EDİTÖR

Doç. Dr. Fulya ASLAY

ARALIK
2024

İmtiyaz Sahibi / Yaşar Hız
Yayına Hazırlayan / Gece Kitaplığı
Birinci Basım / Aralık 2024 - Ankara
ISBN / 978-625-388-131-3

Gece Kitaplığı
Kızılay Mah. Fevzi Çakmak 1. Sokak
Ümit Apt No: 22/A Çankaya/ANKARA
0312 384 80 40
www.gecekitapligi.com / gecekitapligi@gmail.com

Baskı & Cilt
Bizim Büro
Sertifika No: 42488

BİLGİSAYAR MÜHENDİSLİĞİ ALANINDA ARAŞTIRMALAR VE DEĞERLENDİRMELER

EDİTÖR

Doç. Dr. Fulya ASLAY

gece kitaplığı

İÇİNDEKİLER

BÖLÜM 1

ÇOCUKLAR İÇİN YAPAY ZEKÂ EĞİTİM PLATFORMU

Emin BORANDAĞ[1]

1 Doç .Dr., Department of Software Engineering, Hasan Ferdi Turgutlu Technology Faculty, Manisa Celal Bayar University, Manisa 45400, Turkey; emin.borandag@cbu.edu.tr, Ordid No: 0000-0001-5553-2707

1.GİRİŞ

Son yıllarda gelişen teknoloji ile, yapay zekâ uygulamalarının pek çok farklı alanda kullanıldığı görülmektedir. Bu yapay zekâ temelli uygulamaların pek çoğunun geliştirilmesinde, yapay zekânın bir alt alanı olan makine öğrenim algoritmaları kullanılmaktadır. Makine öğrenimi geniş bir uygulama sahasına sahiptir. Sağlık alanından, finans alanına, lojistik, otomotiv, tarım, e-ticaret ve eğitim gibi pek çok farklı alanda, giderek daha fazla bir oranda kullanılmaktadır (Jordan vd.,2015)

Yapay zekanın özellikle eğitim alanındaki uygulamalarına bakacak olursak; kişiselleşmiş öğrenme, otomatik değerlendirme sistemleri ve akıllı eğitim asistanları gibi geliştirilen uygulamaları görebiliriz. Bu uygulamalar öğrencilerin çalıştıkları alandaki bilgi ve becerilerini arttırmayı amaçlayan, analitik yöntemler kullanan aplikasyonlardır. Yapay zekâ destekli bu araçlar, öğrencinin daha kısa sürede güçlü ve zayıf yönlerini belirlemesini amaçlamaktadır (Luckin vd.,2015)

Yaşadığımız çağda çocuklar teknoloji ile iç içe büyümektedir. Bununla beraber günümüzde yapay zekâ ve makine öğrenmesi uygulamaları da hayatımızın birçok alanında kullanılmaktadır. Bu sistemlere örnek olarak; yüz tanıma sistemleri, plaka tanıma sistemleri, çeşitli görüntü işleme sistemleri ve fiyat tahmin algoritmaları verilebilir. Yapay zeka sistemlerinin bir alt alanı olan makine öğrenmesi adı geçen bu sistemlerin geliştirilmesinde etkin rol oynamaktadır. Yapay zekâ, günümüzde hemen hemen bütün alanlara nufus etmiş ve o sahalarda kullanımını arttırmaktadır (Geron,2012).

Teknolojinin gelişim hızının artmasından en çok etkilenen kesimler genellikle çocuklar olmaktadır. Günümüzde okul yada sınıf kavramını sanal eğitim araçları ve bireysel eğitim yazılımları almaya başlamıştır (Holmes vd., 2019). Bu bağlamda geliştirilen bu bölümde; çocukların gelişen yeniliklere uyum sağlaması ve oyunlarla birlikte gelişen teknolojinin bir parçası olan makine öğrenim kavramlarını gençlere aşılamak amaçlanmaktadır. Çocukların sevebilecekleri bir platformda AI ve makine öğrenmesi gibi konular ile tanışması ve teorik bilgileri metotlu bir şekilde tekrarlayarak konuyu pekiştirmeleri hedeflenmektedir.

Bu bölümde, geliştirilen bir proje ile çocuklara makine öğrenmesinin mantığını anlatmak için bir platform geliştirilmiştir. Geliştirilen bu platformun amacı; çocukların makine öğreniminin mantığını eğlenceli bir şekilde öğrenmesidir. Makine öğreniminin mantığının eğlenceli bir şekilde çocuklara anlatılması için geliştirilen eğitici ve öğretici özelliği olan platformun içerisinde 10 farklı oyun bulunmaktadır.

Çalışmanın ikinci bölümünde literatür taraması yapılmış, üçüncü bölümde geliştirilen platform ve platform içerisindeki uygulamalara yer ve-

rilmiştir. Son bölüm olan dördüncü bölümde ise çalışma sonucunda elde edilen bilgilere yer verilmiştir.

2. LİTERATÜR

Son yıllarda yapay zekânın önemi oldukça artmıştır. Yapay zekâ alanında pek çok farklı teknolojiler geliştirilmiş ve geliştirilmeye de devam edilmektedir. Literatüre bakıldığında oyun oynayarak yazılım mantığını öğreten çeşitli çalışmalar bulunmaktadır. Örnek olarak Code.Org sitesinde 83 milyonun üzerinde yapay zekâ ve makine öğrenmesi dâhil olmak üzere eğitsel birçok uygulama bulunmaktadır (Wilson vd.,2010). Bu alanda geliştirilen önemli teknolojilerden biri de ML.NET teknolojisidir. ML.NET Microsoft'un geliştirmiş olduğu açık kaynak kodlu bir makine öğrenmesi kütüphanesidir (Milford,2018). Bir diğer uygulama ise Code Combat uygulamasıdır. Bu uygulama ile çevrimiçi programlamayı öğrenebileceğiniz bir alt yapıya sahiptir (Wilson vd.,2010). Bir başka yapı ise Scratch uygulamasıdır. Scratch ile pek çok farklı uygulama oluşturulabilir ve bu uygulamalar herkesle paylaşılabilmektedir. Özellikle 8 ila 15 yaş aralığındaki bireylerin rahatlıkla kullanabilecekleri pek çok uygulamayı geliştirmesine olanak sağlamaktadır. Scratch diğer kod öğrenimi platformlarından farkı ise kullanan kişinin hiçbir kod satırı yazmadan, Lego'lara benzeyen Scratch bloklarını ile uygulamayı geliştirmesidir (Resnick,2009). Geliştirilen bu uygulamaların temel amacı genel olarak çocuklara bilgisayar programcılığı ile ilgili bilgiler vermektir. Konu ile ilgili yol gösterici niteliğinde bulunan Dr. Michael Milford ve Dr. Dhoot De'nin yazdığı kitaplarda da yapay zeka ve makine öğreniminin erken yaşlarda çocuklara anlatılmasının öneminden söz edilmektedir (Dhoot,2020).

3. Eğitim Platformu

Son yıllarda hem ülkemizde hem de Dünya genelinde, makine öğrenmesi ve yapay zekâ alanına ilişkin büyük bir farkındalık oluşmuştur. Bu doğrultuda eğitim bakanlıkları alana özgü bir farkındalık ile daha fazla yazılımcı yetiştirmek için çocukların yazılıma olan ilgisini makul olan en küçük yaştan başlatarak arttırmaya çalışmaktadır. Bu amaçla çalışmaya konu olan platformun temel amacı; özellikle makine öğrenmesi ve yapay zekâ kavramlarının çocuklar tarafından eğlenerek iyice kavranmasıdır.

3.1 Geliştirilen Uygulamaların Kapsam ve Kullanımı

Projenin kapsamı ve kullanımı 3 adımdan oluşmaktadır. Bu adımlar aşağıda listelenmiştir.

Programın açılıp istenilen oyunun seçilmesi: Kısa bir program açılma animasyonundan sonra oyunların olduğu ana menü kullanıcıyı karşılamaktadır. Bu menüde kullanıcı için 10 adet oyunun kapak resimlerini görüntülenmektedir. Kullanıcı istediği bir oyunu seçerek giriş yapabilir.

Eğitim aşaması: Kullanıcı ana menüde oyunu seçtikten sonra, oyun içinde ilk olarak, eğitim aşaması ile karşı karşıya kalmaktadır. Burada kullanıcıdan oyun aşamasına geçebilmek için modeli eğitmesi beklenmektedir. Model eğitme kısmı ise kullanıcının seçtiği oyuna göre değişiklik göstermektedir. Her oyunun kendine özgün bir eğitim aşaması vardır. Bazı oyunlarda kullanıcıdan klavye yoluyla veri girmesi istenirken bazı oyunlarda ise resim yâda ses verisi girmesi beklenmektedir. Kullanıcı gerekli bilgileri girdikten ya da yükledikten sonra model eğitimini tamamlanır.

Test aşaması: Kullanıcı test aşamasında ise eğitim aşamasındakine benzer bir yapıya sahiptir. Kullanıcıdan eğittiği modelin test etmesi için bilgi girmesi ya da yüklemesi beklenmektedir. Kullanıcı bilgileri girdikten ya da yükledikten sonra uygulama eğitim aşamasında öğrendiği verilere göre çalışır.

3.2. Gereksinimler

Geliştirilen platformun işlevsel ve işlevsel olmayan gereksinimleri aşağıda listelenmiştir.

İşlevsel Gereksinimler: Yazılımın nasıl üretileceğinin belirlenip, yazılıma ait ürününe ilişkin fonksiyonların çıkarılması olarak tanımlanan işlevsel gereksinimler, geliştirilen proje bazlı olarak aşağıda listelenmiştir.

- Kullanıcı sistem içerisinde 10 farklı oyunu görüntüleyebilecektir.
- Kullanıcı 10 farklı oyun içerisinden istediğini seçip oynayabilecektir.
- Kullanıcıların oyunları görüntüleyebilmesi veya oynayabilmesi için üye olması ya da giriş yapması gerekmemektedir.

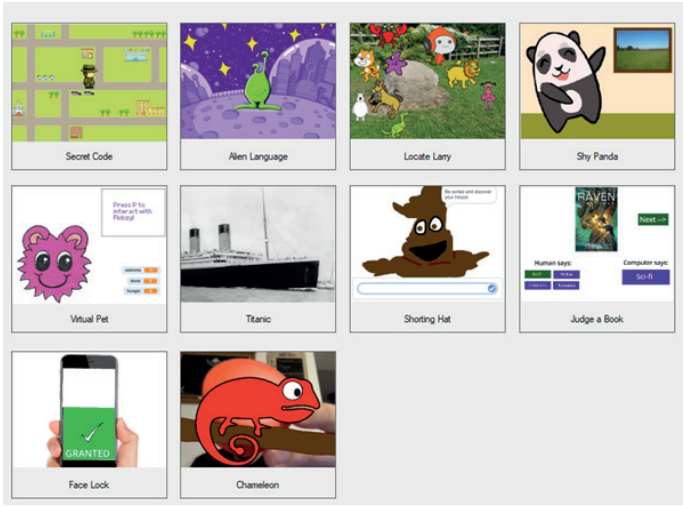
İşlevsel Olmayan Gereksinimler: Yazılımın özellikle performans ve güvenlik gibi spesifik özelliklerini belirler. Bu kapsamda maddeler halinde şu gereksinimler belirlenmiştir;

- Yazılım tüm web tarayıcılarında çalışabilecektir.
- Sistem 7/24 çalışabilecektir.
- Sistemin görünüm kısımları C# kullanılacaktır.
- Sistemin veri tabanı MySQL olacaktır ve makine öğrenimi algoritmaları için ML.NET kullanılacaktır.
- Sistem aynı anda 100 adet kullanıcıyı desteklemektedir.

3.3 Ana Uygulama Ekranı

Öğretmenin en iyi yolu bir konuyu oyunlaştırarak aktarmaktır (Deterding vd.,2011) (Hamari, vd.,2014). Günümüzde birçok kişinin tanımı-

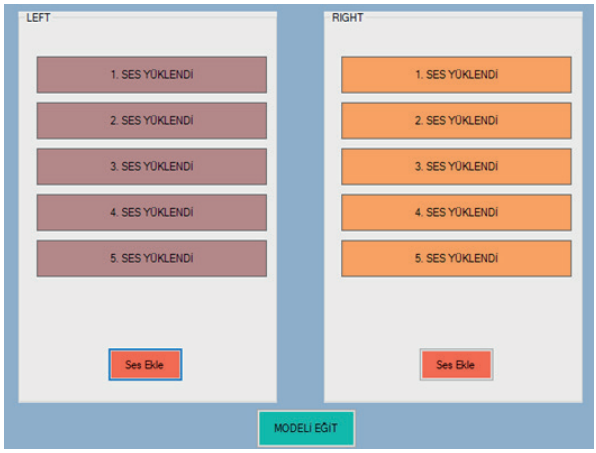
nı anlamakta zorlandığı geleceğin teknolojisi olan yapay zekâ ve makine öğrenmesine yönelik oyun sistemi ile çocuklara bu bilgilerin aşılanması düşüncesiyle geliştirilen platformun ana sayfası Şekil-1'de gösterilmektedir. Platformun, Bölüm 3.2 içerisindeki işlevsel ve işlevsel olmayan gereksinimlere göre geliştirilmiştir. Geliştirilen uygulama içerisinde yer alan birbirinden farklı oyunlar sayesinde çocukların yapay zekâ ve makine öğrenmesi gibi konuları pekiştirmesi hedeflenmektedir.



Şekil-1 Yapay Zekâ Eğitim Platformu Ana Ekranı

Geliştirilen yapay zekâ platformun 5 oyununa ilişkin detaylı bilgiler aşağıda verilmiştir.

Alien Language: Kullanıcın verdiği sağ ve sol yön komutlarına göre hareket eden bir oyundur. Kullanıcı ana sayfada Alien Language oyuna tıkladığında Şekil-2'de göründüğü gibi bir form ekranı açılır. Bu sayfada 'Ses Ekle' butonuna tıklayarak 'Sağa Git' ve 'Sola Git' etiketlerine, kullanıcının seslerini kaydeder. Ses kaydetme işlemi tamamladığında model oluşturulur ve oluşan modelden 'Modeli Eğit' butona tıklayarak eğitilir.

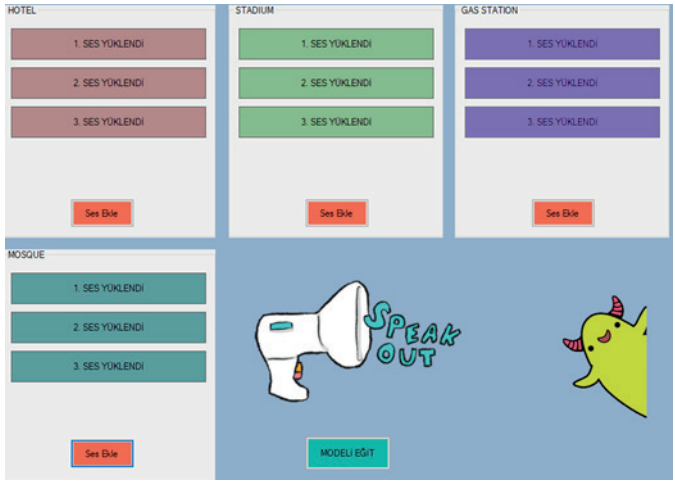


Şekil-2 Alien Language Model Eğitimi

Model eğitimi tamamlandıktan sonra Şekil-3'teki ekran kullanıcı karşılar. Kullanıcı Şekil-3'teki mikrofona tıklayarak sesli komutlarını verir ve komutlara göre karakter sağ yada sola hareket eder.

Şekil-3 Alien Language Oyunu

Secret Code: Kullanıcın verdiği yön komutlarına göre hareket eden bir oyundur. Kullanıcı Ana sayfada Secret Code oyuna tıkladığında Şekil-4'de göründüğü gibi 'Ses Ekle' butonuna tıklayarak 'Hotel', 'Stadyum', 'Benzinlik' ve 'Cami' etiketlerine seslerini kaydeder. Ses kaydetme işlemi tamamladığında model oluşturulur ve oluşan modelden 'Modeli Eğit' butona tıklanarak model eğitilir.



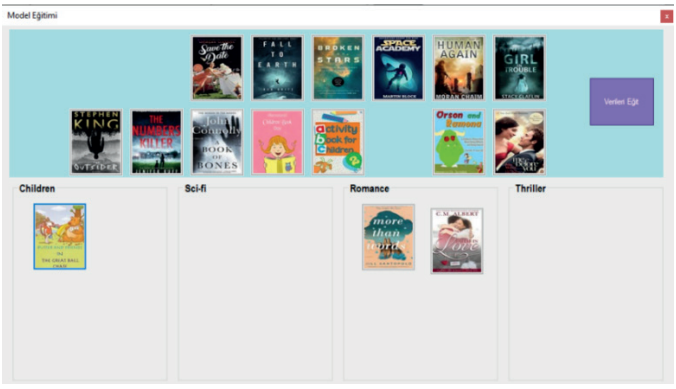
Şekil-4 Secret Code Model Eğitimi

Model eğitimi tamamlandıktan sonra kullanıcı Şekil-5'deki 'Oyuna Başla' butonuna tıklayarak oyuna başlar. Butona tıkladıktan sonra Şekil-5'daki ekran kullanıcı karşılar. Kullanıcı mikrofona tıklayarak sesli komutlarını verir ve komutlara göre karakter hareket eder.

Şekil-5 Secret Code

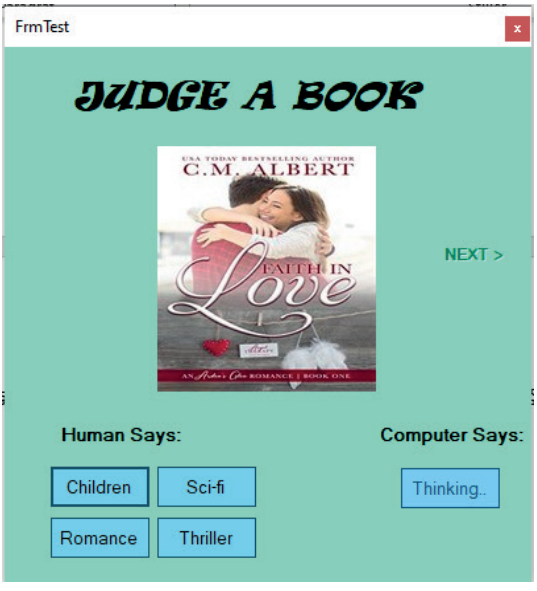
Judge a Book: Kitap resimlerinden kitabın türünün tahmin edildiği bir oyundur. İlk olarak oyuna başlamadan önce kitap resimleri ile bir model oluşturulur. Daha sonra oyun alanında bu kitapların türleri oyuncular ve bilgisayar tarafından tahmin edilir. Kullanıcı ana sayfa üzerinde Judge a Book oyununa tıkladığı zaman Şekil-6'daki form sayfası gelecektir. Bura-

da Eğitim ve Test işlemi sayfalarına ulaşmak için iki ayrı buton bulunmaktadır. Bu sayfa üzerinde başla butonuna basınca, kullanıcının önüme kitap resimleri gelecektir. Kullanıcı bu kitap resimlerini tür isimlerinin bulunduğu alanlara sürükleyip bıraktıktan sonra verileri eğit butonuna basınca eğitim işlemi tamamlanacaktır.



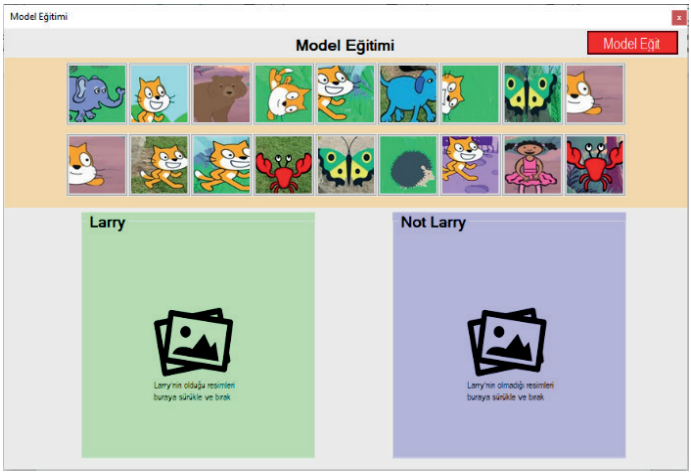
Şekil-6 Judge a Book Model Eğitimi

Daha sonra Test işlemini gerçekleştirmek için Test butonuna tıkladığımızda karşımıza Şekil-7'deki ekran gelecektir. Bu sayfada da oyuncu ve bilgisayar resimdeki kitabın türünü tahmin edecektir.



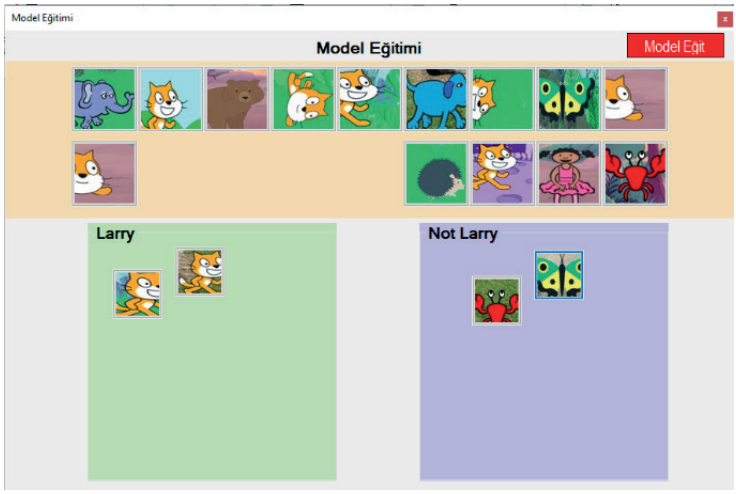
Şekil-7 Judge a Book

Locate Larry: Larry karakterinin, resimler üzerinde bulunarak işaretlendiği bir oyundur. İlk olarak oyuna başlamadan önce Larry karakterinin bulunduğu ve bulunmadığı resimler ile bir model oluşturulur. Makine öğretimi ile eğitilen bu model sayesinde bilgisayar Larry karakterini bulur. Kullanıcı ana sayfa üzerinde Locate Larry oyununa tıkladığı zaman Şekil-8'deki form sayfası ekrana gelecektir. Burada eğitim işlemi ve test işlemi sayfalarına ulaşmak için iki ayrı buton bulunmaktadır.



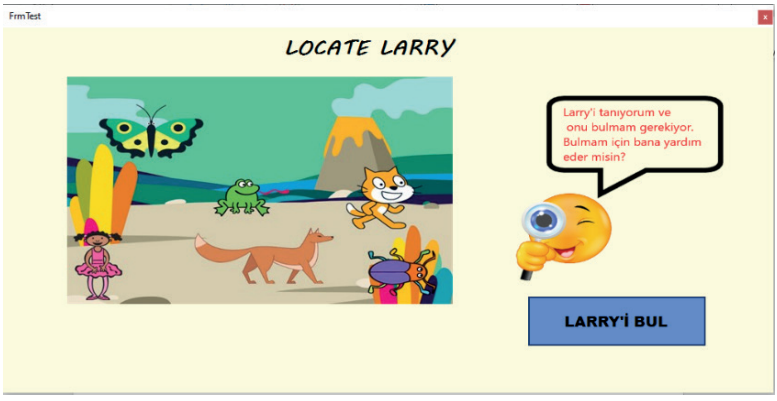
Şekil 8 Locate Larry Model Oluşturma

Eğitim butonuna tıkladığımızda model oluşturma kısmı ekrana gelecektir. Bu sayfa üzerindeki Larry'nin bulunduğu resimler ve bulunmadığı resimler ilgili alanlara sürüklenip bırakıldıktan sonra modeli eğit butonuna tıklanarak model oluşturma işlemi gerçekleştirilecektir.



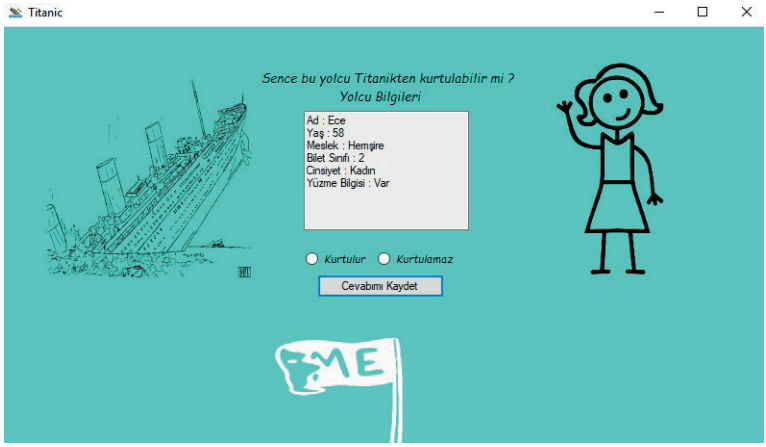
Şekil 9 Locate Larry Model Eğitimi

Daha sonra Test işlemini gerçekleştirmek için Test butonuna tıklandığında Şekil-10'deki sayfa ekrana gelecektir. Bu sayfa da Larry'i Bul butonuna tıklanınca resim üzerinde Larry'nin bulunduğu alan bilgisayar tarafından işaretlenecektir.



Şekil 10 Locate Larry

Titanic: Bilgisayara verilen bilgilere göre kişilerin yaşayıp yaşamayacağını tahmin etmesini isteyen bir oyundur. Kullanıcı Ana sayfada Titanic oyuna tıkladığında, Model eğitimi sayfası açılacaktır. Model eğitimi tamamladıktan sonra Şekil-11'deki gibi oyun başlanır. Kullanıcı verilen bilgilere göre Titanic gemisinde kimlerin yaşayacağını, makine öğrenim algoritmalarını kullanan bilgisayar vasıtası ile tahmin etmeye çalışır.



Şekil 11 Titanic Oyunu

4. Sonuçlar

Gelişen teknolojiyle birlikte yapay zekâ ve makine öğrenmesi gibi konular artık popüler hale gelmiş ve pek çok kişi bu alana yönelmeye başlamıştır. Bu alanların yeni çıkmış olmasından dolayı küresel çapta büyük bir pazar olmaya başlamıştır. Bu doğrultuda bu teknolojisinin öneminin farkında olan aileler hatta devletler bu sektöre yatırım yapmak istemektedir. Maalesef yapay zekâ ve makine öğrenmesi uygulamaların ve oyunların çocuklar için yetersiz sayıda bulunduğu yapılan literatür çalışmasında görülmüştür. Bu sorun doğrultusunda birçok makine öğrenme uygulamalarının bir arada bulunduğu, eğlenceli ve özgün bir makine öğretimi platformu geliştirilmiştir. Geliştirilen program kolay kullanımı sayesinde makine öğrenmesinin temellerini kullanıcılara eğlenceli bir şekilde öğretmeyi amaçlamıştır. Geliştirilen programın özellikleri aşağıda liste halinde verilmiştir;

- Çocukların sistemde bulunan oyunlardan ilgisini ve merakını çekenleri gözlemleyip çalıştırabilir.
- Çalışan uygulama içinde eğitim kısmında ilgili bilgileri sisteme yükleyebilir.
- Eğitim kısmından sonra sisteme girilen bilgiler vasıtasıyla test işlemi gerçekleştirilebilir.
- Programı kullanan çocuk için görsel ve yaşına uygun bir şekilde bilgilendirme sağlanmaktadır.
- Türkiye'de çocuklara yönelik yapılan faaliyetler genelde algoritma, kodlama, robotik ve tasarım alanında eğitimleri içermektedir burada ise yapay zekâ ve makine öğrenimini çocukların dikkatini çekebilecek oyunlarla öğretmek amaçlanmıştır.

- Çocuklar bu oyunları oynarken veri toplama, toplanan verileri bilgisayara öğretme ve bilgisayarı test etme yöntemleri hakkında bilgi sahibi olacaklar.

- Geliştirilen platform, çocuklara analitik düşünme ve problem çözme becerilerinin yanı sıra analitik uygulama geliştirmeyi de amaçlamaktadır.

Referanslar

Aurelien Geron, “Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems”, 2017

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: Defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments* (pp. 9-15). ACM. https://doi.org/10.1145/2181037.2181040

Dhoot “Supervised Machine Learning for Kids - Tinker Toddlers “ , 2020

Hamari, J., Koivisto, J., & Sarsa, H. (2014). Does gamification work?--A literature review of empirical studies on gamification. In *2014 47th Hawaii International Conference on System Sciences* (pp. 3025-3034). IEEE. https://doi.org/10.1109/HICSS.2014.377

Holmes, W., Bialik, M., & Fadel, C. (2019). *Artificial intelligence in education: Promises and implications for teaching and learning*. Center for Curriculum Redesign.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science, 349*(6245), 255-260. https://doi.org/10.1126/science.aaa8415

Luckin, R., Holmes, W., Griffiths, M., & Forcier, L. B. (2016). Intelligence unleashed: An argument for AI in education. Pearson Education. https://doi.org/10.13140/RG.2.2.24094.43842

Michael Milford , “The Complete Guide to Artificial Intelligence for Kids” , 2018

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60-67. https://doi.org/10.1145/1592761.1592779

Wilson, C., Sudol, L. A., Stephenson, C., & Stehlik, M. (2010). Running on empty: The failure to teach K-12 computer science in the digital age. *Association for Computing Machinery (ACM)*. https://doi.org/10.1145/1868358.1868368

BÖLÜM 2

SİBER GÜVENLİK AÇIKLARI VE EKONOMİK BOYUTLARI

Didem İŞSEVER[1]

1 DR. PTT Bilgi Teknolojileri Daire Başkanlığı ORCID NO:0000-0003-0226-1442

GİRİŞ

Siber güvenlik, hızla gelişen teknolojilerin ışığında, dijitalleşme çağında bireylerin, organizasyonların ve bilgilerin korunmasında temel bir rol oynamaktadır. İnternet kullanıcılarının günden güne artış göstermesi, ağların daha hızlı ve geniş hale gelmesi bilgi güvenliği tehditlerinin çeşitlenmesine ve yoğunlaşmasına neden olmuştur. Siber güvenlik yalnızca kişisel verilerin ve finansal bilgilerin korunmasını değil, aynı zamanda kritik altyapıların, ticari operasyonların ve kamusal faaliyetlerin de korunmasını ve sürekliliğini sağlar. Veriler ve sistemler üzerinde meydana gelebilecek potansiyel tehditlere karşı koruma sağlayarak, güvenli bir dijital ortam oluşturmayı hedefler.

Siber güvenlik, kötü niyetli saldırıların önlenmesine ve sistemlerin olası zayıf noktaların tespit edilmesine de katkıda bulunur. Bu sebeple siber güvenlik, günümüz dijital dünyasında yalnızca kişisel gizlilik değil, aynı zamanda toplumsal düzenin ve ekonomik hakların korunması için de kritik öneme sahiptir. Bu bağlamda, siber güvenlik açıklarının ekonomik, bireysel ve kurumsal maliyetlerinin yanı sıra makroekonomik istikrar içinde önemli etkileri bulunmaktadır. Siber saldırıların bankacılık sektörü üzerindeki operasyonel maliyetleri önemli oranda artırdığı ve finansal performansı düşürdüğü belirtilmektedir (Karasoy ve Babaoğlu, 2021).

Uluslararası Para Fonu'nun (IMF) raporlarına göre, siber güvenlik ihlallerinin küresel finans sektörü üzerindeki yıllık maliyeti 90 milyarın üstünde olmaktadır, bu da sektördeki net kârın %9'una tekabül etmektedir. Siber güvenlik eğitiminin yetersizliğine ve yetenek açığına ilişkin kapsamlı çalışmalar ile bu durumun siber risk yönetimini zorlaştığı ifade edilmektedir (Afyonluoğlu, 2020). Artan siber tehditlere karşı yapılan teknoloji yatırımları, azalan getiri yasası gereği ekonomik sürdürülebilirlik açısından sorgulanabilir hale gelmiştir. Sonuç olarak, siber güvenlik açıklarının yarattığı ekonomik ve toplumsal etkiler, yalnızca bireysel veya kurumsal düzeyde değil, aynı zamanda finansal sistemlerin bütünlüğü ve ulusal güvenlik politikalarının sürdürülebilirliği açısından da kritik önem taşımaktadır. Bu nedenle, etkili siber güvenlik stratejilerinin geliştirilmesi, risk yönetimi politikalarının güçlendirilmesi ve uluslararası iş birliklerinin artırılması gerekmektedir.

SİBER GÜVENLİK KAVRAMI

Amerika Birleşik Devletleri Savunma Bakanlığı' na bağlı özel bir birim tarafından bulunan dijital iletişim teknolojisi internetin, 1960' lı yıllarda yaygın biçimde ülkelerce kullanılmaya başlamasıyla ortaya çıkan bilgi yoğunluğunu hedef alan çeşitli tehditler, siber güvenliğin daha ciddi bir kavram olduğunu ve ülkeler için sürdürülmesinin kaçınılmaz bir gereklilik olduğunu ortaya koymuştur (Karasoy ve Babaoğlu, 2021; Afyonluoğlu, 2020).

Siber güvenlik bilgisayar kullanıcıları aracılığıyla ağ sisteminde bulunan tehditlere dikkat çekebilmek için kullanılan bir kavram olup, dijital iletişim teknolojilerinden ortaya çıkan yoğun saldırıların zarar verici etkileri olabileceği görüşü ağır basmaya başlayınca teknik bir kavram olmanın ötesine geçmeyi başarmıştır (Hansen ve Nissenbaum, 2009). Siber suçlular bir bilgisayar ve internet bağlantısının ötesinde sadece düşük maliyetle bu saldırıları gerçekleştirebilmektedirler. Sınırları olmamasından dolayı kimliklerini tespit etmek ve takip etmek oldukça zordur. Bu göz önüne alındığında bilgi teknolojisi sistemlerine yönelik saldırılar çok cazip olduğundan, bu tür saldırıların sayısının ve karmaşıklığının gün geçtikçe artmaya devam edeceği öngörülmektedir (Jang-Jaccard ve Nepal, 2014).

Siber güvenliğin ilk örneklerinden biri, 1971'de ARPANET üzerinde yayılan Creeper virüsüdür. Bu virüs, kendini kopyalayarak diğer sistemlere giriş yaparak mesaj bırakmakta olup bu virüse karşı ise ilk anti-virüs programı olarak Reaper, kullanılmıştır. 1983'te ABD'de RSA ortak anahtar şifreleme denemesi için siber güvenlik patenti alınmış olup 1986'da ise izinsiz olarak başka bilgisayarlara erişim sağlayanlar için hapis cezası içeren bir yasa çıkarılmıştır. 1990'lı yıllarda internet ve web tarayıcılarının yaygınlaşmasıyla korsan e-posta ve web siteleri üzerinden sistemlere zarar vermeye başlamışlardır. 1994'te SSL protokolü geliştirilmiş, 2000'lerde devletler arası siber savaşlar yaşanmış ve farklı ülkelerdeki korsan gruplar, büyük şirketlere yönelik siber saldırılar gerçekleştirmişlerdir (Nasuh B. Karadağ, 2024).

Siber güvenlik, dijital sistemleri, ağları, cihazları ve verileri yetkisiz erişim, saldırı veya hasardan korumak amacıyla uygulanan bir dizi teknik, stratejik bir süreç olarak tanımlanabilir. Siber güvenlik, gizlilik, bütünlük ve erişilebilirlik ilkeleri ile bilginin yetkisiz erişime karşı korunmasına, doğruluğunun garanti altına alınmasına ve yetkili kullanıcıların zamanında erişiminin sağlanmasına hizmet eder. Finansal sistemler ve kritik altyapılar gibi alanlarda operasyonel risklerin azaltılması ve istikrarın sağlanmasında siber güvenliğin önemi gözardı edilmemelidir. Bu, teknik çözümlerin yanı sıra sosyal, ekonomik ve kurumsal boyutlarını da dikkate alan çok disiplinli bir yaklaşım gerektiren çok yönlü bir konudur (Karasoy ve Babaoğlu, 2021). Artan siber tehditler ışığında, bireyler ve kurumlar arasında farkındalık yaratmak, bir güvenlik kültürünü teşvik etmek ve eğitim programlarının yalnızca teknik becerileri değil, aynı zamanda etik hususları ve kriz yönetimini de kapsamasını sağlamak büyük önem taşımaktadır (Afyonluoğlu, 2020; Hansen ve Nissenbaum, 2009).

Siber güvenliğin sağlanması amacıyla uygulanan en önemli bileşenler aşağıdaki gibi tanımlanabilmektedir. Siber güvenliğin ana hedefleri, bilginin erişilebilirliğini, doğruluğunu, bütünlüğünü ve gizliliğini sağlamaktır. Ayrıca, bilginin reddedilemezliğini de güvence altına almaktır (Çakır ve

Uzun, 2021; Ünver, Canbay, ve Mirzaoğlu, 2018).

Gizlilik: Dijital olarak üretilen verilere yalnızca yetkili kişiler ve sistemler tarafından erişilebilmesini sağlar. Bu, verilerin yetkisiz erişimden korunarak gizliliğinin ve güvenliğinin sağlanması anlamına gelir. Gizlilik, şifreleme yöntemleri, erişim kontrol mekanizmaları ve yetkilendirme protokolleri kullanılarak sağlanır.

Bütünlük: Verinin kaynağından alıcısına dış etkilere maruz kalmadan ve içeriği yetkisiz kişiler tarafından bozulmadan veya değiştirilmeden aktarılmasını sağlar. Bütünlük, verilerin güvenilirliğini, doğruluğunu ve tutarlılığını korur. Bütünlük ihlalleri siber saldırılar, kötü amaçlı yazılımlar veya teknik hatalar sonucunda meydana gelebilir ve verilerin doğruluğunu tehdit eder.

Erişilebilirlik: Bu, yetkili kullanıcıların ve sistemlerin ihtiyaç duydukları verilere, ihtiyaç duydukları zamanda, ihtiyaç duydukları kalitede ve sürede erişebilmelerini ifade eder. Erişilebilirlik, sistem sürekliliğinin sağlanması ve hizmet kesintilerinin önlenmesi açısından kritik önem taşır.

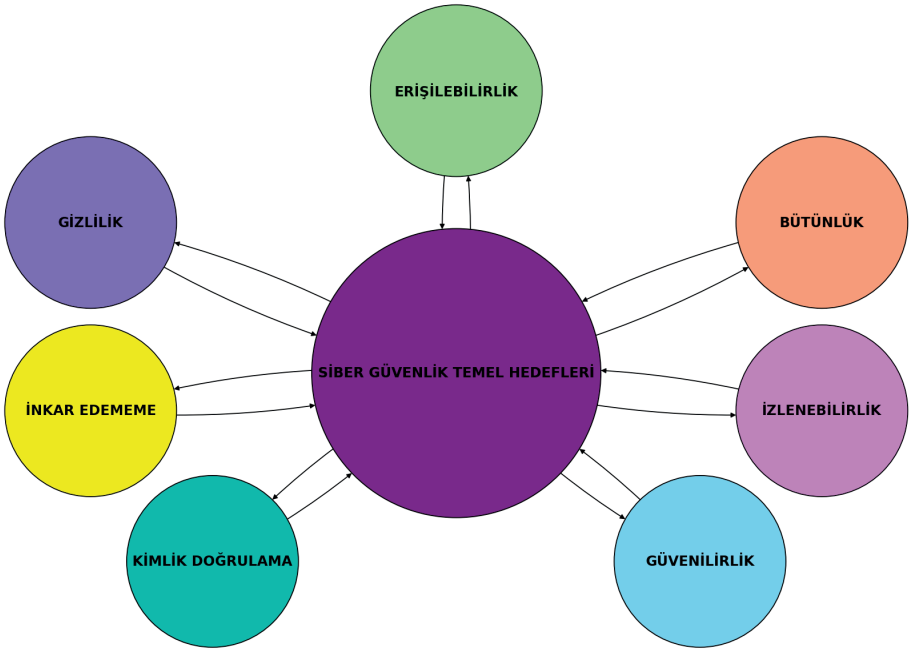
İzlenebilirlik: Tüm sistem olaylarının, kullanıcı işlemlerinin ve erişim faaliyetlerinin kaydedilmesini ve gerektiğinde analiz edilmek üzere saklanmasını sağlar. Bu özellik sistem güvenliği için önemlidir ve olayları tespit etmek, günlük kayıtlarını analiz etmek ve olası ihlalleri belirlemek için kullanılır.

Kimlik Doğrulama: Bir kullanıcının, sistemin veya veri göndericisinin kimliğinin güvenilir bir şekilde doğrulanmasını sağlar. Bu, iletişim veya işlem taraflarının gerçekliğini garanti ederek yetkisiz erişim riskini ortadan kaldırır.

Güvenilirlik: Sistemin tasarlandığı ve amaçlandığı gibi çalışması, dış tehditlerden korunması ve beklenen performansı sunması anlamına gelir. Güvenilirlik, donanım, yazılım ve süreçlerinin istikrarlı çalışmasını garanti eder ve sistem arızalarının en aza indirilmesini sağlar.

İnkar edememe: Gönderici tarafından iletilen bir mesajın veya verinin daha sonra reddedilmesini veya inkar edilmesini önler. Bu özellik iletişimin doğruluğunu ve güvenilirliğini sağlar.

Aşağıda siber güvenlik temel hedeflerine ait şekil görülmektedir.



Şekil 1. Siber Güvenlik Temel Hedefleri

Bir siber saldırıyla ilgili olan riskler genellikle üç güvenlik faktörünü içerir: Bunlar tehditler, zafiyetler ve etkilerdir. Tehditler, saldırıyı gerçekleştiren kişiler veya grupları ifade ederken zafiyetler, saldırıya açık olan zayıf noktaları; etkiler ise saldırının ortaya çıkardığı sonuçların veriler veya kullanıcılar üzerindeki olumsuz etkilerini ifade etmektedir (Fischer, 2014).

Siber güvenlik olayları, kuruluşların sistemleri, ağları veya bireysel cihazları üzerinde güvenlik risklerine neden olabilmektedir. Bu olaylar; dış veya iç tehditler, yazılım açıkları, kötü niyetli saldırılar veya insan hatalarından kaynaklanabilmektedir. Bu tür olaylar, kurumların iş sürekliliğini tehdit edebilmekte, verilerin güvenliğini tehlikeye atabilmekte ve kişisel veya ticari bilgilerin açığa çıkmasına yol açabilmektedir. Yetkisiz erişim, şifrelerin ele geçirilmesi, kötü amaçlı yazılımlar kullanılması veya fiziksel güvenlik açıklarından yararlanılması gibi çeşitli şekillerde gerçekleşebilir. Bu tür eylemler, organizasyonların güvenlik politikalarını ihlal eder ve potansiyel olarak ciddi zararlar doğurabilir (Sun vd., 2018). Siber güvenliği ihlal eden teknikler aşağıda sıralanmıştır.

Kötü amaçlı yazılım (malware): Kasıtlı olarak bir bilgisayar, istemci, sunucu veya bilgisayar ağına zarar vermek amacıyla tasarlanmış her

türlü program veya yazılımdır. Bu tür yazılımlar, örneğin botnetler gibi ağları hedef alarak zarar verebilmektedirler. Kötü amaçlı yazılımlar, bilgisayar virüsleri, solucanlar, truva atları, reklam yazılımları, fidye yazılımları, casus yazılımlar ve kötü amaçlı botlar gibi çeşitli türlerde olabilmektedir (Jang-Jaccard ve Nepal, 2014; McIntosh vd., 2019)

Hizmet reddi (Denial-of-Service, DoS): Bu saldırı türü, bir makineyi veya ağı hedef alarak, aşırı trafikle doldurup çökmesine neden olup, bu makine veya ağı, gerçek kullanıcıların erişimine kapatma amacı güden bir saldırı olarak değerlendirilmektedir (Sun vd., 2018).

Kimlik avı-oltalama (Phishing): Sosyal mühendislik türlerinden biri olup, insan etkileşimleri aracılığıyla gerçekleştirilen birçok kötü niyetli aktiviteyi içerir. Bu saldırılar, e-posta, SMS veya anlık mesajlaşma gibi elektronik iletişim araçlarıyla, güvenilir bir kişi ya da kurum gibi görünerek, banka ve kredi kartı bilgileri, giriş bilgileri veya kişisel kimlik bilgileri gibi hassas verileri çalmaya yönelik yapılan dolandırıcılık girişimleridir (Jang-Jaccard ve Nepal, 2014).

Şebeke trafiğinin dinlenmesi (Sniffing): Trafiğin dinlenmesi verinin yolunu engellemek olarak adlandırılabilir. Bu ihlalde ağdaki paketler yakalanabilmekte ve içeriği okunabilmektedir. Sniffing, yönlendiricilere ulaşan tüm paketleri yakalar ve bilgisayarlar arasındaki tüm dataların yakalanıp saklanmasını sağlar. Bu, en çok kullanılan korsan yöntemlerden birisidir. Bu tehditten korunmak için şifreli bağlantı kullanılması gerekmektedir (Arslan, 2016).

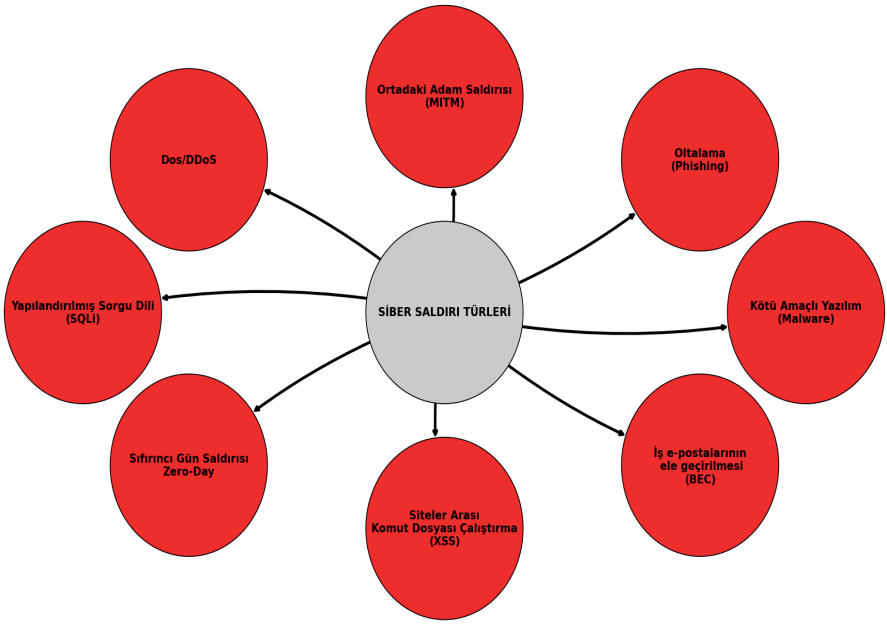
İstem dışı elektronik posta (Spam): Elektronik posta internet vasıtasıyla gönderilen elektronik iletilerdir. Ucuz ve kullanımı basit olmasından dolayı dünya genelinde her gün milyarlarca kez tercih edilmektedir. Spam e-posta ise istenmeyen, önemsiz ve gereksiz e-postaları ifade etmek için kullanılmaktadır. Birden çok alıcıya gönderilen reklam veya ilgisiz içeriğe sahip mesajlar anlamına da gelmektedir (Eryılmaz ve Kılıç, 2020).

Ortadaki Adam Saldırısı (MITM): Ortadaki adam saldırısı, saldırganın hedef kullanıcı ile kaynak arasındaki iletişim ağını gizlice ele geçirerek verilere erişim sağladığı ve bunları değiştirdiği bir saldırı türüdür. Bu saldırının önemli bir yönü, kullanıcının saldırganın varlığından tamamen habersiz olmasıdır. Kullanıcı doğrudan güvenli bir ağa bağlı olduğunu düşünse de tüm ağ trafiği aslında saldırganın kontrolü altındaki bir kanal üzerinden yönlendirmektedir. Sonuç olarak saldırgan, kullanıcı tarafından gönderilen verileri, oturum bilgilerini ve hassas kişisel verileri elde edebilir veya manipüle edebilir. Ağa bağlanırken, kullanıcı cihazı bilgi paketleri gönderir. Ağ, kullanıcının cihaz adresini ve bir şifreleme anahtarını içeren dijital bir sertifika oluşturur. Ancak, bağlantı sırasında kullanılan sertifika güvenilir değilse, saldırgan bu sertifikaya erişebilir ve kullanıcının izniyle

içeriğini değiştirebilir, böylece iletişimin kontrolünü ele geçirebilir (Acar ve Zoray, 2021).

Sosyal Mühendislik Saldırıları: Sosyal mühendislik, insan etkileşimlerindeki ve davranışlardaki açıkları kullanarak güvenlik önlemlerini aşma yöntemlerine verilen isimdir. Bu alanda kullanılan teknikler arasında hedef kişiye güvenilir biriymiş izlenimi verilirken aynı zamanda ortak kişiler aracılığıyla samimiyet oluşturarak, iletişim araçlarını kullanıp bir başkasıymış gibi pek çok ortamda hedef sistemden atılan şahsi verileri toplama esasına dayanan bir saldırı yöntemidir (Çakır ve Uzun, 2021).

Şekil 2'de yaygın olarak kullanılan siber saldırı türleri gösterilmiştir (Letstechiteasy, 2024).



Şekil 2.Siber Saldırı Türleri (Letstechiteasy, 2024).

SİBER GÜVENLİK AÇIKLARI

Siber güvenlik açıkları, yazılım hataları, yanlış yapılandırmalar, donanım problemleri ve insan hataları gibi pek çok teknik ve insan odaklı sebepten kaynaklanmaktadır. Bu güvenlik açıklarının doğru bir şekilde anlaşılması, organizasyonların etkili savunma stratejileri geliştirebilmesi ve hassas bilgilerini kötü niyetli saldırganlardan koruyabilmesi açısından kritik bir öneme sahiptir. Siber saldırıların daha yaygın ve karmaşık hale

gelmesiyle birlikte, bu tür güvenlik açıklarının analizi, modern siber güvenlik önlemlerinin en önemli bileşenlerinden biri olmuştur (The Cyber Express, 2024; Sprinto, 2024)

Siber güvenlik açıklarının ekonomik etkileri, geniş kapsamlı ve önemli sonuçlar doğurabilir. İhlaller, hırsızlık, dolandırıcılık ve maliyetli olay müdahaleleri gibi durumlarla ani mali kayıplara yol açabilir. Örneğin halka açık şirketler, bu tür ihlallerin ardından genellikle hisse senedi fiyatlarında ciddi düşüşler yaşamaktadır. Doğrudan mali etkilerin ötesinde kuruluşlar, tüketici güvenini zedeleyen, tedarik zincirlerini bozan ve genel ekonomik istikrarı olumsuz yönde etkileyen itibar kaybı yaşayabilmektedirler. Ayrıca küresel ekonomiye yılda trilyonlarca dolara mal olması beklenen siber suç tehdidi, bu güvenlik açıklarının ele alınmasının aciliyetini daha da vurgulamaktadır (Keeper Security, 2023; Federal Reserve, 2022; PTM Wealth, 2024; CISA, 2024). Siber güvenlik açıkları aşağıda belirtilen farklı türlere sahiptir.

Yazılım Hataları: Yazılım güvenlik açıkları genellikle kodlama hataları, zayıf tasarım tercihleri veya beklenmeyen kullanım senaryoları nedeniyle ortaya çıkmaktadır. Programlama sırasında yapılan yazım hataları veya mantıksal eksiklikler gibi basit hatalar, siber suçluların istismar edebileceği açıklıklar yaratabilmektedir. Yazılım mimarisindeki kusurlar ise güvenlik önlemlerinde boşluklar bırakarak yetkisiz erişimlere yol açabilmektedir. Ayrıca, dışa bağımlı kütüphaneler veya bileşenler kullanılması ve bu bileşenlerin yeterli güvenlik önlemlerine sahip olmaması durumunda ek güvenlik risklerini de beraberinde getirebilmektedir (The Cyber Express, 2024).

Donanım Hataları: Donanım hataları, ürünün donanımının nem, toz birikmesi ve doğal afetler gibi çevresel faktörlerden kaynaklanan saldırılara karşı duyarlı olması olarak adlandırılabir. Bu durum, donanımın hasar görmesine ve günlük operasyonlarda aksamalara yol açabilmektedir.

Donanım güvenlik açıkları, cihazların fiziksel yapılarındaki tasarım hataları veya eksikliklerden kaynaklanabilir. Yazılım güvenlik açıklarına göre daha zor tespit edilebilen bu hatalar, genellikle daha kalıcıdır ve uzun vadede sistemin güvenliğini ciddi şekilde tehlikeye atabilme potansiyeline sahiptir (Coderspace, 2024).

Ağ güvenlik açıkları: Ağ yapılandırmalarındaki zayıflıklar, saldırganlar için potansiyel giriş noktaları oluşturabilir. **Yamalanmamış ağ cihazları,** tıpkı yazılımlar gibi, bilinen güvenlik açıklarını kapatmak için düzenli güncellemelere ihtiyaç duyabilmektedir. Bu cihazların güncellenmemesi, açıkların kötüye kullanılmasına yol açabilir. **Güvensiz kablosuz ağlar** ise, uygun şifreleme ve diğer güvenlik önlemlerinin eksikliği nedeniyle, yetkisiz kişilerin ağa erişmesine ve verilerin ele geçirilmesine imkân

verir. Bu tür zayıf yapılandırmalar, ağ güvenliğini ciddi şekilde tehdit eder (The Cyber Express, 2024).

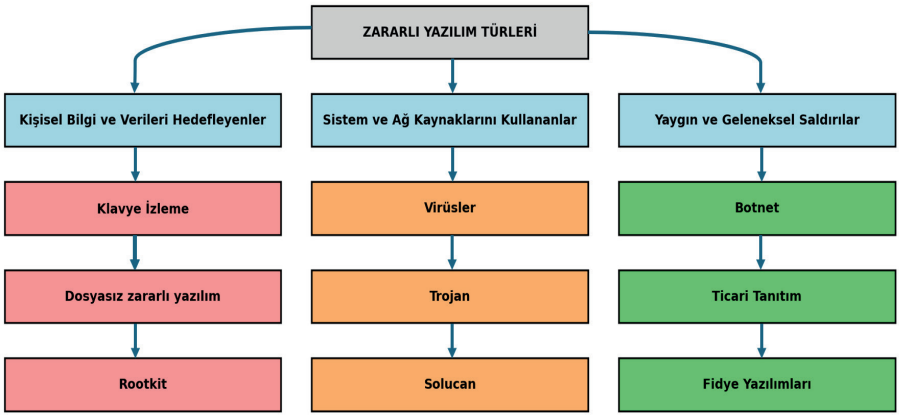
Sıfırıncı Gün Saldırıları: Sıfırıncı gün saldırıları, yazılımdaki bir güvenlik açığının keşfedildiği anda başlar. Yazılım geliştiricileri, bu tür açıkları tespit ettiklerinde, kullanıcıların sistemlerini güvenlik yamalarıyla güncellemeyi amaçlar. Ancak, saldırganlar henüz fark edilmeyen "Sıfırıncı Gün" açıklarını bulmaya çalışır ve bu zafiyetleri kullanarak hedef sistemlere sızmayı amaçlar. Bu tür saldırılar, güvenlik açığı fark edilene kadar ciddi tehditler oluşturabilmektedir (Kara, 2019; Uddin, Ali, ve Hassan, 2020).

İnsani Etkiler: İnsan hataları, siber güvenlik açıklarının meydana gelmesinde kritik bir faktördür. Zayıf parola güvenliği, kimlik avı saldırılarına karşı savunmasızlık ve dikkat eksikliği gibi ihmalkâr davranışlar, siber suçlular tarafından kolayca istismar edilebilecek kişisel ve hassas bilgilerin ifşa olmasına neden olabilmektedir. Bu tür ihmaller, kullanıcıların farkında olmadan güvenlik tehditlerine açık hale gelmelerine yol açarak, tüm sistemin güvenliğini tehlikeye atabilmektedir (Zengrc, 2024).

Zararlı Yazılımlar ve Çeşitleri

- Bilgisayar Virüsleri
- Truva Atı ("trojan")
- Klavye İzleme ("key logger")
- Kurtçuk ("worm")
- İstem dışı olarak gönderilen ticari tanıtım ("adware") yazılımları
- Bilgi toplayan casus / köstebek ("spyware")
- Botnet
- Fidye Yazılımları
- Rootkitler olarak sınıflandırılabilmektedir (Yazılım Türkiye, 2024; Ünver, Canbay, ve Mirzaoğlu, 2018).

Bilgisayar sistemlerine, ağlara veya cihazlara zarar vermek, veri çalmak, gizliliği ihlal etmek veya kontrol sağlamak amacıyla tasarlanmış kötü amaçlı yazılım türleri aşağıdaki şekilde gösterilmiştir (Akamai, 2024).



Şekil 3. Zararlı yazılım türleri (Akamai, 2024)

Siber Güvenlik Açığı Analizi Yöntemleri

- Kaynak Kod İncelemesi
- Güvenlik Açığı Tarayıcıları
- Sızma Testi
- Log incelemesi
- Sosyal mühendislik testleri
- Güvenlik incelemeleri ve denetimi gibi yöntemler siber güvenlik açığı analizinde önem arzetmektedir (Infinitumit, 2023).

SİBER GÜVENLİK AÇIKLARININ EKONOMİK SONUÇLARI

Siber güvenlik açıkları hem bireysel kuruluşlar hem de daha geniş ekonomik sistemler için ciddi ekonomik sonuçlara yol açabilmektedir. Bu açıklar, siber saldırılar için birer zayıf nokta teşkil ederek, saldırganların sistemlere izinsiz erişim sağlamasına ve çeşitli zararlı faaliyetlerde bulunmasına olanak tanımaktadır. Bu tür güvenlik açıkları, işletmelerin operasyonel işleyişini aksatabilir, mali kayıplara yol açabilir ve hatta daha geniş bir ekonomik krizin tetikleyicisi olabilir.

Doğrudan Ekonomik Etkiler: Siber güvenlik ihlallerinin doğrudan maliyetleri, genellikle önemli ve hızla büyüyen bir sorun teşkil etmektedir. Bu tür olaylar, kuruluşlar için yalnızca anlık mali kayıplara değil, aynı zamanda uzun vadeli finansal etkiler yaratacak çeşitli giderlere de yol açmaktadır. Bir siber saldırı veya güvenlik ihlali, doğrudan hırsızlık,

dolandırıcılık, veri kaybı ve hizmet kesintileri gibi finansal zararlara neden olabilmektedir. Bu kayıplar, şirketlerin gelirlerini doğrudan etkilediği gibi kısa vade de likidite problemlerine sebep olmaktadır.

Bir siber güvenlik olayı sonrasında, kuruluşlar sıklıkla olay müdahalesi ve kurtarma çalışmalarına büyük miktarda para harcarlar. Olay müdahalesi, güvenlik açıklarını tespit etme, saldırının etkilerini sınırlama ve verileri koruma gibi acil önlemleri içerir (Keeper Security, 2023). Bu süreçler, genellikle dış danışmanlık hizmetleri ve güvenlik uzmanlarıyla yapılan sözleşmelerle de desteklendiğinden ek maliyetler oluşmaktadır. Ayrıca, şirketlerin sistemlerini eski haline getirmek ve normal işleyişi sağlamak için altyapı ve yazılım güncellemeleri gerekebilir, bu da ciddi bir finansal yük oluşturmaktadır. Örneğin halka açık şirketler için siber güvenlik ihlallerinin etkileri, hisse senedi fiyatlarında belirgin bir düşüşe yol açabilir.

Yapılan araştırmalar, büyük bir siber güvenlik ihlalinin şirketin hisse senedi fiyatlarında ortalama %5 ila %10 arasında bir düşüşe neden olabileceğini göstermektedir. Bu düşüş hem anlık finansal kayıpları hem de gelecekteki güvenlik ve karlılık ile ilgili yatırımcı kaygılarını yansıtmaktadır. Hisse senedi fiyatlarındaki ani düşüş, şirketin piyasa değerini kaybetmesine ve potansiyel yatırımcıların şirkete olan güveninin sarsılmasına yol açabilmektedir. Bu durum, gelecekteki finansman maliyetlerinin artmasına, potansiyel ortaklıkların veya yatırım fırsatlarının azalmasına neden olabilmektedir (Federal Reserve, 2022). Özetle, siber güvenlik ihlallerinin doğrudan mali etkileri yalnızca olayın hemen ardından değil, aynı zamanda uzun vadede de devam edebilir. Şirketlerin bu tür ihlallerin finansal sonuçlarını azaltabilmek için etkili güvenlik önlemleri alması, kapsamlı bir kriz yönetimi planı geliştirmesi ve yasal düzenlemelere uygunluk sağlaması son derece önemlidir. Aksi takdirde, kısa vadeli mali kayıpların yanı sıra uzun vadeli itibar zedelenmesi de şirketin sürdürülebilirliğini ciddi şekilde riske atabilir. Bu durum, yalnızca mevcut iş süreçlerini olumsuz etkileyip aksatmakla kalmaz, aynı zamanda gelecekteki büyüme ve fırsatların da sınırlanmasına yol açabilir.

Dolaylı Ekonomik Etkiler: Doğrudan ekonomik kayıpların yanı sıra, siber güvenlik tehditlerinin dolaylı etkileri de oldukça kapsamlı ve uzun vadeli olabilir. Özellikle bir kuruluşun itibarının zarar görmesi, tüketicilerin güvenini ve sadakatini olumsuz etkileyebilir. Bu durum, ilgili sektördeki satışların düşmesine yol açabilir. Bir veri ihlali ya da güvenlik açığı sonrasında, tüketiciler hizmet aldıkları şirketlere karşı güven kaybı yaşayabilir ve bu olumsuz durum yalnızca mevcut müşteri kitlesini değil, potansiyel müşterileri de etkileyebilir. Güven kaybı, özellikle dijital çağda müşterilerin daha dikkatli seçimler yapmasına neden olur, bu da şirketin pazar payının daralmasına yol açar (Federal Reserve, 2022; PTM Wealth, 2024).

İtibar zedelenmesi satışları olumsuz etkiler ve uzun vadede finansal kayıplara neden olabilir. Ayrıca, şirketler siber güvenlik önlemlerini güçlendirmek adına önemli kaynaklar ayırmak zorunda kaldıklarından kaynakların genellikle inovasyon ve büyüme girişimlerinden uzaklaştırılması söz konusu olur. Şirketler, siber tehditlere karşı savunmalarını pekiştirebilmek için ek güvenlik yatırımlarına yönelmek durumunda kalabilirler.

Bu, kısa vadede şirketlerin teknolojik gelişim ve yenilikçilik çalışmalarını sınırlayabilmektedir. İnovasyon ve geliştirmeye yönelik harcamaların azalması, ekonomik gelişmeyi ve sektörel büyümeyi potansiyel olarak yavaşlatabilir. Bu, özellikle yüksek teknoloji ve dijital dönüşüm süreçlerine dayalı sektörlerde daha belirgin hale gelir. Teknolojik ilerlemenin yavaşlaması hem şirketlerin rekabet gücünü hem de genel ekonomik verimliliği olumsuz yönde etkileyebilir. Finansal hizmetler sektörü, siber güvenlik tehditlerinin yol açtığı dolaylı etkilerden daha fazla etkilenen bir diğer önemli sektördür. Bu sektör, siber güvenlik ihlallerinin neden olduğu zararlara karşı duyarlı olup, ihlallerin karlılık ve rekabet gücü üzerinde ciddi etkileri olabilir.

Finansal hizmetler şirketleri, güvenlik ihlalleri sonucunda sadece doğrudan mali kayıplarla değil, aynı zamanda yatırımcıların ve iş ortaklarının güvenini kaybetme riskiyle de karşı karşıya kalır. Bunun sonucu olarak, finansal kuruluşlar uluslararası pazarlarda rekabetçiliklerini kaybedebilir ve yabancı yatırımlar açısından daha az cazip hale gelebilirler. Bu durum, sadece sektördeki şirketleri değil, tüm ekonomi üzerinde daha geniş çaplı etkiler yaratabilir, çünkü finansal hizmetler sektörü, ekonomik büyüme ve yatırım kararlarının temel taşıyıcı unsurlarındandır (Esecurity Planet, 2024).

Sonuçta siber güvenlik tehditlerinin doğrudan ve dolaylı ekonomik etkileri hem bireysel kuruluşları hem de küresel ekonomik yapıları derinden etkileyebilir. Bu etkiler, sadece kısa vadeli mali kayıplarla sınırlı kalmaz, aynı zamanda uzun vadeli ekonomik büyüme, rekabetçilik ve küresel ticaret üzerinde de önemli bir baskı oluşturur. Bu bağlamda, devletler ve şirketler, siber güvenlik önlemlerini güçlendirmek ve dijital altyapıları daha dirençli hale getirmek için daha fazla kaynak ayırarak, bu tehditlerin olası ekonomik etkilerini minimize etmek adına stratejiler geliştirmelidir.

Daha Geniş Ekonomik Etkiler: Siber güvenlik açıklarının etkileri, yalnızca bireysel şirketleri değil, aynı zamanda ulusal güvenliği ve ekonomik istikrarı da ciddi şekilde tehdit etmektedir. Özellikle enerji, ulaştırma ve iletişim gibi kritik altyapılar, siber suçlular tarafından sıkça hedef alınmaktadır. Bu sektörlerde meydana gelen herhangi bir güvenlik ihlali, temel hizmetlerin aksamasına neden olabilir ve büyük ölçekli ekonomik kayıplara yol açabilir. Örneğin, enerji arzının kesilmesi, ulaşım sistemlerindeki

aksaklıklar veya iletişim ağlarındaki çöküşler, toplumun günlük işleyişini durdurabilir ve geniş çaplı ekonomik zararlar doğurabilir (Federal Reserve, 2022; PTM Wealth, 2024). Küresel ölçekte, siber suçların ekonomik etkilerinin yıllık trilyonlarca dolara ulaşması beklenmektedir. Bu durum, dünya genelinde ekonomik istikrarı tehdit eden büyük bir risk faktörü haline gelmiştir. Siber suçların hızla artan yaygınlığı, yalnızca bireysel işletmelerin değil, tüm ulusal ekonomilerin de bu tehditlere karşı daha savunmasız hale gelmesine neden olmaktadır. Bu nedenle, güvenlik açıklarının ele alınması hem ulusal güvenliği hem de ekonomik bütünlüğü korumak açısından son derece önemli bir aciliyet taşımaktadır (CISA, 2024).

Siber Güvenlik Politikaları: Ülkemizde siber güvenlik stratejileri ve uygulamaları henüz uzun bir geçmişe sahip olmasa da bu alanda atılan önemli adımlar giderek artmaktadır. Özellikle, dijital güvenliğin sağlanması ve altyapıların korunması amacıyla yapılan yasal düzenlemeler, siber güvenlik alanındaki ilk somut girişimler arasında yer almaktadır. 2004 yılında çıkarılan 5070 sayılı Elektronik İmza Kanunu, dijital ortamda güvenli işlemler için gerekli hukuki altyapıyı oluşturmuş ve elektronik imza kullanımını yasallaştırmıştır. Bu kanun, dijital güvenliğin güçlendirilmesi ve e-ticaretin sağlıklı bir şekilde yürütülmesi açısından önemli bir kilometre taşı olmuştur. Sonraki yıllarda, 2008'de yürürlüğe giren Elektronik Haberleşme Güvenliği Yönetmeliği, özellikle haberleşme alanındaki güvenlik tedbirlerini amaçlayarak, kritik iletişim altyapılarının siber tehditlere karşı korunmasına odaklanmıştır (Hekim ve Başıbüyük, 2013). Ayrıca, siber güvenlik becerileri eksikliğine ilişkin analizlerin, işgücü ekonomisi, yetkinlik politikası ve etkililik analizi gibi sosyal bilimlerde yer alan araştırma geleneklerine bağlanması gerekliliği de günümüzde tartışma konusu olmaktadır (Austin, 2020). Bu yönetmelik, Türkiye'nin siber güvenlik stratejilerinin temel unsurlarından biri haline gelmiş ve sektördeki tüm paydaşlar için güvenlik standartlarının belirlenmesinde önemli bir rol oynamıştır.

Türkiye'deki Önemli Siber Güvenlik İhlalleri ve Ekonomik Etkileri

Türk Telekom Altyapı Saldırısı (2016)

Olay: 2016 yılında Türk Telekom'un altyapısına yönelik siber saldırıda Türk Telekom'un internet ve telefon hizmetlerinde kesintiler meydana gelmiştir.

Ekonomik Etkiler: İnternet hizmetlerine erişimin kısıtlanması, ticaretin ve günlük iş faaliyetlerinin aksamasına neden olmuştur. Bu durum, birçok işletmenin gelir kaybı yaşamasına yol açtığı gibi özellikle e-ticaret sektöründe faaliyet gösteren firmalar, satışlarının önemli ölçüde düşmesiyle karşılaşmıştır. Ayrıca, Türk Telekom'un teknik altyapısının onarım ve iyileştirilmesi için ek maliyetler de ortaya çıkmıştır.

Banka Hesaplarına Yönelik Oltalama Saldırıları (2018)

Olay: 2018'de Türkiye'deki büyük bankalara yönelik oltalama saldırıları düzenlendi. Saldırganlar, müşterilerin banka bilgilerini çalarak, hesaplarındaki paraları izinsiz bir şekilde transfer etti.

Ekonomik Etkiler: Bankaların güvenlik açıklarından yararlanan bu saldırı sonucu, birçok müşteri parasal kayıplara uğramıştır. Ayrıca, bankalar için itibar kaybı söz konusu olmuş ve bu, durum banka müşterilerinin güveninin zedelenmesine yol açmıştır. Bankalar, güvenlik önlemleri almak için ek yatırımlar yapmak zorunda kalmış ve sektörün genel güvenlik maliyetleri arttığından bankaların operasyonel giderleri de bu durum paralelinde artmıştır.

Üniversitelere Yönelik Siber Saldırılar (2020)

Olay: Türkiye'deki birkaç üniversiteye yönelik siber saldırılar, üniversitelerin eğitim ve araştırma verilerini hedef almıştır. Bu saldırılarda, akademik veriler çalınmış ve bazı üniversiteler eğitimlerine geçici olarak ara vermek zorunda kalmıştır.

Ekonomik Etkiler: Üniversiteler, eğitim süreçlerini sürdürmekte zorlanmış ve bu da eğitim sektörü için finansal kayıplara yol açmıştır. Ayrıca, saldırı nedeniyle akademik ve araştırma verilerinin güvenliğini sağlamak için üniversiteler yeni yatırımlar yapmak zorunda kalmıştır. Eğitim sektöründeki bu aksaklıklar, eğitimdeki verimliliği olumsuz etkileyerek ekonomik anlamda daha büyük kayıplara yol açmıştır.

Türk Hava Yolları ve Devlet Kurumlarına Yönelik Saldırılar (2021)

Olay: 2021 yılında Türk Hava Yolları ve bazı devlet kurumlarına yönelik ve büyük ölçüde kötü niyetli yazılımlardan oluşan saldırıda kurumların dijital altyapılarının işleyişi geçici olarak engellenmiştir.

Ekonomik Etkiler: Türk Hava Yolları'nın dijital sistemlerinin devre dışı kalması, uçuş iptalleri müşteri hizmetlerinde aksaklıklara yol açmıştır. Bu da müşteri memnuniyeti kaybına ve gelir kaybına neden olmuştur. Ayrıca, saldırı sonrası alınan güvenlik önlemleri, şirketin ek harcamalar yapmasına yol açmıştır. Devlet kurumlarına yönelik saldırıların etkileri hem devletin dijital hizmetlerini hem de kamu güvenliğini tehdit etmiştir. Devletin bu tür olaylara karşı aldığı güvenlik önlemleri de ek bütçe gereksinimlerine neden olmuştur.

Banka Sektörüne Yönelik Fidye Yazılımı Saldırıları (2022)

Olay: 2022 yılında Türkiye'deki birkaç büyük bankaya yönelik fidye yazılımı saldırıları düzenlenmesi şeklinde olmuştur. Saldırganlar, bankaların dijital sistemlerini kilitleyerek, verilerin serbest bırakılması için fidye

talep etmiştir.

Ekonomik Etkiler: Bu saldırıların ekonomik etkileri, sadece doğrudan fidye ödemeleriyle sınırlı kalmamış aynı zamanda bankaların sistemlerini yeniden kurmak ve güvenliği artırmak için yapacakları ek yatırım ve maliyetleri de arttırmıştır. Ayrıca, bankaların müşteri verilerinin güvende olmaması, müşteri güvenini sarsmış ve olası müşteri kayıplarına yol açmıştır. Bankaların saldırıya uğraması, diğer finansal kurumları da daha büyük güvenlik tedbirleri almak zorunda bırakarak, sektörde genel bir güvenlik maliyet artışına neden olmuştur.

SONUÇ VE ÖNERİLER

Siber güvenlik açıklarının yaygınlığı hem ulusal istikrar hem de ekonomik ilerleme için ciddi bir risk oluşturmaktadır. Dijitalleşmenin hızlanmasıyla birlikte finansal ve kritik altyapı sistemlerine yönelik siber saldırılar önemli ölçüde artmış, bu da operasyonel aksaklıklara, güvenin azalmasına ve itibarın zedelenmesine neden olmuştur. Bu risklerden en çok etkilenen sektörler ulaşım, enerji, sağlık ve finans sektörleridir. Uzun vadeli ekonomik istikrarsızlık, daha düşük hizmet kalitesi ve daha yüksek işletme giderleri, bu tür güvenlik ihlallerinin olası sonuçlarıdır. Siber saldırıların parasal kayıpların ötesine geçen yansımaları da vardır; ayrıca paydaş ve tüketici sadakatinde düşüşe ve kamu güveninde kayba neden olmaktadır. Bu faktörler göz önünde bulundurulduğunda, siber güvenliğin iyileştirilmesi için kapsamlı ve çok boyutlu önlemlerin alınması gerekmektedir. Etkili tehdit yönetimi, kamu ve özel sektörün siber güvenlik stratejileri geliştirmek üzere birlikte çalışmasıyla kolaylaşacaktır. Kurumsal güvenlik altyapılarının güçlendirilmesi son teknolojinin bütünleştirilmesi ve sürekli izleme sistemlerinin devreye sokulmasıyla saldırılara karşı dayanıklılıkları artırılacaktır. Ayrıca, bireyler ve çalışanlar arasında farkındalığı artırmak ve siber güvenlik kültürünü yaygınlaştırmak için eğitim programlarının fazlalaştırılması, toplumun bu konuda daha da bilgilendirilmesi için çalışmalar yapılması büyük önem taşımaktadır. Bu strateji, siber güvenlik uzmanlarını eğiterek ve küresel çapta bilgi alışverişinde bulunarak risklerin önlenmesi için güçlü bir temel oluşturacaktır.

KAYNAKLAR

Karasoy, H. A., & Babaoğlu, P. (2021). Türkiye'de siber güvenlik: Yasal ve kurumsal altyapı. Yasama Dergisi, 44, 123-155.

Afyonluoğlu, Mustafa., "Siber Güvenlik ve Kamu Politikaları", Teknoloji ve Kamu Politikaları Kitabı, ss. 379-411, (Ed.: Mete Yıldız-Cenay Babaoğlu), Gazi Kitabevi, Ankara, 2020.

Hansen, L., & Nissenbaum, H. (2009). Digital disaster, cyber security, and the Copenhagen School. International studies quarterly, 53(4), 1155-1175.

Jang-Jaccard, J., & Nepal, S. (2014). A survey of emerging threats in cybersecurity. Journal of computer and system sciences, 80(5), 973-993.

Nasuh B. Karadağ. (2024). Siber güvenliğin tarihsel gelişimi. Nasuh Bugra Karadağ. https://www.nasuhbugrakaradag.av.tr/siber-guvenligin-tarihsel-gelisimi/ Erişim Tarihi: 02.12.2024.

Çakır, H., & Uzun, S. A. (2021). Türkiye'nin siber güvenlik eylem planlarının değerlendirilmesi. Ekonomi İşletme Siyaset ve Uluslararası İlişkiler Dergisi, 7(2), 353-379.

Ünver, M., Canbay, C., & Mirzaoğlu, A. G. (2009). Siber güvenliğin sağlanması: Türkiye'deki mevcut durum ve alınması gereken tedbirler. Bilgi Teknolojileri ve İletişim Kurumu (BTK), Ankara, 8, 2018.

Fischer, E. A. (2014). Cybersecurity issues and challenges: In brief.

Sun, N., Zhang, J., Rimba, P., Gao, S., Zhang, L. Y., & Xiang, Y. (2018). Data-driven cybersecurity incident prediction: A survey. IEEE communications surveys & tutorials, 21(2), 1744-1772.

McIntosh, T., Jang-Jaccard, J., Watters, P., & Susnjak, T. (2019). The inadequacy of entropy-based ransomware detection. In *Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part V 26* (pp. 181-189). Springer International Publishing.

Arslan, M. E. (2016). Siber Güvenlik ve Siber Saldırı Türleri. *Gazi Üniversitesi, Sağlık Bilişim Enstitüsü, Ankara*.

Eryılmaz, E. E., & Kılıç, E. (2020). İstenmeyen Epostaların Tespiti için Kullanılan Yöntemlerin İncelenmesi. *Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Dergisi*, *11*(3), 977-987.

Acar, S. A. M. İ., & Zoray, M. (2021). Ortadaki adam saldırısının teknik yönden incelenmesi.

Letstechiteasy. (2024). Cyber security attacks. https://letstechiteasy.com/blog/cyber-security-attacks/ Erişim Tarihi: 02.12.2024.

The Cyber Express. (2024). What are vulnerabilities? https://thecyberexpress.com/what-are-vulnerabilities/ Erişim Tarihi: 02.12.2024.

Sprinto. (2024). Cyber security vulnerabilities. https://sprinto.com/blog/cyber-security-vulnerabilities/ Erişim Tarihi: 02.12.2024.

Keeper Security. (2023, December 27). Common types of cybersecurity vulnerabilities. https://www.keepersecurity.com/blog/2023/12/27/common-types-of-cybersecurity-vulnerabilities/ Erişim Tarihi: 03.12.2024.

Federal Reserve. (2022, May 12). Implications of cyber risk for financial stability. https://www.federalreserve.gov/econres/notes/feds-notes/implications-of-cyber-risk-for-financial-stability-20220512.html Erişim Tarihi: 04.12.2024.

PTM Wealth. (2024). The economic impact of cybersecurity breaches. https://ptmwealth.com/articles/the-economic-impact-of-cybersecurity-breaches Erişim Tarihi: 04.12.2024.

CISA. (2024). Cybersecurity best practices. https://www.cisa.gov/topics/cybersecurity-best-practicesSon Erişim Tarihi: 05.12.2024

Coderspace. (2024). Siber güvenlik açıkları nedir? https://coderspace.io/blog/siber-guvenlik-aciklari-nedir/Son Erişim Tarihi: 05.12.2024

Kara, İ. (2019). Kaba kuvvet saldırı tespiti ve teknik analizi. Sakarya University Journal of Computer and Information Sciences, 2(2), 61-69.

Uddin, M. H., Ali, M. H., & Hassan, M. K. (2020). Cybersecurity hazards and financial system vulnerability: a synthesis of literature. *Risk Management*, *22*(4), 239-309.

Zengrc. (2024). Top 7 vulnerability mitigation strategies. https://www.zengrc.com/blog/top-7-vulnerability-mitigation-strategies/ Erişim Tarihi: 04.12.2024.

Yazılım Türkiye. (2024). Zararlı yazılımlar nedir, çeşitleri neler? https://www.yazilimturkiye.com/zararli-yazilimlar-nedir-cesitleri-neler/ Erişim Tarihi: 05.12.2024

Infinitumit. (2023). Güvenlik açığı. https://www.infinitumit.com.tr/guvenlik-acigi/ Erişim Tarihi: 06.12.2024

Akamai. (2024). 9 malware types enterprise professionals need to know. https://www.akamai.com/blog/security/9-malware-types-enterprise-professionals-need-to-know Erişim Tarihi: 05.12.2024

Esecurity Planet. (2024). 2024 cybersecurity laws & regulations. https://www.esecurityplanet.com/compliance/2024-cybersecurity-laws-regulations/Son Erişim Tarihi: 06.12.2024

Hekim, H., & BAŞIBÜYÜK, O. (2013). Siber suçlar ve Türkiye'nin siber güvenlik politikalari. *Uluslararası Güvenlik ve Terörizm Dergisi*, *4*(2), 135-158.

Austin, G. (2020). Cyber Security Education. Principles and Policies. New York, NY, UK and New York: Routledge.

CHAPTER 3

THE IMPACT OF SOFTWARE TESTING ACTIVITIES ON SOFTWARE QUALITY

Döne KARHAN[1], Fulya ASLAY[2]

1 Department of Artificial Intelligence and Robotics, Institute of Science, Erzincan Binali Yıldırım University, Erzincan, Türkiye. done.kilic.92@gmail.com, ORCID ID: 0009-0003-9584-1725.
2 Assoc. Prof. Dr., Department of Computer Engineering, Faculty of Engineering and Architecture, Erzincan Binali Yıldırım University, Erzincan, Türkiye. faslay@erzincan.edu.tr, ORCID: 0000-0001-5212-6017.

1.INTRODUCTION

The software sector has become an indispensable part of social and economic life with rapidly developing technology. Today, individuals and businesses want to have quick access to the services and products they need. Accordingly, the functionality, reliability and user-friendliness of software products are of great importance. However, meeting these expectations requires meticulous work not only in the software development process but also in the quality assurance and testing processes. Planning and management processes are crucial for the success of software projects. However, problems such as delays, budget overruns and poor quality are common challenges. For example, in a study conducted by Standish Group International in 2004, it was found that 53% of software projects were completed over the planned budget and time. Only 29% were completed on time and on budget (Seker, 2014). This clearly demonstrates the critical role of effective planning and management processes in software projects in reducing the probability of failure. Software testing is a systematic process to evaluate whether a software works in accordance with specified requirements, to detect defects and to prevent negative consequences that these defects may cause. However, in many software projects, testing processes are neglected due to time and cost pressures. Although it is thought to save on the testing phase in the first place, it is seen that this approach leads to higher costs and customer dissatisfaction in the long run. Untested software has the potential to produce errors after it is presented to the user, and these errors can negatively affect the reliability of the software and the user experience. A quality software development process depends not only on coding, but also on the effective implementation of testing processes. The planned execution of testing processes ensures that the software is not only error-free, but also optimized in terms of performance and fully responsive to user needs. Test automation, in particular, has an important place in modern software projects in terms of accelerating processes and reducing costs. Major companies around the world are investing heavily in test automation and continuous integration processes to reduce costs and improve quality in software projects. In this study, the importance of software testing processes and the techniques used in these processes are discussed in detail. The aim is to show how testing processes can be improved and integrated into the software lifecycle to achieve higher quality results at lower cost in software projects. Testing processes in the software lifecycle are not only a technical requirement but also a strategic priority for the sustainability of software projects. The correct implementation of testing processes increases the value of software products to both businesses and users in the long term. Accordingly, effective management of test processes is a key factor in the success of software projects.

2. WHAT IS SOFTWARE TESTING?

Software testing is a set of methods to evaluate the conformance of a software to specified requirements and to uncover potential defects. This process is performed to verify whether the software meets the desired functionality, performance and user requirements. As an integral part of the development process, software testing aims to both improve the overall quality of the software and ensure user satisfaction. Since it is usually not possible to prove that a software is bug-free, the main goal of the testing process is to minimize defects by simulating the operating conditions of the software. This is achieved by examining the software's responses to inputs, checking the correctness of the outputs and comparing them with the expected results. At the same time, the software's weaknesses and possible error scenarios are tested to predict future risks. Software testing involves two basic approaches: static testing and dynamic testing. Static testing involves examining the software before it is executed, while dynamic testing involves checking the software during its real-time operation. Both approaches are critical for analyzing different aspects of the software and ensuring its correctness. An effective testing process not only detects software defects, but is also an important tool for optimizing the software's performance, improving its ease of use and ensuring its security. Software testing therefore enables the early detection of defects during the development process, resulting in high quality products at a lower cost. As a result, software testing is an integral part of the software development lifecycle and an indispensable process to guarantee the functionality, performance and reliability of a software. It is not only a technical requirement but also one of the cornerstones of a user-centered approach.

3. OBJECTIVES OF SOFTWARE TESTING

The main goal of software testing is not to make software completely bug-free, but to detect and resolve existing bugs. This process is aimed at improving the quality of the software and minimizing potential risks. By correcting bugs, the reliability of the software increases, which increases customer satisfaction, prevents major risks such as loss of life and helps prevent financial losses. In addition, the reliability of the person or company developing the software is directly proportional to the effectiveness of the testing processes. Seven basic principles have been identified in software testing processes. These clearly define the purpose of testing and how it should be conducted: The Purpose of Testing is to Detect Defects: Software testing is not performed to show that defects do not exist, but rather to find and fix existing defects (Amir G., 2018). One Hundred Percent Untestable: It is not practically possible to test every possible situation of the software. Therefore, testing should focus on the most risky areas of the software. Testing should be done at the beginning of the process: Software

testing should be started at the beginning of the development process, not at the end. In this way, bugs can be detected and corrected in the early stages. Bugs are often concentrated in specific areas: Bugs in software are often concentrated in certain areas. Tests focusing on these areas ensure that bugs are detected at less cost. The Pesticide Paradox: As the same test cases are repeated, the software may become immune to testing and defects may not be detected. In this case, test cases need to be renewed and diversified (Vaishali, 2017). Each Software Requires Unique Testing Approaches: Each software has different characteristics depending on factors such as the technology, hardware and security requirements. Therefore, unique testing approaches should be developed for each software. Bug-Free Software Does Not Meet Customer Needs: Just because software is bug-free does not necessarily mean that it meets customer needs. Therefore, not only technical verification but also whether customer requirements are met should be considered in software testing. Software testing not only identifies defects, but also improves the overall quality and reliability of the software, enabling the adoption of a customer-centric approach.

3.1. SOFTWARE TEST TYPES

Software testing methods are used to minimize errors in the details of software development. Each test has its own interstices and limitations. Below, the most common types of testing are described.

3.1.1. Black Box (Black Box) Test

In black-box testing, only the operations between what comes from the outside and what is obtained are examined, without looking at the internal structure and code of the software. This type of testing does not require any indication of how the software is performing; it only tests whether the expected results are met with the given inputs (Srinivas & Jagruthi, 2012). This type of testing is particularly preferred to simulate how the software will be executed by the user. Black box testing does not indicate how much of the software's code has not been tested. This means that the scope of testing can be limited. It can also sometimes be difficult to correctly specify test inputs according to business rules, which can lead to incomplete execution of some test steps (Khan, M., 2011).

3.1.2. White Box (White Box) Test

White box testing is a type of testing that analyzes the internal structure of the software in detail. By examining the functions, procedures, methods and data of the software, this test makes it possible to detect unnecessary code. At the same time, it aims to increase the readability of the software while ensuring the efficiency of use (Srinivas & Jagruthi, 2012). White box testing offers the ability to perform more in-depth analysis

than black box testing. Comprehensive testing of the entire code allows for more detailed evaluations. This minimizes the number of overlooked test cases and significantly improves the code quality of the software. In addition, accelerated defect detection and easier determination of test coverage make white box testing a preferred method in software development processes. However, white box testing is a process that requires technical knowledge. Collaboration with developers and knowledge of certain code standards are important for successful implementation of this test. However, in large software projects, creating test cases that cover the entire code can be time-consuming, which can increase the lead time and complicate project management (Steegmans et al.).

3.1.3. Grey Box Test

Gray box testing, as the name suggests, is a combination of certain features of white box testing and black box testing. In this type of testing, the tester has limited knowledge of the internal structure of the software. That is, they not only perform tests on the user interface, but also go down to the code level to a certain extent. This approach aims to detect bugs in both the functionality and the internal structure of the software. While gray box testing tests the functioning of the software and user interactions, it also requires some knowledge of the technical infrastructure of the software.

3.1.4. Unit Test

Unit testing is a basic test method performed by developers at the beginning of the software development process. These tests aim to check whether the smallest units of the software - for example, functions or code modules - work in accordance with the specified requirements (Myers et al., 2012). Unit testing allows developers to assess the correctness of the code they are creating at an early stage and enables early detection of potential errors. In order to apply these tests effectively, developers need to have a deep knowledge of the details of the code and software design. Today, many software development platforms offer tools and frameworks that support unit testing, making it easier to integrate these tests into software processes. Many organizations have made the use of unit testing mandatory to ensure quality from the early stages of software development (Osherove, 2013). The main purpose of unit testing is to quickly detect defects that may occur during the development or modification of software. While these tests provide insight into the functioning of the code, fake objects or similar techniques are often used to prevent external dependencies from affecting the testing process. Thus, external factors such as network, database and file system access do not interfere with the results of the tests. In addition, unit testing is fast and each test can be performed independent-

ly, making it possible to obtain test results in a short time. These features are of great importance for ensuring the correctness and reliability of each component of the software (Olan, 2003). Unit tests not only detect defects early, but also play a critical role in analyzing the impact of changes made in code refactoring processes on the software. Unit tests, written in accordance with standards, help developers to quickly identify and minimize defects that may occur as a result of changes to the code (Kua). In this context, unit testing is an indispensable tool for improving the reliability of software and supporting the efficiency of the development process.

3.1.5 Integration Test

Integration tests are tests that examine not the individual components of the software, but whether these components work in harmony with each other. For example, an example of integration testing is checking that the accounting and customer relationship management (CRM) modules in a software work properly in both systems. These tests also cover the integration of components such as database access, file reading and writing (Osherove, 2013). The main objective is to test whether small systems that function independently create any incompatibilities or errors when they work together. Individual unit tests may show that each system is error-free, but they do not guarantee that these systems function properly as part of a larger system. Integration testing is necessary to detect errors and problems that may arise from interactions between systems. It should be noted that integration testing should be done after unit testing. These tests are usually considered end-to-end tests and provide control over the state of the environment, dependencies (timing, database, network connections, threads, etc.). This type of testing evaluates the interactions of functional units with each other and the overall performance of the system. Two main approaches can be used for integration testing: Narrow Scope Integration Testing: This type of testing involves only testing other services that interact with a specific service. That is, only the interactions between specific modules are examined, not the entire system. This is a broader version of unit testing (Vocke, 2018). Comprehensive Integration Testing: This is a test to check that all services and the system are working properly. Numerous factors are reviewed, such as network access, file reading and writing, database connections and authorizations. It also simulates how the system should function in a live environment and tests the interactions between multiple services (Vocke, 2018).

3.1.6.Functional Test

Functional testing is testing that checks the functionality of the software, usually through the software interface. This type of testing does not focus on the design or structure of the software, but only checks whether

the software meets the requirements (Young & Pezze, 2008). Functional testing is one of the commonly used test types in companies in Turkey. In some organizations, in the absence of software analysts and testers, software development team members can perform these tests. However, in companies where analysts are employed, software testing is mostly performed by analysts. In companies with test teams, processes such as documentation, planning, execution and reporting of functional tests are managed by test teams. In this process, it is extremely important for the test team to communicate with all stakeholders in the software development lifecycle (developers, analysts, etc.). This communication plays a major role in the creation of test cases. When test cases are categorized according to importance criteria, it is possible to determine which tests should be prioritized and this provides a significant advantage in terms of time management. The purpose of functional testing is to demonstrate how well the software meets its requirements and how well it can reduce error rates, thus improving the quality of the software. Providing appropriate test data is also very important in testing. Because it can often be difficult to obtain appropriate test data in large software systems. Providing the right test data before testing is a critical step for efficient testing.

3.1.7.Smoke Test

A smoke test is a type of test where a few of the most basic test cases are usually selected. The aim is to check that the basic functions are being performed, rather than to test the system or the screen. This test is done to determine whether the application is accepted into the testing process. If the basic functions are not working correctly, the product is sent back to the software team before it is accepted into the testing process (Laboon, 2017). Although this phase is often referred to as acceptance testing, it is not the acceptance of the test that is essential here, but the suitability of the product for testing. It may seem unnecessary at first, but it actually makes a big difference in terms of time and cost. Taking a product that is not ready for testing into the testing process can cause unnecessary revisions. These revisions waste the test team's time and also cause extra effort from developers and analysts.

3.1.8.Regression Test

Regression testing, like smoke testing, usually involves a smaller number of test cases, but is more detailed than smoke testing. This type of testing focuses on testing specific and critical functions, not the entire system. This type of testing is necessary after minor changes to the software. During regression testing, important functions in the software are identified and which functions to test are selected based on a risk assessment. Testing is performed in an environment where the entire system is running,

i.e. not on the developer's computer, but in an environment where all modules and services are integrated (Ngah et al., 2017). Regression testing is not a specific test level, but a type of testing that can be applied at different test levels. This test can be performed during unit, integration and system testing. When determining the regression test set, four main techniques such as coverage, precision, efficiency and generality are used. Testing is done on the existing product instead of starting the software lifecycle. When new requirements or changes arise, testing is done on specific functions. Regression testing can be completed in shorter times than normal functional testing. While functional testing may require the tester to know the system well, regression testing may not require such deep knowledge. While functional testing continues throughout the software development process, regression testing is usually done and finalized when the code is released into the production environment (Arcuri & Briand, 2011).

3.1.9.System Test

System testing is a critical phase in the software development process that enables a holistic evaluation of all components of the software after integration. In this testing phase, the software's ability to adapt to environmental conditions and its performance, as well as functional requirements, are examined in detail. For example, metrics such as the extent to which the software uses computer resources and whether it is compatible with the operating system and other applications are analyzed at this stage. In this context, it is important to perform non-functional testing as well as functional testing. System testing is usually performed with a black box approach. In this approach, only the external behavior of the software is observed during the testing process; the inner workings of the code are not evaluated by the test engineers. This method provides a great advantage in understanding how the system performs from the perspective of the end user. System testing is performed on all integrated components and the effects of these components on each other are observed end-to-end. For this reason, system testing is also called "end-to-end testing" and is used to verify both functional and non-functional requirements. A comprehensive system testing process can involve a variety of test methods. Various test methods such as usability testing, load testing, regression testing, recovery testing, migration testing and hardware testing are applied to evaluate various aspects of the software. These types of testing play a critical role in measuring the software's durability, performance and suitability to user needs. This process not only improves the quality of the software, but also increases confidence in the project.

3.1.10. Acceptance Test

Acceptance tests are not performed by the test team, analyst or developer. These tests should be performed by the people who request the software requirement. The main purpose here is to check whether the requirements are met, rather than to test. It evaluates whether the requested functions are performed correctly and whether the software meets the requirements correctly. After the product owner receives the solution from the software team, the software is accepted with the approval of the product owner. If the product does not reach the desired level, it is sent back to the software team (Melnik et al., 2009). Acceptance testing is also often referred to as "user acceptance testing", where the main purpose is for the user to confirm that the required actions have been taken and that the requirements have been met. Who performs acceptance testing is usually determined at the beginning of the project. Analysts determine which users are responsible for which issues in meetings with the relevant people in the creation of requirements and business rules. Some parts of the product may be used by different users, while other parts may be used by different teams. Therefore, it is important that testing is done according to these terms of use and authorizations. In user acceptance testing, test cases prepared by the test team or the development team are usually not used. This is a common mistake. The user should do their own requirements and testing. This requires good planning. Users may not have used the product before and therefore need training. Once trained, they can do their own testing. Tests should be grouped according to the tasks of the users. Having different people test the same test steps instead of a single person provides a broader feedback about the product and a clearer idea about possible shortcomings. When training the user, it is important not to go into too much technical detail, as this can confuse the user and get in the way of what they really want to see. Especially for test data, the types of data that users want should be created and tests should be based on this data. In these tests, not only business rules are tested, but also usability, user-friendliness, complexity, performance and security. User acceptance testing is an important step, usually performed before the software goes live. However, it is a mistake to perform these tests only at the end of the process. A product that fails to gain user approval can lead to drastic changes in the project. These changes can lead to wasted time, extra effort and loss of employee motivation. In order to avoid such problems, it would be more efficient to conduct tests with the user in the early stages when certain functions of the product are fulfilled. Based on the user's feedback, early changes in the software design can be made more quickly and smoothly. Also, if agile software development methodologies are used, it is very useful to conduct user acceptance tests within specific sprints. This practice not only ensures the accuracy of the

tests, but also allows users to learn more about the product before they receive it. Once the product is delivered, users can more comfortably use the product or adapt quickly with basic training (Wallace, 1986).

3.1.11.Usability Test

Usability is an area that is frequently focused on in software projects and where significant investments are made by companies. Some organizations set up special laboratories for these tests and analyze user behavior in detail. For example, eye movements are tracked to understand which processes users have difficulty with; this method is effective in identifying unnecessary or underutilized areas. In line with the importance of this field, specialists in usability testing are being trained. A software developer can develop good code, but to optimize the user experience, it is necessary to work with design experts. The seven main issues to be considered when designing the interface are as follows: Follows Standards and Guidelines Correctness (Correct) Consistent Intuitive design Usefulness Flexibility (Flexible) Comfortable (Patton, 2000). The tests performed in line with these items evaluate whether the software is user-friendly. Accuracy: The interface should only present actions that the user can perform at that moment. For example, if a transaction needs to be canceled, the relevant button must work. Displaying a button that does not work may give the user a false impression and question the reliability of the software (Patton, 2000). Consistency: Following standards that users are used to makes the software easy to use. For example, while it is generally expected that the "copy" function is located under an "Edit" menu, moving it to another menu may lead to confusion. Consistency makes it easier for the user to navigate through the application (Patton, 2000). Following Standards and Guidelines: If the software runs on platforms such as Windows or MacOS, standards specific to these platforms are already established and documented. These documents contain recommended regulations to increase the usability of the software. If working on a specific platform, the standards of the relevant platform should be researched and test cases should be prepared according to these standards (Patton, 2000). Intuitive Design: Interface design should be intuitive to enable users to easily meet their needs. For example, users' eye and mouse movements are tracked to determine the areas they interact with the most and the design is organized accordingly. Important buttons should be in a prominent position, and attention-grabbing warnings should be designed for actions that should not be taken (Patton, 2000). Flexibility: Users may have different preferred usage patterns. This should be supported by making the software customizable by offering different color schemes, font sizes or levels of detail. For example, some users may prefer to use a dark mode, while others may prefer a light-colored theme. Such options increase the user-friendliness of

the software (Patton, 2000). Convenience: The user wants to know what stage the software is in during a process. If a process is taking a long time, a loading screen or an informational message showing the progress should be provided. In case of errors, the user should be able to return to the process and information loss should be minimized (Patton, 2000). Usability: The functions of the software should add value to the product and make sense to the user. Unnecessary features only complicate the interface and negatively affect the user experience. The functions that enable users to use the software effectively should be clearly defined (Patton, 2000).

3.1.12. Load Test

Load testing is performed to identify performance bottlenecks by examining how software systems behave under a given load. These bottlenecks are uncovered by increasing the load on the system and observing it over a period of time. Performance problems may not only be caused by code; there may also be problems related to hardware or network infrastructure. Therefore, load tests are performed in an environment that reflects real system conditions. Load tests are critical to detect problems in advance, especially in announcement systems where exam results are announced or announcement systems that may receive heavy traffic. For example, bottlenecks that may occur when more users than expected access the system at the same time can only be revealed by load tests. Functional tests cannot identify such problems. After load testing, necessary improvements are made to the software code or infrastructure depending on the data obtained. During the testing process, tests should be carried out over a long period of time and at regular intervals. For example, log records, thread management, file read-write operations, memory usage and network connection details are examined in detail. Continuous processing helps early detection of errors that may occur in the system in the long term (Daniel, 2002).

3.1.13. Stress Test

Stress testing aims to measure how the system reacts to unexpected situations by pushing it to its limits. The system is pushed through maximum user load or compute-intensive functions and the behavior of the entire system is evaluated. The main purpose of this test is to observe if there is data loss and how the system recovers in the event of a crash. For example, in a system with multiple main servers, can the structure sustain itself if one server goes down? With such scenarios, the resilience of the system is tested and potential weak points are identified. In addition, solutions to improve system performance are developed based on the data obtained during stress tests.

3.1.14. Safety Test

Security is one of the most important features of a system. Users do not prefer an insecure system. Therefore, during security testing, both software and hardware reliability of the system are tested by simulating possible cyber attacks. The purpose of security testing is to determine how resilient the system is to external threats. For example, critical points such as the confidentiality of personal information, prevention of fraud risks and prevention of unauthorized access are checked through these tests. Strong security measures ensure that the system is preferred by users with confidence.

3.1.15. Alpha Test

Alpha testing is a testing phase that usually takes place in-house after all the basic testing processes have been completed. Specific user groups use the product and provide feedback. Based on this feedback, the development team fixes the identified bugs. Alpha testing is one of the quality control phases before the software is released to the end user.

3.1.16. Beta Test

Beta testing is a testing process with real users and real data. Unlike other tests, this phase allows the system to be tested in a live environment. The beta version is released to identify and fix remaining problems in the software based on user feedback. When releasing a beta version of a software, companies usually inform users that this version is a "pre-release" and is used for various data collection processes. Feedback and bugs are resolved during beta testing to make the final product a more robust version. These improvements made during the beta phase ensure that the software is more reliable and functional for users.

4. QUALITY IN SOFTWARE

Quality in software is defined as the ability of a software product to meet user needs and to provide key elements such as reliability and maintainability. Ensuring quality requires a disciplined and systematic approach at every stage of the software development process. As software projects increase in complexity, the level of formality and discipline in processes must increase to successfully complete these projects. Often overlooked by small developers, process management is a key determinant of quality in large-scale projects. Unplanned and random approaches jeopardize both the functionality and reliability of the software. To ensure quality in software: • Analysis and Requirements Management: Accurate definition of the software's objectives. • Design and Planning: Modeling the building in accordance with both functional and technical requirements. • Testing and Review: Detecting and correcting bugs in the early stages. Improving processes increases both quality and productivity. This is not only a technical

requirement, but also an approach that guarantees the software's compliance with user requirements and its longterm success.

4.1 Quality Aspects

Software quality is directly related to the functionality, reliability, performance and maintainability of a software product. Developing quality software is not only about technical excellence, but also about creating a product that meets user expectations. The main elements that determine the quality of software are listed below:

• ISO Standards and Compliance

Quality in software is assessed not only by technical specifications, but also by compliance with international standards. ISO/IEC 25010 provides a quality model for software products and defines quality criteria in a systematic way. Compliance with ISO standards increases the universal acceptance and reliability of software.

• Security

Security is about the ability of software to protect against unauthorized access and data breaches. Preventing vulnerabilities ensures both the protection of user data and the integrity of the system.

• Reliability

Reliability refers to the software's fault tolerance and ability to operate without interruption. Reliable software both increases user satisfaction and minimizes the risk of failure in critical systems.

• Usability and Ease

Usability refers to the ease with which software can be understood, learned and used by users. Ease enables users to use the software in a practical way and shortens the learning process.

• Usability

Usability relates to the practical usability of the software and its ability to meet user requirements. A usable software increases user satisfaction and reduces learning time.

• Portability

Portability refers to the capacity of a software to run on different platforms. This feature ensures smooth operation, especially between different devices and operating systems.

• Flexibility

Flexibility refers to the capacity of software to easily adapt to chang-

ing needs and environmental conditions. The ability of a software to be reconfigured or extended in different contexts without losing functionality is one of the keys to long-term success. For example, the modular design of a software allows new features to be easily added.

• Efficiency and Effectiveness

Efficiency and effectiveness are determined by the software's capacity to perform specific tasks with minimal resource use and time consumption. These elements affect both the user experience and operational costs.

• Documentation

Effective documentation is essential for a correct and complete understanding of the software. User manuals, white papers and training materials increase the learnability and maintainability of the software.

• Cohesion and Integration

The extent to which software is compatible with other systems and platforms is important. Good integration increases the usability of the software in a wider context.

• Functional Relevance

Functional conformance ensures that the software fully meets the specified requirements. This guarantees correct, complete and reliable operation of the software.

• Performance

Efficiency It refers to how effectively the software uses resources and how quickly it responds to meet user expectations. Performance directly affects both user experience and system reliability.

• Sustainability

Maintainability is determined by whether the software is suitable for future maintenance, development and changes. Good software can be easily updated and maintain its functionality in the long term.

• Trust and Risk

Aversion A key indicator of software quality is that the software does not create unexpected problems and provides users with confidence. Risk aversion reinforces this trust by minimizing system errors.

• Context and Scope

It is important that the software meets the business needs identified in the context in which it is developed and that its scope is aligned with those needs. A clear context and scope definition avoids unnecessary workload.

• Performance Efficiency

Efficient use of resources and fast response times indicate the extent to which the software meets user needs. Performance ensures that the software remains stable under all load conditions.

• Individual Capabilities

The knowledge, skills and experience of the development team directly affect the quality of the software. Talented individuals can produce innovative solutions to complex problems.

• Team Cohesion and Communication

Strong collaboration and effective communication between team members play a critical role in the successful completion of projects. Disharmony and lack of communication have a negative impact on quality.

• Complexity of the Product

Complex systems have more potential for error and are difficult to manage. Simplicity in software design is an important element that improves quality.

• Systematic Approaches

Planned and systematic approaches at every stage, from analysis to testing, ensure software reliability and

• Time Required

Allocating sufficient time is essential for developing quality software. Time constraints are often a factor that reduces quality. Evaluating software quality based on these elements provides a valuable perspective not only from a technical point of view, but also in terms of user satisfaction and system sustainability. This multidimensional approach to quality provides a framework that increases success in both individual and corporate projects.

5. THE IMPACT OF SOFTWARE TESTING ON SOFTWARE QUALITY

Software testing plays a critical role in quality assurance and is an effective tool at all stages of the software development process. Both functional and performance-oriented testing make it possible to assess the software's conformance to requirements and identify potential defects at an early stage. This process supports not only the development of technically high-quality software, but also the improvement of user satisfaction and the long-term sustainability of the software. Failure rates in software projects clearly show the negative impact of the lack of quality control and testing

processes on projects. In this context, extensive studies in the literature analyzing the effects of software testing strategies on quality demonstrate the importance of this process.

Software testing is a critical process that evaluates the conformance of software to functional and non-functional requirements and improves software quality. Basili and Selby (1987), in their study comparing software testing strategies, found that the code reading method is more effective than other methods in detecting defects (Basili & Selby, 1987). Gelperin and Hetze (1988) divided software testing models into two groups as "phase models" and "life cycle models" and emphasized the contributions of both models to the prevention and detection of requirements, design and implementation defects (Gelperin & Hetze, 1988).

Whittaker (2000), by dividing software testing processes into phases such as modeling the software environment, selection and evaluation of test cases, emphasized the critical importance of these phases in increasing the reliability of software (Whittaker, 2000). Tassey (2002) stated that deficiencies in software testing can lead to quality degradation, cost increases and delays in project deliveries, and also stated that testing in software development processes has shifted from the coding phase to analysis and design (Tassey, 2002).

Software testing plays a critical role not only in defect detection but also in improving the effectiveness of the software development process and project success. For example, Beşli and Çavdar (2010) stated that detailed consideration of test procedures in data warehouse projects increases the reliability of the software (Beşli & Çavdar, 2010). Kılıç and Öztürk (2010) examined integration testing processes in three phases as installation, module compatibility and functionality tests in the integrated system and emphasized the importance of planning these processes (Kılıç & Öztürk, 2010).

Vural and Sagiroglu (2011) emphasized the impact of security tests on software durability and reliability, and stated that the classification and implementation of these tests within the framework of standards and guidelines is critical for improving software quality (Vural & Sagiroglu, 2011).

Nasir and Sahibuddin (2011) stated that clear definition of requirements, realistic time planning and user satisfaction are critical for success in software projects (Nasir & Sahibuddin, 2011).

According to the statistics of the Standish Group, between 1994 and 2004, many software projects did not achieve the desired efficiency, could not be delivered on time, or were shelved because changes had to be made after completion (Atagören, 2012). However, in the analysis of the same

group between 2004 and 2011, it was observed that the success rates of software projects increased. This increase is directly related to the advances in software engineering and the increased importance given to software quality control processes. In 1994, the success rate was only 16%, while in 2012, this rate increased to 39% (The Standish Group International Inc., 2013).

Software testing is an important step in the software development process that aims to improve the quality of the product and ensure customer satisfaction. An effective software testing process helps to fix defects that occur during the software development phases at lower costs, which directly affects the success of the project. Since 2000, studies show that testing is the most widely used approach among software quality assurance methods (Orso & Rothermel, 2014). Uzun and Koruyan (2019), working on the analysis of software test reports, stated that reports based on defect type, severity and priority contribute to decision processes (Uzun & Koruyan, 2019). Yücalar and Borandağ (2019) underlined the effectiveness of making software testing processes more systematic with the Test Maturity Model Integrated (TMMI) and the effectiveness of this model in preventing software defects (Yücalar & Borandağ, 2019).

Zhao, Hu, and Gong (2021) emphasized the important role of the software industry in global economic development and that technical innovations in this field are the main engine of today's progress (Zhao, Hu, & Gong, 2021). Researchers state that software quality is a critical element for the sustainable growth of the industry and that software testing is one of the most reliable methods to ensure quality. Software developers, testers, independent evaluators, and quality assurance teams emphasize that clearly defining and implementing software testing and quality standards contributes greatly to the success of the processes (Zhao, Hu, & Gong, 2021).

The software testing process aims to detect unintentional errors that occur during the design and development of software systems or models, or to verify that the model meets the specified objectives. Unlike traditional software development processes, artificial intelligence (AI) and machine learning (ML) models are built using an inductive approach (De Silva & Alahakoon, 2022).

Salahat et al. (2023) stated that the quality assurance process ensures that a software project is successfully executed in accordance with established guidelines, concepts and goals (Salahat et al., 2023). They also emphasized that a deeper understanding of the software analysis process and research on type and calibration in this area is still an important goal. Challenges in current software practices include lack of analysis techniques, user arrogance and logistical culture (Salahat et al., 2023).

Safaat and Tjhin (2024) found that automated testing using Katalon Studio is faster and more efficient than manual testing. In their study, they compared both testing methods using nine test cases on a stock exchange trading website application. The test cases created in Katalon Studio are recorded as the tests are run and the test results are logged. In manual testing, each step is documented by taking screenshots or recording. Comparisons show that automated tests give faster results than manual tests. Specifically, the automated tests were completed in 3,177 seconds, while the manual tests took 3,492 seconds. These findings show that automated testing saves time in software testing processes and allows for faster turnarounds (Safaat & Tjhin, 2024).

Khilari et al. (2024) comprehensively examined the strengths and capabilities of various machine learning algorithms used in software testing and quality assurance processes. The research addresses how these algorithms can be used to analyze various metrics in software projects and uncover hidden insights in projects. In particular, they investigated the application of algorithms to predict and improve factors such as code complexity, project completion times, and developer productivity in software projects (Khilari et al., 2024). The researchers emphasize that each algorithm offers different advantages for specific software testing processes and that these algorithms, when properly selected, can make significant contributions to increasing quality and improving productivity in software development processes. According to the findings of these studies, software testing is not only a defect detection activity, but also a critical mechanism that increases the effectiveness of software development processes, assures quality and directly affects the success of projects. When literature reviews focus on the roles of software testing in different dimensions such as integration, security, process analysis and maturity models, the importance of the contribution of each testing phase to software quality becomes clear. Each of these processes makes it possible to develop more efficient, error-free and reliable products at every stage of the software lifecycle.

6. CONCLUSIONS AND RECOMMENDATIONS

Software testing, regardless of its comprehensiveness, is not sufficient on its own to guarantee the success of software projects. The quality of testing within software development processes is crucial for ensuring software reliability and enhancing user satisfaction. A systematic approach to planning and implementing testing from the requirements phase onward helps identify defects early, thereby controlling project costs effectively. In this regard, the integration of traditional testing methodologies with modern approaches offers an efficient strategy for improving the overall quality of software projects.

The future of software testing lies in the effective use of artificial intelligence (AI) and machine learning (ML) technologies. These advancements will facilitate the development of smarter and more predictive testing strategies. Research indicates that AI-based testing tools have the potential to enhance testing efficiency by making more accurate defect predictions (Vahid, Bauer & Felderer, 2020; Bertolini & Mota, 2008). Additionally, incorporating natural language processing (NLP) techniques into software testing processes can further optimize the efficiency and effectiveness of testing. Studies in the NLP field provide significant contributions that can help refine software testing methods.

It is recommended to explore innovative approaches in software testing, both for academic research and industrial applications, to foster strategies that encourage the effective integration of emerging technologies. A critical factor in the success of software development and testing processes is accurately understanding customer needs and requirements. In this context, NLP techniques can serve as a powerful tool for automating the transformation of customer requests into clear, actionable requirements.

Furthermore, Unified Modeling Language (UML) diagrams play a vital role in software testing processes. These diagrams assist in the early detection and resolution of defects that may emerge during the software design phases. Test cases derived from UML diagrams are instrumental in verifying the software's compliance with requirements. They not only assess the software's functionality but also ensure that it meets user expectations. Therefore, UML diagrams contribute significantly to improving software quality by enhancing the coverage of testing processes. The systematic generation of test cases based on UML diagrams is a crucial step in confirming the accuracy of the software and its adherence to user requirements.

In conclusion, the combination of classical and contemporary software testing methodologies will lead to more effective testing processes and improved software quality. Adopting these strategies will help ensure that software development projects meet both functional and user-centric objectives.

REFERENCES

Amir, G. (2018). Seven principles of software testing. Testing Excellence. Retrieved December 6, 2024, from https://www.testingexcellence.com/seven-principles-of-software-testing

Arcuri, A., & Briand, L. (2011). A practical guide for using statistical tests to assess randomized algorithms in software engineering. Simula Research Laboratory. P.O. Box 134, 1325 Lysaker, Norway.

Atagören, Ç. (2012). Error measurement, root cause analysis and a sample case study in software engineering projects (Doctoral dissertation). Ankara.

Basili, V. R., & Selby, R. W. (1987). Comparing the effectiveness of software testing strategies. IEEE Transactions on Software Engineering, 13(12), 1278-1296. https://doi.org/10.1109/TSE.1987.6312967

Beşli, O., & Çavdar, İ. H. (2010). Testing in data warehouse software development process. Academic Informatics'10 - XII.

De Silva, D., & Alahakoon, D. (2022). An artificial intelligence life cycle: From conception to production. Patterns, 3(10), 100489. https://doi.org/10.1016/j.patter.2022.100489

Gelperin, D., & Hetze, B. (1988). The growth of software testing. Communications of the ACM, 51(6). https://doi.org/10.1145/56625.56629

Khan, M. E. (2011). Different approaches to black box testing technique for finding errors. International Journal of Software Engineering & Applications, 32.

Khilari, S., Bhamango, B., Dabade, T., Kale, R., Hendre, A., & Shaikh, Z. (2024). Analysis of strength and capabilities of major machine learning algorithms used in software testing and quality assurance. Naturalista Campano, 28(1), 1414. https://doi.org/10.2223/naturalista.campano.28

Kılıç, E., & Öztürk, S. (2010). Integration testing in large-scale software projects. 2nd Software Quality and Software Development Tools Symposium, Istanbul.

Kua, P. (2003). Unit testing.

Laboon, B. (2017). A friendly introduction to software testing.

Melnik, G., Meszaros, G., & Bach, J. (2009). Acceptance test engineering guide.

Myers, G., Sandler, C., & Badgett, T. (2012). The art of software testing (3rd ed.). Canada: John Wiley & Sons, Inc.

Nasir, M. H. N., & Sahibuddin, S. (2011). Critical success factors for software projects: A comparative study. Scientific Research and Essays, 6(10), 2174-2186. https://doi.org/10.5897/SRE11.0672

Ngah, A., Munro, M., & Abdallah, M. (2017). An overview of regression testing. Journal Name.

Olan, M. (2003). Unit Testing: Test Early, Test Often.

Orso, A., & Rothermel, G. (2014). Software testing: A research travelogue (2000–2014). Proceedings of the Future of Software Engineering, 117-132. https://doi.org/10.1109/FSE.2014.213

Osherove, R. (2013). The art of unit testing (2nd ed.). United States of America: Manning Publications.

Patton, R. (2000). Software testing (1st ed.). Indiana: Sams Publishing.

Pressman, R. S., & Maxim, B. R. (2015). Software engineering: A practitioner's approach (8th ed.).

Safaat, G. I., & Tjhin, V. U. (2024). Analysis of quality assurance performance in the application of manual testing and automation testing for software product testing. Indonesian Interdisciplinary Journal of Sharia Economics, 7(2), 1987-1996. https://doi.org/10.2139/ssrn.3567634

Salahat, M., et al. (2023). Software testing issues improvement in quality assurance. 2nd International Conference on Business Analytics for Technology and Security, ICBATS 2023.

Seker, S. E. (2014). Software Development Life Cycle. YBS Encyclopedia, 1(2), 2-5.

Srinivas, N., & Jagruthi, D. (2012). Black box and white box testing techniques – A literature review. International Journal of Embedded Systems and Applications, July 2012.

Steegmans, E., et al. (n.d.). Black & white testing: Bridging black box testing and white box testing.

Tassey, G. (2002). The economic impacts of inadequate infrastructure for software testing. RTI for National Institute of Standards and Technology.

The Standish Group International Inc. (2013). Manifesto, CHAOS: Think Big, Act Small.

Uzun, B., & Koruyan, K. (2019). Overview and approaches to status reporting in software testing process. Journal of Management Information Systems, 5(1), 52-63.

Vaishali, B. (2017). Seven fundamental principles of testing.

Vocke, H. (2018). The practical test pyramid.

Vural, Y., & Sağıroğlu, Ş. (2011). Security tests and recommendations in corporate information security. Gazi University Journal of Engineering and Architecture Faculty, 26(1), 89-103. https://doi.org/10.1016/j.theriogenology.2020.12.030

Wallace, D. R. (1986). An overview of computer software acceptance testing. Journal Name.

Whittaker, J. A. (2000). What is software testing? And why is it so hard? IEEE Software, 17(1), 70-79. https://doi.org/10.1109/52.841303

Young, M., & Pezzè, M. (2008). Software testing and analysis: Process, principles and techniques (1st ed.).

Yücalar, F., & Borandağ, E. (2019). Improving quality in software projects: TMMi. AURUM Journal of Engineering Systems and Architecture, 3(2). https://doi.org/10.1016/j.cose.2017.06.015

Zhao, Y., Hu, Y., & Gong, J. (2021). Research on international standardization of software quality and software testing. Proceedings - 2021 IEEE/ACIS 21st International Fall Conference on Computer and Information Science, ICIS 2021-Fall. https://doi.org/10.1109/ICIS.2021.9601902