Fırat Üniversitesi Müh. Bil. Dergisi
34(2), 517-529, 2022

Araştırma Makalesi
https://doi.org/10.35234/fumbd.1076870

Yazılım Güvenlik Açıklarının Evrişimsel Sinir Ağları (CNN) ile Sınıflandırılması

Edanur ÖZDEMİR[1*], İbrahim TÜRKOĞLU[2]
[1] Yazılım Mühendisliği, Teknoloji Fakültesi, Fırat Üniversitesi, Elazığ, Türkiye
[2] Yazılım Mühendisliği, Teknoloji Fakültesi, Fırat Üniversitesi, Elazığ, Türkiye
[*1] edanurozdemir23@gmail.com, [2] iturkoglu@firat.edu.tr

(Geliş/Received: 21/02/2022; Kabul/Accepted: 25/08/2022)

Öz: Kötü niyetli saldırılara karşı yazılım sisteminin doğru şekilde çalışmaya devam etmesini sağlayacak teknik uygulamalar yazılım güvenliği kapsamındadır. Yazılımda bulunan ve saldırıya uğrama riskine sebep olacak zayıflıklar ve kusurlar yazılım güvenlik açığı olarak tanımlanır. Yazılım güvenlik açıkları, bir yazılımın gizliliğini, bütünlüğünü, kullanılabilirliğini, erişilebilirliğini tehlikeye atarak ciddi zararlara sebep olabilir. Yazılım güvenlik açıkları erken aşamada tespit edilip, etkin şekilde yönetilirse sistemin saldırıya uğrama ve hasar görme riski azaltılabilir. Yazılım güvenlik açıklarını otomatik yönetmek için son yıllarda yapay zekâ tabanlı yöntemler kullanılmaya başlanmıştır. Bu çalışmada, güvenlik açıklarının kategorilerinin belirlenerek yönetilmesine fayda sağlamak için derin öğrenmeye dayalı otomatik sınıflandırma yöntemi önerilmiştir. Geliştirilen yöntem, derin sinir ağ modellerinden biri olan evrişimli sinir ağı (ESA) modeli üzerine inşa edilmiştir. Önerilen yöntemin başarımını test etmek için ABD Ulusal Güvenlik Açığı Veri tabanı (NVD) kullanılmıştır. Çalışmada kullanılan ESA modelinin yazılım güvenlik açıklarının otomatik sınıflandırılmasında etkili olduğu gözlemlenmiştir.

Anahtar kelimeler: Yazılım güvenliği, yazılım güvenlik açıkları, yapay zekâ, derin öğrenme.

Classification of Software Vulnerabilities with Convolutional Neural Networks (CNN)

Abstract: The technical applications that will ensure that the software system continues to work correctly against malicious attacks is called software security. Weaknesses and flaws in the software system that cause the risk of being hacked are defined as software vulnerability. Software vulnerabilities can cause serious damage by compromising the confidentiality, integrity, usability and accessibility of a software. If software vulnerabilities are detected at an early stage and managed effectively, the risk of system attack and damage can be reduced. Artificial intelligence-based methods have been used in recent years to automatically manage software security vulnerabilities. In this study, a deep learning-based automatic classification method has been proposed to help identify and manage the categories of security vulnerabilities. The developed method is built on the convolutional neural network (CNN) model, which is one of the deep neural network models. The National Vulnerability Database (NVD) was used to measure the performance of the proposed model. It has been observed that the CNN model used in the study was effective in automatically classifying software safety vulnerabilities.

Key words: Software security, software vulnerabilities, artificial intelligence, deep learning.

1. Giriş

Son zamanlarda internetin yaygınlaşması ve bilişim teknolojilerinin çoğalmasıyla çeşitli endüstrilerin dijital ortamdaki faaliyetleri artmaktadır. Bu durum yazılıma olan bağımlılığı artırmakta ve yazılım sistemlerinin sürdürülmesini gerekli kılmaktadır. Teknolojik gelişmelerin sunmuş olduğu bu geniş kullanım alanı, sistemlerin karmaşıklığını arttırarak saldırganların açık hedefi haline getirmektedir. Buna bağlı olarak güvenlik sorunları giderek daha belirgin hale gelmektedir. Bir yazılımın saldırı veya tehdit altındayken işlevlerini doğru bir şekilde yerine getirmesini sağlayacak şekilde korunmasına yazılım güvenliği denir [1]. Yazılım güvenliği bir yazılımın çalışması için işlevsel olmayan bir gereklilik olarak tanımlanır [2]. Yazılım güvenliğinde amaç, güvenlik saldırılarına karşı yazılımın hatasız ve daha güçlü çalışmasını sağlamaktır. Yazılım güvenlik açıkları, bir yazılımın tasarım ve uygulama aşamalarında meydana gelmektedir [3-4]. Yazılım güvenliği, yazılım projelerinin başarıya ulaşması ve ekonomik değere dönüşmesini doğrudan etkileyen en önemli kalite ölçütlerinden biridir. Yazılımda bulunan her bir güvenlik açığı; gizlilik, erişilebilirlik, kullanılabilirlik ve bütünlük gibi güvenlik nitelikleri üzerinde farklı etkilere sahiptir. Yazılımın güvenliğini sağlamak için yapılan çalışmalar, geliştirmenin sonraki aşamalarında yazılıma entegre edilirse, yazılım geliştirmenin çeşitli aşamalarında güvenlik açıklarının riskini arttırabilir. Artan güvenlik açıkları riski, zaman, maliyet, kalite ve itibar açısından daha yüksek potansiyel riskler oluşturabilir [5]. Bu sebeple önemli ve riskli işlemlerin yapılabildiği yazılım sistemlerinin korunması ve güvenli

[*] Sorumlu yazar: edanurozdemir23@gmail.com. Yazarların ORCID Numarası: [1] 0000-0002-0311-9838, [2] 0000-0003-4938-4167

yazılım geliştirmek en az teknolojinin sağladığı avantajlar kadar dikkate alınması gereken bir konudur. Güvenli yazılım geliştirebilmenin en önemli noktalarından biri yazılım geliştirme yaşam döngüsünün ilk aşamasından itibaren güvenliğin dahil edilerek geliştirilmesidir [6,7]. Yazılım kalitesini arttırmak için yapılan tüm test, ölçüm ve analizler de yazılımın güvenilirliğini arttırmaktadır [6]. Yazılım güvenlik açıklarının türleri, özellikleri, yerleri belirlenerek analiz edilmesi; yazılım geliştiricilerinin, güvenlik uzmanlarının ve araştırmacıların çalışmalarında daha güvenli sistemler geliştirmeleri için fayda sağlamaktadır [8]. Bu nedenle yazılım güvenlik açıkları, güvenlik uzmanları tarafından detaylı bir şekilde sınıflandırılıp, önem derecesi belirlenerek raporlanmaktadır.

Yapay zekâya dayalı yöntemler kullanılarak yazılım güvenlik açıklarının otomatik sınıflandırılması, önem derecesinin tahmini, kaynak kodda bulunan anlamsal ve sözdizimsel güvenlik açıklarının tespit edilmesi gibi çalışmalar giderek artmaktadır [8]. Son yıllarda yazılım güvenliğinin yönetimini sağlayabilmek için kullanılan yapay zekâ yöntemlerinin performansı arttırdığı görülmektedir. Aota ve ark. [9] tarafından yapılan çalışmada, NVD veri tabanı kullanılarak güvenlik açığı açıklamaları sınıflandırılmıştır. Boruta algoritması ile Destek Vektör Makinesi, Lojistik Regresyon, Karar Ağacı ve Rastgele Orman sınıflandırma algoritmaları ayrı ayrı uygulanmıştır. En iyi performans, %96.92 doğruluk değerinde Boruta ve Rastgele Orman algoritmasından elde edilmiştir. Ancak sınıflandırma için veri tabanının daha dengeli ve geliştirilmesi gerektiği önerilmiştir. Han ve ark. [10] yaptıkları çalışmada, bir güvenlik açığının önem derecesini (CVSS puanı) tahmin etmek için derin öğrenme yöntemi kullanmışlardır. NVD veri tabanından alınan güvenlik açığı bilgilerinden oluşan veriler kritik, yüksek, orta, düşük olarak 4 kategoriye ayrılmıştır. Word2vec ve tek katmanlı Evrişimli Sinir Ağı kullanarak geliştirilen modelde sonuç olarak %81.6 f1-ölçütü elde edilmiştir. Rehman ve Mustafa [11] yaptıkları çalışmada yazılım geliştirme yaşam döngüsünün tasarım aşamasında meydana gelen güvenlik açığı türlerini sınıflandırmışlardır. NVD veritabanından elde ettikleri 427 güvenlik açığı açıklaması üzerinde metin ön işleme adımlarını gerçekleştirdikten sonra sınıflandırmak için Destek Vektör Makinesi kullanmışlardır. Sınıflandırmanın doğruluğu 6 güvenlik açığı kategorisi için tatmin edici bulunmuştur. Mısırlı tarafından yapılan çalışmada [12], yazılım güvenilirliğini tahmin edebilmek için Hibrid Bayes ağlarını kullanılmıştır. Yazılım projelerinin sürüm sonrası hatalarını tahmin etmede başarılı sonuç elde edildiği belirtilmiştir. Wu ve ark. [13] yaptıkları çalışmada, derin öğrenme ile yazılım güvenlik açığı tespiti yapabilmek için Linux işletim sisteminden alınan işlev çağrıları üzerinde CNN, LSTM ve CNN-LSTM olmak üzere 3 model kullanmışlardır. Deneysel sonuçlar derin öğrenme modellerinin çok katmanlı algılayıcı gibi klasik modellere göre tahmin doğruluğunun daha yüksek olduğunu göstermiştir. Clemente ve ark. [14], yazılım güvenliğini ölçmek için Rastgele Orman, Naif Bayes, Destek Vektör Makinesi ve Karar ağaçları algoritmaları ile Derin Sinir Ağı modelini karşılaştırmışlardır. Çalışma sonucunda Derin Sinir Ağı'nın daha yüksek doğruluk sonucu verdiği gösterilmiştir. Kaya ve ark. [15] tarafından yapılan çalışmada yazılım güvenlik açıklarını sınıflandırmak için veri dengeleme teknikleri ile Rastgele Orman, Destek Vektör Makinesi, RUSBoosted trees, LDA, SD, AdaBoost, W-kNN sınıflandırma algoritmaları kullanılmıştır. PHPMyAdmin, Drupal ve Moodle projeleriyle ilgili üç genel veri kümesi üzerinde uygulanmıştır. Deneysel sonuçlar, Rastgele Orman algoritmasının küçük veri kümeleri için, RusboostTree algoritmasının büyük veri kümeleri için daha iyi performans sağladığını göstermiştir. Russel ve ark. [16] tarafından C/C++ kaynak kodlarındaki güvenlik açıklarının Evrişimsel Sinir Ağı (ESA) ve Tekrarlı Sinir Ağları derin öğrenme yöntemlerini kullanılarak otomatik algılanması çalışması yapılmıştır. Çalışma sonucuna göre ESA modeli daha iyi performans göstermiştir. Huang ve ark. [17] tarafından yapılan çalışmada yazılım güvenlik açıklarını sınıflandırabilmek için yeni bir derin öğrenmeye dayalı TFI-DNN modeli önerilmiştir. NVD veri tabanı kullanılarak güvenlik açığı açıklamaları üzerinde uygulanan TFI-DNN; TFI-KNN, TFI-SVM, TFI-NB modelleri ile karşılaştırıldığında daha iyi sonuç verdiği gözlemlenmiştir. Davari ve ark. [18] tarafından yapılan çalışmada Bohrbug ve Mandelbug sınıflandırması gerçekleştirilmiştir. Bohrbug, test aşamasında kolayca tekrarlanabilir girdilere dayalı programdaki bir hatadır. Mandelbug, giriş verilerinin sistem durumları ile etkileşimi, işlemlerin zamanlaması, işlem sırası, sistemlerin iç yapısı gibi karmaşık koşullar altında tetiklenebilen bir yazılım hatasıdır. Mozilla Foundation Security Advisory'den toplanan ve metinlerden oluşan 580 veri seti üzerinde belirli güvenlik açığı kategorileri seçilmiştir. C4.5 Karar Ağacı, Rastgele Orman, Lojistik Regresyon ve Naif Bayes makine öğrenme yöntemleri uygulanarak sınıflandırmanın etkinliği değerlendirilmiştir. Sonuç olarak C4.5 Karar Ağacı'nın %69 F-ölçütü ile görünmeyen güvenlik açıkları kategorisini tanımlayabildiği belirtilmiştir. Şahin ve ark. [19] yaptıkları çalışmada derin öğrenme ile SYMbiyotik genetik algoritma kullanan yazılım güvenlik açığı tahmini üzerine çalışmışlardır. Derin özellik öğrenimi için Derin Sinir Ağı kullanılmıştır. Drupal, Moodle ve PHPMyAdmin projelerinden elde edilen veri kümeleri üzerinde model değerlendirilmiştir. Sonuç olarak GRUSYMbiyotik GA-II modelinin güvenlik açığı tahminini geliştirdiği sonucuna varmışlardır. Chernis ve Verma [20], C kaynak kodundaki büyük bir hata yüzdesini yakalamanın nasıl mümkün olduğunun gösterimi üzerine çalışılmıştır. Github'da bulunan savunmasız koda sahip C diliyle yazılmış açık kaynak kodlu 100 proje kullanılmıştır. Sınıflandırmak için Naive Bayes, K-En Yakın Komşu, K-Ortalama, Sinir Ağı, Destek

Vektör Makinesi, Karar Ağacı ve Rastgele Orman dahil olmak üzere birkaç algoritma kullanılmış ve sonuçları yorumlanmıştır.

Bu çalışmada güvenlik açıklarını tespit etme ve yönetimini sağlamak için araştırmacıların, güvenlik uzmanlarının, geliştiricilerin çalışmalarına fayda sağlaması amacıyla yazılım güvenlik açıklarının otomatik sınıflandırılması gerçekleştirilmiştir. Bunun için, 75.870 veri setinden oluşan ABD Ulusal Güvenlik Açığı Veri tabanı'ndan (NVD) elde edilen yazılım güvenlik açığı verileri kullanılmıştır [21]. Çalışmada, öncelikle güvenlik açığı verilerinden elde edilen güvenlik açığı açıklamaları ön işlem ve metin işleme yöntemleri kullanılarak sayısallaştırılmıştır. Sonrasında, büyük veri işlemede başarısı ile ön plana çıkan derin öğrenme yöntemi evrişimli sinir ağı kullanılarak yazılım güvenlik açıklarının, yazılım güvenlik açığı türlerine göre kategorisi tanımlanmıştır. Sonuç olarak önerilen modelin başarım performansı elde edilmiştir.

2. Yazılım Güvenlik Açıkları

Yazılım sistemlerinin çeşitli teknolojiler, kütüphaneler kullanılarak geliştirilmesi büyük ve karmaşık yapıların oluşmasına sebep olmuştur. Bu karmaşıklık yazılım sistemlerinde farklı türde güvenlik açıklarının oluşmasına sebep olmaktadır. Güvenlik açıklarının sebep olduğu risk düzeyi ve önlemek için gereken çözümler, farklı türdeki güvenlik açıkları için farklıdır. Bir sistem için tehlike oluşturan güvenlik açıkları, birden fazla sebepten kaynaklanabilir. Güvenlik açıkları genel olarak; yazılım, donanım, ağ, sunucu, veri tabanı, işletim sistemi, fiziksel ortam, geliştirme ekibi ve yönetim gibi ilgili oldukları alanlara göre sınıflandırılabilir.

Tablo 1'de OWASP 2021 yılı top 10 listesine göre [22] en yaygın güvenlik açıkları belirlenmiştir. OWASP (Açık Web Uygulama Güvenliği Projesi), dünya çapında web uygulama güvenliğini artırmak amacıyla güvenlik uzmanları, araştırmacılar ve geliştiriciler için ücretsiz olarak araçlar, kaynaklar sunan bir topluluktur.

Tablo 1. OWASP 2021 İlk 10'da Bulunan Güvenlik Açıkları Kategorileri

OWASP 2021 İlk 10'da Bulunan Güvenlik Açıkları Kategorileri
A01:2021-Bozuk Erişim Kontrolü (Broken Access Control)
A02:2021-Kriptografik hatalar (Crytographic Failures)
A03:2021-Enjeksiyon (Injection)
A04:2021-Güvensiz Tasarım (Insecure Design)
A05:2021-Güvenlik Hatalı Yapılandırması (Security Misconfiguration)
A06:2021-Kırılgan ve Güncel Olmayan Bileşenler (Vulnerable and Outdated Components)
A07:2021-Tanımlama ve Kimlik Doğrulama Hataları (Identification and Authentication Failures)
A08:2021-Yazılım ve Veri Bütünlüğü Arızaları (Software and Data Integrity Failures)
A09:2021-Güvenlik Günlüğü ve İzleme Arızaları (Security Logging and Monitoring Failures)
A010:2021-Sunucu Tarafı İstek Sahteciliği (Server-Side Request Forgery)

OWASP tarafından kategorilere göre SQL enjeksiyonu A03 kategorisinde, siteler arası komut çalıştırma (XSS) A03 kategorisinde, siteler arası istek sahteciliği (CSRF) A01 kategorisinde, kod enjeksiyonu (Code Injection) A03 kategorisinde, dizin geçişi (directory traversal) A01 kategorisinde, kimlik doğrulama atlama (Authentication Bypass) A01 kategorisinde bulunmaktadır [22]. Bu çalışmada ABD Ulusal Güvenlik Açığı Veri Tabanı'ndan (NVD) [21] elde edilen ve yaygın olan güvenlik açığı türleri araştırılmıştır.

2.1 SQL Enjeksiyonu

Yapılandırılmış Sorgu Dili (SQL), veritabanı yönetimini sağlamak için kullanılan bir alt dildir. SQL sorguları üzerinden veritabanı olan sistemlerdeki açıklardan yararlanarak gerçekleştirilen siber saldırı türüne SQL Enjeksiyonu (SQL Injection) denir. SQL Enjeksiyonu, OWASP 2021 top 10 listesinde en yaygın güvenlik açıkları arasında yer almaktadır [22]. SQL Enjeksiyonu saldırısı ile saldırgan; bir sistemin veri tabanında bulunan verileri okuyabilir, değişebilir, yetkisini yükseltebilir, veritabanını silebilir [23]. Bu sayede kimlik sahtekârlığı, verilerin ifşa edilmesi, işlemlerin geçersiz kılınması ve verileri yok ederek kullanılamaz hale getirilmesi gibi olaylar yaşanabilir [23].

2.2. Siteler Arası Komut Dosyası Çalıştırma

Siteler Arası Betik Çalıştırma (XSS) saldırıları ile web uygulamalarına kötü niyetli komut dosyaları enjekte edilmektedir. XSS saldırıları, genellikle istemci tarafından kötü amaçlı komut dosyasını kullanıcıya göndermek için web uygulamasını kullandığında meydana gelir [24]. Kötü amaçlı komut dosyası ile saldırgan, web uygulamasını kullanan kullanıcının tüm tanımlama bilgilerine, oturum bilgilerine, çerezlere ve diğer hassas bilgilere erişebilir [24]. OWASP 2021 en yaygın güvenlik açıkları listesinde yer almaktadır [22].

2.3. Hizmet Reddi

Bir sunucuya veya uygulamaya belirli kapasitesinin üstünde istek göndererek kullanıcılar için kullanılamaz hale getirilmesine Hizmet Reddi (DDoS) saldırısı denir. DDoS saldırıları sadece kapasite üstü isteklerle sınırlı değildir. Aynı zamanda sistemde bulunan ağ paketlerinden, programlamasından, mantıksal yapısından kaynaklanan güvenlik açıklarını manipüle ederek sistemin kullanıcıları tarafından erişimini engelleyebilir [25]. Birçok türde gerçekleştirilen DDoS saldırıları; yazılım hataları, donanım arızaları, çevresel koşullar, kaynak yetersizliği gibi nedenlerden kaynaklanabilir [26]. Bir sistemin elektrik kaynağının kasıtlı olarak engellenmesi fiziksel DDoS saldırısına örnek verilebilir. DDoS saldırıları hizmet kalitesini ciddi ölçüde düşürebilir.

2.4. Siteler Arası İstek Sahteciliği

Siteler arası istek sahteciliği (CSRF), bir web uygulamasında oturum açmış kullanıcının oturum bilgilerini ve diğer kimlik doğrulama bilgilerini kullanarak, savunmasız bir web uygulamasına sahte HTTP isteği göndermeye zorlayan bir saldırıdır [27]. Web uygulamasına giden isteklerin nasıl ve hangi kaynaktan gönderildiği kontrol edilmeyen sistemlerde meydana gelir. Saldırganlar, sosyal mühendislik teknikleriyle de CSRF saldırısı gerçekleştirilebilir [28]. Başarılı bir CSRF saldırısı sonucunda kullanıcının bilgilerini değiştirme, banka hesabından para transferi, uygulamaya yönetici ekleme gibi işlemler gerçekleştirilebilir. OWASP 2021 en yaygın güvenlik açıkları listesinde yer almaktadır [22].

2.5. Arabellek Taşması

Bir sistemin veri boyutu, ayrılmış olan arabellek kapasitesini aştığında arabellek taşması meydana gelir. Arabellek taşması (Overflow), 1980'lerden beri en popüler güvenlik açıklarından biri olmuştur. Arabellek taşması sonucunda; kullanılabilirlik eksikliği, rastgele kod yürütme, hizmet reddi, yönetici izni gibi tehlikeler meydana gelebilir. Bu tehlikeler sonucunda diğer güvenlik hizmetleri de savunmasız hale getirilebilir [29].

2.6. Kod Enjeksiyonu

Kod enjeksiyonu (Code Injection), saldırgan tarafından bir ağ üzerinden hedef sunucuda kötü amaçlı kod çalıştırmasıyla meydana gelir. Saldırgan hedef sistemi yeterince ele geçirdikten sonra sunucuda bulunan dosyalar, veritabanları gibi tüm bilgilere erişebilir ve sistemin kontrolünün kaybolmasına sebep olabilir [30]. Web uygulamalarında yaygın olarak bilinir ve tespit etmek zordur. OWASP 2021 en yaygın güvenlik açıkları listesinde yer almaktadır [22].

2.7. Dizin Geçişi

Dizin geçişi veya yol geçiş saldırısı (Directory Traversal), saldırganın web kök klasörünün dışında kısıtlı dosya ve dizinlere erişerek web sunucusunun kök dizini dışında komutlar yürütmeyi amaçlayan bir HTTP saldırısıdır. Saldırgan, web sitesi kullanıcısını taklit ederek komutlar çalıştırabilir. Kök dizinden çıkarak dosyalara erişebilir, bilgi edinebilir. OWASP 2021 en yaygın güvenlik açıkları listesinde yer almaktadır [22].

2.8. Kimlik Doğrulama Atlama

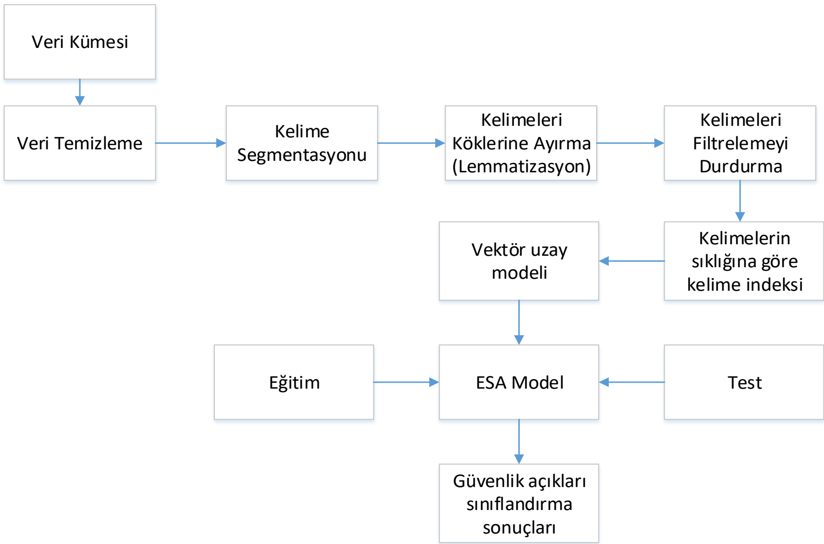
Kimlik doğrulama atlama (Authentication Bypass) güvenlik açığı, saldırganların sistemde bulunan kimlik doğrulama zafiyetinden yararlanarak, kimlik doğrulama aşamasını atlayıp kötü amaçlı birçok işlem gerçekleştirmesine sebep olmaktadır. Saldırgan, yetkili bir kullanıcının ayrıcalıklarıyla sisteme erişimi elde edebilir. OWASP 2021 en yaygın güvenlik açıkları listesinde yer almaktadır [22].

3. Yazılım Güvenlik Açıklarının Sınıflandırılması

Yazılım güvenliği, iç ve dış saldırıları içerir. Saldırganlar genellikle bir yazılım sistemini kendi iç zayıflığından, yani yazılım açıklarından yararlanarak bozmaktadırlar [31]. Güvenlik uzmanları; sistem açıklarının bulgu adı, kategorisi, önem derecesi, etki alanı, etkilenen sistemler, erişim noktası gibi detaylı özelliklerini analiz ederek rapor etmektedir. Analiz edilen özellikler ile yazılım durumu tespit edilerek güvenlik açıkları yönetilmektedir. Yazılım güvenlik açıklarının daha etkin yönetilmesi için güvenlik açığı kategorilerinin otomatik bir şekilde belirlenmesi, güvenlik uzmanlarının ve araştırmacıların iş yükünü azaltarak yapılan işlemlerin daha hızlı ve verimli hale gelmesini sağlayabilir. Ayrıca yazılım sistemlerinin güvenlik sorunlarının çözülmesinde fayda sağlayarak, saldırıya uğrama ve hasar görme riskini de azaltılabilir. Bu nedenle yazılım güvenlik açıklarını yapay zekâ yöntemleri kullanarak sınıflandırma çalışmaları, güvenlik araştırmacıları tarafından daha fazla yapılmaktadır [17]. Yapay zekâ, makinelerin insan beynini taklit etmesini sağlayan bir tekniktir. Yapay zekânın alt alanı olan makine öğrenmesi, veriler üzerinden öğrenme işlemi gerçekleştirerek deneyim kazanan ve bu deneyimlere göre hareket eden sistemlerdir. Derin öğrenme ise çok daha karmaşık problemler için yapay sinir ağlarını kullanan makine öğrenmesinin alt kümesidir.

Şekil 1. Yapay zekâ, makine öğrenmesi, yapay sinir ağı, derin öğrenme arasındaki ilişki

Güvenlik açığı açıklamasına göre güvenlik açıklarını sınıflandırmak bir tür metin sınıflandırmasıdır. Metin sınıflandırması alanında son zamanlarda makine öğrenmesi yöntemleri giderek artmaktadır. Bu nedenle, güvenlik açıklarının otomatik olarak sınıflandırılması sorunu, makine öğrenmesi yöntemleri kullanılarak da çözülebilir. Makine öğrenimi sınıflandırma algoritmaları, birçok alanda başarılı sonuçlar elde etmesine rağmen, çok sayıda güvenlik açığı verisi ve kısa açıklama nedeniyle oluşturulan kelime vektör alanını yüksek boyutlu hesaplayarak seyrek özellikler elde eder. Aynı zamanda, güvenlik açığı sınıflandırma çalışmalarında belirli güvenlik açığı bilgilerini göz ardı ederler ve sınıflandırma doğruluğu yüksek değildir. Bu sebeple bu çalışmada doğal dil alanında ve birçok alanda kullanılarak önemli etkisi görülen derin öğrenme yöntemi kullanılmıştır. Bu bölümde, yazılımın güvenilirlik düzeyinin temel göstergeleri haline gelen yazılım açığı verileri ile derin sinir ağı kullanılarak otomatik sınıflandırma modeli oluşturulmuştur.

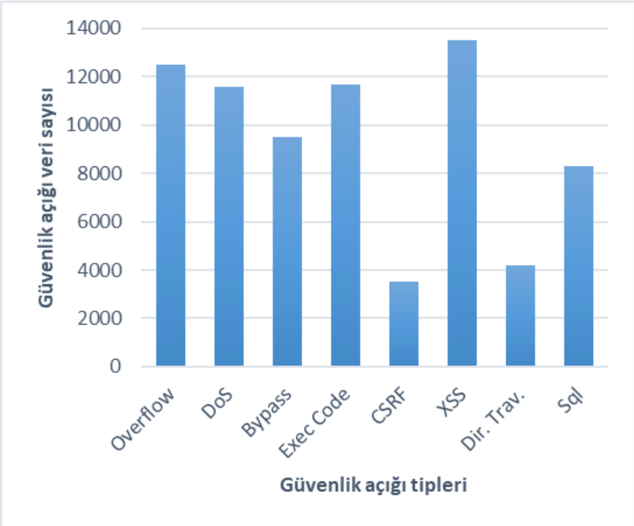


Şekil 2. Yazılım güvenlik açıklarını sınıflandırma sürecinin akış şeması

Şekil 2'de akış şeması verilen model, Windows 10 işletim sistemi ile çalışan Intel (R) Core (TM) i7-3537U işlemci, 2.50 GHz ve 8.00 GB RAM özelliklerine sahip bir bilgisayarda gerçekleştirilmiştir. Model, Anaconda3 Spyder ortamında Python 3.8 ile TensorFlow, Keras kütüphaneleri kullanılarak programlanmıştır. Ulusal Güvenlik Açığı Veri Tabanı'ndan (NVD) [21] alınan güvenlik açığı verilerinin güvenlik açığı açıklamaları ve güvenlik açığı tipi Excel dosyasından çıkarılarak metin ön işleme sürecinden geçirilmiştir. Ön işlem sürecinden geçirilen veriler, derin öğrenmeye dayalı evrişimli sinir ağı modeli ile Ulusal Güvenlik Açığı verileri üzerinde eğitilmiş ve test edilmiştir. Gerçekleştirilen işlemler ve kullanılan algoritmalar bu bölümde detaylı sunulmuştur.

3.1. Veri Kümesi

Çalışmada oluşturulan modeli doğrulamak için kullanılan veri seti, 1999 yılından beri uluslararası kabul görmüş en popüler ve en büyük veri tabanlarından biri olan Ulusal Güvenlik Açığı Veri Tabanı'ndan (NVD) elde edilmiştir [21]. NVD, Ulusal Standartlar ve Teknoloji Enstitüsü (NIST) Bilgisayar Güvenliği Bölümü'nün bir ürünüdür [21]. NVD'de bulunan her bir güvenlik açığı kaydı, güvenlik açığının teknik açıklaması olan kimlik numarası, önem derecesi, yayınlanan tarih, kaynak, güvenlik açığı türü, güvenlik açığı açıklaması gibi birçok bilgi içermektedir. Bu çalışmada, kullanılan güvenlik açığı kayıtları 8 kategoriden oluşmaktadır ve veri setinin toplam sayısı 75.870'dir. Her kategoride bulunan güvenlik açıkları kayıtlarının sayısı değişkendir. Veri setinde bulunan her kategorideki güvenlik açığı kayıtlarının dağılımı Şekil 3'te gösterilmiştir.



Şekil 3. Modelde kullanılan güvenlik açığı tiplerinin dağılımı

Model için güvenlik açığı metin açıklaması ve güvenlik açığı kategori verileri kullanılmıştır. Veri seti, Python'da yazılmış program kodları kullanılarak eğitim için %80 ve test için %20 oranında kullanılmıştır. Veri örnekleri Tablo 2'de gösterilmiştir.

Tablo 2. Modelde kullanılan veri seti örnekleri

CVE_ID	Vulnerability Type	Description
CVE-2019-1010302	DoS	Tildeslash Monit Version 5.25.2 and earlier is affected by: Buffer Over-read. The impact is: Disclosure of memory contents in an HTTP response, and Denial of Service…
CVE-2018-6878	XSS	Cross Site Scripting (XSS) exists in the review section in PHP Scripts Mall Hot Scripts Clone Script Classified 3.1 via the title or description field.
CVE-2019-1010163	Exec Code	Socusoft Co Photo 2 Video Converter 8.0.0 is affected by: Buffer Overflow - Local shell-code execution and Denial of Service…
CVE-2019-16119	Sql	SQL injection in the photo-gallery (10Web Photo Gallery) plugin before 1.5.35 for WordPress exists via the admin/controllers/Albumsgalleries.php album_id parameter.

3.2. Ön İşlem

Ön işlem, modelin başarısını etkileyen önemli süreçlerden biridir. Ön işlem sürecinde verinin analiz edilebilir yapıya dönüştürülmesi için veriyi temizleme, eksik veriyi tamamlama, veriyi düzeltme, veri boyutunu küçültme, veriyi normalleştirme, veriyi sayısallaştırma gibi işlemler yapılır.

Model de ön işlem sürecinin ilk aşamasında veri temizleme işlemi gerçekleştirilmiştir.

Veri temizleme (Data clean): Bu aşamada güvenlik açığı açıklamasında bulunan güvenlik açığı kategorilerine ait anahtar kelimeler kaldırılmıştır. Örnek olarak; “SQL Injection”, “cross site scripting”, “code execution”, “bypass”, “buffer overflow”, “CSRF”, “XSS”, “DoS”, “Directory Traversal”, “denial of service”, “vulnerability” gibi kelimeler güvenlik açığı açıklaması verilerinden kaldırılmıştır.

Kelime bölütleme (Word segmentation): Ön işlem sürecindeki en önemli aşamalardan biridir. Kelime bölütleme, bir metni kelimelere bölme işlemidir. Bu aşama da, metinlerdeki kelimelerin hepsi öncelikle küçük harfe dönüştürülmüştür. Güvenlik açığı açıklaması verilerinde bulunan noktalama işaretleri, boşluk gibi ifadeler kaldırılarak tek kelime haline getirilmiştir. Daha sonra verilerin morfolojik analizi yapılarak kök forma dönüştürülmesi için kelimeleri köklerine ayırma (lemmatizasyon) işlemi gerçekleştirilmiştir.

Kelimeleri köklerine ayırma (Lemmatization): Bir kelimenin farklı çekimli biçimlerini tek bir öğe olarak analiz edilebilmeleri için bir araya getirme işlemidir. Kök oluşturmaya benzer ancak kelimelere bağlam getirir. Böylece benzer anlama sahip kelimeleri bir kelimeye bağlar. Bu işlemden sonra kelime filtrelemeyi durdurma işlemi gerçekleştirilmiştir.

Kelime filtrelemeyi durdurma (Stop word): Metinde sıkça görünen ve metin bilgilerinin içeriğine veya sınıflandırılmasına çok az katkıda bulunan veya hiç katkı sağlamayan sözcüklerin temizlenmesini sağlar. Bu modelde, WordNetLemmatizer kütüphanesi kullanılmıştır. Ön işlem aşamasından sonra temizlenen güvenlik açığı açıklama verilerinden örnek veriler Tablo 3'de gösterilmiştir.

Tablo 3. Ön işlem aşamasında temizlenen verilerden sonra elde edilen veri örnekleri

Güvenlik Açığı Türü	Güvenlik Açığı Açıklama Verileri
Bypass	*cybozu garoon allows remote authenticated attacker access restriction alter content application address without modify privilege via application address*
Exec Code	*apache netbeans incubating netbeans proy autoconfiguration pac interpretation remote command rce using nashorn script engine environment javascript eecution proy autoconfiguration leak privileged object used circumvent eecution limit different script engine used eecution limit place vector allow*

Gerçekleştirilen işlemlerden sonra Keras kütüphanesine ait Tokenizer sınıfı ve Gömme katmanı (Embedding layer) kullanılarak kelimelerin indekslerine göre vektör uzay modeli oluşturulmuştur.

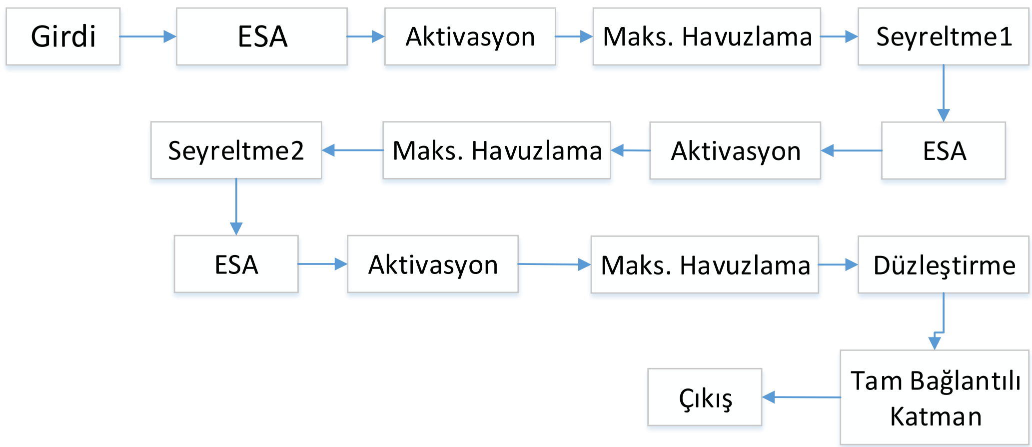
Tokenizer: Bu yöntemde öncelikle fit_on_texts() metodu ile cümleler kelimelere ayrılarak kelime yoğunluğuna göre dizin oluşturulmuştur. Her kelimenin benzersiz bir değer alması için kelime->indeks şeklinde bir veri kümesi oluşturulmuştur. Daha sonra texts_to_sequences() metodu ile her benzersiz kelime belirtecine karşılık gelen tamsayı değeri oluşturulup, dizi haline getirilmiştir. Veri setinde bulunan her metin aynı uzunluğa sahip değildir. Oluşturulan tüm dizilerin aynı uzunluğa sahip olması için pad_sequences metodu kullanılarak belirlenen maksimum kelime uzunluğuna göre her dizinin başına 0 doldurulmuştur.

Gömme Katmanı (Embedding Layer): Gömme katmanı, her kelimeyi tanımlanmış boyutta ve sabit uzunlukta gerçek değerlere sahip vektöre dönüştürülmesini sağlar. Çalışmada gerçekleştirilen modelde pad_sequences() metodu ile tam sayılardan oluşan dizilerde bulunan her kelimeyi sabit uzunlukta vektöre dönüştürmek için gömme katmanı kullanılmıştır. Böylece kelimelerin boyutu küçültülmüştür.

3.3 Evrişimli Sinir Ağı-ESA (CNN) ile Sınıflandırma

Çok katmanlı algılayıcıların bir türü olan evrişimli sinir ağları bir görüntü verisini alabilen, görüntüdeki çeşitli piksellere özellikler tanımlayan ve birini diğerinden ayırt edebilen bir derin öğrenme algoritmasıdır. Görüntü verilerini öğrenmek için tasarlanmıştır [32]. Görüntü ve ses işleme, doğal dil işleme, biyomedikal gibi birçok alanda kullanılmaktadır. Evrişimli sinir ağlarının mimarisi genellikle giriş katmanı, evrişim katmanı, havuzlama katmanı, tam bağlantılı katman ve çıkış katmanından oluşmaktadır. ESA'nın evrişim katmanında, yakındaki piksellerden anlamsal olarak benzer özellikleri öğrenmek için filtreleme işlemi gerçekleştirir. ESA modellerinde birden fazla evrişim katmanı bulunabilir. Havuzlama katmanı, evrişim katmanında belirlenen özelliklerin uzamsal boyutunu küçültmekten sorumludur. Boyutsallık azaltmada amaç, verileri işlemek için gereken hesaplama gücünü azaltmaktır. Ayrıca bu katmanda baskın özellikler çıkarılarak modelin etkin bir şekilde eğitilmesi süreci sürdürülür. Tam bağlantılı katman, öğrenilen özellikleri, etiket uzayına eşleyerek aktivasyon fonskiyonu aracılığıyla sınıflandırma işlemini gerçekleştirir. ESA'nın görüntü verisindeki filtreleme işleminde yakındaki piksellerden özellikleri öğrenebilmesi, doğal dil işleme görevlerini de kolaylaştırabilir. Örneğin, metin sınıflandırma işleminde, bir cümleye uygulanan ESA filtreleri, cümle içerisindeki kelimeleri, vektörlerin bulunduğu bağlamsal özellik uzayındaki yerel bağlamsal özellik vektörlerine yansıtabilir [32].

Çalışmada oluşturulan modelde de evrişimli sinir ağı (ESA) kullanılmıştır. Güvenlik açığı açıklamasının vektörleştirme işleminden sonra, doğal dille tanımlanan güvenlik açığı metin verileri, makine tarafından tanınabilen ve istatistiksel öğrenme yoluyla ifade edilebilen veri yapısına dönüştürülmüştür. ESA modelini oluşturmak için derin öğrenme kütüphanesi TensorFlow kullanılmıştır. Çalışmada oluşturulan ESA modelinin yapısı Şekil 4'te gösterilmiştir.

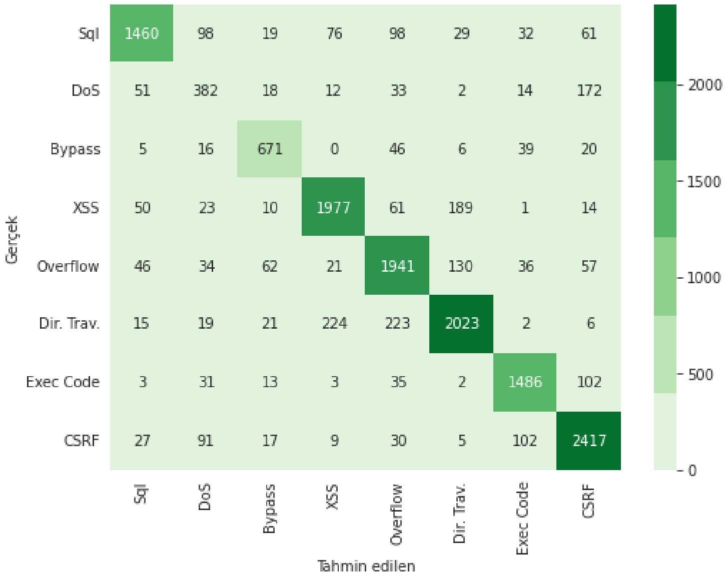


Şekil 4. Modelde kullanılan evrişimli sinir ağı

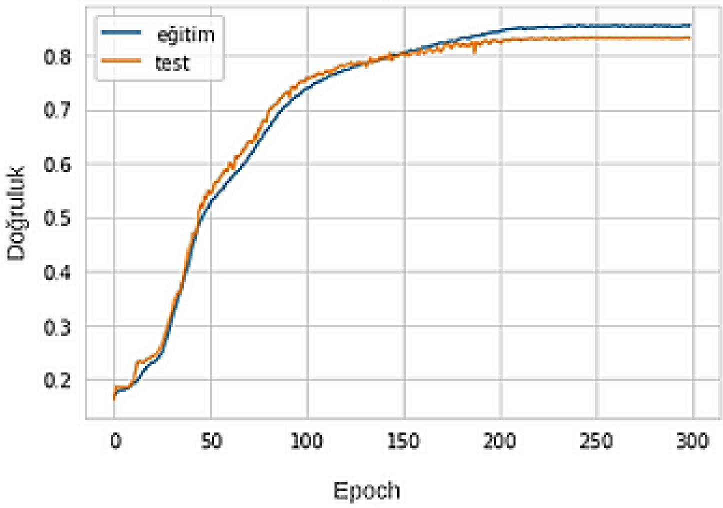
ESA modelini eğitmek için eğitim seti kullanılmıştır. Test için ayrılan veriler model performansını değerlendirmek için kullanılmıştır. Oluşturulan modelde 3 tane evrişim katmanı kullanılmıştır. Veri seti metinlerden oluştuğu için Keras kütüphanesine ait Conv1D katmanı kullanılmıştır. Evrişim işlemi için sırasıyla 128, 64, 32 çıktı filtresi kullanılmıştır. Evrişim işleminde özellik haritası üzerinde açıklayıcı özelliklerin çıkarılması için her bir evrişim katmanında konvolüsyon filtresinin değeri (kernel_size) 4x4 olarak ayarlanmıştır. İşlem yükü daha az olduğundan aktivasyon fonksiyonu olarak ReLU kullanılmıştır. Daha sonra modelin hesapsal boyutunu azaltmak için havuzlama katmanı (MaxPooling1D) kullanılmıştır. Havuzlama katmanı, özellik haritası üzerinde belirlenen filtre boyutunda pencereler seçip o alandaki en büyük değeri almamızı sağlar. Aralarda ise modelin aşırı öğrenmesini engellemek amacıyla Seyreltme (Dropout) katmanı kullanılmıştır. Seyreltme katmanı ile belirlenen değerde girdi rastgele seçilerek sıfıra eşitlenir böylece modelin veriyi ezberlemesinin önüne geçilir. Son olarak sinir ağını beslemesi için Düzleştirme (Flatten) işlemi uygulanmıştır. Tam bağlantılı katmanda 128 düğüm ve ReLU aktivasyon fonksiyonu kullanılmıştır. Sınıflandırma işlemi için 8 farklı sınıf bulunmaktadır. Çoklu sınıflandırma gerçekleştirdiğimiz için çıkış katmanında 8 düğüm ve aktivasyon fonksiyonu olarak softmax kullanılmıştır.

4. Bulgular ve Tartışma

Çalışmada oluşturduğumuz ESA modelinin performansını değerlendirmek için çok sınıflı karışıklık matrisi kullanılmıştır. Güvenlik açığı veri setini sınıflandırmak için kullanılan modelin karışıklık matrisi Şekil 5'te ve doğruluk değerine ait grafiği Şekil 6'da gösterilmiştir.



Şekil 5. Modelin Karışıklık Matrisi



Şekil 6. Modelin Doğruluk Grafiği

Bu çalışmada, metin sınıflandırma alanında yüksek başarı gösteren Evrişimli sinir ağı önerilmiştir. Şekil 6'da görüldüğü gibi modelin eğitimi belli bir eğitim tur (epoch) değerine kadar doğrusal bir şekilde artmaktadır. Evrişimli sinir ağı'nın güvenlik açığı sınıflandırmasındaki performansında %84 doğruluk değeri sonucuna ulaşılmıştır. Deneysel sonuçlar Tablo 4'te gösterilmiştir. Elde edilen sonuçları literatürde bulunan çalışmalarla karşılaştırmak doğru bir sonuç vermeyecektir. Çünkü kullandığımız veri setinde bulunan güvenlik açığı türleri ve sayıları bu alanda yapılan çalışmalardaki veri seti ile aynı değildir. Ayrıca oluşturduğumuz model yapısı ise literatürde bulunmamaktadır.

Tablo 4. ESA modelin deneysel sonuçları

Model	Kesinlik (Precision)	Geri Çağırma (Recall)	F1-Skor (F1-score)	Test Veri Sayısı
Sql	0.88	0.78	0.94	1873
Bypass	0.81	0.84	0.82	803
DoS	0.55	0.56	0.55	684
XSS	0.85	0.85	0.85	2325
Overflow	0.79	0.83	0.81	2327
Directory Traversal	0.85	0.80	0.82	2533
Execution Code	0.87	0.89	0.88	1675
CSRF	0.85	0.90	0.87	2698
Doğruluk (Accuracy)			0.84	14918
Makro Ortalama (Macro Average)	0.81	0.80	0.80	14918
Ağırlıklı Ortalama (Weighted Average)	0.83	0.84	0.84	14918

Modelde kullandığımız güvenlik açığı veri sayısında kategoriler arasında dengesiz bir dağılım bulunmaktadır. Bu sebeple modeli doğrulamak için ayrılan test veri sayısı ise her güvenlik açığı türü için farklıdır. Tablo 4'te görüldüğü gibi her güvenlik açığı kategorisinin kesinlik, geri çağırma, f1-skor değerleri değişmektedir.

Sonuç olarak; 75.870 veri setinde ESA modelinin güvenlik açığı otomatik sınıflandırma performansı ile %84 doğruluk değeri elde edilmiştir.

5. Sonuçlar

Güvenlik açıkları, farklı türdeki yazılım ve donanım sistemlerinde farklı düzeylerde bulunur. Bu sebeple güvenlik açıklarından tamamen kaçınmak zordur. Yazılım güvenliğini sağlamak için her bir güvenlik açığı için farklı çözüm yöntemleri gerçekleştirilmektedir. Bir güvenlik açığı türünü önleyebilmek için türlerinin, özelliklerinin, yerlerinin analiz edilmesi araştırmacılara ve geliştiricilere yardımcı olmaktadır. Yazılımın güvenlik açıklarının detaylı bir şekilde analiz edilmesi, firmaların güvenlik açıklarından kaçınmalarını sağlayabilir. Güvenlik açıklarını otomatik bir şekilde analiz etmek için son yıllarda yapay zekâ tabanlı çeşitli metodolojiler vardır.

Bu çalışmada güvenlik açığı türlerinin otomatik tespit edilmesi için çoklu metin sınıflandırma tekniği kullanılmıştır. Çalışmanın amacı, yazılımların güvenlik performansını iyileştirmek için güvenlik açıklarını derin sinir ağları ile ait oldukları sınıflara göre otomatik olarak daha iyi analiz etmek ve yönetmektir. Oluşturulan otomatik sınıflandırma modelinde evrişimli sinir ağı kullanılmıştır. NVD [21] veritabanından alınan toplam 75.870 ve 8 kategoriden oluşan güvenlik açıkları bilgilerinin bulunduğu veriler, güvenlik açığı açıklamaları kullanarak ait oldukları kategorilere göre sınıflandırılmıştır. Sınıflandırma sonucunun %84 doğruluk değerinde performansa sahip olduğu görülmüştür. Çalışmada evrişimli sinir ağı (ESA) kullanılarak oluşturulan modelde, güvenlik açıklarını türlerine göre sınıflandırmada olan etkinliği gösterilmiştir.

Gelecek çalışmada veri setinde bulunan sınıf dengesizliğinin giderilmesi ve farklı metin sınıflandırma teknikleri ve farklı derin öğrenme algoritmaları ile çalışmak hedeflenmektedir.

Teşekkür

Bu çalışma TEKF.21.14 nolu Proje ile Fırat Üniversitesi Bilimsel Araştırmalar Projesi Birimi tarafından desteklenmiştir.

Kaynaklar

[1] Bulut GF. Predicting Software Vulnerabilities Using Topic Modeling With Issues, MSc, İstanbul Technical University, İstanbul, 2019.

[2] Dangler YJ. Categorization of Security Design Patterns, MSc, East Tennessee State University, Sam Wilson Hall, 2013.

[3] Hosseinzadeh S, Rauti S, Lauren S, Makela M.J, Holvitie J, Hyrynsalmi S, Leppanen, V. Diversification and obfuscation techniques for software security: A systematic literature review, Information and Software Technology 2018; 104(1): 72-93.

[4] Baran O. Yazılım Güvenliği Sistemlerinin İncelenmesi Ve Bir Yazılım Koruma Uygulaması, Yüksek Lisans Tezi, Gazi Üniversitesi, Ankara, 2011.

[5] Sarıman G. Yazılım Güvenliği Test Ve Değerlendirme Aracı Geliştirilmesi, Doktora Tezi, Süleyman Demirel Üniversitesi, Isparta, 2015.

[6] Kaçtıoğlu S., Keskinkılıç M., Kahveci F. Yazılım Kalitesi Faktörü Olarak Yazılım Güvenliğinin Öğrenci Bilgi Sisteminde Rıps Testi İle Değerlendirilmesi. Atatürk üniversitesi İktisadi ve İdari Bilimler Dergisi, 2019; 33(4): 1261-1278.

[7] Assal H., Chiasson S. Security in the Software Development Lifecycle. Symposium on Usable Privacy and Security (SOUPS), August 12–14, 2018; Baltimore, MD, USA. 281-294.

[8] Hanif, H., Md Nasir, M. H. N., Ab Razak, M. F., Firdaus, A., & Anuar, N. B. The rise of software vulnerability: Taxonomy of software vulnerabilities detection and machine learning approaches. Journal of Network and Computer Applications, 2021; 179(9).

[9] Aota, M., Kanehara, H., Kubo, M., Murata, N., Sun, B., & Takahashi, T. Automation of Vulnerability Classification from its Description using Machine Learning. 2020 IEEE Symposium on Computers and Communications (ISCC), November 10, 2020; Tokyo, Japan.

[10] Han, Z., Li, X., Xing, Z., Liu, H., & Feng, Z. Learning to Predict Severity of Software Vulnerability Using Only Vulnerability Description. 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), September 17-22 2017; Shanghai, China. 125-136.

[11] Rehman, S., Mustafa, K. Software Design Level Vulnerability Classification Model. International Journal of Computer Science and Security (IJCSS), 2012; 6(4): 238-255.

[12] Mısırlı A. Modelling Software Reliability Using Hybrid Bayesian Networks, Doktora Tezi, Boğaziçi Üniversitesi, İstanbul, 2012.

[13] Wu F, Wang J, Liu J, Wang W. Vulnerability Detection with Deep Learning. 3rd IEEE International Conference on Computer and Communications, 2017; Nanjing, China. 1298-1302.

[14] Clemente JC, Jaafar F, Malik Y. Is Predicting Software Security Bugs using Deep Learning Better than the Traditional Machine Learning Algorithms?, IEEE International Conference on Software Quality, Reliability and Security, 2018; Lisbon.; 95-102.

[15] Kaya A, Keceli AS, Catal C, Tekinerdogan B. The impact of feature types, classifiers, and data balancing techniques on software vulnerability prediction models. Journal of Software: Evolution and Process, 2019; 31(1): 1-25.

[16] Russell R, Kim L, Hamilton L, Lazovich T, Harer J, Ozdemir O, McConley M. Automated Vulnerability Detection in Source Code Using Deep Representation Learning, IEEE International Conference on Machine Learning and Applications, 2018; Canada; 757-762.

[17] Huang G, Li Y, Wang Q, Ren J, Cheng Y, Zhao XA. Automatic Classification Method For Software Vulnerability Based On Deep Neural Network, IEEE Access 2019; 1(1): 1-9.

[18] Davari, M., Zulkernine, M., & Jaafar, F. An Automatic Software Vulnerability Classification Framework. 2017 International Conference on Software Security and Assurance (ICSSA), July 24-25 2017; Altoona, PA, USA; 44-49.

[19] Şahin, C. B., Dinler, Ö. B., & Abualigah, L. Prediction of software vulnerability based deep symbiotic genetic algorithms: Phenotyping of dominant-features. Applied Intelligence, 2021; 51(11):8271-8287.

[20] Chernis, B., & Verma, R. Machine Learning Methods for Software Vulnerability Detection. Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics, March 21, 2018; Tempe, AZ, USA. 31-39.
[21] NVD, https://www.cvedetails.com/vulnerabilities-by-types.php, Erişim: 11-1-2021.
[22] OWASP Top Ten, https://owasp.org/www-project-top-ten/, Erişim: 20-07-2022.
[23] https://owasp.org/www-community/attacks/SQL_Injection, Erişim Tarihi: 7.07.2021.
[24] https://owasp.org/www-community/attacks/xss/, Erişim Tarihi: 7.07.2021.
[25] https://owasp.org/www-community/attacks/Denial_of_Service, Erişim Tarihi: 7.07.2021.
[26] Wood AD, Stankovic JA. Denial of service in sensor networks. Computer 2002; 35(10), 54–62.
[27] https://owasp.org/www-community/attacks/csrf, Erişim Tarihi: 7.07.2021.
[28] Yadav P, Parekh CD. A report on CSRF security challenges & prevention techniques. 2017 International Conference on Innovations in Information, Embedded and Communication Systems, March 17-18, 2017; Coimbatore, India. 1-4.
[29] https://owasp.org/www-community/attacks/Buffer_overflow_attack, Erişim Tarihi: 7.07.2021.
[30] Oriyano SP, Shimonski R. Web Application Attacks. Client-Side Attacks and Defense, 2018; 195–221.
[31] Wang A, Wang H, Guo M, Xia M. Security metrics for software systems. Proceedings of the 47th Annual Southeast Regional Conference, 2009; Marietta, ABD. 1-6.
[32] Lin, G., Wen, S., Han, Q.-L., Zhang, J., & Xiang, Y. Software Vulnerability Detection Using Deep Neural Networks: A Survey. Proceedings of the IEEE, 2020; 108(10): 1825-1848.