılmıştır. Gereksinim, tasarım, kodlama, uygulama, test ve bakım aşamalarında kullanılan güvenlik metrikleri ortaya konmuş bu metrik değerleri büyük bir proje içerisinde incelenmiş ve değerler paylaşılmıştır. Metriklerin kullanımı, uygunluğu ve seçimi için bir çerçeve oluşturulmuş ve bu çerçeve ile metrik seçimine yardımcı olunmuştur. Bu çerçeve kullanılan projelerde güvenlik açıkları azalır ve bu metriklerle ortaya konur. Bu çalışmada oluşturulan çerçeve ile güvenlik açıklarının azaldığı tespit edilmiş ve sonuçları paylaşılmıştır.

Anahtar Kelimeler: Yazılım Güvenliği, Güvenlik Metrikleri, Yazılım Yaşam Döngüsü, Metrik Çerçevesi.

SUMMARY

In this work, software security metrics were examined, and the criteria for measuring software security of the analyzed metrics were revealed. In order to measure software security in a large project, software security metrics were calculated and the results were shared. The security metrics used in the requirements, design, coding, implementation, testing and maintenance stages were revealed, and these metric values were analyzed within a large project and the values were shared. A framework has been created for the use, suitability and selection of metrics, and with this framework, metric selection has been assisted. Security vulnerabilities are reduced in project using this framework are revealed by these metrics. With the framework created in this study, it was determined that the security vulnerabilities were reduced and the results were shared.

Key Words: Software Security, Security Metrics, Software Life Cycle, Metric Framework.

TEŞEKKÜR

Başta, yüksek lisans eğitimimde ve akademik hayatımda ilgisini, bilgisini ve yardımlarını hiçbir zaman esirgemeyip her zaman yön veren Prof. Dr. İbrahim SOĞUKPINAR'a, bütün çalışmam boyunca yanımda olan, bilgi ve tecrübelerini benimle paylaşan, her zaman yol göstericim olan değerli Öğr. Gör. O. Erdal DEDELEROĞLU'na ve göstermiş olduğu tüm desteklerinden dolayı sevgili eşim E. Melike GÖKŞEN'e en içten teşekkürlerimi sunarım.

İÇİNDEKİLER

	Sayfa
ÖZET	v
SUMMARY	vi
TEŞEKKÜR	vii
İÇİNDEKİLER	viii
SİMGELER ve KISALTMALAR DİZİNİ	x
ŞEKİLLER DİZİNİ	xi
TABLOLAR DİZİNİ	xii
1. GİRİŞ	1
1.1. Tezin Amacı, Katkısı ve İçeriği	1
1.1.1. İlgili Çalışmalar	3
2. GÜVENLİK GEREKSİNİM ANALİZİ VE GÜVENLİK METRİKLERİ	5
2.1. Gereksinim Aşamasındaki Güvenlik Metrikleri	6
2.2. Tasarım Aşamasındaki Güvenlik Metrikleri	6
2.3. Kodlama ve Entegrasyon Aşamasındaki Güvenlik Metrikleri	7
2.4. Test Aşamasındaki Güvenlik Metrikleri	8
2.5. Bakım Aşamasındaki Güvenlik Metrikleri	10
3. GÜVENLİK METRİKLERİNİN YAZILIM YAŞAM DÖNGÜSÜ AŞAMALARINA GÖRE KARŞILAŞTIRILMASI	12
3.1. Güvenlik Metriklerinin Kullanımında Yaşanan Zorluklar	15
3.2. Kodlama Test ve Bakım Aşamasında Paydaşlarla Yapılan Çalışmalar	16
3.3. Kodlama ve Entegrasyon Aşamasında Yapılması Gerekenler	20
3.3.1. Yazılım Geliştiricilere Kodlama ve Entegrasyon Metrikleri İle İlgili Eğitim Verilmesi	20
3.3.2. Test Sorumlularına Kodlama ve Entegrasyon Metrikleri İle İlgili Eğitim Verilmesi	20
3.3.3. BT Güvenlik Sorumlularına Kodlama ve Entegrasyon Metrikleri İle İlgili Eğitim Verilmesi	21

3.3.4. Verilerin Konsolide Edilmesi Metrikleri 21
3.3.5. Metriklerin Karşılaştırılması 21
3.4. Test Aşamasında Yapılması Gerekenler 22
3.4.1. Yazılım Geliştiricilere Test Metrikleri İle İlgili Eğitim Verilmesi 22
3.4.2. Test Sorumlularına Test Metrikleri Metrikleri İle İlgili Eğitim Verilmesi 22
3.4.3. BT Güvenlik Sorumlularına Test Metrikleri İle İlgili Eğitim Verilmesi 23
3.4.4. Verilerin Konsolide Edilmesi Metrikleri 23
3.4.5. Metriklerin Karşılaştırılması 23
3.5. Bakım Aşamasında Yapılması Gerekenler 24
3.5.1. Yazılım Geliştiricilere Bakım Metrikleri İle İlgili Eğitim Verilmesi 24
3.5.2 BT Güvenlik Sorumlularına Bakım Metrikleri İle İlgili Eğitim Verilmesi 24
3.5.3. Verilerin Konsolide Edilmesi Metrikleri 25
3.5.4. Metriklerin Karşılaştırılması 25
4. TEST VE BAKIM AŞAMASINDA YAPILAN ÖLÇÜMLER VE SONUÇLARI 26
4.1. Test Aşamasında Yapılan Ölçümler ve Sonuçları 33
4.2. Bakım Aşamasında Yapılan Ölçümler ve Sonuçları 37
5. SONUÇ 40

KAYNAKLAR 41
ÖZGEÇMİŞ 43

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler ve Kısaltmalar		**Açıklamalar**
Ccp	:	Eşleşen bozulma yayılma
Cer	:	Kritik eleman oranı
Ndd	:	Güvenlik tasarım karar sayısı
Nerr	:	Sistem içerisinde uygulama hata sayısı
Nex	:	Sistem güvenliğini gidermek için kullanılan istisna sayısı
Nlsr	:	En az öncelikli güvenlik gereksinimlerinin sayısı
Noex	:	Atlanan istisna sayısı
Nosr	:	İhmal edilen güvenlik gereksinimleri sayısı
Npsr	:	Öncelikli güvenlik gereksinimleri sayısı
Nsa	:	Güvenlik algoritması sayısı
Nsdf	:	Güvenlik tasarım kusurları sayısı
Nserr	:	Güvenlikle ilgili uygulama hata sayısı
Nsr	:	Güvenlik gereksinimlerinin sayısı
Ntcp	:	Başarısız sistem güvenlik testi sayısı
Rdd	:	Güvenlik tasarım kararları oranı
Rdf	:	Güvenlik tasarım kusurlarının oranı
Roex	:	İhmal edilen istisna sayısı
Rosr	:	Öncelikli güvenlik gereksinimleri oranı
Rp	:	Güvenlik açıkları için yayınlanan yamaların oranı
Rsc	:	Güvenlik değerlendirmesine bağlı yazılım değişikliklerinin oranı
Rserr	:	Güvenliği etkileyen uygulama hatalarının oranı
Rsr	:	Güvenlik gereksinimlerinin oranı
Rtc	:	Güvenlik test senaryoları oranı
Sr	:	Durma oranı
Ttc	:	Sistemin toplam güvenlik testi sayısı

ŞEKİLLER DİZİNİ

Şekil No:		Sayfa
1.1:	Dr. Raees Khan güvenli kod yazılım modeli.	2
2.1:	Yazılım süreçlerinde yazılım açığı giderme maliyeti.	5
2.2:	Güvenli bakım için gerekli adımlar.	11
4.1:	Bulgu sebebine göre dağılım.	32
4.2:	Yazılım güvenlik açığı örneği.	32
4.3:	Test aşamasındaki ilk metrik sonuçları.	33
4.4:	Birinci metrik ölçümlerinde test aşamasında yapılan güvenlik testlerinin dağılımı.	34
4.5:	Test aşamasındaki ikinci metrik sonuçları.	35
4.6:	İkinci metrik ölçümlerinde test aşamasında yapılan güvenlik testlerinin dağılımı.	36
4.7:	Bakım aşamasındaki ilk metrik sonuçları.	37
4.8:	Bakım aşamasındaki ikinci metrik sonuçları.	38

TABLOLAR DİZİNİ

Tablo No:		Sayfa
2.1:	Güvenli test için gerekli her adımdaki kuralcı faliyetler.	9
3.1:	Güvenli yazılım geliştirme aşamaları için güvenlik aktiviteleri.	12
3.2:	Yazılım aşamasında güvenlik metrikleri.	13
3.3:	Yazılım geliştiriciler ile yapılan anket sonuçları -1.	16
3.4:	Yazılım geliştiriciler ile yapılan anket sonuçları -2.	17
3.5:	Test sorumluları ile yapılan anket sonuçları.	18
3.6:	BT güvenlik uzmaları ile yapılan anket sonuçları.	19
4.1:	Güvenlik test sonuçları risk seviyelendirilmesi.	26
4.2:	Bulguların önem derecesi ve kategorisi.	28
4.3:	Bulguların önem derecesi ve kategorisi.	29
4.4:	Bulguların önem derecesi ve kategorisi.	30
4.5:	Bulguların önem derecesi ve kategorisi.	31

1. GİRİŞ

Sistem kullanıcıları sistemin ve uygulamanın her zaman tamamen güvenli olmasını isterler. Bu her zaman mümkün değildir. Güvenli olmayan sistemlere neden olan temel nedenler ise tasarım ve dizayn aşamalarında yapılan güvenlik açıkları, sistemle kullanıcılar arasındaki etkileşim karmaşıklığı ve bu iletişimde yaşanan sıkıntılar, saldırganların birçok ücretsiz araçlar dahil saldırı araçlarına kolayca ulaşabilmedir. İnsan, doğası gereği hataya açıktır, sistem açıklarını azaltmak ancak tecrübe ve mesleki yeterlilikle giderilebilir. Güvenlik açıklarının yazılım kalitesi üzerindeki etkisi, yazılım geliştirmenin karmaşıklığı, dağıtılmış ve eş zamanlı olmasıyla doğru orantılıdır[1]. Güvensiz yazılım geliştirmek yazılımı geliştiren kişi ya da kurumlara, yazılımı kullanan kullanıcılara, uygulamanın sahibi olanlara zarar verir. Bu zarar hem maddi hem de itibar kaybı olarak görülebilir. Aksine, güvenli yazılım bir işletmeyi çökmelerle ilgili sorunlardan koruyabilir ya da işletmenin daha çevik ve aktif olmasını sağlayabilir[2]. Yazılım güvenlik açıkları, diğer yazılım kusurlarına benzemez, farklı olarak ele alınması gerekir[3]. Araştırmalara göre diğer her şey eşit olduğunda kod satırı sayısı ile ilişki olduğunu gösteriyor. Bu sebeple mümkünse yazılım mümkün olan en basit şekilde tasarlanmalıdır[4].

1.1. Tezin Amacı, Katkısı ve İçeriği

Yazılım güvenliği metrikleri bazı çalışmalarla belirlenmiş ve hangi süreçlerde hangi metriklerin kullanılacağı ortaya konmuştur.

Bu metrikler gereksinim aşamasında güvenlik gereksinimlerinin sayısı(Nsr), güvenlik gereksinimlerinin oranı(Rsr), öncelikli güvenlik gereksinimleri sayısı(Npsr), en az öncelikli güvenlik gereksinimlerinin sayısı(Nlsr), ihmal edilen güvenlik gereksinimleri sayısı(Nosr), öncelikli güvenlik gereksinimleri oranı(Rosr)'dır.

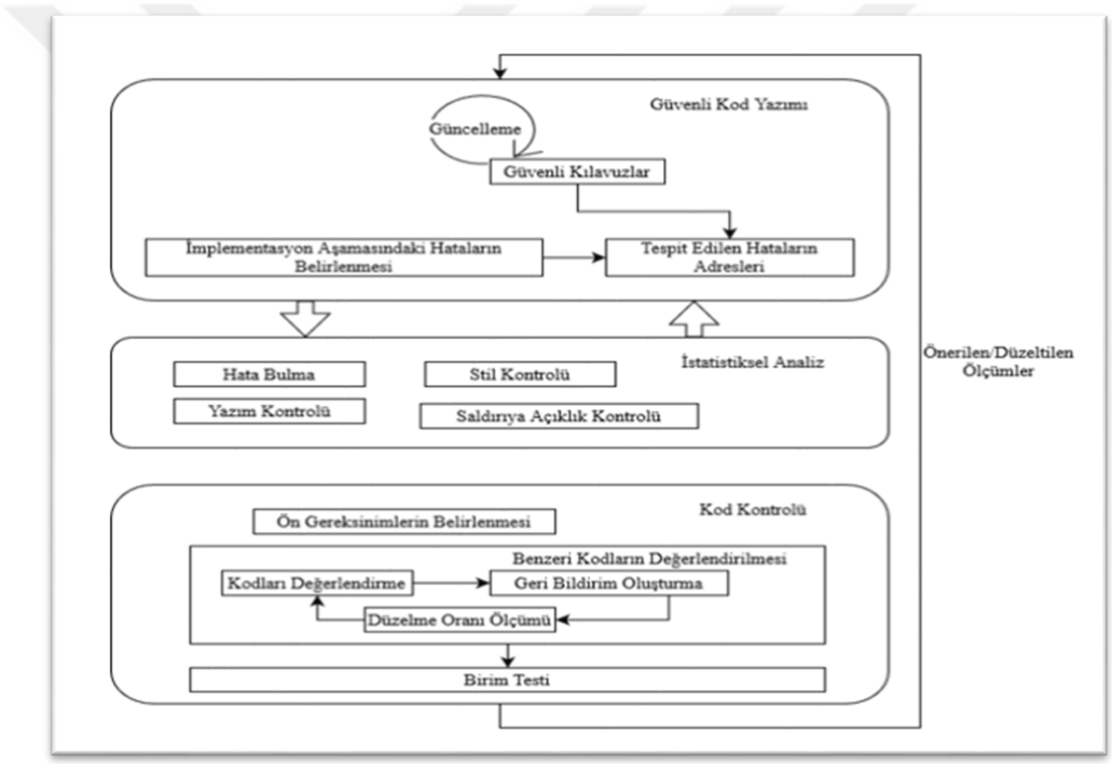
Tasarım aşamasındaki güvenlik metrikleri : Güvenlik tasarım karar sayısı(Ndd), güvenlik tasarım kararları oranı(Rdd), güvenlik algoritması sayısı(Nsa), güvenlik tasarım kusurları sayısı(Nsdf), güvenlik tasarım kusurlarının ortanı(Rdf).

Tasarım aşamasındaki güvenlik metrikleri : Güvenlik tasarım karar sayısı(Ndd), güvenlik tasarım kararları oranı(Rdd), güvenlik algoritması sayısı(Nsa), güvenlik tasarım kusurları sayısı(Nsdf), güvenlik tasarım kusurlarının ortanı(Rdf).

Uygulama aşamasındaki metrikler: Sistemde bulunan hata sayısı(Nerr), sistem güvenliğini gidermek için kullanılan istisna sayısı(Nex), atlanan istisna sayısı(Noex), ihmal edilen istisna sayısı(Roex)

Test aşamasındaki güvenlik metrikleri: Sistemin toplam güvenlik testi sayısı(Ttc), başarısız sistem güvenlik testi sayısı(Ntcp), güvenlik test senaryoları oranı(Rtc)

Bakım aşamasındaki metrikler: Güvenlik değerlendirmesine bağlı yazılım değişikliklerinin oranı(Rsc), güvenlik açıkları için yayınlanan yamaların oranı(Rp)



Şekil 1.1: Raees Khan güvenli kod yazılım modeli.

Şekil 1.1'de belirtilen güvenli kod yazılım modeli ilk olarak Dr. Raees Khan tarafından ortaya atılmıştır[5]. Bu modelin amacı yapılan hatalardan ders çıkararak bir daha tekrarlanmaması ve mümkün olan en fazla sayıda testin bir düzen içinde yapılmasını sağlamaktır. İstatiksel analizler ile elde edilen bulguların matematiksel modellemesinin yapılmasıdır. Böylece pek çok hata istatistiki bir şekilde tutulmakta ve güvenli kod yazımı için kodların sınıflandırılması akılcı bir bakış açısı ile ele alınmaktadır.

1.1.1. İlgili Çalışmalar

Raees Ahmad Khan 2011 yılında yazdığı Secure Software Development : A presicriptive Framework makalesinde güvenlik metriklerinin kullanımına dair bir çerçeve berlirlemiş ve güvenlik metriklerinin nasıl kullanılması gerektiğine dair önerilerde bulunmuştur. Yazılım geliştiricileri de güvenlikle ilgili sürece dahil etmiş ve her aşamada metrik ölçümleri yapılması gerektiğine dair önerilerde bulunmuştur. Bu makalede yalnızca öneriler yer almış, sonuçlar paylaşılmamıştır[5].

Smriti Jain ve Maya Ingle 2011 yılında yazdıkları makalede yazılım yaşam döngüsü içerisinde kullanılabilecek metrikleri listelemişler ve bu metriklerin matematiksel formülleriyle nasıl kullanılabileceğine dair önerilerde bulunmuşlardır. Herhangi bir sonuç paylaşılmayan bu makalede bir çerçeve oluşturulmuş ve öneriler listelenmiştir[20]. Smriti Jain ve Maya Ingle 2014 yılında yazdıkları Security Metrics and Software Development Progression adlı makalede ise güvenlik metrikleri yazılım yaşam döngüsü içerisinde listelenmiş, 3 ayrı kategoride 3 ayrı vaka çalışması yapılmış ve etkililik faktörleri listelenmiş ve paylaşılmıştır. Web tabanlı, masaüstü tabanlı ve sunucu tabanlı 3 ayrı vaka incelenmiş metrik sonuçları tek aşamalık paylaşılmış ve etkililik faktörü incelemesi yapılmıştır[21].

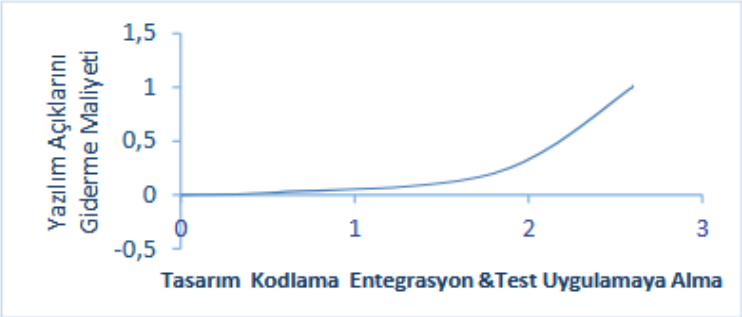
M. Sahinoglu, S. Stockton, S. Morton, P. Vasudev ve M. Eryilmaz 2014 yılında yazdıkları Metrics to Access and Manage Software Application Security Risk adlı makalelerinde olasılık ve oyun teorilerini de kullanarak nicel risk yönetimi için otomatik bir yazılım aracı oluşturmuştur. Bu uygulama ile riski, ölçülebilir riski ve maliyeti azaltmayı hedeflediler[22].

Bu tez kapsamında, öncelikle daha önce belirlenen güvenlik metrikleri listelenmiş bu metriklerin açıklamaları yapılmış, nasıl kullanılabileceği belirtilmiştir. Yazılım döngüsü içerisinde aşama aşama metrikler belirlenmiş, belirlenen metrikler büyük bir proje içerisinde incelenmiş ve veriler paylaşılmıştır. İncelenen proje büyüklüğü birçok sistem, kod parçası, sunucu, birçok IP ve dosya yolu içermektedir. Aynı proje içerisinde tüm metrikler incelenmiş ve veriler değerlendirilmiştir. İkinci bölümde güvenlik gereksinim analizi ve güvenlik metrikleri listelenmiş ve açıklanmış, üçüncü bölümde metrikler yazılım yaşam döngüsü içerisinde karşılaştırılmıştır. Son aşama olan dördüncü bölümde ise sonuç paylaşılmıştır. Bu çalışma kapsamında bir çerçeve oluşturulmuş ve bu çerçeve kullanıldığında güvenlik açıklarının azaldığını gösteren

sonuçlar paylaşılmıştır. Diğer çalışmalardan farklı olarak aynı aşamalardaki metrikler birden fazla ölçülmüş ve sonuçlar paylaşılmış, ölçümlerden elde edilen sonuçla sürekli iyileştirmeye dair önerilerde bulunulmuştur.

2. GÜVENLİK GEREKSİNİM ANALİZİ VE GÜVENLİK METRİKLERİ

Yazılımın bütünlüğünü, kalitesini ve işlevini korumak için güvenlik gereksinim analizi yapılmalıdır. Güvenlik ve gizlilik gereksinim analizi projenin başında yapılır ve minimum güvenlik gereksinimlerinin özelliklerini içerir. Güvenli yazılım geliştirmenin başlangıç aşaması gereksinim analizi yapmaktır[7]. Bu aşamada fayda/maliyet göz önünde bulundurularak bazı tespitler yapılmalı ve bunlar net olarak ortaya koyulmalıdır. İşlevsel ve işlevsel olmayan gereksinimler ayrılmalı, gereksinimler bazı araçlarla gösterilmeli ve gerekirse sınıflandırılmalı, ortak paydaşlarla çalışma yapılmalı, kullanım aşamaları ve detayları belirlenmelidir. Sınıflandırılan güvenlik gereksinimleri önceliklendirilmeli, olası saldırı senaryoları tespit edilmeli ve bu konuda çalışma yapılmalıdır. Bu aşamada tespit edilen güvenlik açıklarının onarılması çok daha kolaydır. Yazılım güvenliği ile uygulama güvenliği birbiriyle karıştırılmamalıdır. Yazılım geliştirilip devreye alındıktan sonra uygulanan güvenlik uygulama güvenliğidir. Kullanıcı kimlik doğrulama, saldırı tespit sistemi, güvenlik duvarları uygulama güvenliğidir.



Şekil 2.1: Yazılım Süreçlerinde Yazılım Açığı Giderme Maliyeti

Şekil 2.1 yazılım geliştirme süreçlerinde yazılım açığı giderme maliyetlerini içermektedir[7].

2.1. Gereksinim Aşamasındaki Güvenlik Metrikleri

Bu aşamadaki metriklerin amacı gereksinim analizi aşamasında güvenlik nasıl değerlendirilir sorusuna yanıt aramaktır.Güvenlik Gereksimlerinin Sayısı(Nsr) diğer metriklerin kullanılması ve değerlendirilmesinde de önemli rol oynar. Sistemde güvenlik gereksinimlerinin toplamıdır. Güvenlik Gereksinimlerinin Oranı(Rsr) tanım olarak güvenlik gereksinimlerinin sayısının toplam gereksinimlerinin sayısına oranı olarak tanımlanabilir. Güvenlik gereksinimine örnek olarak erişim yetkisi olarak tanımlanan yetkilerin önceden tanımlanması ve kontrollerinin yapılması verilebilir. Rsr = |SR|/R olarak ifade edilecel olursa, Rsr; güvenlik gereksinimlerinin oranı, SR; sistemin güvenlik gereksinimlerinin sayısı, R ise sistemin tüm gerekliliklerinin sayısıdır. Güvenlik gereksinimlerinin ve güvenlik kaygısının tüm sistem içerisindeki oranıyla ilgili değerler üretir ve güvenlik seviyesi hakkında bilgi verir. Öncelikli Güvenlik Gereksinimleri Sayısı(Npsr) diğerlerinden daha fazla önceliğe sahip olan ve sisteme yapılan saldırılarda sistemi en çok etkileyecek veya yok edecek gereksinimleri listeleyip bu listenin sayısını tutar. En az Öncelikli Güvenlik Gereksinleri Sayısı(Nlsr) sistemde bulunan en az öncelikli güvenlik gereksinimleri sayısını ifade eder. Nlsr = Nsr – Npsr olarak ifade edilebilir. Öncelikli Güvenlik Gereksinleri Oranı(Rosr) Rosr = Nosr(Nosr+Nsr) olarak ifade edilir.

2.2. Tasarım Aşamasındaki Güvenlik Metrikleri

Güvenlikle ilgili tasarım aşamasındaki kusurları bulmak; daha az maliyetle sistemlerdeki açıklıkları gidermek ve sistemi güvenli hale getirebilmek demektir. Bu aşamadaki metriklerin amacı tasarım aşamasında güvenlik nasıl değerlendirilir sorusuna yanıt aramaktır.[8] Bu sorulara yanıt bulmak için kullanılan metriklerden ilki güvenlikle ilgili tasarım kararı sayısıdır. Güvenlik Tasarım Kararı Sayısı(Ndd) yazılım geliştiriciler aynı soruna birden fazla çözüm yöntemi getirebilir. Bu, tasarım aşamasında da böyledir. Bu metrik güvenlik gereksinimlerini ele alan tasarım kararları sayısını ele alır ve yazılım geliştiricilerin tasarım aşamasındaki güvenlik kaygılarını, güvenlikle ilgili değerlendirme yapmalarını sağlamayı hedefler. Tasarım aşamasında kullanılabilecek ikinci metrik ise tasarım kararlarının oranıdır. Güvenlik Tasarım Kararları Oranı(Rdd) metriğinin amacı tüm sistem içerisindeki güvenliğe ayrılan

tasarım kısmını bulmak ve yorumlamaktır. Rdd, güvenlikle ilgili tasarım kararlarının sayısının tüm sisteminin kararlarının sayısına bölünmesidir. Rdd=Ndd/Nd olarak ifade edilebilir. Nd, tüm sistemdeki tasarım karar sayısıdır. Güvenlik Algoritması Sayısı(Nsa) tüm sistemde güvenlikle ilgili kullanılan algoritmaların sayısını verir. Güvenlik Tasarım Kusurları Sayısı(Nsdf) eğer sistem içerisindeki güvenlik açısından tasarımı sıkıntılı yerlerin sayısı bulunabilirse, onarılması ve giderilmesi gereken sorunların tespiti daha kolay olacaktır. Güvenlik Tasarım Kusurlarının Oranı(Rdf) Nsdf'in tüm sistem içerisindeki tasarım kusurlarına oranıdır. Güvenliğin tüm sistemdeki değeri hakkında bilgi verebilir. Rdf= Nsdf/Nd olarak tanımlanabilir. Nd; tüm sistem içerisindeki tasarım kusurlarının sayısıdır.

2.3. Kodlama ve Entegrasyon Aşamasındaki Güvenlik Metrikleri

Bu aşamadaki metrikler kaynak kodun güvenliğini arttıran kalite özelliklerini ölçer. Bu metrikler aşağıda verilmiştir[9]. Durma Oranı(Sr) ile programın ilerlemesini etkileyen, engelleyen ya da geciktiren aktiviteler ölçülür. Programın ilerlemesini engelleyen bazı gereksiz ifadeler olabilir[9][10]. Cer = Sınıftaki kritik eleman sayısı/ Sınıftaki toplam eleman sayısı olarak ifade edilir. Eşleşen Bozulma Yayılma (Ccp) kritik bir parametrenin belirli bir değerde kalması için empoze edildiğinde potansiyel güvenlik açığı arızaları ve diğer parametreler değiştirilse bile, arızalar aynı kalır[9][10][11]. Ccp = Parametrelerle çağrılan alt yöntem sayısı/Parametrelere bağlı orjinal çağrı sayısı olarak ifade edilir. Sistem İçerisindeki Uygulama Hata Sayısı(Nerr) sistem içerisinde bulunan uygulama hata sayısı bulunur. Uygulama hata sayısı güvenlikle ilgili hatalar hakkında fikir verir. Güvenlikle İlgili Uygulama Hata Sayısı(Nserr) sistem içerisinde bulunan uygulama sayılarının özelleşmiş halidir. Güvenlikle ilgili uygulama hata sayısını verir. Yaygın olarak görülen güvenlik uygulama sayıları hakkında fikir verir. Güvenliği Etkileyen Uygulama Hatalarının Oranı(Rserr) güvenlikle ilgili hata sayısının sistem içerisindeki toplam hata sayısına oranı olarak ifade edilir. Formal olarak tanımlarsak; Rserr = Nserr/Nerr olarak ifade edebiliriz. Eğer Rserr 1 sayısına yakınsarsa sistemin güvenliğini tehlikeye atacak önemli sayıda uygulama hatası var demektir[8]. Kodlama aşamasında bu metriklerin kullanımına ek olarak statik analizler yapılmalı, kaynak kod ile ikili analiz araçları kullanılmalı ve yaygın güvenlik açıklarını tespit için değerlendirilmelidir. Statik analiz

araçları kodun yazılma aşaması bittiğinde, yani bütün halinde entegre edildiğinde kullanılır. Kodun yazılma aşamasında güvenlik açıklarının tespiti için ayrıca kod inceleme yapılmalı ve farklı gözlerin denetiminden geçirilmelidir. Bu aşama otomatikleştirilmek istenirse FxCop, Gendarme, StyleCop, CheckStyle, Findbugs, Hammurapi ve PMD araçları kullanılabilir. Bu aşamanın otomatikleştirilmesi bazı adımları gözden kaçırabilir. Bu nedenle elle de yapılmalıdır. Şekil 1'de güvenli kodlama ile ilgili aşamalar verilmiştir. Güvenli kod yazımı aşamasında statik analiz ve kod analizinin önemi büyüktür[5]. Sistem Güvenliğini Gidermek İçin Kullanılan İstisna Sayısı(Nex) güvenlik konusunda etkisi olan ve sistem arızalarını gidermek için koda dahil edilen istisnaların sayısı ölçülür[8]. İhmal Edilen İstisna Sayısı Oranı(Roex) güvenlikle ilgili ihmal edilen istisna sayısının toplam ihmal edilen istisna sayısına oranı ile ilgilenir. Matematiksel olarak Roex=Noex(Noex+Nex) şeklinde ifade edilebilir.

2.4. Test Aşamasındaki Güvenlik Metrikleri

Test aşamasında genel olarak 2 test yapılır: İşlevsel test ve güvenlik testi. Her iki test de önemli olmasına rağmen ikisi de birbirinden çok farklıdır. Güvenlik testinde saldırgan gibi sisteme sızmaya, açık aramaya ve gerekirse sisteme zarar vermeye çalışılır ve açıklıklar tespit edilmeye çalışılır. Güvenlik testi sistem geliştirmenin temel adımlarından biri olmalıdır. İşlevsel test yazılımın yapması gereken işlemleri test eder, güvenlik testi ise yapmaması gereken testidir.

Tablo 2.1: Güvenli test için gerekli her adımdaki kuralcı faliyetler.

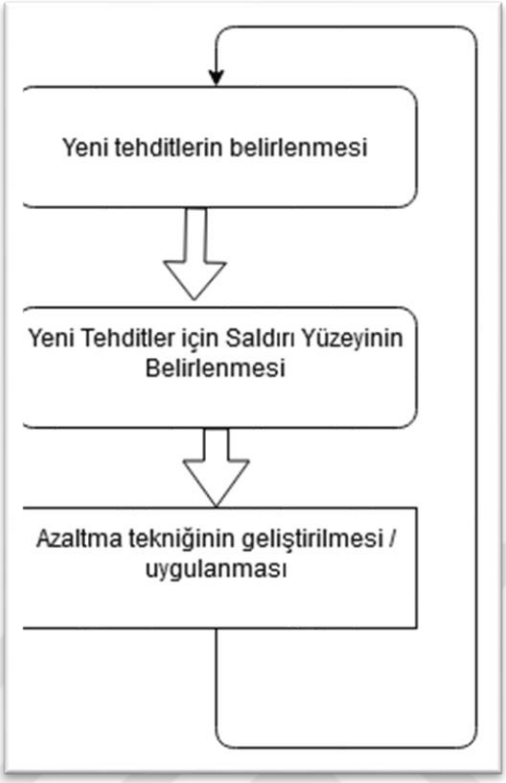
Adımlar	Uygulanacak Kuralcı Faliyetler
Güvenli Test Aktiviteleri	Olası saldırıları ve etkilerini belirlemek için kötüye kullanım durumlarını, tehdit modellemesini ve tasarım bölgerini kullanın. Bir istismarın en kolay olacağı program alanlarını sıralayın. Anonim kullanıcıların saldırabileceği alanları belirleyin. Programın en yüksek etkisini derelendirin.
Güvenlik Test Durumları	Yazılımlara başarılı bir şekilde saldırmak için test senaryoları tasarlayın. Test senaryolarına öncelik verin. Tüm varsayımları ve iş süreçlerini göz önünde bulundurun. Dinamik analiz sırasında test senaryolarını kullanın. Yazılımı bir test ortamında yükleyin ve çalıştırın ve tasarlanan test senaryolarını kullanın.
Güvenlik Test Belgeleri	Güvenlik testi durumlarını belgeleyin. Öncelikli güvenlik testi durumunu belgeleyin. Otomatik ve manuel dinamik analizden öncelikli güvenlik açıkları listesini belgeleyin.

Tablo 2.1'de güvenli test için kuralcı bir çerçeve oluşturulmuş ve testin her aşaması detaylandırılmıştır[5]. Test aşamasında sistemin güvenliğini test etmek için şu metrikler kullanılır[8]. Sistemin Toplam Güvenlik Testi Sayısı(Ttc) sistemde yapılan tüm güvenlik testi sayısını verir. Başarısız Sistem Güvenlik Testi Sayısı(Ntcp)

uygulama sırasında meydana gelen hata sayısı ve bu hataları yakalayan test sayısıdır. Güvenlik Test Senaryoları Oranı(Rtc) geliştiricilerin güvenlik test oranını belirlemelerine yardımcı olur. Matematiksel olarak Rtc = RT/T olarak ifade edebiliriz. RT güvenlik odaklı yapılan test durumlarının sayısı, T ise yapılan tüm test durumlarının sayısıdır. Test Metriklerinin Kendi İçinde Oranı daha önce yapılan çalışmalarda test aşamasında kullanılması gereken güvenlik metrikleri paylaşılmıştır. Paylaşılan güvenlik metrikleri test için yapılan güvenlik testleri çeşitliliğini içermemektedir. Birim testleri, sosyal mühendislik testleri, penetrasyon testleri, port taraması ve servis araştırması, güvenlik duvarı kural testleri, işletim sistemi sertleştirme, SQL enjeksiyon testleri, ağ taraması, zayıf şifreleri belirleme güvenlikle ilgili yapılan başlıca testlerdir. Bu testlerin metrikleri kendi aralarında değerlendirilmeleri gerekir. Bu değerlendirme yapılmazsa sosyal sistemin toplam güvenlik testi sayısı fazla olmasına rağmen bu testlerden birisi atlanabilir ya da az sayıda yapılmasından dolayı güvenlik açığı giderilmemiş olabilir. Her bir test metriğinin kendi içerisinde tutarlı olması ve test çeşitliliğinin sağlanması gerekir. Daha önce tespit edilen bir güvenlik açığı varsa güvenlik açığı incelenmeli ve hangi testin eksik yapıldığı tespit edilmelidir. Sonraki ölçümlerde tespit edilen güvenlik testi eksikliği giderilmelidir. Örneğin SQL enjeksiyon ile ilgili tespit edilen bir güvenlik açığı var ise SQL enjeksiyon test sayısı arttırılmalıdır. Diğer güvenlik testlerinin sayısı azalmayacak şekilde SQL enjeksiyon ile yapılan test sayısının toplam yapılan güvenlik testine oranı arttırılmalıdır. Bu tür testleri yapmak için birçok yöntem bulunur. Bu yöntemlerdış ağ testleri, iç ağ testleri, kör test, habersiz kör test, hedefli test, kara kutu testi, beyaz kutu testi ve gri kutu testidir.

2.5. Bakım Aşamasındaki Güvenlik Metrikleri

Uygulama dağıtımında önce yöneticiler yazılımın güvenlik durumunu anlamalıdır. Dağıtımdan sonra daha önce dikkate alınmayan ve göz ardı edilen hususlar yeniden ele alınmalı ve değerlendirilmelidir. Yazılım %100 güvenli olamaz, yeni tehditler oluşabilir ve bu tehditlerin tespit edilmesi gerekmektedir. Saldırı yüzey alanları tespit edilmeli, gelebilecek saldırıların tahminini yapabilmek için uygulama yazılımının zayıf alanlarını bilmek önemlidir. Dağıtımdan sonra ele alınan güvenlik açıkları tespiti ile beraber yeni tehditlerden ve tespit edilen kusurlardan korunmak amacıyla yamaların yazılması ve zamanında eksiksiz dağıtılması çok önemlidir.



Şekil 2.2: Güvenli bakım için gerekli adımlar.

Şekil 2.2'de güvenli bir bakım aşmasından bahsetmek için uyulması gereken adımlar listelenmiştir[5]. Buna göre dağıtımdan sınra yeni tehditler belirlenmeli, yeni tehditlerin saldırı yüzeyi tespit edilmeli ve güvenlik kusurlarının tespitinden sonra gerekli yama yazılımları gerçekleştirilmeli ve dağıtımı yapılmalıdır. Bakım aşamasında da güvenlikle ilgili bize fikir veren metrikler vardır. Güvenlik Değerlendirmesi Bağlı Yazılım Değişikliklerinin Oranı(Rsc) sistemde gerçekleştirilen işin kapsamını belirlemeye yardımcı olur[8]. Matematiksel olarak Rsc = Nsc/Nc olarak ifade edilebilir. Nc tüm sistemin değişiklik sayısını, Nsc ise yeni bir güvenlik gereksinimiyle tetiklenen değişikliklerin sayısını ifade eder. Güvenlik Değerlendirmesi Bağlı Yayınlanan Yamaların Oranı(Rp) güvenlik açıklarını kapatmak amacıyla yayınlanan yamaların oranı hesaplanır[8]. Rp = Nsp/Np olarak ifade edilebilir. Np: Tüm sistemin yama sayısı, Nsp ise sistemin güvenliği ile ilgili yama sayısıdır.

3. GÜVENLİK METRİKLERİNİN YAZILIM YAŞAM DÖNGÜSÜ AŞAMALARINA GÖRE KARŞILAŞTIRILMASI

Yazılım güvenliği ile ilgili fikir elde etmek için güvenlik metrikleri kullanılır. Bu metrikler yazılım yaşam döngüsü içerisinde farklılaşır ve bulunduğu aşama hakkında güvenlikle ilgili fikirler verir. Her aşama kendine özgüdür ve bu sebeple metrikler de farklıdır. Yaşam döngüsü içerisinde güvenlikle ilgili bir zafiyet ne kadar erken tespit edilirse maliyeti o oranda düşük olur ve tolere edilebilir. Şekil 2.1 bu konuda bilgi vermektedir.

Tablo 3.1: Güvenli yazılım geliştirme aşamaları için güvenlik aktiviteleri.

Yazılım Geliştirme Aşamaları	Güvenlik Aktiviteleri
Ön Gereksinim Aşaması	Güvenlik Eğitimi, Risk Yönetimi için Planlama ve Geliştirme Çerçevesi
Gereksinim Aşaması	Güvenlik Gereksinimlerini Belirleyin, Kullanım Durumlarını Geliştirin, Kötüye Kullanım Durumlarını Geliştirin, Güvenlik Kullanım Durumlarını Geliştirin, Gereksinim Dokümantasyonu
Tasarım Aşaması	Güvenlik Mimarisi Oluşturun, Etkileşim Noktalarını, Varlıkları ve Yazılım Saldırı Yüzeyini En Aza İndirin, Tehdit Modellerini Tanımlayın
Uygulama Aşaması	Güvenli Kod Yaz, Statik Analiz ve Kodun Gözden Geçirilmesi
Test Aşaması	Güvenlik Testi Planlama, Güvenlik Testi
Yayın ve Dağıtım Aşaması	Güvenlik İncelemesi, Güvenlik Denetimi, Güvenlik Dağıtımı

Tablo 3.2: Yazılım aşamasında güvenlik metrikleri.

Yazılım Geliştirme Aşamaları	Güvenlik Aktiviteleri
Ön Gereksinim Aşaması	Sistem için güvenlik testi mümkün mü, Sistemde güvenlik gerekli mi, Yazılım Geliştirme Sürecinin her aşamasında olası güvenlik ihtiyacı sayısı, Yazılım Geliştirme Sürecinde Toplam Güvenlik İhtiyacı Sayısı
İhtiyaç Toplama ve Analizi	Öncelikli güvenlik gereksinimleri sayısı(Npsr), En az öncelikli güvenlik gereksinimleri sayısı(Nlsr), Toplam güvenlik gereksinimi sayısı(Nsr), İhmail edilen güvenlik gereksinimleri sayısı(Rsr) ve Sayı oranı ihmal edilen güvenlik gereksinimleri(Rossr)
Yazılım Tasarımı	Güvenlikle ilgili tasarım kararlarının sayısı(Ndd), Güvenlik algoritmaları sayısı(Nsa), Tasarım kararlarının oranı(Rdd), Güvenlikle ilgili tasarım kusurlarının sayısı(Nsdf) ve Güvenlikle ilgili tasarım kusurlarının oranı(Rfd)
Kodlama	Durma oranı(Sr), Kritik eleman oranı(Cer), Kaplin bozulması yayılımı(Ccp)
Entegrasyon	Sistemde bulunan uygulama hatalarının sayısı(Nerr), Güvenlikle ilişkili uygulama hatalarının sayısı(Nserr), Güvenliği etkileyen uygulama hatalarının oranı(Rserr), Güvenlikle ilgili hataların üstesinden gelmek için uygulanan istisna sayısı(Nex),

Tablo 3.2: Devam.

Entegrasyon	Güvenlikle ilgili yürütme hatalarını ele almak için atlanan istisna sayısı(Noex) ve ihmal edilen istisna sayısının oranı(Roex)
Test	Toplam güvenlik testi vakası sayısı(Ttc), Başarısız olan güvenlik testi vakalarının sayısı(Ntcp), Güvenlik testi vakalarının oranı(Rtc) ve Başarısız olan güvenlik testi vakalarının oranı(Rtcp)
Bakım	Güvenlik değerlendirmesinden kaynaklanan yazılım değişikliklerinin oranı(Rsc) ve güvenlik açıklarını ele almak için yayınlanan yamaların oranı(Rp)
Belgeleme	Kalite değerlendirmesi için GQM yaklaşımını kullanan teknik belgeler

Tablo 3.1 yazılım geliştirme aşamaları içerisinde güvenlik aktivitelerini gösteririr[8][12][13][14][15]. Tablo 3.2 yazılım geliştirme aşamalarındaki güvenlik metriklerini göstermektedir[8][16][17]. Tablo 3.1'deki adımları gerçekleştirilmeli, sonrasında yapılan işlemlerin neticesinde güvenlik metriklerine göre fikir edinilmelidir. Bu çıktılar ve yorumlamalarla güvenlik aktiviteleri gözden geçirilmeli ve metrikler de yeniden ölçülmelidir. Yazılım yaşam döngüsü içerisinde bu süreç devam etmeli ve sonlanmamalıdır. Projelerin sadece %16.2'si vaat edilen tüm işlevsellik ile zamanında ve bütçede tamamlanarak başarılı sayıldı. Projelerin %52.7'si maliyetin üzerinde, zaman içinde ve vaat edilen işlevlerden yoksundu. Bu, %31.1'in başarısız olarak sınıflandırılmasına neden oluyor. Bu da onların terk edildiği ve ya iptal edildiği anlamına geliyor[18]. Maliyeti aşan projelerde güvenlikle ilgili yamaların ve güvenlik sorunlarının yol açtığı maliyetler de önemli seviyededir. Bu sebeple gereksinim analizi aşamasında güvenlik tahmini yapmak çok önemlidir.

3.1. Güvenlik Metriklerinin Kullanımında Yaşanan Zorluklar

Metrikler güvenlikle ilgili bilgiler verdiği gibi, sorumlu olunan kişilere raporlamalar konusunda da yardımcı olur. Yazılım güvenlik metrikleri kurum ve kuruluşların siber güvenlik alanında güçlü ve zayıf yönlerini ortaya koymak için kullanılabilir. Metrik seçimleri ve ölçümleri konusunda yaşanan zorluklar vardır. FireMon adına Ponemon Ensitüsü güvenlik metrikleri ile ilgili yaklaşık 600 BT ve güvenlik uzmanıyla Nisan 2014'te bir anket düzenledi. Bu ankete göre 10 katılımcıdan en az 8'i iş hedefleriyle uyumlu metriklere sahip olmanın önemli olduğunu söyledi. Katılımcıların %43'ü kullanılan metriklerin güvenlikle ilgili çok az şey yaptığını iletirken, %11'i güvenlik metriklerinin çok önemli ve etkili olduğunu ilettiler.[19] Ponemon Enstitüsü'nün yapmış olduğu anket sonucundan doğru metriklerin kullanım oranının düşük olduğunu ve metrik sonuçlarının BT ve güvenlik uzmanları içerisinde sınırlı sayıda kişi tarafından uygun ve yeterli metriklerin kullanıldığı çıkarılabilir. Uygun kullanılmayan metrikler ya da yetersiz metrik kullanımı ile güvenlik açıkları tespiti yapmak güvenlik zafiyetlerinin oluşmasına ve sonrasında güvenlik saldırılarına sebep olabilir. Güvenlik metrikleri kurumların siber güvenlikle ilgili güçlü ve zayıf yönlerini gösterse de kullanımında yaşanan zorluklar vardır. SonarQube ve Checkmarx gibi yazılım güvenliği ile ilgili kullanılabilecek ücretli uygulamalar mevcut ancak bazı firmalar kurulum ve uygumala maliyetinden kaçınmak için bu uygulamaları tercih etmeyebiliyor. Bu çalışmada kullanılan metrikler bu uygulamaların dışında kullanılan metrikleri ele alır. Yazılım güvenliği ile ilgili güvenlik açığı tespiti yapan araçlar genelde kod zayıflığı ve tespit üzerine yoğunlaşıp zafiyetin hangi kod satırından kaynaklandığını göstermeyi hedeflerken bu çalışmada güvenlik metrik kullanımları güvenlik açığı tespitini doğrudan yapmayı değil güvenlik ile ilgili kaygıları ve yazılım güvenliğini yazılım yaşam döngüsü içerisinde bütünüyle ele almayı ve bir kültür oluşturmayı hedefler. Güvenlik metriklerini kullanmadaki bir diğer zorluk ise metrik ölçümlerinin yazılım yaşam döngüsü içerisinde şeffaf bir şekilde yapılamaması ve ölçüm için gerekli otomasyonun yapılamamış ve güvenlik metrik ölçümlerinin kıyaslamalarının yeterli veri ya da kıyaslama ile yapılmamasıdır. Bu çalışma kapsamında kullanılan güvenlik metriklerinden herhangi birisinin sonucu tek başına herhangi bir anlam taşımaz. Önemli olan metrik sonuçlarını doğru yorumlamak ve güvenlik zafiyeti konusunda kurum ve gerekirse uygulama özelinde

doğru kıyaslamaları yapabilmektir. Bu çalışma kapsamında kodlama, test ve bakım aşamasındaki güvenlik metrikleri ile ilgili 4 uygulama geliştirici, 4 test sorumlusu, 4 BT güvenlik uzmanı ile anket çalışması yapılmış ve sonuçları paylaşılmıştır.

3.2. Kodlama, Test ve Bakım Aşamalarında Paydaşlarla Yapılan Çalışmalar

Tablo 3.3: Yazılım geliştiriciler ile yapılan anket sonuçları -1.

	Güvenlik metrikleri konusunda yeterli bilgiye sahip olduğunuzu düşünüyor musunuz?	Kod güvenlik analizi metrikleri yeterli mi?	Metriklerin ölçülebilmesi için yeterli katkıyı sağlıyor musunuz?	Kodlama yaparken yazılım güvenliği kaygısı taşıyor musunuz?	Güvenlik metriklerinin ölçülebilmesi için katkı sağlamak ister misiniz?
Katılımcı 1	Hayır	Hayır	Hayır	Hayır	Evet
Katılımcı 2	Evet	Hayır	Hayır	Evet	Evet
Katılımcı 3	Hayır	Hayır	Hayır	Evet	Evet
Katılımcı 4	Hayır	Hayır	Hayır	Hayır	Evet

Tablo 3.3 yazılım geliştiriciler ile yapılan anket sonuçlarını içermektedir. Ankete katılan 4 kişiden 3'ü güvenlik metrikleriyle ilgili yeterli bilgiye sahip olmadığını düşünüyor. Ankete katılan tüm yazılımcılar güvenlik metriklerinin yeterli olmadıklarını düşünüyorlar. 4 kişiden 3'ü güvenlik metrikleri konusunda yeterli bilgiye sahip olmasa da güvenlik metriklerinin de yetersiz olduğunu düşünüyor. Bu konuda böyle düşünmelerinin nedeni kodlama aşamasında yazılımcının yeterli bilgiye ve tecrübeye sahip olmaması, güvenlik metriklerinin sonucunda yazılım geliştiricilerin metrikleri yorumlayamaması yatıyor. Kodlama yaparkem yazılım güvenliği kaygısı taşıyan katılıcı sayısı ise yalnızca ikidir. Yazılım güvenliğinin de BT güvenlik aşamalarında tespit edilip geribildirimin yazılımcıya yapılması ve sonucunda aksiyon alınması bu kaygıyı yazılım geliştiricinin değil doğrudan BT güvenlik ekibinin sorumluluğu olarak görülmesinden kaynaklanıyor. Yazılım geliştiriciler güvenlik metriklerinin incelenmesi ve güvenlikle ilgili gerekli aksiyon alması gereken kişilerin

öncelikle yazılımcılar olmadığını düşünüyor. Tablo 3.4'te sonuçları paylaşılan yazılım güvenliği metriklerinin incelenmesi ve aksiyon alma görevinin ilk sorumlusunun yazılım geliştiriciler olduğunu düşünüyor musunuz sorusuna 4 katılımcıdan 3'ü hayır olarak cevap verdi. Bu sonuçlara göre yazılım geliştiriciler yazılım metrikleri ölçümlerinin ve sonuçlarını ilk sorumlusu kendileri olmasa da, yeterli bilgi ve tecrübeye sahip olduklarını düşünmeseler de bu konuda yardımcı olmayı ve eğitim alabileceklerine dair cevaplar verdiler. Yazılım geliştricilere yöneltilen bir diğer soru ise test için kodlama yapıp yapmadıkları idi. Bu konuda ise bütün yazılımcılar test yazmadıklarını ve test yapan kullanıcılar olduklarını ilettiler.

Tablo 3.4: Yazılım geliştiriciler ile yapılan anket sonuçları -2.

	Devreye alım öncesi herhangi bir güvenlik metriği aracı kullanıyor musunuz?	Test için kodlama yapıyor musunuz?	Metriklerin ölçülebilmesi için yeterli katkıyı sağlıyor musunuz?	Yazılım güvenliği metriklerini incelenmesi ve aksiyon alma görevinin ilk sorumlusunun yazılım geliştiriciler olduğunu düşünüyor musunuz?
Katılımcı 1	Hayır	Hayır	Hayır	Hayır
Katılımcı 2	Evet	Hayır	Hayır	Hayır
Katılımcı 3	Hayır	Hayır	Hayır	Hayır
Katılımcı 4	Evet	Hayır	Hayır	Evet

Tablo 3.4 yazılım geliştiricilere yöneltilen diğer soruları listelemektedir.

Tablo 3.5: Test sorumluları ile yapılan anket sonuçları.

	Güvenlik metrikleri konusunda yeterli bilgiye sahip olduğunuzu düşünüyor musunuz?	Yaptığınız test sayılarını düzenli olarak tutuyor musunuz?	Uygulama hata sayılarını ve sonuçlarını düzenli olarak tutuyor musunuz?	Metriklerin ölçülebilmesi için yeterli katkıyı sağlıyor musunuz?	Güvenlik metriklerinin ölçülebilmesi için katkı sağlamak ister misiniz?
Katılımcı 1	Hayır	Hayır	Hayır	Hayır	Evet
Katılımcı 2	Hayır	Hayır	Hayır	Hayır	Evet
Katılımcı 3	Hayır	Hayır	Hayır	Hayır	Evet
Katılımcı 4	Hayır	Hayır	Evet	Hayır	Evet

Tablo 3.5'te test sorumluları ile yapılan anket sonuçları paylaşılmıştır. Uygulama testi yapan 4 katılımcıdan 3'ü iş analisti 1'i ise testçi olarak çalışmaktadır. Katılımcıların tamamı güvenlik metrikleri ile ilgili yeterli bilgiye sahip olmadığını ve yapılan test sayılarını düzenli olarak tutmadıklarını ilettiler. Uygulama hata sayılarını ve sonuçlarını tutan katılımcı sayısı ise yalnızca 1. Katılımcıların tamamı metriklerin ölçülebilmesi için yeterli katkıyı sağlamadıklarını ama katkı sağlamak için destek vermeye hazır olduklarını iletttiler. Yazılım geliştiricilere yöneltilen test için kodlama yapıyor musunuz sorusuna tüm katılımcıların hayır yanıtını verdiğini de göz önünde bulundurarak test sorumlularının test aşamasındaki güvenlik metriklerinin tam ölçülebilmesi ve sonuçların değerlendirilebilmesi için test sonuçlarını, yapılan test sayısını ve başarısız test sayısını düzenli olarak tutmaları ve sonuçları konsolide etmeleri gerekmektedir.

Tablo 3.6: BT güvenlik uzmanları ile yapılan anket sonuçları.

	Güvenlik metrikleri konusunda yeterli bilgiye sahip olduğunuzu düşünüyor musunuz?	Yama sayılarını tutuyor musunuz? Güvenlikle ilgili olan yama sayılarını ayırt edebiliyor musunuz?	Güvenlikle ilgili kod değişikliği sayılarını tutuyor musunuz?	Güvenlik test sayılarını ve sızma testi sayılarını ve sonuçlarını saklıyor musunuz?	Güvenlik metriklerinin ölçülebilmesi için katkı sağlamak ister misiniz?
Katılımcı 1	Evet	Evet	Evet	Evet	Evet
Katılımcı 2	Evet	Evet	Evet	Evet	Evet
Katılımcı 3	Evet	Evet	Evet	Evet	Evet
Katılımcı 4	Hayır	Evet	Evet	Evet	Evet

Tablo 3.6'da BT güvenlik uzmanları ile yapılan anket sonuçları paylaşılmıştır. Anket sonuçlarına göre ankete katılan 4 BT güvenlik uzmanından 3'ü güvenlik metrikleri ile ilgili yeterli bilgiye sahip olduğunu düşünüyor. Tüm katılımcılar yapılan yama sayılarını sakladıklarını, güvenlikle ilgili yapılan yama sayılarını ve tüm yama sayılarını ayırt edebildiklerini ve sızma testi gibi güvenlik test sayılarını ve sonuçlarını sakladıklarını ve gerekirse güvenlik metrikleri konusunda işbirliğine açık olduklarını ifade ettiler.

Yazılım geliştiriciler, test sorumluları ve BT güvenlik uzmanlarının katıldığı anket sonuçlarına göre BT güvenlik uzmanlarının asıl sorumluluklarının güvenlik olmasının da sağladığı katkıyla 3 grup içerisinde güvenlik ve güvenlik metrikleri konusunda en yetkin ve bilgili kişiler olduğu görülüyor. Bu proje kapsamında test ve bakım aşamasındaki güvenlik metrikleri sonuçları paylaşılmıştır. Bu çalışma kapsamında sonuçların paylaşılacak olmasından dolayı test sonuçları ve test sayıları test sorumluları tarafından paylaşılmıştır ancak ankete katılan katılımcılar özelinde genel bir anlayış olmadığı ve test sonuçlarının düzenli olarak saklanmadığı görülmüştür. Bu çalışma kapsamında güvenlik metriklerinin ölçülebilmesi ve sonuçların değerlendirilebilmesi için bir model oluşturulmuş ve bu model kodlama, test ve bakım aşamaları için paylaşılmıştır.

3.3. Kodlama ve Entegrasyon Aşamasında Yapılması Gerekenler

Kodlama ve entgrasyon aşamasında metriklerin ölçülebilmesi ve sağlıklı şekilde değerlendirilebilmesi için yapılması gerekenler aşağıda listelenmiştir.

3.3.1. Yazılım Geliştiricilere Kodlama ve Entegrasyon Metrikleri İle İlgili Eğitim Verilmesi

Yazılım geliştiriciler ile yapılan anket sonuçlarına göre yazılım geliştiricilerin güvenlik metrikleri ile ilgili yeterli bilgiye sahip olmadıkları görülmüştür. Kodlama aşamasında metriklerin tanıtılması ve ölçülebilmesi için yeterli bilgilendirme yapılmalıdır. Bu bilgilendirmeler sınıf içi, görsel ya da belge şeklinde olabilir. Her yazılım geliştiricinin bu eğitimi alması ve kullanılacak metrikleri tanıması ve sistemde sorumluluk alabilmesi için bu eğitim mutlaka verilmelidir. Metriklerin ölçülmesi ve kullanımı aşamasında yazılım geliştiricilerin rolü de net olarak belirlenmeli, test ve BT güvenlik ekipleriyle iş birliği yaparak bu sürece katkı sağlamaları sağlanmalıdır.

3.3.2. Test Sorumlularına Kodlama ve Entegrasyon Metrikleri İle İlgili Eğitim Verilmesi

Test sorumluları ile yapılan anket sonuçlarına göre ankete katılan 4 test sorumlusu çalışan da güvenlik metrikleri hakkında yeterli bilgiye sahip olmadığını belirttiler. Ankete katılan 4 test sorumlusundan 3'ü ise uygulama hata sayılarını düzenli olarak saklamadıklarını ve verileri tutmadıklarını ilettiler. Test sorumlularına kullanılacak metriklerle ilgili eğitimler verilmesi, eğitimler sonrasında uygulama hata sayılarını düzenli olarak saklamaları ve listelenebilir hale getirmeleri metriklerin kullanımı açısından önemli bir yer tutuyor. Test sorumluları tarafından düzenli olarak tutulan uygulama hata sayıları ile yazılım geliştiriciler tarafından tutulacak olan uygulama sayılarının toplamı toplam uygulama hata sayısını verir. Uygulama hata sayıları ile güvenlikle ilgili alınan uygulama hata sayıları sayısı ve oranı ise uygulama hakkında güvenlikle ilgili bilgiler verir.

3.3.3. BT Güvenlik Sorumlularına Kodlama ve Entegrasyon Metrikleri İle İlgili Eğitim Verilmesi

BT güvenlik ekipleri ile yapılan anket çalışmasına göre BT güvenlik çalışanlarının sorumluluğu gereği metrikleri düzenli olarak kullandıkları, elde edilen verileri düzenli olarak sakladıkları ve sistem takibi yaptıkları anlaşılıyor. Diğer ekiplerle işbirliğine hazır oldukları ve güvenlik kaygısıyla hareket ettikleri anlaşıldı. BT güvenlik ekiplerinin de katkısıyla yazılım geliştiriciler ve test sorumluları arasında işbirliği için eğitim verilebilir, verilen eğitim sonrası ortaya çıkan veriler güvenlikle ilgili kullanılabilir.

3.3.4. Verilerin Konsolide Edilmesi

Yazılım geliştiriciler ve test kullanıcıları ile uygulama hata sayılarına ait verilerin konsolide edilmesi ve saklanması gerekir. Konsolide edilen uygulama hata sayısı(Nerr) elde edilir ve güvenlikle ilgili kullanılır. BT güvenlik ekipleri tarafından yapılan güvenlikle ilgili uygulama hata sayıları(Rserr), ihmal edilen güvenlikle ilgili hata oranı(Roex) güvenlikle ilgili bilgiler verir. Kodlama ve entegrasyon aşamasında metriklerin ölçülmesi ve konsolide edilebilmesi için yazılım geliştiriciler, test sorumluları ve BT güvenlik ekiplerinden alınan veriler aynı havuzda toplanmalı ve değerlendirilmelidir. İstisna oranları ve sayılarıyla ilgili raporlamalarda ve metriklerin konsolide edilmesinde yazılım geliştiricilerin entegrasyon aşamasında kullandıkları Checkmarx ve SonarQube gibi uygulamalar da entegre edilerek kullanımı sağlanabilir.

3.3.5. Metriklerin Karşılaştırılması

Yazılım geliştiriciler, test sorumluları ve BT güvenlik ekiplerinin eğitim sürecinin tamamlanması, verilerin konsolide edilmesi ve kullanılan bazı araçlarla uyumlu hale getirme işlemlerinin yapılmasından sonra elde edilen metrikler saklanmalı ve her entegrasyon aşamasında metriklerin karşılaştırılması yapılmalıdır. Bu metrikler başta çok net sonuçlar vermese de test ve bakım aşamasında ortaya çıkan/çıkmayan güvenlik zafiyeti sayısı ile paralellik gösterecek ve bir standarda yaklaşacaktır.

3.4. Test Aşamasında Yapılması Gerekenler

Test aşamasında metriklerin ölçülebilmesi ve sağlıklı şekilde değerlendirilebilmesi için yapılması gerekenler aşağıda listelenmiştir.

3.4.1. Yazılım Geliştiricilere Test Metrikleri İle İlgili Eğitim Verilmesi

Yazılım ekipleri ile yapılan ankette ankete katılan 4 katılımcı da test için kodlama yapmadıklarını ilettiler. Ankete katılan yazılım geliştiriciler test için kodlama yapmasalar ve bu testleri kendileri yapmasalar da ekibe yeni katılanlar ya da diğer ekiplerde bulunanlar test için kodlama yapabilir ve kendi testlerini kendileri yapabilirler. Bu konuda bilinçli olmaları ve yapılan testlerin sayısını ve sonuçlarını düzenli olarak tutmaları gerekir. Bu çalışma kapsamında da benzer bir bilgilendirme ve eğitim süreci yapılmış ve yazılım geliştiricilere süreçlerle ilgili bilgi verilmiştir.

3.4.2. Test Sorumlularına Test Metrikleri İle İlgili Eğitim Verilmesi

Test sorumluların ile yapılan anket çalışmasına göre test sorumlularının genel olarak yapılan test sonuçlarını ve test sayılarını düzenli olarak tutmadığı tespit edildi. Başarılı ve başarısız olan test sonuçlarının düzenli olarak saklanması, verilerin BT güvenlik ekipleri ve yazılım geliştiriciler tarafından elde edilen verilerle konsolide edilmesi ve sonuçların paylaşılabilmesi için test sorumlularına gerekli eğitimler verilmeli, düzenli olarak takibi yapılmalı ve sisteme ve takıma yeni katılan her çalışana bu kültür aşılanmalıdır. Güvenlik testi vakalarının oranı(Rtc) metriğinin ölçülmesinde test sorumlularına verilecek sorumluluk ve eğitim büyük önem arz etmektedir. Bu çalışma kapsamında test sorumlularına test metrikleri ile ilgili bilgilendirmeler yapılmış ve ölçümler bölüm 4'te paylaşılmıştır.

3.4.3. BT Güvenlik Sorumlularına Test Metrikleri İle İlgili Eğitim Verilmesi

BT güvenlik sorumluları ile yapılan ve Tablo 3.6'da sonuçları paylaşılan anket çalışmasına göre BT güvenlik sorumluları genel olarak metriklerle ilgili yeterli bilgiye sahip olduğunu ancak ankete katılan 4 katılımcıdan 1'i yeterli bilgiye sahip olmadığını iletti. Güvenlik ve sızma testleri ile ilgili olan metriklerle ilgili gerekli bilgilendirmenin yapılması ve BT güvenlik sorumluları ile düzenli olarak işbirliği yapılması gerekmektedir. Bu çalışma kapsamında da BT güvenlik sorumluları metriklerle ilgili bilgilendirilmiştir. Toplam güvenlik testi sayısı(Ttc), başarısız güvenlik testi sayısı(Ntcp), güvenlik testi vakalarının oranı(Rtc) ve başarısız olan güvenlik testi vakalarının oranı(Rtcp) metriklerinin ölçülmesinde, saklanmasında ve değerlerin konsolide edilmesinde BT güvenlik sorumluların doğrudan rol alması ve işbirliği yapması gerekir. Bu çalışma kapsamında yapılan testler ve sonuçları bölüm 4'te paylaşılmıştır.

3.4.4. Verilerin Konsolide Edilmesi

Yazılım geliştiriciler, test sorumluları ve BT güvenlik sorumlularına verilen eğitimler sonucunda tutulan değerlerin konsolide edilmesi ve metrik ölçümlerinin yapılması gerekir. Özellikle güvenlik testi vakalarının oranı(Rtc) metriği için yazılım geliştiriciler tarafından yapılan test sayıları, test sorumluları tarafından yapılan test sayıları ve BT güvenlik sorumluları tarafından yapılan güvenlik testlerinin toplamına ihtiyaç duyulur ve bu metrik ancak her 3 birimde yapılan testlerin konsolide edilmesi ile ölçülebilir.

3.4.5. Metriklerin Karşılaştırılması

Yazılım geliştiriciler, test sorumluları ve BT güvenlik sorumluları tarafından konsolide edilen test metrikleri ölçümleri saklanmalı, sonraki uygulama testleri ve genel metrikler ile kıyaslanmalıdır. Genel test anlayışı ve güvenlik seviyesi hakkında bilgiler değerlendirilebilir. Metrikler karşılaştırılır iken konsolide veriler bir önceki metrik sonuçları ile karşılaştırılmalıdır. Burada dikkat edilmesi gereken diğer husus ise her güvenlik metriğinin kendi içerisinde de karşılaştırmasının yapılmasıdır.

3.5. Bakım Aşamasında Yapılması Gerekenler

Bakım aşamasında metriklerin ölçülebilmesi ve sağlıklı şekilde değerlendirilebilmesi için yapılması gerekenler aşağıda listelenmiştir.

3.5.1. Yazılım Geliştiricilere Bakım Metrikleri İle İlgili Eğitim Verilmesi

Bakım aşamasında güvenlik değerlendirmesinden kaynaklanan yazılım değişikliklerinin oranı(Rsc) hesaplanırken yazılım geliştiricilerin işbirliğine ihtiyaç duyulur. Matematiksel olarak Rsc = Nsc/Nc olarak ifade edilebilir. Nc tüm sistemin değişiklik sayısını, Nsc ise yeni bir güvenlik gereksinimiyle tetiklenen değişikliklerin sayısını ifade eder. Yazılım geliştiriciler tarafından yapılan yazılım değişikliklerinin toplamını ifade eden Nsc'nin ölçülmesinde yazılım geliştiricilerin bilgilendirilmesi gerekir. Bu toplam sayısı Git ve TFS gibi kod yönetim sistemlerinden de alınabilir. Bu çalışma kapsamında da TFS ve Git üzerinden yazılım geliştiricilerin desteği ile ölçümler yapılmış ve sonuçlar bölüm 4'te paylaşılmıştır.

3.5.2. BT Güvenlik Sorumlularına Bakım Metrikleri İle İlgili Eğitim Verilmesi

Bakım aşamasında güvenlik değerlendirmesinden kaynaklanan yazılım değişikliklerinin oranı(Rsc), güvenlik açıklarını ele almak için yayınlanan güvenlik yamalarının oranı(Rp) ve bu metriklerin hesaplamalarında kullanılan yeni bir güvenlik gereksinimiyle tetiklenen değişikliklerin sayısını(Nsc), toplam yayınlanan yama sayısı(Nsp) metriklerinin ölçümleri için BT güvenlik sorumlularına eğitim verilmeli, bu verilerin düzenli saklanması için işbirliği yapılmalıdır. Bu çalışma kapsamında yapılan çalışma sonuçları bölüm 4'te paylaşılmıştır.

3.5.3. Verilerin Konsolide Edilmesi

Yazılım geliştiriciler ve BT güvenlik sorumlularına verilen eğitimler ve metrik bilgilendirme sonucunda her iki ekibin de ölçtüğü değerler konsolide edilmeli ve saklanmalıdır.

3.5.4. Metriklerin Karşılaştırılması

Yazılım geliştiriciler ve BT güvenlik sorumluları tarafından konsolide edilen bakım metrikleri saklanmalı ve ölçüm sonuçları sonraki metrik ölçüm sonuçları ile kıyaslanmalıdır. Bu kıyaslamalar ve sonuçlar bakım aşamasındaki güvenlik sonuçları hakkında bilgi verebilir.

4. TEST VE BAKIM AŞAMASINDA YAPILAN ÖLÇÜMLER VE SONUÇLARI

Bu çalışma ile bir belirli bir sektörde kullanılan bir kurumun yazılım ürünleri 2020 ve 2021 yılları içerisinde test ve bakım aşamasında incelenmiştir. Yapılan güvenlik test süreçleri yakından incelenmiş ve süreçler bu çalışma paralelinde işletilmiştir. Standart güvenlik testleri, sızma testleri, yapılan kod değişiklikleri ve güvenlik yamaları yakından incelenmiş, 8 web sitesi, 2 masaüstü uygulama, 300 dış IP, 980 iç IP, 4 farklı kablosuz ağ ile e-posta suncusu güvenlik açısından incelenmiş ve sonuçları paylaşılmıştır.

Tablo 4.1: Güvenlik test sonuçları risk seviyelendirilmesi.

Risk Seviyesi	Risk Puanı	Detaylı Açıklama
ACİL	5	Acil öneme sahip zafiyetler, saldırganlar tarafından uzaktan gerçekleştirilen ve sistemin tamamen ele geçirilmesi ile sonuçlanabilecek ataklara sebep olan zafiyetlerdir. Bir saldırgan bu tür zafiyetleri kullanarak sunucuyu root veya administrator hakları ile ele geçirebilir. Ayrıca arka kapıların (backdoor) ve trojanlarin varlığı da bu kategoridedir.
KRİTİK	4	Bu tür zafiyetler saldırgana root veya administrator hakları olmaksızın sunucuya erişim imkanı sağlar. Saldırgan bu durumda dosya sistemine kısmi erişim imkanı sağlar. Ayrıca kritik öneme sahip bilgilerin ifşası da bu kategoriye girmektedir.

Tablo 4.1: Devam.

YÜKSEK	3	Bu tür zafiyetler saldırgana sunucu üzerindeki güvenlik ayarları da dahil olmak üzere spesifik bilgilere ulaşma imkanı sağlar. Bazı dosyalara erişim, dizin listeleme, servislerin yetkisiz kullanımı, filtreleme kurallarının ifşası gibi zafiyetler bu tür zafiyetlere örnek verilebilir. Ayrica DoS saldırıları bu gruba girmektedir.
ORTA	2	Bu tür zafiyetler servislerin versiyon bilgisi ifşası gibi öneme sahip bilgilerin ifşasıdır. Bu zafiyetleri kullanan bir saldırgan sunucuya ait potansiyel saldırılar arayabilir, bu bilgileri kullanarak bir saldırı planlayabilir
DÜŞÜK	1	Düşük öneme sahip zafiyetler ise etkilerinin tam olarak belirlenemediği ve literatürdeki en iyi sıkılaştırma yöntemlerinin (best practices) izlenmemesinden kaynaklanan eksikliklerdir

Tablo 4.1 güvenlik testi sonuçlarında kullanılan risk seviyesini göstermektedir. Tabloya göre risk seviyesi en kritik olan olan işlemler için risk puanı 5, en düşük risk seviyesine sahip işlemler için risk puanı 1 ile temsil edilir. 1'den 5'e doğru risk seviyesi kritik ve acil önlem gerektirecek düzeyde artmaktadır.

Tablo 4.2: Bulguların önem derecesi ve kategorisi.

Bulgu Adı	Önem Derecesi	Bulgu Kategorisi
E-posta ile Oltalama Saldırısı	Acil	Sosyal Mühendislik
Apache Tomcat Manager Öntanımlı Hesap Kullanımı(CVE-2009-3099)	Kritik	Sistem
SPN Yapılandırma Zafiyeti	Kritik	Sistem
Öntanımlı Postgresql Veritabani Kullanıcı Hesabı	Kritik	Veritabanı
Yetkisiz Erişilen Web Servis Dokümantasyon Dosyası	Yüksek	Web
Dizin Dolaşma (Directory Traversal)/Dosya Dahil Etme Zafiyeti (OTG-AUTHZ-001)	Yüksek	Web
Bir Fonksiyonun Limitsiz Kullanilabilmesi (OTG-BUSLOGIC-005)	Yüksek	Web
Kimlik Bilgilerinin Şifresiz Kanaldan Aktarılması (OTG-AUTHN-001)	Yüksek	Web
Eksik/Hatalı Yapılandırılmış HTTP Başlıkları	Yüksek	Web
Yetkisiz Erişilen Web Sayfaları (OTG-AUTHZ-002)	Yüksek	Web
Slow HTTP DOS Servis Dışı Bırakma Açıklığı (OTG- CONFIG-002)	Yüksek	Web
İnternetten Erişilebilen Hassas Bilgiler (OTG-INFO-001)	Yüksek	Web
Desteği Olmayan Windows İşletim Sistemi Kullanımı	Yüksek	Sistem
Apache Tomcat Çoklu Güvenlik Açıklığı	Yüksek	Sistem
Genele Açık NFS Paylaşımı Kullanımı	Yüksek	Sistem
İstemci Sistemler Üzerinde Tespit Edilen Güvenlik Zafiyetleri	Yüksek	Sistem

Tablo 4.3: Bulguların önem derecesi ve kategorisi.

Bulgu Adı	Önem Derecesi	Bulgu Kategorisi
Host Tabanli Güvenlik Kontrollerinin Atlatilmasi	Yüksek	Sistem
Tahmin Edilebilir / Öntanimli Hesap Bilgisi Kullanimi	Yüksek	Sistem
Microsoft IIS Tilde Ad Çözümleme Zafiyeti	Yüksek	Sistem
Desteği Olmayan Web Sunucu Yazılımları	Yüksek	Sistem
Üçüncü Parti Yazılımların Güncelleme Eksikliği	Yüksek	Sistem
Microsoft Windows İşletim Sistemi Güncelleme Eksiklikleri	Yüksek	Sistem
Parola Korumasız Web Uygulamaları/Arayüzleri	Yüksek	Sistem
Desteği Olmayan Unix İşletim Sistemi Kullanımı	Yüksek	Sistem
Aynı Kullanıcı Adı Hesabının Farklı Sistemlerde Kullanımı	Yüksek	Sistem
ARP Önbellek Zehirlenmesi(MITM Saldırısı)	Yüksek	Network
Yetkisiz E-posta Gönderim Zafiyeti(Mail Relay)	Yüksek	E-Posta

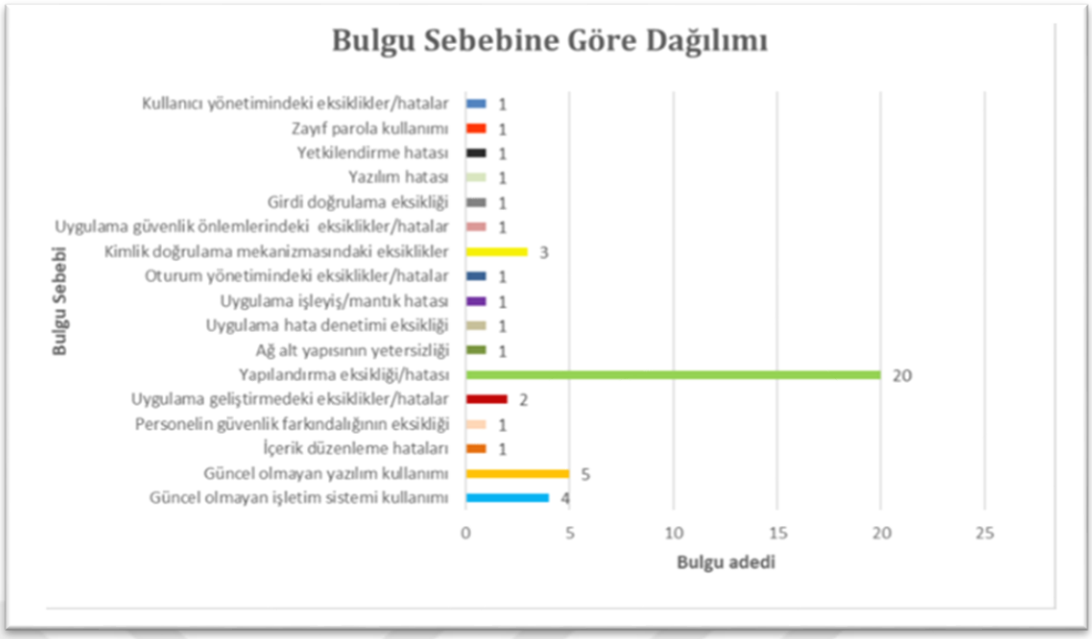
Tablo 4.4: Bulguların önem derecesi ve kategorisi.

Bulgu Adı	Önem Derecesi	Bulgu Kategorisi
NetBIOS ve LLMNR Zehirlenmesi	Yüksek	DNS
Cisco Wireless LAN Controller Güncelleme Eksiklikleri	Yüksek	Kablosuz Ağ
Dağıtık Servis Dışı Bırakma(DdoS) Zafiyeti	Yüksek	Ağ Altyapısı
Zayıf Parola Politikası Kullanımı(OTG-AUTHN-007)	Orta	Web
Özelleştirilmemiş Hata Sayıları	Orta	Web
Altyapı ve Uygulama Yönetim Panellerinin İfşası (OTG-CONFIG-005)	Orta	Web
Güncel Olmayan Web Uygulama Bileşenleri	Orta	Web
Oturum Çerezi Güvenlik Eksiklikleri(OTG-SESS-002)	Orta	Web
Web Sunucuda Telikeli HTTP Metodu Kullanımı (OTG-CONFIG-006)	Orta	Web
Kullanıcı Hesaplarının Tespit Edilebilmesi(OTG-IDENT-004)	Orta	Web

Tablo 4.5: Bulguların önem derecesi ve kategorisi.

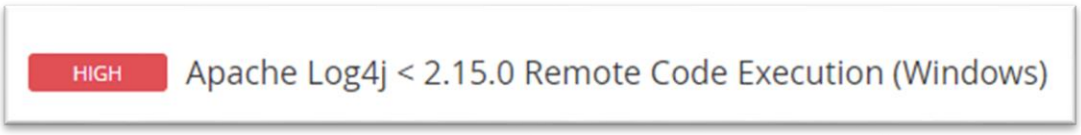
Bulgu Adı	Önem Derecesi	Bulgu Kategorisi
Anonim FTP Hesabı Kullamı	Orta	Web
IPMI v2.0 Parola Özeti İfşası	Orta	Sistem
Öntanımlı SNMP Bilgileri Kullanımı	Orta	Network
Sahte Kabosuz Ağ Yayını Yapabilme	Orta	Kablosuz Ağ
HTTP Başlık Bilgisinde Hassas Veri İfşası(OTG-INFO)	Düşük	Web

Tablo 4.2, Tablo 4.3, Tablo 4.4 ve Tablo 4.5 bulguların önem derecesini ve kategorisini göstermektedir. Bu çalışma kapsamında yapılan güvenlik testleri ile en önemli ve kritik olarak sosyal mühendislik saldırıları özellikle de e-posta ile yapılan oltalama saldırıları olarak belirlenmiştir. Sosyal mühendislik saldırıları sonrası en önemli riskler ise Apache Tomcat Manager öntanımlı hesap kullanımı, SPN(Service Principal Name) yapılandırma zafiyeti ve öntanımlı PostgreSQL veritabanı kullanıcı hesabı tanımlamaları olarak gösterilmiştir.



Şekil 4.1: Bulgu sebebine göre dağılım.

Şekil 4.1 bulgu sebebine göre dağılımı göstermektedir. Bu tablo 2020 yılında yapılan güvenlik testleri sonucunda elde edilen verileri içermektedir. Yazılım hatası olarak yalnızca 1 hata tespit edilmiş, güncel olmayan yazılım kullanımı 4, uygulama geliştirmedeki eksiklikler/hatalar 2 olmak üzere yazılımla ilgili güvenlikle ilgili bulgu sayısı toplam 7 iken yapılandırma eksikliği/hatası bulgu sayısı 20'dir.

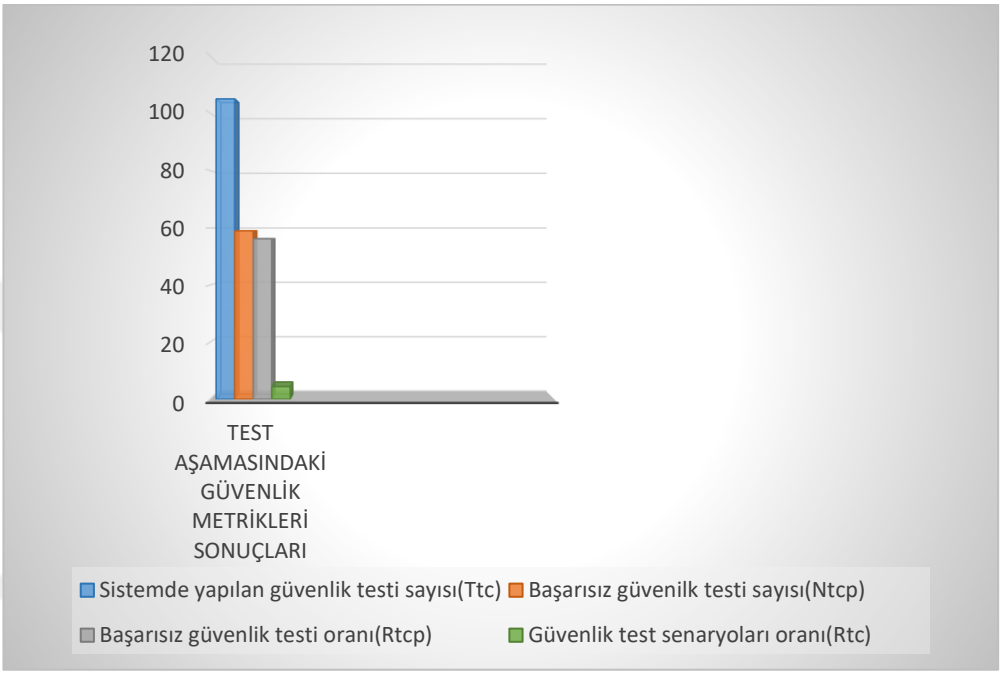


Şekil 4.2: Yazılım güvenlik açığı örneği.

Şekil 4.2 Kod aşamasında kullanılan ve Apache Log4j versiyonu 2.15.0'dan daha eski olan versiyonlarda güvenlik açığı tespit edilen ve bu uyarıyı gösteren test sonucunu içermektedir. Bu çalışma kapsamında incelenen üründe eski versiyonların tespit sürecine dahil olunmuş ve güvenlik testi sonucunda versiyon güncellemeler yapılmıştır.

4.1. Test Aşamasında Yapılan Ölçümler ve Sonuçları

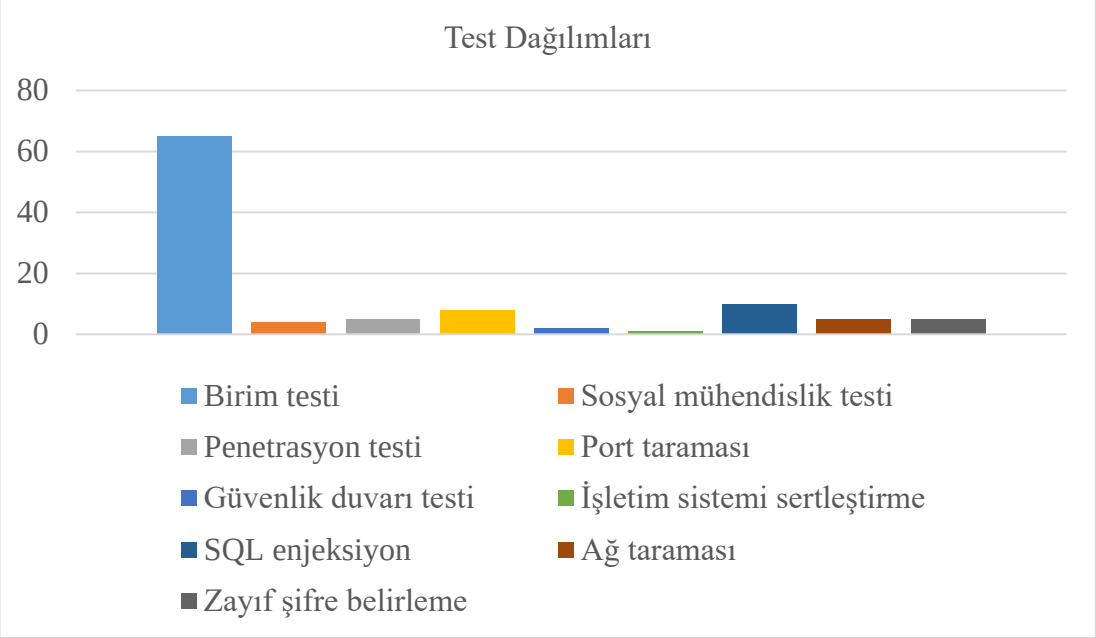
Bölüm 3.4'te test aşamasında yapılması gerekenler listelenmiştir. Bu çalışma kapsamında yazılım geliştiriciler, test sorumluları ve BT güvenlik sorumluları ile test aşamasındaki güvenlik metrikleri ile ilgili bilgilendirmeler yapılmış ve çalışma sonucunda metrikler ölçülmüştür.



Şekil 4.3: Test aşamasındaki ilk metrik sonuçları.

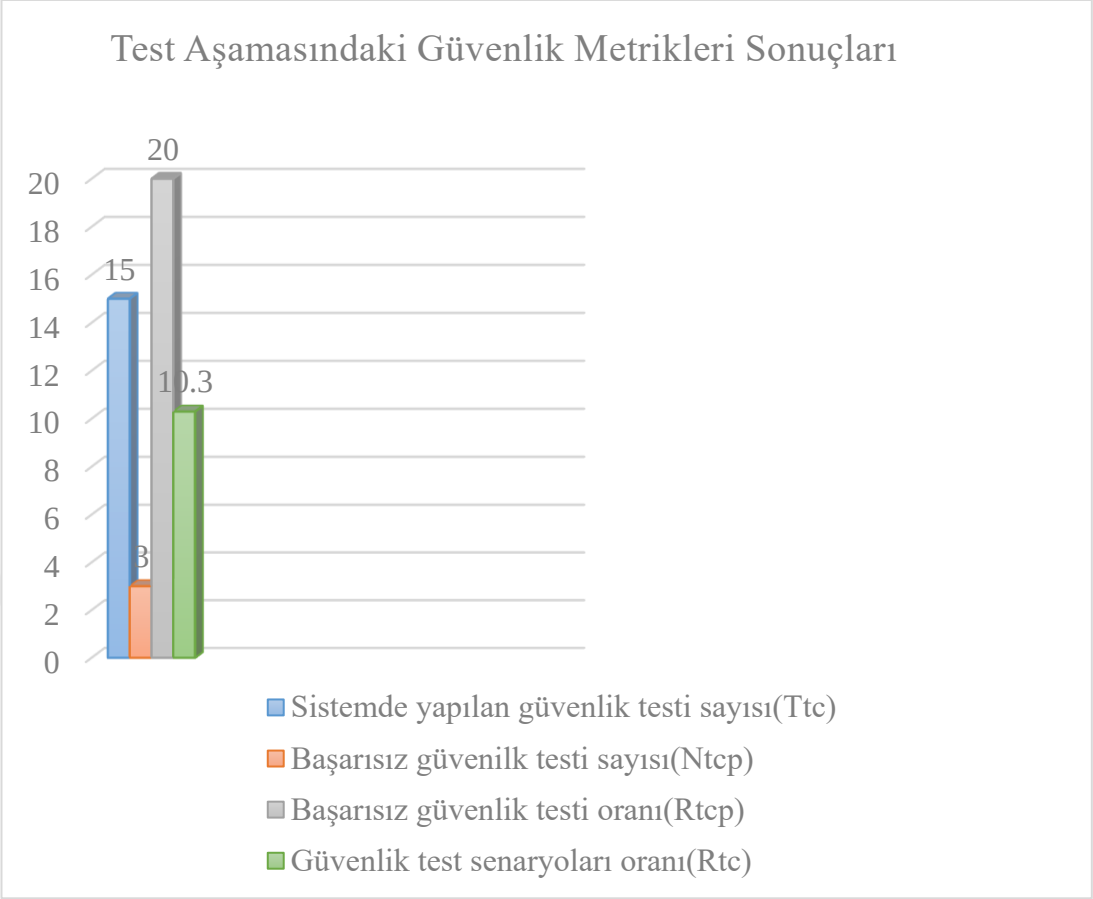
Yapılan çalışma kapsamındaki ölçümlere göre güvenlik test sayısı 105, tüm sisteme yapılan test sayısı tutulan test dökümanı ve yapılan görevlere göre 2357'dir. BT güvenlik ekipleri tarafından yapılan 105 test ve yazılım geliştiriciler ve test sorumluları tarafından yapılan 2252 testinin toplamı olarak 2357 toplam test sayısına ulaşılmıştır. 105 güvenlik testinin 59'u uygulamayı devreye alım sırasında başarısız olmuş ve bu ölçümler Checkmarx sonuçları neticesinde çıkarılmıştır. Bu testler sonucunda 13 yazılım değişikliği yapılmış, yazılım değişikliği sonrası checkmarx sonuçlarına göre 2 kez daha yazılım değişikliği yapılmak durumunda kalınmıştır. Bu sonuçlarla yapılan toplam kod değişikliği sayısı 15'tir. Yazılım kod değişikliği sayısı bakım aşamasındaki metriklerde kullanılmıştır. Bakım aşamasındaki güvenlikle ilgili yama ve kod değişikliği sayısını azaltmak için güvenlikle ilgili yapılan test sayısı arttırılmalı ve başarısız test sayısını azaltmak için de kodlama ve entegrasyon aşamasındaki metrikler daha iyi kullanılmalıdır. Metrik ölçümleri sonrasında yapılan

testlerde elde edilen veriler sonraki verilere ışık tutmalı ve test sayıları ve başarısız test sayılarının azaldığı gözlenmelidir. Şekil 4.3 ilk metrik ölçümlerini ve sonuçlarını içermektedir.



Şekil 4.4: Birinci metrik ölçümlerinde test aşamasında yapılan güvenlik testlerin dağılımları.

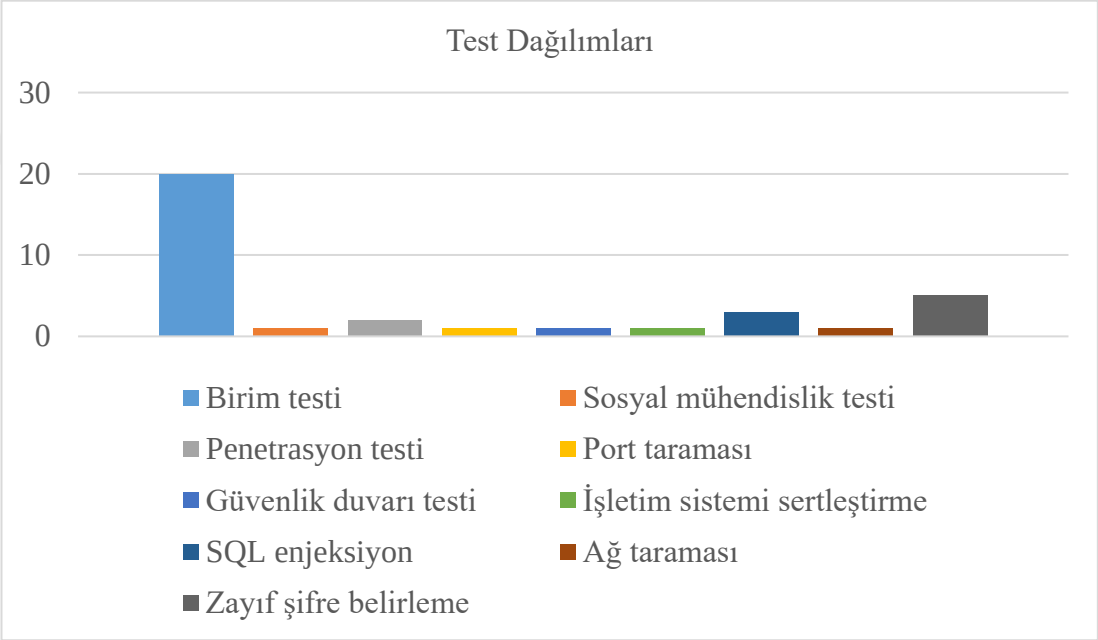
Şekil 4.4'te test aşamasında güvenlikle ilgili yapılan testlerin dağılımları gösterilmektedir. İlk metrik ölçümlerine kadar 65 birim testi, 4 sosyal mühendislik testi, 5 penetrasyon testi , 8 port taraması, 2 güvenlik duvarı testi, 1 işlem sistemi sertleştirme, 10 SQL enjeksiyon testi, 5 ağ taraması testi ve 5 zayıf şifre belirleme testi yapıldı. 59 başarısız test incelendiğinde büyük çoğunluğunun SQL enjeksiyon nedeniyle kaynaklandığı ve bazılarının ise global değişken ve zayıf veritabanı şifrelendirmesinden kaynaklandığı görüldü. Bu sebeple sonraki aşamalarda yazılım değişiklikleri ve daha güçlü parola kullanımına ve SQL enjeksiyonu azaltacak değişiklikler yapıldı. Güvenlik testleri içerisinde birim test, SQL enjeksiyon ve zayıf şifre belirleme ile sızma testlerinin oranının arttırılmasına karar verildi.



Şekil 4.5: Test aşamasındaki ikinci metrik sonuçları.

Yapılan ilk ve kapsamlı metrik çalışması Şekil 4.4'te paylaşılmıştır. Bu metriklerin ölçümlerine ve değerlendirmelerine göre bakım aşamasında yapılan kod değişikliği sayısı ve yama sayısı da göz önünde bulundurularak güvenlik metrikleri gözden geçirilmiş ve sonraki süreçte tüm sistem içerisinde yapılan güvenlik testi sayısı arttırılmıştır. Şekil 4.5'te paylaşılan verilere göre sistemde toplam yapılan test sayısı 162, sistemde yapılan güvenlik testi sayısı(Ttc) 31, başarısız güvenlik testi sayısı(Rtcp) 3, güvenlik test senaryaları oranı(Rtc) %19.13 olarak ölçülmüştür. Başarısız güvenlik test senaryolar oranını azaltmak için de entegrasyon aşamasında Checkmarx sonuçları titizlikle incelenmiş ve rapora dikkat edilmiş, kodlama checkmarx bulgu sonuçlarını aşacak düzeye gelmiş olsa da tüm yönlendirmelere dikkat edilmiş ve bu sayede güvenlik risk minimize edilmeye çalışılmıştır. Şekil 4.3'te sonuçları paylaşılan metrik öçümlerinde başarısız güvenlik testi oranı %56.2 iken Şekil 4.5'te sonuçları paylaşılan metrik ölçümlerine göre bu oran %9.67 olarak hesaplanmıştır. İkinci ölçüm sonuçları daha kısıtlı test sayıları ve verileri ile yapılmış olsa da bu azalmanın temel sebeplerinden birisi kodlama ve entegrasyon aşamasında

verilen eğitimler ve yönlendirmeleri uygulama ve verileri konsolide etme sonucunun etkisi de sayılabilir. Şekil 4.3'e göre güvenlik kaygısı ile yapılan test sayısının sistem testleri içerisindeki oranı %4.45 iken Şekil 4.5'te ilk veriler de göz önünde bulundurularak bu oran arttırılmış ve %19.13'e ulaşmıştır. İlk metrik ölçümlerinde %4.45 güvenlik test senaryoları oranı olsa da bakım aşamasında yapılan güvenlik testleri sırasında güvenlikle ilgili bulgular tespit edilmiş olmasında dolayı güvenlik test sayısı oranının daha yukarıya çıkarılması gerektiği anlaşılmaktadır. Bu sebeple sonraki ölçümlerde bu konuda hassas davranılmış ve güvenlik testi sayısı ve oranı arttırılmıştır.

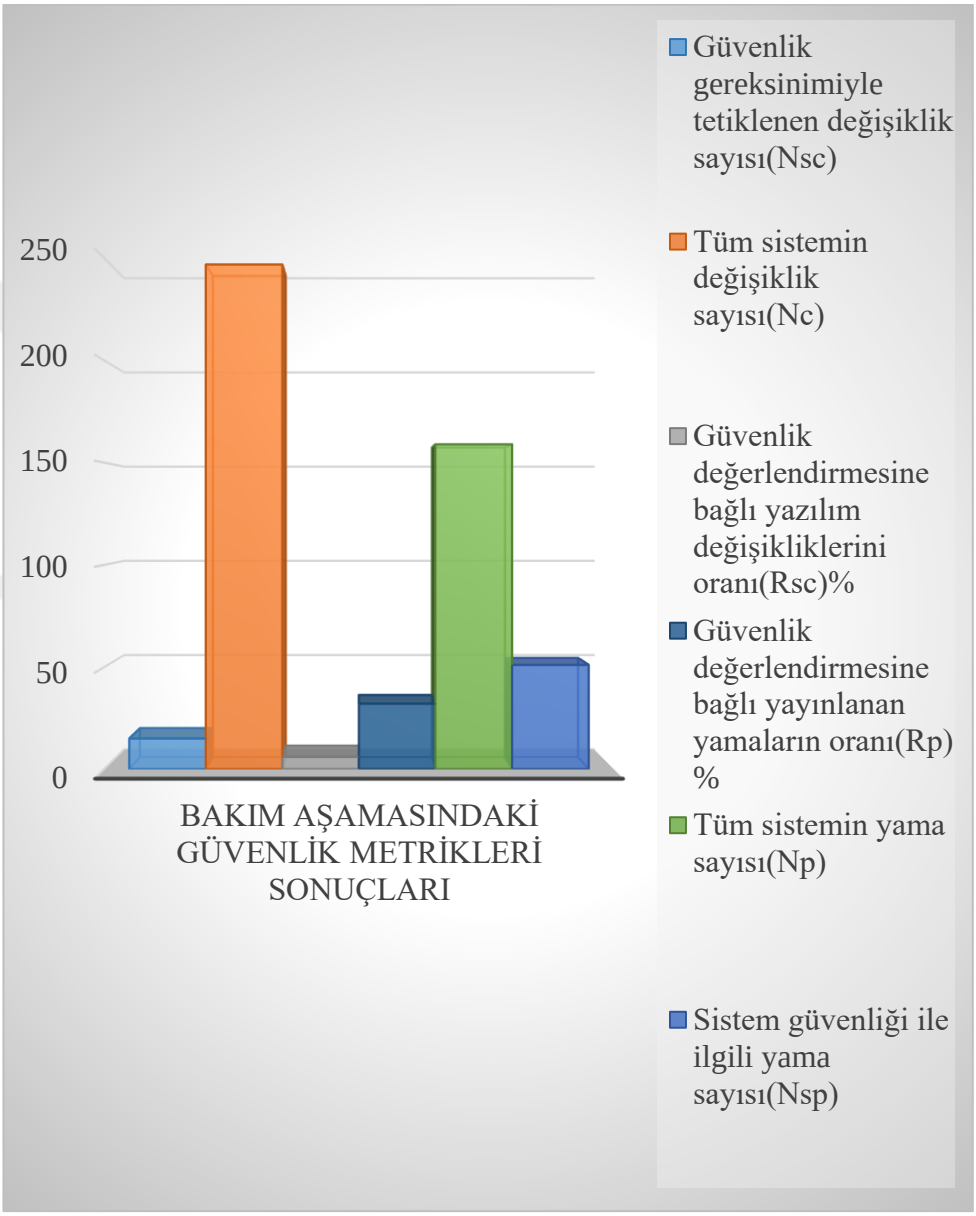


Şekil 4.6: İkinci metrik ölçümlerinde test aşamasında yapılan güvenlik testlerin dağılımları.

Şekil 4.6 birinci metrik ölçümleri ile ikinci metrik ölçümleri arasında kalan zamanda yapılan güvenlik testlerinin dağılımlarını göstermektedir. Bu dağılıma göre 20 birim testi, 1 sosyal mühendislik testi, 2 penetrasyon testi, 1 port taraması, 1 güvenlik duvarı taraması, 1 işletim sistemi taraması, 3 SQL enjeksiyon testi, 1 ağ tarama ve 1 zayıf şifre belirleme testi yapıldı. İlk metrik ölçümleri aşamasında birim testinin tüm güvenlik testlerine oranı %61.9 iken ikinci metrik ölçümlerinde bu oran %64.51 olarak ölçülmüştür. Penetrasyon testinin toplam güvenlik testine oranı ilk ölçümlerde %4.7 iken ikinci ölçümlerde %6.45 olarak ölçülmüştür. SQL enjeksiyon testlerinin toplam güvenlik testlerine oranı ilk metrik sonuçlarında %9.5 iken ikinci metrik ölçümlerinde %9.67 olarak ölçülmüştür.

4.2. Bakım Aşamasında Yapılan Ölçümler ve Sonuçları

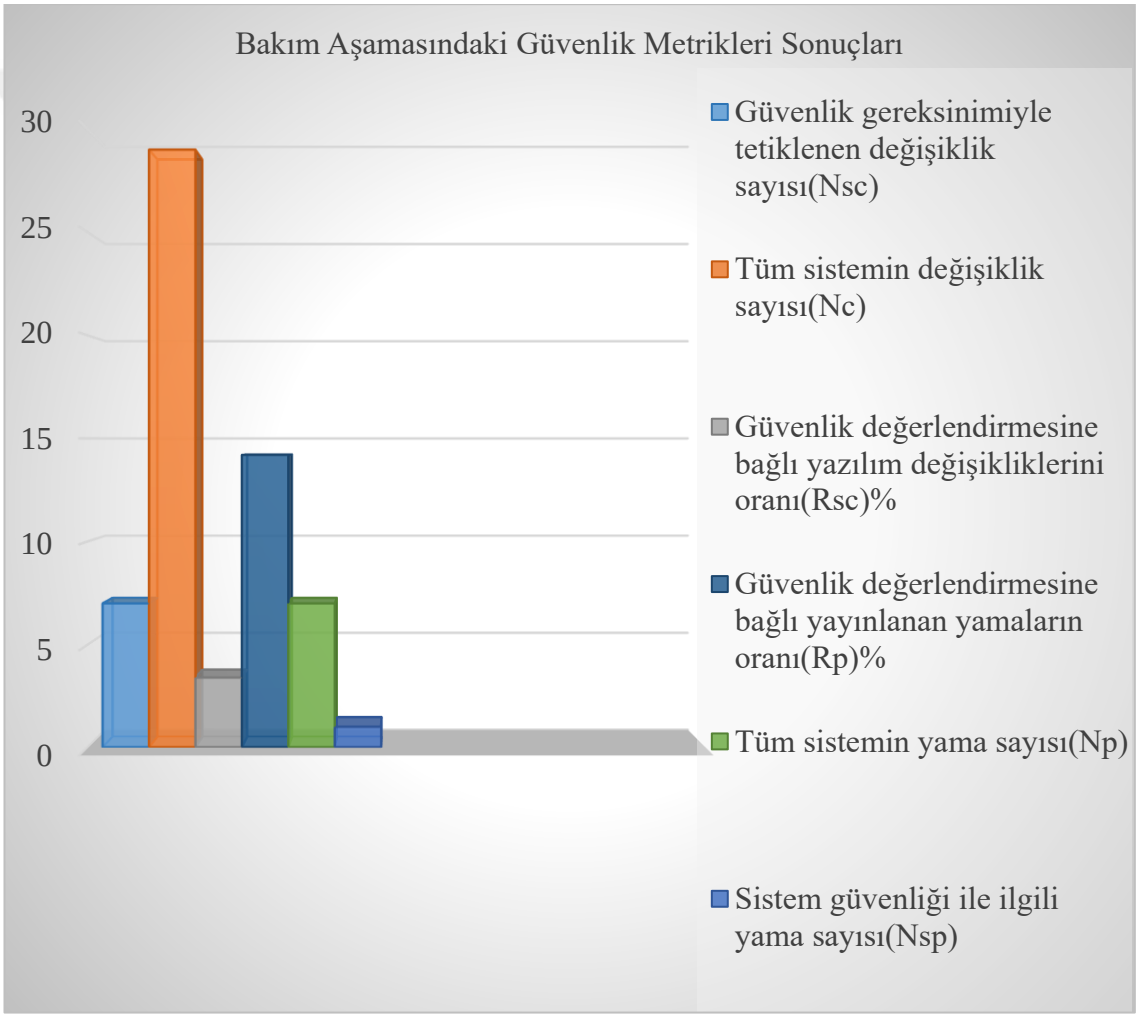
Bölüm 3.5'te bakım aşamasında yapılması gerekenler listelenmiştir. Bu çalışma kapsamında yazılım geliştiriciler ve BT güvenlik sorumluları ile bakım aşamasındaki güvenlik metrikleri ile ilgili bilgilendirmeler yapılmış ve çalışma sonucunda metrikler ölçülmüştür.



Şekil 4.7: Bakım aşamasındaki ilk metrik sonuçları.

Şekil 4.7'de bakım aşamasında ilk ölçülen güvenlik metrikleri sonuçları yayınlanmıştır. Bu metrik ölçümlerine göre güvenlik gereksinimiyle tetiklenen gereksinim sayısı(Nsc) 15, tüm sistemin değişiklik sayısı(Nc) 247, güvenlik

değerlendirmesine bağlı yazılım değişikliklerinin oranı(Rsc) %6, tüm sistemin yama sayısı(Np) 159, sistem güvenliği ile ilgili yama sayısı(Np) 51, güvenlik değerlendirmesine bağlı yayınlanan yamaların oranı(Rp) %32 olarak ölçülmüştür. Güvenlik gerekçesiyle yayınlanan yama sayısını ve kod değişikliğini daha aza indirmek için ilk metrik değerleri konsolide edilip bu çalışma kapsamında kullanılmıştır. İyileştirme için bu çalışma kapsamında yapılan ilk çalışma kodlama ve entegrasyon çalışmaları sırasında kullanılan Checkmarx verilerini daha etkin kullanmak, güvenlik testi sayısı oranının yükseltmek olmuştur. Bu çalışmalar sonucunda bakım aşamasındaki ikinci metrik sonuçları Şekil 4.8'de paylaşılmıştır.



Şekil 4.8: Bakım aşamasındaki ikinci metrik sonuçları.

Şekil 4.8'de paylaşılan sonuçlara göre güvenlik gereksinimiyle tetiklenen değişiklik sayısı(Nsc) 1, tüm sistemin değişiklik sayısı(Nc) 29, güvenlik değerlendirmesine bağlı yazılım değişikliklerinin oranı(Rsc) %3.4, tüm sistemin yama

sayısı(Np) 1, sistem güvenliği ile ilgili yama sayısı(Np) 1, güvenlik değerlendirmesine bağlı yayınlanan yamaların oranı(Rp) %14.2 olarak ölçülmüştür. Kodlama ve entegrasyon aşamaları ile test aşamasındaki metrikleri değerlendirerek bu doğrultuda güvenlik odaklı test sayılarını arttırıp uygulama hata sayıları azaltıldığında bunun sonucu olarak bakım aşamasındaki güvenlik kaynaklı yazılım değişikliği oranının ve güvenlik değerlendirmesine bağlı yayınlanan yama oranının azaldığı görüldü. İkinci metriklerin ölçümü için kısıtlı verilerle işlem yapılmış olsa da oranlarda ciddi iyileştirmeler olduğu görüldü.

5. SONUÇ

Bu çalışma kapsamında yazılım yaşam döngüsü içerisinde metrikler listelenmiş, bu metriklerin kodlama ve entegrasyon, test ve bakım aşamalarında kullanılabilmesi için bir model oluşturulmuş ve bu modelin sonucunda metrikler hesaplanmış ve ilk sonuçlar paylaşılmıştır. İlk sonuçlardan hareket ederek kodlama ve entegrasyon ile test aşamasında iyileştirmeler yapılmış ve metriklerle konsolide edilerek paylaşılmıştır. İlk güvenlik metrik değerlendirmelerinde test aşamasındaki ilk ölçümlerde güvenlik test senaryaları oranı(Rtc)%4.45, başarısız güvenlik testi oranı(Rtcp) %56.2 olarak paylaşılmıştır. İkinci ölçümde ise güvenlik test senaryaları oranı(Rtc)%19.13 ve başarısız güvenlik testi oranı(Rtcp) %9.67 olarak paylaşılmıştır. Bu çalışma kapsamında yapılan metrik değerlendirmelerinde güvenlik test senaryoları oranı(Rtc) %193.77 artarken, başarısız güvenlik testi oranı(Rtcp) %82.79 azalmıştır. Kodlama ve entegrasyon ile test aşamasında yapılan iyileştirmelerden sonra bakım aşamasında da ciddi oranda iyileştirmeler tespit edilmiştir. İlk ölçüm sonuçlarına göre güvenlik değerlendirmesine bağlı yazılım değişikliklerinin oranı(Rsc) %6, güvenlik değerlendirmesine bağlı yayınlanan yamaların oranı(Rp) %32 olarak ölçülmüştür. İyileştirmeler sonrası ikinci ölçümde ise güvenlik değerlendirmesine bağlı yazılım değişikliklerinin oranı(Rsc) %3.4, güvenlik değerlendirmesine bağlı yayınlanan yamaların oranı(Rp) %14.2 olarak ölçülmüştür. güvenlik değerlendirmesine bağlı yazılım değişikliklerinin oranı(Rsc) %43.3 oranında azalmış ve bu konuda iyileştirme sağlanmıştır. Güvenlik değerlendirmesine bağlı yayınlanan yamaların oranı(Rp) %55.6 oranında azalmış ve bu konuda da iyileşme sağlanmıştır.

KAYNAKLAR

[1] Du J., Yang Y., Wang Q., (2009), "An Analysis for Understanding Software Security Methodologies", Third IEEE International Conference on Secure Software Integration and Reliability Improvement, 141-149, Shangai/China, July.

[2] Paul M., (2014), "The Ten Best Practices for Secure Software Development", isc2.org, pp.5, Romania, 2014/06/02.

[3] Web 1, (2022), https://www.itnews.com.au/feature/building-security-into-your-software-development-lifecycle-102315, (Erişim Tarihi: 05/01/2022).

[4] Allen J., (2010), "Measuring Software Security", Extracted from the 2009 CERT Research Annual Report, Carnegie Mellon University, 64- 65.

[5] Khan D., (2011), "Secure Software Development: A Prescriptive Framework", Computer Fraud & Security, 2011(8), 12-20.

[6] Gregoire J., Buyens K., Win B., Scandariato R., Joosen W., (2007), "On the Secure Software Development Process: CLASP and SDL Compared", Third International Workshop on Software Engineering for Secure Sistems, 1-1, Minneapolis/Minesota/USA, May.

[7] Web 2, (2020), http://www.bilgiguvenligi.gov.tr/teknik-yazilar-kategorisi/guvenliyazilim-gelistirme-modelleri.html, (Erişim tarihi: 12/01/2020).

[8] Sultan K., En-Nouaary A., Hamou-Lhadi A., (2008), "Catalog of Metrics for Assesing Security Risks of Software throughout the Software Development Life Cycle", International Conference on Information Security and Assurance, 461-465, Busan/Korea, April.

[9] Chowdhury I., Chan B., Zulkernine M., (2008), "Security metrics for source code structures", In Proceedings of the Fourth International Workshop on Software Engineering for Secure Sistems (ICSE'08), 57-64, Leipzig/Germany, May.

[10] Siddiqui S.,(2015) "Comparative Study and Design of Software Requirement Tools for Secure Software Development", Ph. D Thesis, Department of Computer Science, A. M. U.

[11] Hadavi M. A., Sangchi H. M., Hamishagi V. S., Shirazi H., "Software security: A vulnerability-activity revisit", (2008), In Proceedings of the 2008 Third International Conference on Availability, Reliability and Security(ARES'08), 866-872, Barcelona/Spain, March.

[12] Ahmed S., (2007), "Secure software development - Identification of security activities and their integration in software development lifecycle", Master's

Thesis, School of Engineering, Blekinge Institute of Technology.

[13] Howard M., (2006), "A Process of performing security code reviews", IEEE Security & Privacy Magazine, 2006(4), 74-79.

[14] Whittaker J., (2003) "Why secure applications are difficult to write", IEEE Security & Privacy Magazine, IEEE Computer Society, 2003(1), 81-83.

[15] Daud M., (2010), "Secure software development model: A guide for secure software life cycle", In Proceedings of the International MultiConference of Engineers and Computer Scientists (IMESC'10), 2010(1), 724-728.

[16] Jain S., Ingle M., (2011), "Review of security metrics in software development process", International Journal of Computer Science and Information Technologies, 2011(2), 2627-2631.

[17] Caldiera G., Basili V., Rombach H., (1994), "Encyclopedia of software engineering", Second Edition, John Wiley & Sons.

[18] Web 3, (2021), http://www.standishgroup.com/sample_research_files/chaos_report_1994.pdf, (Erişim Tarihi: 15/03/2021).

[19] Web 4, (2022), https://www.darkreading.com/risk/the-four-big-problems-with-security-metrics, (Erişim Tarihi: 10/01/2022).

[20] Sampada G., Sake T., Amrita (2021), "Smart Computing", First Edition, CRC Press.

[21] Jain S., Ingle M., (2014), "Security Metrics and Software Development Progression", International Journal of Engineering Research and Applications, 2014(5), 161-167.

[22] Sahinoglu M., Stockton S., Morton S., Vasudev P., Eryilmaz M., (2014), "Metrics to Access and Manage Software Application Security Risk", International Conference Security and Management, 2014(14), 275-282.

ÖZGEÇMİŞ

Ali Gökşen 2010 yılında başladığı Dokuz Eylül Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümünü 2016 yılında başarıyla tamamlayarak özel sektörde yazılım geliştirme uzmanı olarak çalışmaya başladı. 2018 yılında yüksek lisans eğitimine Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında başladı. 2018 yılından bu yana Softtech'te yazılım mühendisi olarak çalışmaktadır.