KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

DOCKER İMAJLARININ RİSK TABANLI GÜVENLİK ANALİZİ

SÜLEYMAN ERGEN

KOCAELİ 2024

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

DOCKER İMAJLARININ RİSK TABANLI GÜVENLİK ANALİZİ

SÜLEYMAN ERGEN

Prof.Dr./Doç.Dr./Dr./Öğr. Üyesi **Danışman, Kocaeli Üniv.**	**.....**
Prof.Dr./Doç.Dr./Dr./Öğr. Üyesi **Jüri Üyesi, Kocaeli Üniv.**	**.....**
Prof.Dr./Doç.Dr./Dr./Öğr. Üyesi **Jüri Üyesi, İTÜ**	**.....**
Prof.Dr./Doç.Dr./Dr./Öğr. Üyesi **Jüri Üyesi, Sakarya Üniv.**	**.....**
Prof.Dr./Doç.Dr./Dr./Öğr. Üyesi **Jüri Üyesi, İstanbul Üniv.**	**.....**

Tezin Savunulduğu Tarih: 28.06.2024

ETİK BEYAN VE ARAŞTIRMA FONU DESTEĞİ

Kocaeli Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez/proje çalışmasında,

- Bu tezin/projenin bana ait, özgün bir çalışma olduğunu,
- Çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamasında bilimsel etik ilke ve kurallara uygun davrandığımı,
- Bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi,
- Bu çalışmamın Kocaeli Üniversitesi'nin abone olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş ölçütlere uygun olduğunu,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Tezin/projenin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez/proje çalışması olarak sunmadığımı,

beyan ederim.

☑ Bu tez/proje çalışmasının herhangi bir aşaması hiçbir kurum/kuruluş tarafından maddi/alt yapı desteği ile desteklenmemiştir.

☐ Bu tez/proje çalışması kapsamında üretilen veri ve bilgiler ..... tarafından ..... no'lu proje kapsamında maddi/alt yapı desteği alınarak gerçekleştirilmiştir.

Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

.....

Süleyman ERGEN

YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI

Fen Bilimleri Enstitüsü tarafından onaylanan lisansüstü tezimin/projemin tamamını veya herhangi bir kısmını, basılı ve elektronik formatta arşivleme ve aşağıda belirtilen koşullarla kullanıma açma izninin Kocaeli Üniversitesi'ne verdiğimi beyan ederim. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin/projemin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanımı bana ait olacaktır.
Tezin/projenin kendi özgün çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin/projenin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.
Yükseköğretim kurulu tarafından yayınlanan ***"Lisanüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"*** kapsamında tezim aşağıda belirtilen koşullar haricinde YÖK Ulusal Tez Merkezi/ Kocaeli Üniversitesi Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

☐ Enstitü yönetim kurulu kararı ile tezimin/projemin erişime açılması mezuniyet tarihinden itibaren 2 yıl ertelenmiştir.
☐ Enstitü yönetim kurulu gerekçeli kararı ile tezimin/projemin erişime açılması mezuniyet tarihinden itibaren 6 ay ertelenmiştir.
☑ Tezim/projem ile ilgili gizlilik kararı verilmemiştir.

.....

Süleyman ERGEN

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışması, Docker imajlarındaki güvenlik risklerinin tespit edilmesi ve zaman içerisindeki değişimi ve Docker imaj tarama araçlarının efektifliğini değerlendirmek amacı ile gerçekleştirilmiştir.

Tez çalışmamın her aşamasında bana desteğini esirgemeyen, çalışmalarımı yönlendiren, bana güvenen ve beni yüreklendiren hocam Dr. Öğr. Üyesi İrfan KÖSESOY'a en içten teşekkürlerimi sunarım. Sizin rehberliğiniz ve katkılarınız olmadan bu çalışmayı tamamlamam mümkün olmazdı.

Haziran - 2024 Süleyman ERGEN

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar

API	:	Application Programming Interface, Uygulama Programlama Arayüzü
CLI	:	Command Line Interface; Komut satırı arabirimi
CNCF	:	Cloud Native Computing Foundation
CPU	:	Central Process Unit; Merkezi işlem birimi
CVE	:	Common Vulnerabilities and Exposures; Ortak Zaafiyetler ve Açıklar
CVSS	:	Common Vulnerability Scoring System; Ortak Güvenlik Açığı Puanlama Sistemi
DHR	:	Detection Hit Ratio; Tespit İsabet Oranı
IT	:	Information Technology; Bilgi Teknolojisi
OS	:	Operating System; İşletim Sistemi
RAM	:	Random Access Memory; Rastgele Erişimli Hafıza
RHEL	:	Red Hat Enterprise Linux; Red Hat Kurumsal Linux
VSM	:	Vulnerability Scanning Metric, Zafiyet Tarama Metriği
YZ	:	Yapay Zeka

DOCKER İMAJLARININ RİSK TABANLI GÜVENLİK ANALİZİ

ÖZET

Docker 2013 yılında tanıtıldığı tarihten itibaren, bilişim sektöründe oldukça popüler bir standart haline geldi. Bunun en büyük nedeni Docker'ın geleneksal sanallaştırma teknolojilerine göre çeşitli avantajlar sağlaması oldu. Bu avantajlar sayesinde Docker imajları günümüzde bulut teknolojilerinin ve bulut teknolojilerinde kullanılan uygulamaların temelini oluşturmaktadır.

Birçok kullanım alanı ve avantajı olmasına rağmen Docker imajları çeşitli güvenlik riskleri içermektedir. Yamalanmamış Docker imajlarındaki güvenlik açıkları kurumlar için ciddi tehditler oluşturmakta, mali kayıplara ve özel verilerin ifşa edilmesine neden olabilmektedir. Ayrıca, kötü amaçlı konteyner imajları kullanıldığında bu güvenlik riskleri daha da kötüleşebilmektedir.

Docker imajları konteynerli uygulamaların temelini oluşturduğundan, bu imajların içindeki yapısal zafiyetler tüm kurulumu veya uygulamayı saldırıya açık hale getirebilmektedir. Dahası, yazılım geliştirmenin dinamik doğası gereği risk ortamı sürekli değişmektedir. Her gün yeni güvenlik açıkları keşfedilmekte ve bu açıkları yamamak için güvenlik güncellemeleri yayınlanmaktadır. Bu süreç, yeni keşfedilen güvenlik açıkları hakkında bilgi sahibi olmayı ve imajları zafiyetlere karşı efektif araçlar ile taramayı içermektedir.

Tez kapsamında iki önemli çalışma yapıldı. Çalışmaların ilki Docker imajlarının taranmasında kullanılan güvenlik tarama araçlarının efektifliğinin ölçülmesi ve başarılarının kıyaslanmasıdır. Bu çalışmada ölçümler literatürde kullanılan değerlendirme metrikleri yanında tezde önerilen iki yeni metrik kullanılarak yapıldı. İkinci çalışmada ise Docker imajlarının sürümleri arasında zamansal iyileştirmeler araştırıldı. Bu amaçlar doğrultusunda Docker Hub'dan en çok kullanılan 439 docker imajı analiz edildi. Bu imajlar ilk çalışma kapsamında Trivy, Grype ve Snyk araçları ile taranıp değerlendirildi. İkinci çalışma kapsamında ise her bir imaj için en az altı ay arayla üç farklı versiyon seçildi ve aynı araçlar ile toplamda 1092 tarama gerçekleştirdi. Bu çalışmada değişen zaman dilimlerindeki sonuçlar analiz edilerek, Docker imajlarının güvenlik açığı değişimine ilişkin analizler yapıldı.

Anahtar Kelimeler: Docker, Güvenlik, Güvenlik Açığı Tespiti, Docker İmajları, Docker Güvenliği, Docker İmaj Tarama, Konteyner Güvenliği.

RISK BASED SECURITY ANALYSIS OF DOCKER IMAGES

ABSTRACT

Since its introduction in 2013, Docker has become a very popular standard in the IT industry. The biggest reason for this is that Docker provides various advantages over traditional virtualisation technologies. Thanks to these advantages, Docker images today form the basis of cloud technologies and applications used in cloud technologies.

Although they have many uses and advantages, Docker images contain various security risks. Security vulnerabilities in unpatched Docker images pose a serious threat to organisations and can lead to financial losses and disclosure of private data. In addition, these security risks can be worsened when malicious container images are used.

Since Docker images form the basis of containerised applications, structural weaknesses within these images can make the entire installation or application vulnerable to attack. Moreover, due to the dynamic nature of software development, the risk environment is constantly changing. New vulnerabilities are discovered every day and security updates are released to patch these vulnerabilities. This process involves being informed about newly discovered vulnerabilities and scanning images with effective tools against vulnerabilities.

Two important studies were carried out within the scope of the thesis. The first study is to measure the effectiveness of security scanning tools used in the scanning of Docker images and to compare their effectiveness. In this study, measurements were made using the evaluation metrics used in the literature as well as two new metrics proposed in the thesis. In the second study, temporal improvements between versions of Docker images were investigated. For these purposes, 439 most used docker images from Docker Hub were analysed. These images were scanned and evaluated with Trivy, Grype and Snyk tools in the first study. In the second study, three different versions were selected for each image at least six months apart and a total of 1092 scans were performed with the same tools. In this study, the results in varying time periods were analysed and analyses were made regarding the vulnerability change of Docker images.

Keywords: Docker, Security, Vulnerability Detection, Docker Images, Docker Security, Docker Image Scanning, Container Security.

1. GİRİŞ

Docker, geliştiricilerin uygulamalarını ve bağımlılıklarını tek bir pakette bir araya getirmelerini sağlayan ve uygulamanın Linux tabanlı herhangi bir ortamda sorunsuz çalışmasını sağlayan bir konteyner teknolojisidir (Haque ve diğ., 2020). Docker 2013 yılında PyCon'da tanıtıldı ve aynı yılın Mart ayında Solomon Hykes (Hykes, 2013) tarafından açık kaynaklı hale getirildi. Bu tarihten itibaren, bilişim ve teknoloji (IT) sektöründe oldukça popüler bir endüstri standardı haline gelmiştir. Bunun en büyük nedeni Docker imajlarının geleneksel sanal makinelere ve sanallaştırma teknolojisine göre çeşitli avantajlar sağlamasıdır.

Docker imajlarının kullanıma sunulması modern yazılım geliştirmede önemli bir başarıdır (Mikkonen ve diğ., 2022). Docker imajları klasik sanal makinelere göre çeşitli avantajlar sağlamaktadır (Zhang ve diğ., 2018). Docker imajları sanal makinelere kıyasla çok küçüktür, çünkü sanal makineler tüm işletim sistemini içermekte, ancak Docker imajları sadece uygulama ve uygulamanın çalışması için gerekli olan kütüphaneleri ve bağımlılıkları içermektedir. Docker imajları Linux tabanlı herhangi bir işletim sisteminde çalışabilir çünkü Docker imajların harici bir bağımlılığı yoktur. Birden fazla sanal makine başlatıldığında, her biri için yarı bir Linux çekirdeği oluşturulmaktadır. Bu, sanal makinelerin daha fazla kaynak (RAM ve CPU) kullandığı anlamına gelir. Ancak, docker imajları için durum böyle değildir, çalışan tek bir Linux çekirdeği birden fazla Docker imajını çalıştırmak için yeterlidir.

Docker imajları ayrıca mükemmel/gelişmiş ölçeklenebilirlik sağlamaktadır (Khan, 2017). Docker Swarm, Hashicorp Nomad, Apache Mesos ve Kubernetes otomatik ölçeklendirme sağladığı için Docker imajlarını ölçeklendirmek sanal makineleri ölçeklendirmekten daha kolaydır. Ayrıca, Docker imajları uygulamaların hızlı ve sürekli dağıtımını sağlar. Konteyner orkestrasyon araçları ile Docker imajları sanal makinelere kıyasla çok hızlı bir şekilde canlıya konuşlandırılabilmektedir.

Docker imajları bu avantajlar sayesinde son yıllarda popülerlik kazanmıştır. CNCF (Cloud Native Computing Foundation) bulut tabanlı anketine göre 2022 yılında şirketlerin %45'i canlı (production) ortamda çoğunlukla Docker imajlarını

kullanmaktadır. Şirketlerin diğer %35'i Docker imajlarını canlı ortamlarında kullanmaya başlamıştır. Yine aynı raporda, tüm şirketlerin %30'u geliştirme ve dağıtım (deployment) süreçleri için bulut tabanlı teknolojilerini kullanmaktadır (CNCF, 2022).

StackOverflow 2023 anketine göre, Docker %51,55 kullanım oranıyla en çok kullanılan geliştirme aracıdır (StackOverflow, 2023). Benzer şekilde, Kubernetes konteyner orkestrasyon platformu dünya genelindeki kuruluşların %49'u tarafından kullanılmaktadır (CNCF, 2022). Bu istatistik ve bilgilerlerden yola çıkarak, Docker imajlarının dünya çapında BT alanında yaygın olarak kullanıldığı sonucuna varabiliriz.

Birçok kullanım alanı ve avantajı olmasına rağmen Docker imajları güvenlik riskleri içerebilmektedir. Yamalanmamış Docker imajlarındaki güvenlik açıkları güvenlik için ciddi bir tehdit oluşturur, mali kayıplara ve özel verilerin ifşa edilmesine neden olabilmektedir. Ayrıca, kötü amaçlı veya güncel olmayan konteyner imajları kullanıldığında bu güvenlik riskleri daha da kötüleşebilir. Ek olarak, yanlış yapılandırılmış ve aşırı ayrıcalıklı konteyner imajları, Linux kerneli ve Runc tarafından sağlanan korumaları aşabilir sistemin genel bütünlüğünü tehlikeye atabilmektedir.

CVE-2019-5736 gibi keşfedilen bazı güvenlik açıkları, Runc Container Escape zafiyeti nedeniyle konteyner içerisinde çalışan bir uygulamanın yetkisini yükseltmesine ve ana bilgisayar işletim sistemine geçmesine neden olabilmektedir (*CVE-2019-5736* 2019). Bu güvenlik açığı, saldırganların yönetici (root) olarak bir komut çalıştırmasına, runc ve kernel tarafından sağlanan güvenlik korumalarını aşmasına ve ana bilgisayar üzerindeki runc dosyası üzerine veri yazmasına imkan vermektedir.

2017 yılında bir saldırgan Docker Hub'da birkaç kötü niyetli Docker imajı yükledi. Bu imajlar kaldırılmadan önce 5 milyondan fazla kez indirilmiş ve 90.000$ mali kayba neden olmuştur (Goodin, 2018). Benzer şekilde, başka bir saldırgan, Docker Hub'da "alpine" ve "alpine2" adlı kötü amaçlı imajlar oluşturarak resmi "alpine" imajının popülerliğinden yararlandılar. Bu yanıltıcı taktik, geliştiricileri bilmeden zafiyetli imajları kullanmaları için kandırmayı amaçlıyordu (*Malicious Docker Hub Container Images Used for Cryptocurrency Mining* 2020). Saldırganlar bu imajlar ile kripto para madenciliği yaptılar ve zafiyetli Docker sunucularını tespit etmek için

masscan (*MASSCAN* 2021) aracı ile network taraması gerçekleştirdiler.

Markus ve arkadaşları, Docker Hub ve diğer özel kayıt defterlerinden 337,171 imajı analiz ettiler ve imajların %8.5'inin API anahtarları, özel anahtarlar, parolalar vb. gibi bilgiler içerdiğini tespit ettiler. 28.621 Docker imajında 52.107 özel anahtar ve 3158 sızdırılmış API anahtarı buldular (Dahlmanns ve diğ., 2023). Bu gizli bilgiler kurumlara büyük bir saldırı yüzeyi açmakta ve verilerin gizliliğini riske atmaktadır.

Yukarıdaki güvenlik olaylarında görüldüğü gibi Docker imajları ile ilgili bazı güvenlik sorunları bulunmaktadır. Bu sorunları çözmek için bazı projeler ortaya çıkmıştır. Chainguard projesi sıfır CVE'li ve zafiyetsiz Docker imajları oluşturmayı hedeflemektedir (ChainGuard, 2023). Böylece saldırı yüzeyini azaltabilir ve güvenlik riski en aza indirebilir. Docker Hub ayrıca Docker CLI, Docker desktop ve kendi kayıt defteri servisi Docker Hub üzerindeki imaj güvenlik açıklarını belirlemek için Scout projesini başlatmıştır (Docker, 2023). Benzer şekilde, Redhat kendi Quay imaj kayıt defteri için Clair projesini başlattı (Redhat, 2023).

Ancak hala bu çözümlerin etkinlikleri yeterince bilinmemektedir. Bütün bu gelişmelere rağmen konteyner güvenliği yeterince gelişmemiştir. RedHat Kubernetes Güvenliğinin 2023'teki durumu raporunda katılımcıların %38'i konteyner operasyonlarında güvenlik yatırımlarının yetersiz olduğunu belirtmektedir ve yine aynı raporda katılımcıların %67'si güvenlik endişeleri nedeniyle bulut tabanlı teknolojilerin benimsenmesini yavaşlatmak zorunda kaldığı vurgulanmıştır (*The State of Kubernetes Security in 2023* 2023). Bu alanda çeşitli çalışmalar var ancak daha fazla çalışmaya ihtiyaç duyulmaktadır.

Docker imajları konteynerli uygulamaların temelini oluşturduğundan, imajların içindeki yapısal zayıflıklar tüm kurulumu veya uygulamayı saldırıya açık hale getirebilir. Dahası, yazılım geliştirmenin dinamik doğası gereği risk ortamı sürekli değişmektedir. Her gün yeni güvenlik açıkları keşfedilmekte ve bu açıkları düzeltmek için güvenlik güncellemeleri yayınlanmaktadır. Devam eden süreç, yeni keşfedilen güvenlik açıkları hakkında bilgi sahibi olmayı, güvenlik güncellemelerini temel (base) imajlara ve bağımlılıklara derhal uygulamayı ve özelleştirilmiş imajları potansiyel zayıflıklara karşı düzenli olarak taramayı gerektirmektedir.

Güvenlik açığı tespiti, Docker imajların güvenliğini sağlamada kritik bir savunma hattıdır. Bunun için ise öncelikle güvenlik açıklarını güvenilir bir şekilde tespit etmek gerekmektedir. Tez çalışmasının ilk aşamasında, Docker imaj tarama araçları araştırılmış ve analiz edilmiştir. Analiz sonucunda tarama araçlarının etkinlikleri mevcut literatürdeki farklı metrikler kullanılarak değerlendirilmiş ve CVSS risk puanı tabanlı iki metrik önerilmiştir. Önerilen iki metriğin CVSS tabanlı olması sayesinde kritik ve yüksek şiddetli güvenlik açığı tespiti öne çıkarılmıştır.

Tez çalışmasının ikinci aşamasında, popüler Docker imajlarında güvenlik açıklarının zaman içinde nasıl değiştiği incelendi. Docker Hub'dan en çok kullanılan 364 docker imajı belirlendi ve her bir imaj için en az altı ay arayla üç farklı versiyon seçildi ve toplamda 1092 tarama gerçekleştirildi. Tez kapsamında üç zafiyet tarama aracı, Trivy, Grype ve Snyk, kullanıldı ve imajların zaman içinde üç farklı noktada tarandı. Farklı zaman dilimlerindeki sonuçlar analiz edilerek, Docker imajlarının zaman içerisinde güvenlik açığı değişimine ilişkin bilgiler elde edildi.

Araştırmamız en etkili Docker imaj tarama aracının mevcut zafiyetlerin ciddi bir bölümünü tespit edemediğini gösterdi. Benzer şekilde güncellenmemiş Docker imajlarının güncellenmiş olanlarına göre yaklaşık iki katı zafiyet içerdiği görüldü.

Tez Çalışmasının Katkıları

Tez kapsamında literatüre yapılan katkılar şunlardır:

- Araştırma, yaygın olarak tanınan üç Docker konteyner tarama aracı olan Trivy, Grype ve Snyk üzerinde gerçekleştirilmiş, bu araçlar çalışma kapsamında önerilen ve literatürde geçen metriklerle kıyaslanmıştır.

- Bildiğimiz kadarıyla bu tez, Snyk konteyner tarama aracını değerlendiren ilk çalışmadır.

- Docker Hub'da bulunan 439 docker konteyner imajı taranarak büyük ölçekli bir docker imaj değerlendirmesi gerçekleştirildi. İmajların zaman içerisindeki zafiyet değişimlerini tespit etmek için 364 docker imajı seçilmiş ve üç zaman diliminde

1092 docker imajı taranmıştır.

- Çoğu konteyner imajının yüksek veya orta şiddette güvenlik açığı içerdiği tespit edildi.
- Tarama araçlarının etkinliğinin karşılaştırılmasına olanak tanıyan yeni iki değerlendirme metriği önerildi.
- Deneysel sonuçlar, birçok imaj tarafından kullanılan işletim sistemi imajlarının dikkatlice seçilmesi gerektiğini ortaya koymuştur. Resmi CentOS imajı gibi bazı işletim sistemi imajlarının bir yıldan uzun süredir güncellenmediği görülmüştür.
- Araştırmamız, birçok imaj tarafından kullanılan işletim sistemi imajlarının dikkatlice seçilmesi gerektiğini ortaya koydu. Resmi CentOS imajı gibi bazı işletim sistemi imajlarının bir yıldan uzun süredir güncellenmediği görüldü.
- Ayrıca centos'un en çok güvenlik açığı içeren docker imajı olduğu, alpine ve photon'un ise en güvenli imajlar olduğu gözlemlenmiştir.
- CVE-1999-0236 gibi 2000 yılından önceki güvenlik açıklarının günümüzde hala keşfedilebildiği gözlemlenmiştir.
- Güncellenmiş docker imajlarının saldırı yüzeyini önemli ölçüde azaltabildiği gözlemlenmiştir.

Tez Çalışmasının Organizasyonu

Tez aşağıdaki şekilde düzenlenmiştir: Bölüm 2'de bu alanda yapılan benzer çalışmalar tartışıldı, Bölüm 3'da tezde kullanılan araştırma metodolojisinin nasıl yürütüldüğü anlatıldı, Bölüm 4'de Docker imaj tarama araçlarının etkinliği çeşitli metrikler ile değerlendirildi, Bölüm 5'de popüler Docker imajlarının zaman içerisindeki zafiyet değişimi araştırıldı ve Bölüm 6'da tezde elde edilen bulgular kısaca değerlendirilerek çalışma sonlandırılmıştır.

2. LİTERATÜR TARAMASI

Konteyner güvenliği her zaman ilgi çekici bir konu olmuş ve bu alanda çeşitli çalışmalar yapılmıştır. Bunlardan bazıları imaj tarama araçlarını kullanarak konteynerlerin güvenliğini değerlendirmiş, bazıları konteyner imajlarına yönelik tedarik zinciri saldırılarını değerlendirmiş, bazıları konteyner orkestrasyon platformlarını değerlendirmiş ve bazıları Kubernetes ortamındaki CVE'leri değerlendirmiştir. Tez kapsamında bunlardan bazılarına değinilmiştir.

Konteyner imaj tarama araçları, Docker imajlarının güvenlik açıklarını değerlendirmek için çok önemlidir. Uygulamalarının geliştirilmesi ve dağıtımında Docker konteyner imajlarını kullanan birçok kuruluş tarafından tarama araçları kullanılmaktadır. Bu nedenle bazı araştırmacılar özellikle imaj tarama araçlarının etkinliğine odaklanmıştır. Omar ve diğerleri (Javed ve Toor, 2021) imaj tarama araçlarının efektifliğini incelemiştir. Çalışmalarında, 57 Java tabanlı Docker konteyner imajını değerlendirmişlerdir. İmaj taraması için Clair, Anchore ve Microscanner, statik kod analizi için ise SpotBugs eklentisini kullanmışlardır. En iyi tarama aracının tüm güvenlik açıklarının %34'ünü gözden kaçırdığını tespit etmişlerdir. Çalışmalarında tespit edilen zafiyet sayısına bağlı tespit isabet oranı (Detection Hit Ratio) isminde bir metrik önermişlerdir. M. Varun ve ark. (Varun ve diğ., 2023) araştırmaları için Trivy ve Grype imaj tarama aracını kullanmıştır. Altı Docker imajının (Ubuntu, Python, Alpine, Consul ve Postgres) taranması yoluyla temel olarak Trivy ve Grype araçlarını karşılaştırmışlardır. Bu çalışmada, Grype'nin verimliliği Trivy Güvenlik Açığı Veritabanı ile birleştirilerek artırılmıştır. Shay vd. (Berkovich ve diğ., 2020) imaj tarama araçlarını değerlendirmiştir. Çalışmalarında 3 Docker imajını (debian:10.2, alpine:3.9.4, ubuntu:18.04) 3 imaj tarama aracı (Trivy, Anchore ve Clair) kullanarak taramışlardır. Çalışmalarında F-Skor formülüne dayanan bir metrik kullandılar. Daha sonra yanlış pozitifleri bulmak için her bir güvenlik açığını manuel olarak incelediler.

Bulut güvenliği öncelikle Docker imajlardaki güvenlik açıklarının tespit edilmesine dayanmaktadır. Bu nedenle bazı araştırmacılar Docker imajlarındaki güvenlik açıklarına odaklanmıştır. Katrine ve ark. (Wist ve diğ., 2020)'da güvenlik açığı taramasına dayalı olarak Docker Hub imajlarındaki zafiyetleri incelemiştir. Çalışmalarında 2500 docker

konteyner imajını incelediler ve güvenlik açıklarını bulmak için Anchore konteyner tarama aracını kullandılar. Sertifikalı (verified) imajların %82'sinde en az bir yüksek veya kritik güvenlik açığı bulunduğunu keşfettiler. Markus ve ark. (Dahlmanns ve diğ., 2023) Docker Hub ve diğer kayıt defterlerinden 330.000'den fazla imajı incelemiştir. Docker imajlarının %8,5'inde parolalar, özel anahtarlar ve API anahtarları da dahil olmak üzere gizli bilgilerin bulunduğunu keşfettiler. Araştırmacılar Docker imajlarında 52.107 özel anahtar ve 3158 sızdırılmış API sırrı tespit etmiştir. Ancak, imaj güvenlik açıklarına değil, yalnızca gizli bilgilerin tespitine odaklandılar.

Tehdit modellemesi, herhangi bir sistemdeki çeşitli güvenlik açıklarını ve tasarım hatalarını tespit etmek ve önlemek için çok önemlidir. Ann ve ark. (Wong ve diğ., 2023), STRIDE tehdit modelleme çerçevesini kullanarak önce tehdit modellemesi yaparak konteyner güvenliğini incelemişlerdir. STRIDE kullanarak her bir sistem bileşenindeki güvenlik açıklarını tespit etmişler ve ardından bir anket gerçekleştirmişlerdir.

Docker imajlarındaki güvenlik açıkları zaman içinde değişmektedir. Yeni güvenlik açıkları keşfedilmekte ve güvenlik güncellemeleri sürekli olarak yayınlanmaktadır. Alan ve ark. (Mills ve diğ., 2023) 380 docker imajını analiz ederek bir çalışma gerçekleştirmiştir. 380 docker imajını 3 farklı kategoride analiz etmişlerdir: Resmi, doğrulanmış ve açık kaynak. Çalışmalarının yürütülmesini üç aşamaya ayırmışlardır. Bu yöntemle zaman içinde duyarlılıktaki değişiklikleri gözlemlediler. Tez kapsamında, Docker imajlarının güvenliği için benzer bir yaklaşım kullanılmıştır.

Bu bölümde değinilen çalışmalar konteyner ve imaj güvenliği üzerine yapılan araştırmaların yalnızca bir bölümünü temsil etmektedir. Literatürde çok sayıda benzer çalışma mevcuttur.

3. MATERYAL VE YÖNTEMLER

Tez kapsamında, Docker imaj tarama araçlarının etkinliği ve Docker imajlarının güvenlik açıkları araştırılmıştır. Güvenlik açıkları ve imaj güvenliği sürekli olarak değişmektedir ve Docker imajlarının güvenlik durumunun zaman içerisinde nasıl değiştiğini değerlendirmek önemlidir. Bunu yapmak için şu iş akışı takip edildi:

- Alt bölüm 3.1'da, bir grup imaj tarama aracı araştırılmış ve seçilmiştir.
- Alt bölüm 5.1.1'de Docker Hub'dan üç kategoriden olmak üzere en popüler Docker imajları seçilmiştir.
- Alt bölüm 3.3'te seçilen Docker imajlarının nasıl tarandığı ve tarama süreci anlatılmıştır.
- Alt bölüm 3.4'de tarama sonuçlarının nasıl işlendiği açıklanmıştır.

3.1. Tarama Aracı Seçim Stratejisi

Docker imajlarını taramak için çok sayıda araç online olarak bulunabilir. Bazıları ücretsiz, açık kaynak kodlu ve Github/Gitlab üzerinden erişilebilirken, diğerleri ticari olarak satılmaktadır. İlk olarak, Github'dan başlanarak diğer sitelerde, bloglarda (Doerrfeld, 2023) (Bhat, 2023) (Avi, 2023) imaj tarama araçları araştırılmıştır. İlk araştırmadan sonra, tarama araçları kullanılabilirlik, popülerlik, güncellik ve fiyatlandırma temelinde değerlendirildi. Mevcut tarama araçları tablo 3.1'daki gibi listelendi:

Tarama Aracı	Kullandığı Dil	Geliştirici	Fiyatlandırma
Trivy	golang	Aqua Security	Ücretsiz
Grype	golang	Anchore	Ücretsiz
Snyk	type script	Snyk Security	Ücretsiz
Clair	golang	Redhat	Ücretsiz
Dagda	python	eliasgranderubio	Ücretsiz
Anchore	golang	anchore	Ücretli
Aqua Scanner	Golang	Aqua Security	Ücretli
Xray	c	JFrog Xray	Ücretli
Qualys Container Security		Qualys Security	Ücretli

Tablo 3.1. Docker imajı zafiyet tarama araçları.

Tez çalışmasında, kurulumu ve kullanımı zor olduğu, basit bir CLI arayüzüne ve iyi bir dokümantasyona sahip olmadığı için Clair hariç tutulmuştur. Dagda (son commit 2021 (Grande, 2023)) gibi güncel olmayan imaj tarama araçları, güncel güvenlik açıklarını bulamadıkları için hariç tutulmuştur. Snyk tabanlı Docker taraması da doğrudan Snyk kullanıdığı için değerlendirme dışı bırakılmıştır. Ticari araçlar, deneme sürümleri mevcut olmayabileceğinden veya yalnızca sınırlı işlevselliğe sahip olabileceğinden dolayı dahil edilmemiştir. Sonuç olarak, tez için Trivy, Grype ve Snyk tarama araçları seçilmiştir.

3.2. Docker İmajlarının Seçim Stratejisi

Docker Hub üzerinde 3,5 milyondan fazla docker imajı bulunmaktadır (Wist ve diğ., 2020). Benzer şekilde, Github konteyner kayıt defteri, Quay kayıt defteri, Amazon ECR, Google konteyner kayıt defteri ve özel kayıt defteri gibi diğer kayıt defterlerinde de çok sayıda Docker imajı barındırılmaktadır. Her imajın taranması mümkün ve uygulanabilir değildir. Bu nedenle tez kapsamında Docker Hub'dan indirilme (pull) sayısına göre en popüler Docker imajları seçilmiştir.

Docker imajlarını listelemek için Docker Hub API'si kullanılmıştır. İlk olarak, tüm Docker resmi (official) imajları sorgulandı ve ardından hepsi seçildi. Bazı resmi imajların "latest" etiketi kullanmadığını tespit edildi. Bu imajlar için imaj etiketlerini manuel olarak kontrol edildi ve tarama için en yeni ve kararlı etiketi kullanılmıştır. Bazı resmi imajların, "Library/cheer" gibi, indirilemediği tespit edildi ve bu imajlar atlandı. OSS (Open Source Software) ve doğrulanmış (verified) imajlar için Docker Hub API'si sorgulandı ve ardından 50 milyondan fazla indirilen imajlar seçildi. Bazı Docker imajlarının da "latest" etiketi kullanmadığını tespit edildi ve bu imajların etiketleri resmi imajlarda olduğu gibi seçildi.

Toplamda, Docker Hub'da barındırılan ve Tablo 3.2'da listelenen en çok kullanılan 439 docker imajı seçildi.

3.3. Tarama İşlemi

Tarama için toplamda 439 olmak üzere çok sayıda docker imajı seçildi. Bu kadar çok sayıda imajı manuel olarak taramak çok uzun zaman gerektirmesinden dolayı mümkün değildir. Sonuç olarak tarama işlemi Python ve shell/bash betikleri ile otomatize edildi.

İmaj Kategorisi	Taranacak İmaj Sayısı
Library	169
Verified	141
Open Source	129

Tablo 3.2. Tez çalışması için başlangıçta seçilen Docker imajları.

Tarama işlemini otomatikleştirilse bile oldukça zaman almaktadır. Bunun üstesinden gelmek için tarama işlemi bulutta gerçekleştirildi, bir VPS (Virtual private server) sunucusu 2 GB RAM, 2 CPU ve 50GB depolama alanı ile kullanılmıştır.

İlk olarak, Docker Hub'dan imajları indiren, ardından imajları 3 farklı araçla (Trivy, Grype ve Snyk) tarayan ve tarama sonuçlarını JSON dosyaları olarak kaydeden bir shell scripti oluşturuldu. JSON'ı ayrıştırmak ve işlemek kolay olduğu için çıktı formatı olarak JSON seçildi. Daha sonra disk alanından tasarruf etmek için Docker imajı silindi. Bir imaja ait indirme işlemi başarısız olursa, o imaj için tarama atlandı.

İmajları tararken Docker Hub'ın imaj indirmeleri için hız sınırlaması uyguladığı tespit edildi. Anonim kullanıcılar için sınır IP adresi başına 6 saatte 100 imaj indirme iken, kimliği doğrulanmış kullanıcılar için sınır 6 saatte 200 imaj indirme olarak belirlendiği görüldü. Sonuç olarak Docker Hub'a `docker login` komutu ile giriş yapıldı ve kullanılan shell betiğinde hız sınırlaması uygulandı.

Bu işlemlerin sonunda, seçilen tüm Docker imajları başarıyla tarandı ve sonuçları kaydedildi.

3.4. Tarama Sonuçlarının İşlenmesi

Bir önceki adımda Docker imajlarının nasıl tarandığı belirtildi. Şimdi tarama sonuçlarının CSV dosya formatına dönüştürülmesi gerekmektedir, böylece sonuçlar Python ve Pandas ile işlenebilecek, Mathplotlib ve Seaborn ile görselleştirilebilecektir. İlk olarak, JSON dosyalarını ayrıştırmak için bir Python betiği oluşturuldu, ardından sonuçlardan bazı alanlar seçildi. Seçilen alanlar şu şekilde listelenebilir: tarama aracı, imaj türü (resmi, doğrulanmış veya açık kaynak), imaj adı, Güvenlik Açığı CVE ID'si, paket adı, paket sürümü, zafiyet şiddeti (severity) ve CVSS puanı.

Çoğu alanı seçmek kolaydır, fakat CVSS puanı özel dikkat gerektirmektedir. Öncelikle,

her güvenlik açığı bir CVE ID'sine ve CVSS puanına sahip olmayabilmektedir. Bazı güvenlik açıkları CVSS2'ye sahipken, CVSS3'e sahip değildir. Benzer şekilde, bazı güvenlik açıkları CVSS3 puanına sahiptir, CVSS2 puanına sahip değildir. CVE ID'si olmayan güvenlik açıkları tarama sonuçlarından hariç tutuldu. Sonuçların geri kalanı için, mevcutsa CVSS3 puanı, mevcut değil ise CVSS2 puanını seçildi. CVSS sonraki bölümlerde detaylarıyla açıklanmıştır(bakınız bölüm 4.1.2). Tarama dosyalarının işlenmesinin ardından tüm sonuçlar tek bir CSV dosyasında toplandı.

Kod dosyaları, veri erişilebilirliği bölümünde (bakınız bölüm 7) JSON dosyaları ile birlikte paylaşılmıştır.

3.5. Kısıtlar

Docker imaj güvenliği üzerine yaptığımız çalışma, Docker imaj güvenliği konusunda değerli bilgiler sunsa da, bazı yapısal kısıtları kabul etmek önemlidir. Bu bölümde, tez çalışması sonuçlarımızı yorumlarken göz önünde bulundurulması gereken bazı temel sınırlamaları ele alacağız.

439 Docker imajı üzerinde yaptığımız analizin tüm Docker imajlarının güvenlik risklerini temsil etmeyebileceğini göz önünde bulundurmak önemlidir. Bu da bulgularımızın daha geniş Docker imajları için değişebileceği anlamına gelmektedir.

Ayrıca, tarama sonuçları değerli olmakla birlikte sınırlamalara sahiptir. Güvenlik açığı veritabanları tamamen güncel olmayabilir ve potansiyel olarak yeni keşfedilen zafiyetleri kaçırabilir. Tarama araçlarının kendileri kusurlu olabilir, yanlış pozitifler (gerçekte olmayan güvenlik açıklarını bildirme) veya negatifler (gerçek güvenlik açıklarını kaçırma) üretebilir. Ayrıca, çoğu tarama aracı tarafından kullanılan statik analiz, çalışma zamanı (runtime) güvenlik açıklarını tespit etmede sınırlamalara sahiptir. Güvenlik, sadece güvenlik açıklarını tespit etmek değildir. Tarama araçları, yanlış yapılandırmaları, önceden kodlanmış gizli bilgileri veya konteyner kaçış zafiyetlerini gözden kaçırabilir.

Dikkate alınması gereken bir diğer sınırlama da Docker imajlarının dinamik yapısıdır. Güncellemeler veya yeni zafiyetlerin ortaya çıkması nedeniyle imajların güvenlik durumları zaman içinde değişebilir. Bu nedenle, tarama sonuçlarının bir imajın mevcut güvenlik durumunu doğru bir şekilde yansıtmasını sağlamayı zorlaştırabilir.

Docker imajı güvenliğinin, analizimizde kullandığımız belirli metriklerin ötesine geçtiğini de göz önünde bulundurmamız gerekir. Güvenliği etkileyebilecek başka birçok faktör vardır, bu nedenle bulgularımız Docker imajlarının tamamını yansıtmayabilir.

Bu kısıtlamalara rağmen, tez çalışmamız Docker imaj güvenliğine ilişkin değerli bilgiler vermektedir. Bu kısıtlamaları kabul ederek, tezde ortaya konulan Docker ekosistemindeki güvenlik riskleri değerlendirilebilir.

4. ARAŞTIRMA 1

Docker zafiyet tarama araçları, Docker imajı zafiyetlerinin neden olduğu artan saldırı yüzeyi nedeniyle çok önemlidir. Tarama araçlarının önemi, konteynerli ortamlardaki güvenlik açıklarını proaktif olarak tespit etme, değerlendirme ve ele alma becerilerinde yatmaktadır. Kuruluşlar bu araçları geliştirme ve dağıtım süreçlerine entegre ederek genel güvenlik durumlarını iyileştirebilir ve uygulamalarını ve verilerini daha iyi koruyabilirler (Sultan ve diğ., 2019).

Docker imajlarında yüksek risk taşıyan güvenlik açıklarını tespit etmek çok önemlidir. Bu güvenlik açıklarını tespit eden tarama araçlarının test edilmesi, yeteneklerini anlamak için gereklidir. Bu bölüm, IT sektöründe yaygın olarak kullanılan tarama araçlarının Docker imajlarındaki güvenlik açıklarını tespit etmedeki etkinliğini değerlendirmeyi ve bu araçları birbirleriyle karşılaştırmayı amaçlamaktadır.

Tez kapsamında, imaj tarama araçlarının efektifliğini değerlendirmek için Docker Hub'dan 439 Docker imajı seçilmiştir (imaj seçim stratejisi detayları için bakınız bölüm 3.2). Bu imajlar daha sonra Trivy, Grype ve Snyk araçları kullanılarak taranmıştır (daha fazla bilgi için araç detayları bölümüne bakınız 3.1). Tarama sonuçları, araçların etkinlik seviyeleri dikkate alınarak karşılaştırılmıştır.

4.1. Değerlendirme Sonuçları

Değerlendirme sonuçları bölümünde, 439 Docker imajından elde edilen verilerin taranması, depolanması ve analiz edilmesinin ardından bulgularımızın değerlendirmesi sunulmuştur. Değerlendirmede, tarama araçlarının etkinliğinin ve performansının kapsamlı bir incelemesini sağlamak için üç farklı metrik kullanılmaktadır.

4.1.1. Değerlendirme Metrikleri

Değerlendirme metrikleri bölümünde, tarama araçlarının performansını ve etkinliğini değerlendirilmek için kullanılan nicel metrikler tanıtılmaktadır. Değerlendirmemiz, Docker imaj tarama araçlarının doğruluğunu ve güvenilirliğini değerlendirmek için nesnel metrikler olarak hizmet eden üç farklı metrik kullanmaktadır.

İlk metrik, her bir araç tarafından tespit edilen toplam güvenlik açığı sayısına odaklanmaktadır. Bu metrik, araçların Docker imajlarındaki potansiyel güvenlik

tehditlerini tespit etme kapasitesine ilişkin temel bir değerlendirme sağlar.

Kullanılan ikinci metrik ise Javed vd. (Javed ve Toor, 2021) tarafından önerilen Tespit İsabet Oranı (DHR) yöntemidir. Bu yöntemde, imajlar mevcut tüm tarama araçları kullanılarak taranır ve tespit edilen güvenlik açıkları ortak bir havuzda toplanır. Daha sonra, bu ortak havuzdaki her bir tarama aracı tarafından tespit edilen (Hit) ve gözden kaçırılan (Miss) güvenlik açıklarının sayısı hesaplanır. İsabet ve ıskalama oranları belirlendikten sonra metrik Denklem 4.1'e göre hesaplanır. Denklem 4.1'de, 'Hit' tarama aracı tarafından tespit edilen güvenlik açıklarının sayısını gösterirken, 'Miss' ortak havuzda bulunan ancak değerlendirilen araç tarafından gözden kaçırılan güvenlik açıklarının miktarını ifade eder.

$$DHR = \frac{Hit}{(Hit + Miss)} \tag{4.1}$$

Yukarıda bahsedilen metrikler sadece güvenlik açığı sayısına odaklanmakta ve güvenlik açığının ciddiyetini göz ardı etmektedir. Bu eksiği ele almak için, tez kapsamında önem derecesine dayalı bir yaklaşım (VSM) bölüm 4.1.2'de önerilmiştir.

4.1.2. Zafiyet Tarama Metriği

Güvenlik açığı tarama araçlarını yalnızca tespit sayılarına göre değerlendirmek en etkili yaklaşım olmayabilir. Bu sorunu etkili bir şekilde ele almak için, tespit ettikleri güvenlik açıklarının ciddiyetine ilişkin tarama araçlarının karşılaştırmalı analizini yapmak zorunludur. Bu nedenle tez kapsamında, güvenlik açıklarının ciddiyetine dayalı yeni bir metrik önerilmektedir.

Güvenlik açıklarının önem dereceleri Ortak Güvenlik Açığı Puanlama Sistemine (CVSS) göre kategorize edilmektedir. CVSS, bir güvenlik açığının temel özelliklerini yakalamak ve ciddiyetini yansıtan sayısal bir puan üretmek için bir yol sağlar (*Common Vulnerability Scoring System SIG* 2023). Şu anda CVSS'nin 4 versiyonu bulunmaktadır. En son versiyon olan 4. versiyon 2023 yılında yayınlanmıştır; ancak mevcut verilerin sınırlı olması nedeniyle bu tez kapsamında kullanılmamıştır. Bunun yerine, CVSS puanlarının sürüm 3 ve sürüm 2'si kullanılmıştır. Her sürüm şiddet ve ciddiyet puanları içerir ve tüm puanlar tablo 4.1 ve (*CVSS v3.1* 2023)'da belirtildiği gibi nitel

Değerlendirme	CVSS Skoru
Yok (None)	0.0
Düşük (Low)	0.1 - 3.9
Orta (Medium)	4.0 - 6.9
Yüksek (High)	7.0 - 8.9
Kritik (Critical)	9.0 - 10.0

Tablo 4.1. Niteliksel şiddet derecelendirme ölçeği.

derecelendirmelerle eşleştirilebilir.

Bu tez kapsamında CVSS tabanlı bir Güvenlik Açığı Tarama Metriği (VSM), tarama araçlarını kendi aralarında karşılaştırmak için, keşfettikleri güvenlik açıklarının risk seviyelerini dikkate alarak kullanılmıştır. DHR yöntemine benzer şekilde VSM yöntemi de Hit ve Miss güvenlik açıklarını kullanır. Bir tarama aracının N güvenlik açığı tespit ettiğini ve tüm tarama araçları tarafından tespit edilen tüm güvenlik açıklarından M güvenlik açığını kaçırdığını düşünelim. Her bir güvenlik açığına, i'inci güvenlik açığı için n_i ile gösterilen bir CVSS puanı atanır. VSM, Denklem 4.2 kullanılarak hesaplanabilir:

$$VSM = \frac{\sum_{i=1}^{N} n_i}{(N+M) \times 10} \tag{4.2}$$

Denklem 4.2'de, VSM hesaplanan Güvenlik Açığı Tarama Metriğini temsil eder. Pay $\sum_{i=1}^{N} n_i$ tarama aracı tarafından tespit edilen tüm güvenlik açıklarının CVSS puanlarının toplamını ifade eder. Payda $(N+M) \times 10$, VSM'nin 0 ila 1 aralığında olmasını garanti ederek metriği standartlaştırır ve tarama araçları arasında karşılaştırma yapmayı kolaylaştırır.

DHR tabanlı bir yöntem olan VSM, birden fazla tarama aracını karşılaştırmak için uygun bir metrik olarak hizmet vermektedir. Esasen, birden fazla tarama aracı taramalarını gerçekleştirdikten sonra, ortak bir havuzda toplanan güvenlik açıklarına dayalı olarak Hit (N) ve Miss (M) sayıları belirlenir.

Güvenlik açıklarını bu sayılara ihtiyaç duymadan sadece risk seviyelerine göre değerlendirmek için ikinci bir metrik olarak Tekil Güvenlik Açığı Tarama Metriğini (SVSM) öneriyoruz. SVSM yönteminin matematiksel ifadesi Denklem 4.2'de

verilmiştir.

$$SVSM = \frac{1}{N}\frac{\sum_{i=1}^{N} n_i}{N \times 10} \tag{4.3}$$

Denklem 4.3'de ortalama CVSS puanı, CVSS puanlarının toplamının toplam güvenlik açığı sayısına bölünmesiyle hesaplanır. CVSS puanı 0 ila 10 arasında değiştiğinden, sonuçlar 10'a bölünerek 0 ila 1 aralığına normalleştirilir. Mevcut haliyle formül, güvenlik açığı puanlarının ortalamasını 0 ile 1 arasında bir aralığa normalleştirmektedir. Başka bir deyişle, tespit edilen güvenlik açığı sayısından bağımsız olarak bağımsız bir değer hesaplamaktadır. Tespit edilen güvenlik açığı sayısını metriğe yansıtmak için bu değer toplam güvenlik açığı sayısına (N) bölünür. Bu durumda, 0'a yakın bir SVSM değeri daha etkili bir tarama aracına işaret eder.

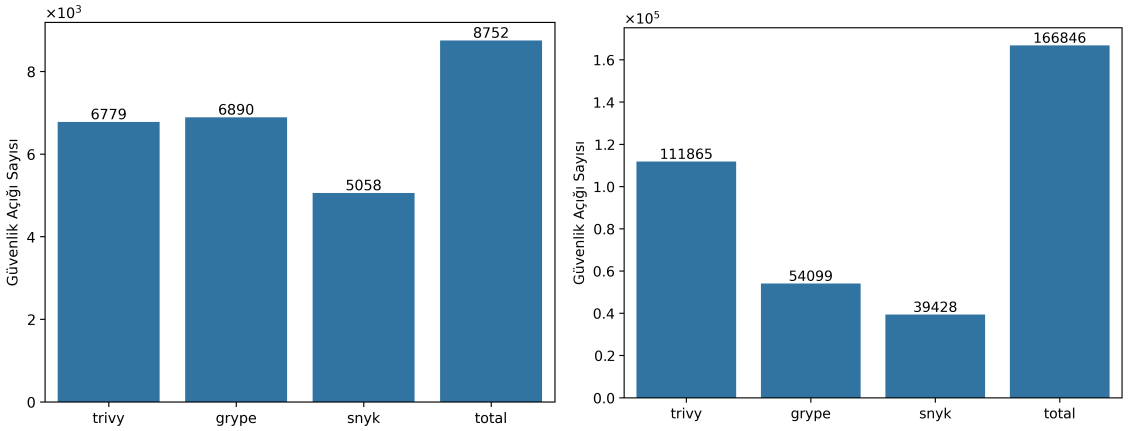
4.2. Araştırma 1 Sonuçları

Aşağıdaki bölümde önceki bölümlerde önerilen metodolojilerin ve metriklerin uygulanmasından elde edilen sonuçlar sunulmaktadır. Bu sonuçlar, incelenen zafiyet tarama araçlarının etkinliği ve performansı hakkında fikir vermektedir. Kapsamlı analiz ve yorumlama ile, güvenlik açıklarının belirlenmesi ve değerlendirilmesinde her bir tarama aracının güçlü ve zayıf yönlerine ışık tutmak amaçlanmıştır.

4.2.1. Zafiyet Tespit Sayılarına Göre Sonuçlar

Araçların karşılaştırılmasına, tespit edilen güvenlik açıklarının incelenmesiyle başlanabilir. Bu metrik yalnızca güvenlik açıklarının sayısını dikkate alır ve önemli bilgiler sunar. Şekil 4.1'de bu bulgular görsel olarak temsil edilmektedir.

Şekil 4.1-a'da, her bir tarama aracının tüm imajlarda CVE ID'lerine göre bulduğu benzersiz güvenlik açığı sayısı ve tüm tarama araçlarının bulduğu toplam benzersiz güvenlik açığı sayısı verilmiştir. Tüm imajların taranması sonucunda bulunan benzersiz güvenlik açığı sayıları incelendiğinde, grype'ın en yüksek sayıda güvenlik açığı tespit ettiği görülmektedir. Trivy çok az bir farkla Grype'ı yakından takip etmekte, onu da Snyk izlemektedir. Bu sayılar, 439 imajın taranmasının ardından bulunan farklı CVE-ID değerlerine sahip benzersiz güvenlik açıklarının sayısını temsil etmektedir (ayrıntılar için Tablo 3.2'a bakın). Buna göre, tüm imajların tüm araçlarla taranması toplam 8752



(a) Eşsiz zafiyet sayıları

(b) Zafiyet sayıları

Şekil 4.1. Her bir araç tarafından tespit edilen eşsiz ve toplam güvenlik açığı sayıları.

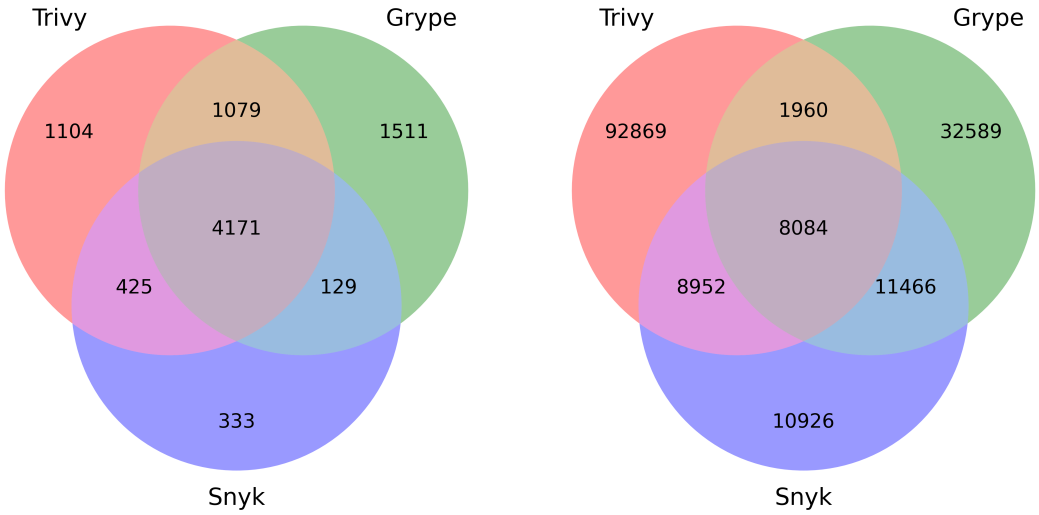
Tarama Aracı	Tespit Edilen Zafiyet Sayısı
Trivy	111.865
Grype	54.099
Snyk	39.430
Total	166.846

Tablo 4.2. Tarama araçları tarafından tespit edilen zafiyet sayıları.

benzersiz güvenlik açığı ortaya çıkarmıştır.

Güvenlik açıkları ayrıca imaj türü, imaj adı, paket adı ve paket sürümüne göre etiketlenmiştir. Bu etiketlemenin ardından, her bir araç tarafından bulunan toplam güvenlik açığı sayısı ve tüm araçların toplamı Şekil 4.1-b'de gösterilmektedir. Şekil 4.1'de verilen grafikler karşılaştırıldığında, Grype tekil güvenlik açığı sayısı bakımından önde gelirken, Trivy imajlarda bulunan toplam güvenlik açığı sayısında önemli ölçüde üstünlük sağlamaktadır. Daha fazla açıklık getirmek gerekirse, Trivy'nin 439 docker imajını taradıktan sonra bulduğu farklı güvenlik açığı sayısı 6779 iken, Trivy'nin tüm imajlarda bulduğu toplam güvenlik açığı sayısı 111865'tir. Benzer bir durum diğer araçlar için de geçerlidir. Grafikler arasında Trivy ve Grype araçları arasındaki sıralama değişse de Snyk her iki grafikte de sürekli olarak son sırada yer almaktadır.

Şekilde görüldüğü gibi 4.1 Trivy en iyi performansı göstermektedir, Grype ikinci, Snyk ise son sırada yer almaktadır.



(a) vulCountsVennUniq (b) vulCountsVenn

Şekil 4.2. Tarama araçları tarafından tespit edilen zafiyet sayısının Venn diyagramı.

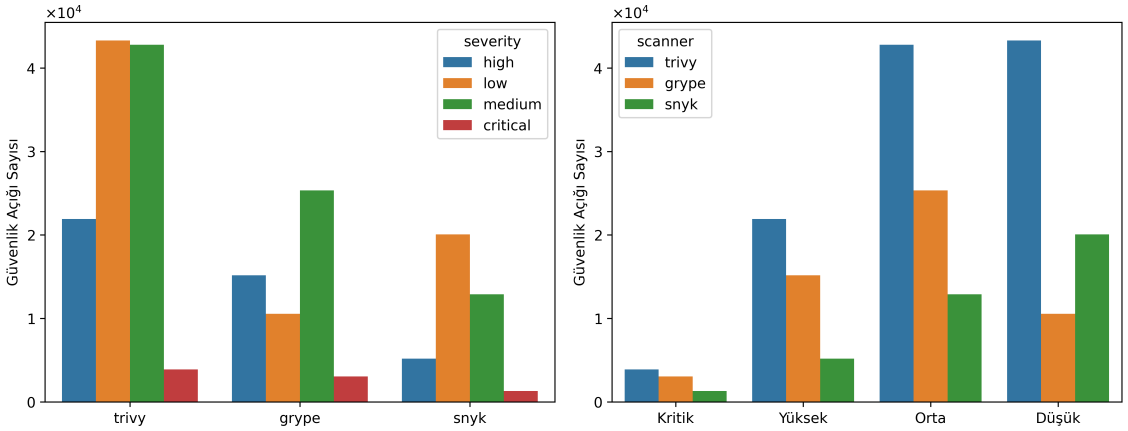
Tarama aracı	Kritik	Yüksek	Orta	Düşük
Trivy	3879	21.904	42.778	43.304
Grype	3041	15.166	25.339	10.553
Snyk	1302	5184	12.880	20.064

Tablo 4.3. Tarayıcı aracına ve önem derecesine göre tespit edilen güvenlik açıkları.

Tarama araçlarının tekil ve toplam güvenlik açığı sayılarının kesişim sayıları sırasıyla Şekil 4.2-a ve Şekil 4.2-b'de Venn diyagramlarında gösterilmiştir. Şekil 4.2-a'daki Venn diyagramına bakıldığında, tüm araçlar tarafından tespit edilen ortak güvenlik açığı sayısının 4171 olduğu görülebilir. Bu sayı toplam güvenlik açığı sayısının (8752) yarısından azdır. Bu, docker imajlarındaki güvenlik açıklarının birden fazla araç tarafından daha iyi kapsanabileceğini göstermektedir. Benzer bir sonuç, Şekil 4.2-b'de verilen ve tüm imajlar tarandıktan sonra bulunan güvenlik açıklarını gösteren Venn diyagramı incelenerek de çıkarılabilir.

Şekil 4.3a ve tablo 4.3'de güvenlik açığı tespitlerini ciddiyete (severity) göre ayrılmış olarak görebiliriz. Tarama sürecinde, ihmal edilebilir ve bilinmeyen güvenlik açıklarını hariç tutuldu, çünkü ihmal edilebilir güvenlik açıkları ciddi hasara işaret etmez ve bilinmeyen güvenlik açıklarının CVSS puanları yoktur.

Kritik güvenlik açıkları en tehlikeli ve ciddi güvenlik açıklarıdır. Bu nedenle bu açıkları tespit etmek son derece önemlidir. Şekil 4.3a ve tablo 4.3'de Trivy'nin 3879 kritik



(a) Tarama aracına göre (b) şiddete göre

Şekil 4.3. Zafiyetlerin tarama aracına ve şiddete göre dağılımı.

güvenlik açığı tespit ettiğini, Grype'nin 3041 kritik şiddette güvenlik açığı tespit ettiğini ve Snyk'in 1302 tespit ettiğini görebiliriz. Trivy ve ardından Grype kritik güvenlik açıklarını tespit etmede iyi performans göstermektedir. Ancak Snyk, Trivy ve Grype'a kıyasla yeterince iyi performans göstermemiş ve yaklaşık 2000 kritik güvenlik açığını gözden kaçırmıştır.

Yüksek şiddetteki güvenlik açıkları kritik güvenlik açıklarından daha az şiddetlidir ancak yine de tespit edilmeleri önemlidir. Aynı şekilde 4.3a ve tablo 4.3 da Trivy 21.904, Grype 15.166 ve snyk 5184 yüksek şiddette güvenlik açığı tespit etmiştir. Bu istatistiklerden Trivy'nin en iyi performansı gösterdiği ve onu Grype'ın izlediği sonucunu çıkarabiliriz.

Orta şiddette güvenlik açıkları için Trivy 42.778, Grype 25.339 ve Snyk 12.880 güvenlik açığı tespit etmiştir. Düşük önem derecesine sahip güvenlik açıklarında Trivy 43.304, Grype 10.553 ve Snyk 20.064 güvenlik açığı tespit etmiştir.

Genel olarak, Trivy en fazla sayıda güvenlik açığı tespit etmiş, onu kritik, yüksek ve orta önem derecesi kategorilerinde Grype ve Snyk izlemiştir. Ancak, şekil 4.3'te gösterildiği gibi, düşük önem dereceli güvenlik açıkları için Trivy en iyi performansı göstermiş, onu Snyk izlemiş ve Grype en düşük tespit oranına sahip olmuştur.

4.2.2. İsabet ve Iskalama Metriğine Göre Sonuçlar

Bu bölümde, isabet ve ıskalama sonuçlarını hesapladık. İlk olarak, tespit edilen tüm güvenlik açıklarından imaj adı, paket adı, paket sürümü ve CVE ID'sini filtreleyerek

Tarama Aracı	Zafiyet Tespit Sayısı	Zafiyet Iska Sayısı	DHR
Trivy	111.865	54.981	0.67
Grype	54.099	112.747	0.32
Snyk	39.430	127.418	0.24

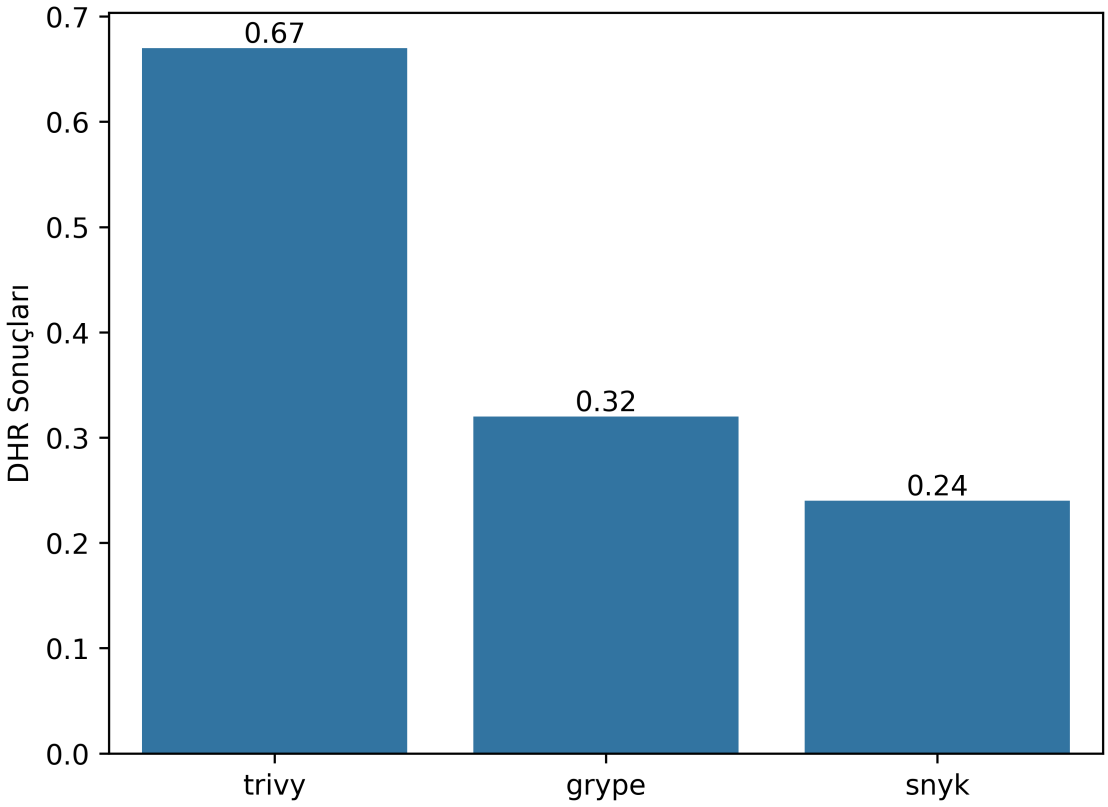
Tablo 4.4. Araçların tespit isabet oranları.

eşsiz (unique) güvenlik açıkları tespit ettik. Bu yöntemle toplamda 166.846 benzersiz güvenlik açığı tespit edildi. Daha sonra denklem 4.1 uygulayarak zafiyet tespit oranları değerlendirildi. Sonuçlar tablo 4.4 ve şekil 4.4'de görülebilir.

Bu sayılardan ve istatistiklerden, Trivy'nin %67 isabet oranıyla en iyi performansı gösterdiği sonucunu çıkarabiliriz. Trivy 111.865 adet güvenlik açığı tespit etmiştir, ancak yine de güvenlik açıklarının %33'sını kaçırmaktadır; kaçırdığı zafiyetlerin tam sayısı 54.981'dir. Yani en iyi tarama aracı güvenlik açıklarının %33'sını kaçırmaktadır. Sonra, Grype 54.099 güvenlik açığı tespit etmiştir ve 112.747 güvenlik açığını kaçırmıştır. Grype'ın DHR'si %32'dir ve güvenlik açıklarının %68'ini kaçırmaktadır. Son olarak, Snyk 39.430 güvenlik açığı tespit etmiştir ve %24 DHR ile 127.418 güvenlik açığını kaçırmıştır. Bu istatistikler şekil 4.4'de görülebilir.

4.2.3. Zafiyet Puanlama Metriğine (VSM) Göre Sonuçlar

Tespit İsabet Oranı (DHR) sonuçların değerlendirmesi için önemli bir metrik olarak hizmet vermektedir. Bununla birlikte, güvenlik açıklarının ciddiyeti ve puanı arasında bir ayrım yapmaması ve kritikliklerine bakmaksızın tüm güvenlik açıklarını eşit olarak ele alması, bu metriğin sınırlılığını oluşturmaktadır. Bu soruna yanıt olarak, Güvenlik Açığı Tarama Metriğini (VSM) öneriyoruz. Bu metrik Denklem 4.2'de açıklandığı şekilde hesaplanmaktadır. VSM, DHR yöntemine benzer bir yaklaşım izlemektedir. VSM, tüm tarama araçları tarafından bulunan tüm güvenlik açıklarının yer aldığı ortak bir zafiyet havuzu oluşturur. Bir tarama aracının başarısı, bu havuzda bulunan güvenlik açıklarını ne kadar iyi tespit ettiği ile belirlenir. DHR'den farklı olarak VSM yöntemi, tespit edilen güvenlik açıklarının risk seviyelerini dikkate alır. Denklem 4.2'de, n_i güvenlik açığının CVSS puanını (bkz. Tablo 4.1) temsil eder. Tarama sürecinde, bir güvenlik açığının risk seviyesi için CVSS2 ve CVSS3 olmak üzere iki farklı puan elde edilebilir. Bir güvenlik açığı CVSS3 skorunun mevcut olmadığı durumlarda CVSS2 skoru kullanılır. VSM sonuçları Tablo 4.6 ve Şekil 4.5'de görülebilir.



Şekil 4.4. Tarama araçları zafiyet tespiti oranları.

Tarama Aracı	Zafiyetlerin CVSS Puan Toplamları
Trivy	736.142,5
Grype	319.417,0
Snyk	256.872,0

Tablo 4.5. Bir araç tarafından tespit edilen zafiyet CVSS puanlarının toplamı.

Tarama Aracı	DHR	VSM	SVSM
Trivy	0.67	0.442	0.0000006
Grype	0.32	0.219	0.0000012
Snyk	0.24	0.158	0.0000017

Tablo 4.6. Önerilen metrik ve DHR ile hesaplanan sonuçlar.

VSM yöntemi DHR yöntemi ile benzer bir yaklaşımla çalışmaktadır. Bu metriklerin hesaplanabilmesi için birden fazla tarama aracının kullanılması gerekmektedir. Bütün tarama araçlarının bulduğu güvenlik açıkları ortak bir havuza toplanıp araçların Miss (M) değerleri bir aracın havuzda olup da bulamadığı güvenlik açığına göre hesaplanmaktadır. Bu çalışmada tarama araçlarını birbirlerinden bağımsız olarak değerlendirmek için SVSM metriği önerilmiştir. Bu metrik VSM'de olduğu gibi zafiyetlerin risk seviyelerini göz önünde bulundurmaktadır. VSM'den farklı olarak sadece aracın bulduğu güvenlik zafiyetlerini kullanarak bir sayısal değer üretmektedir. VSM metriğinde bulunan sayısal değerin bire yakın olması tarama aracının daha iyi olduğunu göstermektedir. Ancak Denklem 3'ten de anlaşılacağı gibi SVSM metriğinde 0'a yakın değere sahip tarama aracının daha iyi olmaktadır. SVSM sonuçları Tablo 4.6 ve Şekil 4.5'de görülebilir.

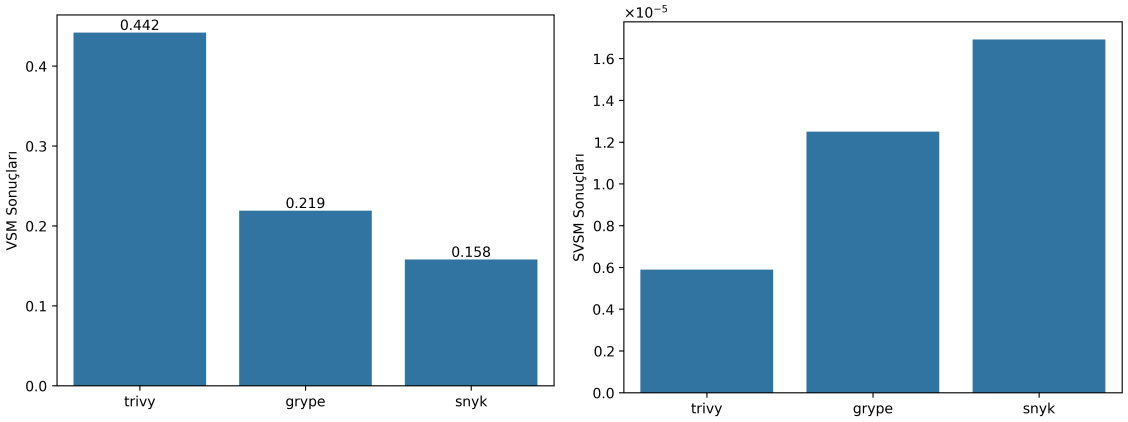
Bir sonraki adımda, bölüm 4.1.2'deki denklemi uygulayarak güvenlik açığı puanı metriklerini hesapladık. Sonuçlar tablo 4.6 ve şekil 4.5'da görülebilir.

Bu rakamlar, Trivy'nin %44 oranla en iyi performansı gösterdiğini, Grivy'nin %22 oranla ikinci sırada yer aldığını ve Snyk'in %16 oranla üçüncü sırada olduğunu açıkça göstermektedir.

4.2.4. Metrikler Arası Karşılaştırma

Önceki bölümlerde sonuçlar 2 farklı metrik ile analiz edildi. Bu bölümde metrik sonuçları karşılaştırılmıştır. Başlangıçta sonuçlar aynı görünüyor ancak bazı önemli farklılıklar mevcut.

İlk olarak VSR'miz DHR'den daha az oran göstermektedir. Bunun nedeni DHR'nin

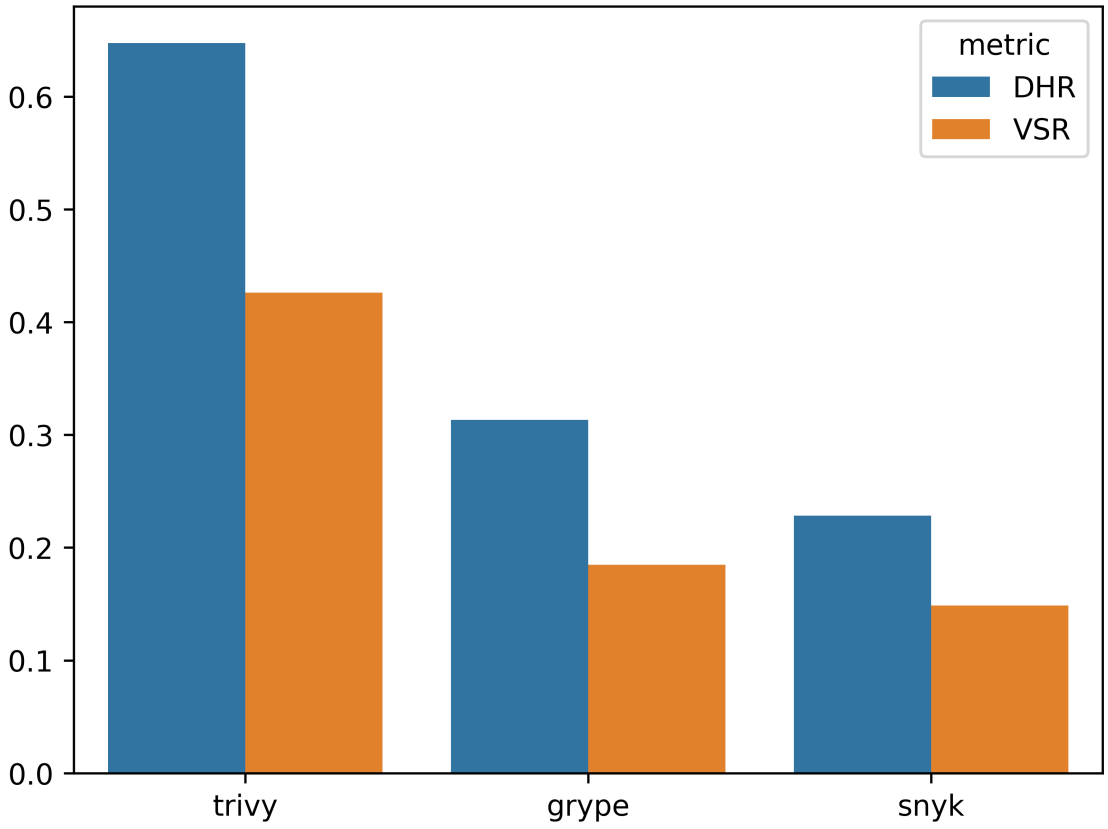


(a) VSM ile hesaplanan sonuçlar

(b) SVSM ile hesaplanan sonuçlar

Şekil 4.5. Önerilen metrik ile hesaplanan sonuçlar.

yalnızca tespit edilen güvenlik açığı sayısına dayanması, ancak bizim metriğimizde ortaya çıkan oranın güvenlik açığının CVSS puanlarına dayanmasıdır. Örneğin tablo 4.2'da Trivy'nin 3879 kritik, 21.904 yüksek, 42.778 orta ve 43.304 düşük şiddette güvenlik açığı tespit ettiği görülebilir. Bu önem dereceleri DHR için önemli değildir, ancak VSR için bu önem derecesi farkı sonuç oranını düşürür çünkü tarama araçları daha az kritik güvenlik açıkları tespit eder. Sonuç grafiğini şekil 4.6'de görebiliriz.



Şekil 4.6. DHR ve VSR'nin Karşılaştırılması

5. ARAŞTIRMA 2

Her gün yeni güvenlik açıkları keşfedilmekte ve bu açıkları düzeltmek için güvenlik güncellemeleri yayınlanmaktadır. Dolayısıyla bir Docker imajının zaman içerisindeki güvenlik zafiyet durumu farklı olmaktadır. Tezin bu bölümde Docker imaj zafiyetlerinin zaman içerisindeki değişimine odaklanılmıştır. Temel olarak en popüler 360 docker imajının 3 farklı zaman dilimindeki sürümleri taranmış ve zaman dilimleri arasındaki zafiyet durumları incelenmiştir.

5.1. Yöntem ve Materyal

5.1.1. İmaj Seçim Stratejisi

Tez çalışmasında Docker imajlarının zaman içindeki güvenlik açıkları incelemek hedeflendi, bu nedenle bölüm 3.1'de belirtilen imaj seçiminden sonra Docker Hub API'si kullanılarak seçilen imajların tüm imaj etiketleri listelendi ve indirildi. İndirilen imaj etiketleri üç kategoride gruplandırıldı: 2023.01.01 tarihinden öncekiler (2023_01) (YYYY.MM.DD tarih formatında), 2023.01.01 ile 2023.07.01 tarihleri arasındakiler (2023_07) ve 2024.02.01 tarihli güncel Docker imajları (2024_01). Daha sonra, 2023.01 dönemi için en güncel imaj etiketleri ve 2023.07 dönemi için en güncel imaj etiketleri seçilmiştir. Son olarak, 2024.01 tarihli en güncel imajlar seçilmiştir.

Seçim işlemi, otomatikleştirmenin bir yolu olmadığı için manuel olarak yapılmıştır. Bunun en büyük nedeni ise Docker imajlarının sürümlerini belirlemek için kullanılan etiketlerin (tag) bir standardının olmamasıdır. Sonuç olarak geliştiriciler Docker imajlarını istedikleri herhangi bir şekilde etiketleyebilmektedirler. Örneğin bazı Docker imajlarının RC (release candidate), beta ve test sürümleri bulunmaktadır, bazı imajlar kararsız sürümler için debug, devel ve unstable etiketlerini kullanabilmektedir. Bu durum stabil sürümler ile test için oluşturulan sürümleri ayırt etmeyi zorlaştırmaktadır. Bazı imajlar bir imaj sürümü için openjdk8, openjdk11, openjdk17 gibi farklı base imajlar kullanabilmektedir. Bazı imajlar etiketler için git commit hash değerlerini (7 karakterli), bazıları Archlinux ve OpenJDK'da olduğu gibi tarih ve önekleri kullanmaktadır. Unit, spark ve Clear Linux gibi bazı imajların Docker Hub'da eski sürümleri yoktur. Yine bazı Docker imajları, Ubuntu gibi, LTS (Long Term Support, Uzun Süreli Destek) sürümlerini kullanır. "zap2docker" imajında olduğu gibi bazı imajların etiket bilgileri

İmaj Türü	Taranan İmaj Sayısı
Library	143
Verified Publisher	101
OSS	120

Tablo 5.1. Son olarak bu tez için seçilen Docker imajları.

Docker Hub API'sinde görünmektedir fakat indirilememektedir.

Yukarıda bahsedilen nedenler seçim sürecini otomatikleştirmeyi imkansız hale getirmiştir. Bu nedenle etiket seçim işlemi manuel olarak el ile yapılmıştır. Seçim sürecindeki herhangi bir hatayı tespit etmek için ortaya çıkan imaj etiketlerini iki kez kontrol edildi ve bir Python betiği kullanarak etiketlerin geçerliliği doğrulandı.

Tezin bu bölümünde Docker imajlarının zaman içindeki güvenlik açığı değişikliklerine odaklanıldı. Bazı Docker imajları kullanımdan kaldırılmış ve bir yıldan uzun bir süre boyunca herhangi bir güncelleme almamıştır. Sonuç olarak, bu imajlar tüm tarama dönemlerinde aynı sonucu üretmektedir. Bu durum tezin amacımıyla çelişmektedir, Bundan dolayı bir sonraki adımda güncellenmemiş imajlar tespit edildi ve listeden kaldırıldı. Bunu yapmak için, imaj ve etiket bilgileri Docker Hub API'sinden sorgulandı. Sorgudan imaj ID'leri ve hash değerleri elde edilmeye çalışıldı ancak sorgu sırasında herhangi bir imaj ID bilgisine rastlanmadı. Bundan dolayı karşılaştırmaya imaj ID'leri yerine hash değerleri ile devam edildi. Her bir imaj etiketinin amd64 imajına ait hash değeri API üzerinden elde edilmiş ve hash değeri üzerinden karşılaştırma yapılmıştır. Amd64'ün seçilmesinin nedeni, en yaygın olarak kullanılan mimarisi olmasıdır ve tüm imajlar tarafından desteklenmesidir. Karşılaştırma sonucunda aynı hash değerine sahip imajlar hariç tutuldu, tablo 5.1'da detaylandırıldığı gibi toplamda 364 imaj seçildi ve seçilen imajlar taramaya hazırlandı.

5.1.2. Tarama İşlemi

Bir önceki adımda tarama işlemi için toplamda 1092 Docker imajı seçilmiştir. Bu miktarda imajın manuel olarak taranması mümkün değildir ve oldukça zaman alıcıdır. Sonuç olarak, tarama sürecini otomatikleştirmek için Bölüm 3.3'de belirtilen strateji uygulanmıştır ve tarama işlemi gerçekleştirilmiştir. Ardından tarama sonuçları Bölüm 3.4'te belirtildiği gibi işlenmiş ve değerlendirmeye hazırlanmıştır.

5.2. Araştırma 2 Sonuçları

İmaj seçimi ve seçilen imajların taranması, tarama sonuçlarının kaydedilmesi ve birleştirilmesi işlemlerinin tamamlanmasının ardından, bulguların değerlendirilmesi aşamasına geçilebilir.

Başlangıçta sonuçlar tarama araçları tarafından tespit edilen güvenlik açıklarının sayısına göre değerlendirdi ve karşılaştırıldı. Bu metrik, önemli bilgiler sağlamakla birlikte, yalnızca güvenlik açığı sayısına dayanır ve önem derecesini hesaba katmaz. Bunu gidermek için, sadece güvenlik açıklarının sayısını değil aynı zamanda potansiyel etkilerini de tanımlayan önem derecesine dayalı bir yaklaşım benimsenmiştir.

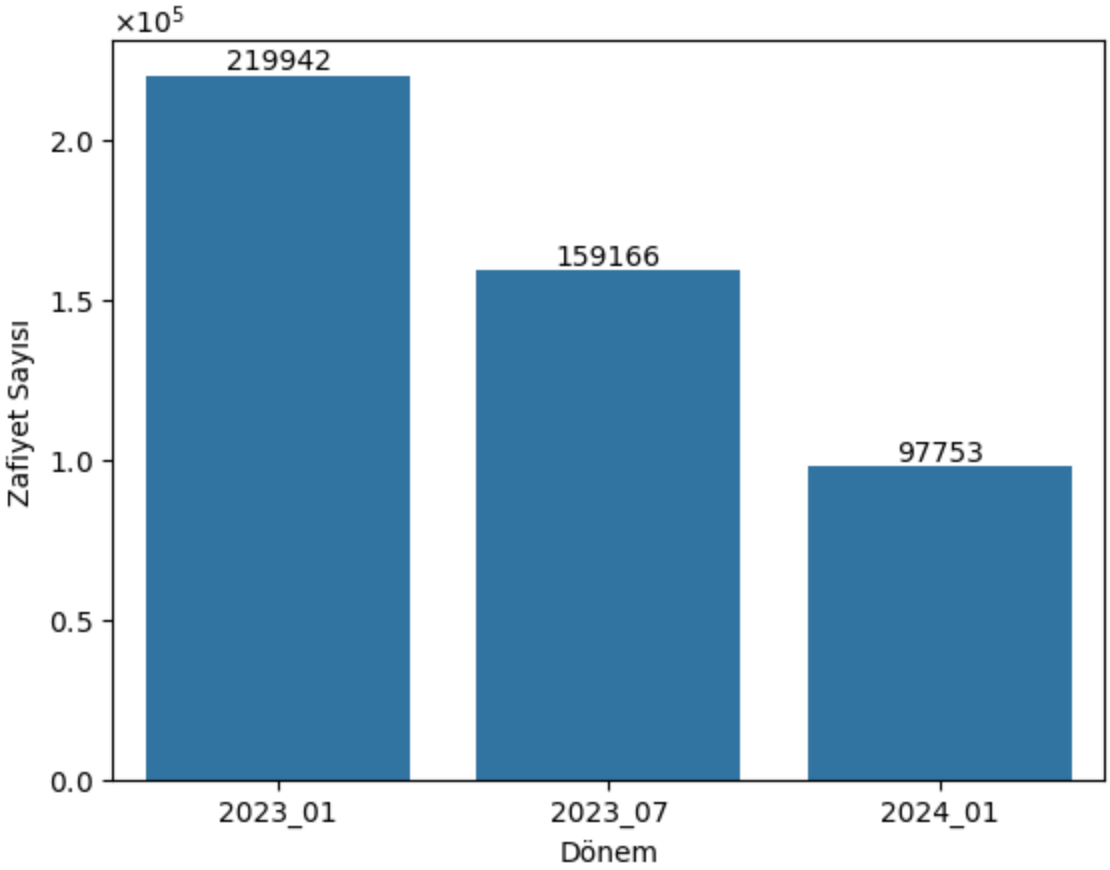
İkinci olarak, değerlendirme basit sayımların ötesine geçmiştir. Her bir periyodda en yaygın CVE'lerle birlikte her dönemdeki en çok zafiyet içeren Docker imajları da listelenmiştir. Bu analiz imaj kategorilerine göre daha da genişletildi.

Son olarak, analizin kapsamı, işletim sistemleri ve programlama diline özgü imajlar arasında güvenlik açığı dağılımındaki eğilimleri ortaya çıkarmak için genişletilmiştir. Bu analizler, Docker imajlarının değişen güvenlik açığı durumuna ilişkin değerli bilgiler sağlamıştır.

5.2.1. Tespit Sayısına Göre Sonuçlar

İlk adımda, sonuçlar güvenlik açığı tespit sayısına göre değerlendirilmiştir. Dönem-1'de tarama araçları 219.942 güvenlik açığı tespit ederek en yüksek sayıya ulaşmış, bunu 159.166 güvenlik açığı ile Dönem-2 ve 97.753 güvenlik açığı ile Dönem-3 izlemiştir. Sonuçlar şekil 5.1'da gösterilmektedir. Şekilde görüldüğü gibi 5.1 güncel olmayan docker imajlarındaki güvenlik açığı sayıları önemli ölçüde artmaktadır. Dönem-2, Dönem-3'e göre %62,82 daha fazla güvenlik açığı içermekte ve benzer şekilde Dönem-1, Dönem-3'e göre %125 daha fazla güvenlik açığı içermektedir. Bu rakamlardan, 1 yıl boyunca güncellenmemiş imajların güncellenmiş olanlara kıyasla iki kattan fazla güvenlik açığı içerdiğini söyleyebiliriz.

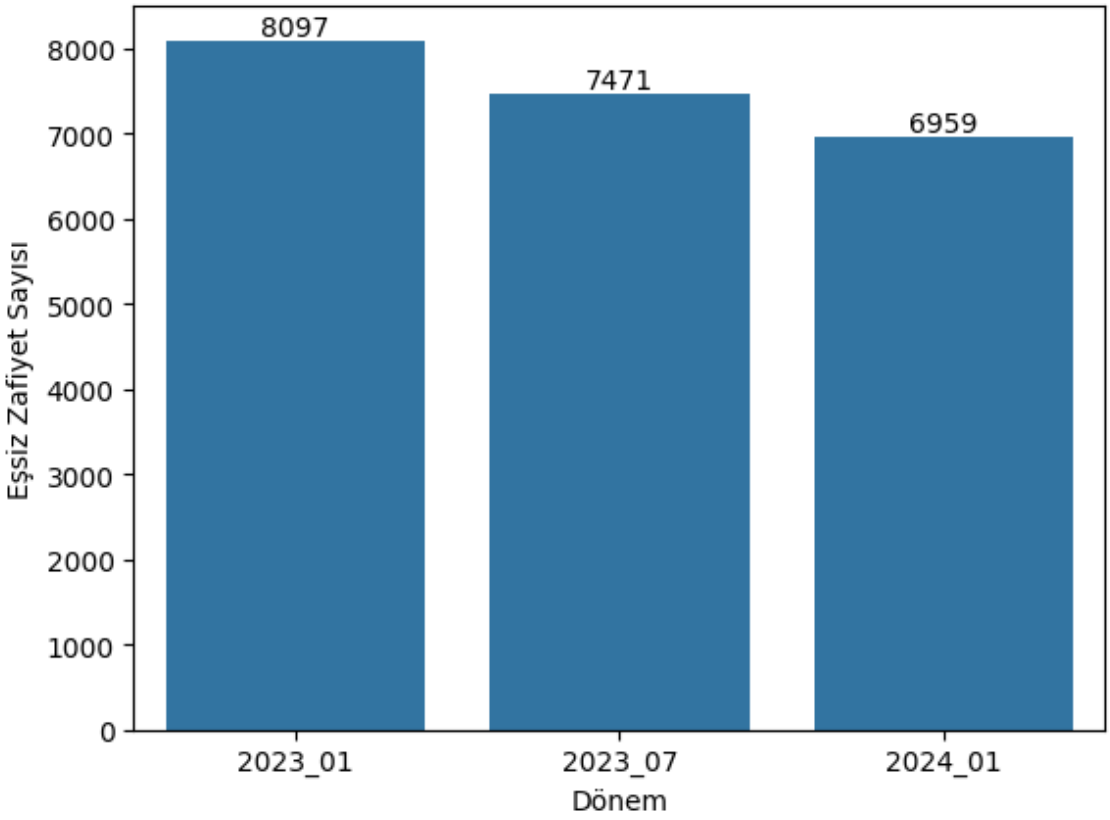
Şekil 5.2 ve tablo 5.2 dönemlere göre eşsiz (unique) zafiyet tespitlerini gösterilmektedir. Dönem-1'den Dönem-3'e kadar tüm imajlarda 1138 benzersiz CVE zafiyeti tamamen çözülmüştür.



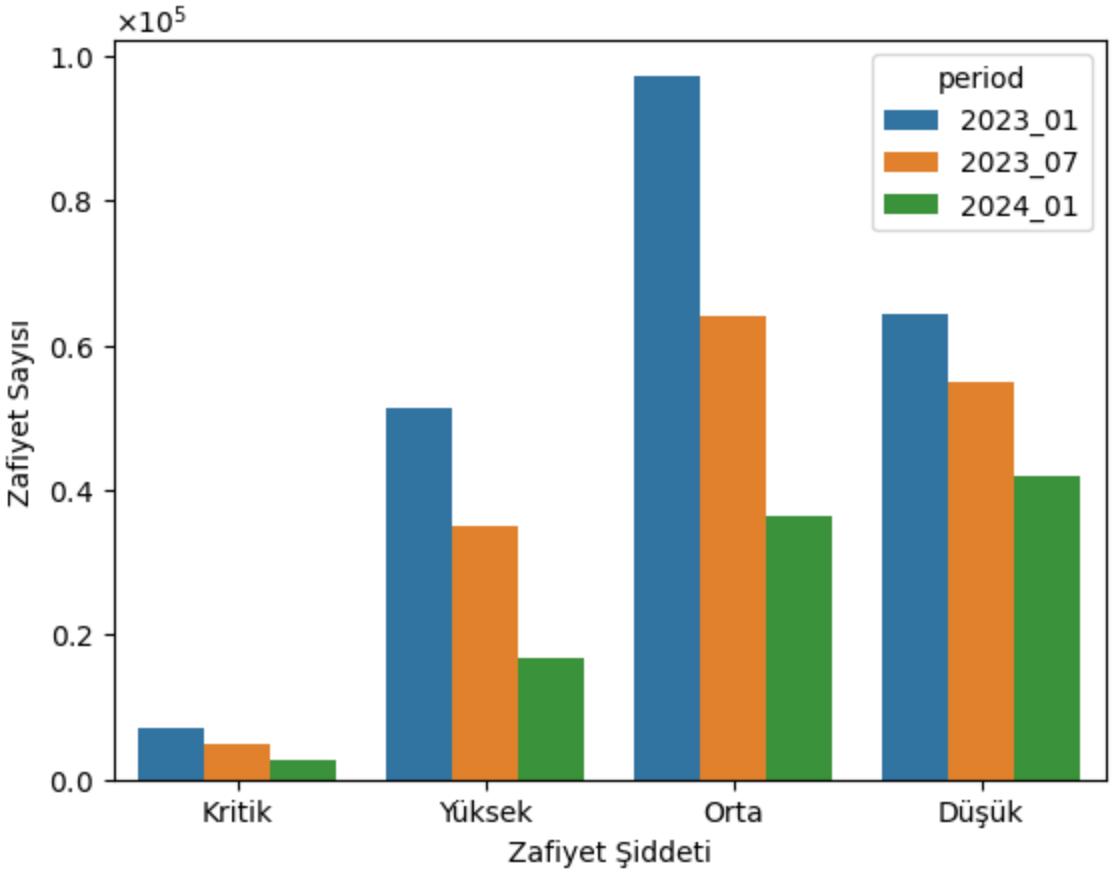
Şekil 5.1. Tüm dönemlerdeki tarama araçlarına ait tespit edilen zafiyetler.

Dönemler	Tespit Edilen Benzersiz Zafiyet Sayısı
Dönem-1	8097
Dönem-2	7471
Dönem-3	6959

Tablo 5.2. Tüm dönemlerde tespit edilen benzersiz güvenlik açıkları



Şekil 5.2. Tüm dönemlerde tespit edilen benzersiz güvenlik açıkları



Şekil 5.3. Şiddet ve döneme göre sonuçlar

Dönemler	Düşük	Orta	Yüksek	Kritik
Dönem-1	64263	97244	51442	6993
Dönem-2	55019	64094	35110	4943
Dönem-3	41802	36477	16728	2746

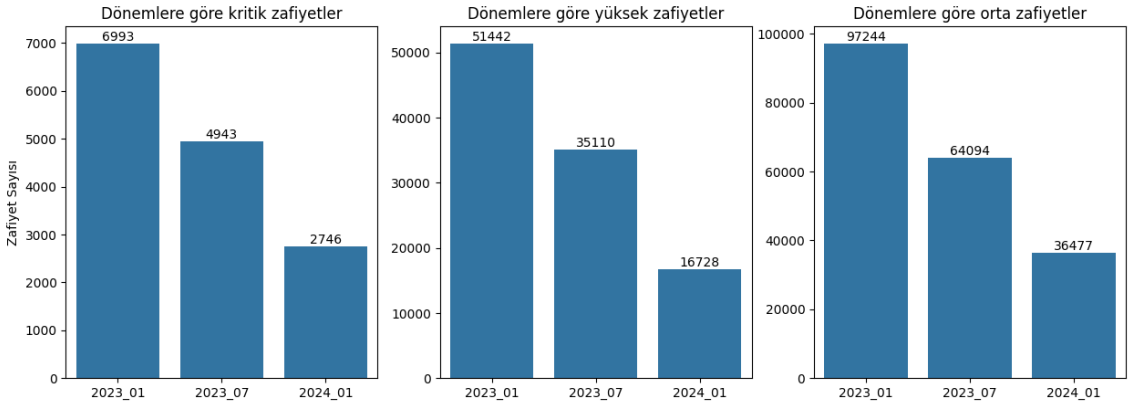
Tablo 5.3. Tarama aracına ve şiddete göre tespit edilen güvenlik açıkları.

Bu analizden elde edilen sonuçlar, Docker imajların düzenli olarak güncellenmesinin kritik öneminin altını çizmektedir.

5.2.2. Şiddet ve Döneme Göre Sonuçlar

Şekil 5.3 ve tablo 5.3 güvenlik açığı tespitlerini önem derecesine göre ayrılmış olarak görülmektedir. Tarama işlemi sırasında, ihmal edilebilir ve bilinmeyen güvenlik açıkları hariç tutulmuştur, çünkü bu güvenlik açıkları önemli bir riske sahip değildir.

Kritik güvenlik açıkları en tehlikeli ve ciddi olanlardır, bu da tespit edilmelerini çok önemli kılar, yüksek ciddiyetli güvenlik açıkları kritik olanlardan daha az ciddi olsa da tespit edilmeleri yine de önemlidir. Şekil 5.3 ve tablo 5.3'te, tüm dönemlerde güvenlik



Şekil 5.4. Zafiyet şiddetlerinin dönemler içerisindeki dağılımı

açığı sayısı zaman içinde azalmaktadır. Kritik ve yüksek güvenlik açıkları için güncel olmayan imajlar, güncellenmiş olanlara kıyasla iki kat daha fazla güvenlik açığı içermektedir.

Bu rakamlardan, bir yıllık güncellenmemiş Docker imajlarının güncellenmiş olanlara göre %154 daha fazla kritik güvenlik açığına sahip olduğunu, altı aylık güncellenmemiş docker imajlarının güncellenmiş olanlara göre %80 daha fazla kritik güvenlik açığına sahip olduğunu söylenebilir. Benzer şekilde, bir yıllık güncellenmemiş docker imajları güncellenmiş olanlara kıyasla %207 daha fazla yüksek güvenlik açığına sahipken, altı aylık güncellenmemiş docker imajları %109 daha fazla yüksek güvenlik açığına sahiptir. Yine benzer şekilde, bir yıllık güncellenmemiş docker imajları güncellenmiş olanlara göre %166 daha fazla orta düzey güvenlik açığına sahipken, altı aylık güncellenmemiş docker imajları %75 daha fazla orta düzey güvenlik açığına sahiptir.

5.2.3. Periyotlar Halinde En Savunmasız Docker İmajları

Docker imajlarındaki yüksek sayıdaki güvenlik açıkları önemli zararlara neden olabilir. Bu nedenle farklı dönemlerdeki en fazla güvenlik açığına sahip olan Docker imajları listelendi. Dönemlere göre güvenlik açığı değişikliklerini gözlemlemek için aynı imajların farklı dönemlerdeki güvenlik açıklarını da tabloya dahil edildi.

Tablo 5.4 Dönem-1'e ait en fazla zafiyet sayısına sahip Docker imajları "circleci/python:3.9.9-buster", ardından "apache/superset:2.0.1" ve sonra "library/hipache:0.3.1" olark gelmektedir. Bunlar Docker Hub'da en çok kullanılan Docker imajlarıdır. Bazı imajların güvenlik açıklarının "apache/superset",

Docker İmajı	**Dönem-1**	Dönem-2	Dönem-3
circleci/python	**4834**	4834	3987
apache/superset	**3731**	927	454
library/hipache	**3577**	3577	3578
library/haskell	**3433**	2266	1866
library/gazebo	**3250**	775	907
circleci/openjdk	**3199**	3199	3367
library/perl	**3170**	2171	967
library/elixir	**3059**	2211	1032
library/gcc	**2978**	2028	968
cimg/ruby	**2917**	2574	960

Tablo 5.4. Dönem-1'deki en zafiyetli docker imajları

Docker İmajı	Dönem-1	**Dönem-2**	Dönem-3
circleci/python	4834	**4834**	3987
library/hipache	3577	**3577**	3578
circleci/openjdk	3199	**3199**	3367
library/ros	2602	**2917**	349
library/silverpeas	2808	**2808**	725
cimg/ruby	2917	**2574**	960
redash/redash	2526	**2526**	3878
library/owncloud	2295	**2295**	2313
library/haskell	3433	**2266**	1866
library/elixir	3059	**2211**	1032

Tablo 5.5. Dönem-2'deki en zafiyetli docker imajları

"library/haskell", "library/perl" gibi önemli ölçüde azaldığını görebiliriz. İlginçtir ki bazı imajların güvenlik açıkları "circleci/openjdk" ve "library/hipache" gibi dönemler boyunca artmaktadır. Normal koşullarda daha eski Docker imajları daha fazla güvenlik açığı içermesi gerekir fakat güncel sürümlerde eklenen yeni bağımlılıklar, paketler ve yeni özellikler daha fazla güvenli açığının oluşmasına neden olabilmektedir. Bu durum, yeni Docker imajlarının, eski imajlarda olmayan yeni türde güvenlik zafiyetlerine maruz kalma olasılığını artırmıştır.

Tablo 5.5'de görüldüğü gibi Dönem-2'de "circleci/python:3.9.9-buster" hala en zafiyetli Docker imajıdır ve bu imaj önceki dönemden itibaren herhangi bir güvenlik güncellemesi almamıştır. İkinci olarak "library/hipache:0.3.1" ve üçüncü olarak "circleci/openjdk:17-buster" gelmektedir. Genel olarak çoğu imajın güvenlik açıkları dönemler içinde azalmaktadır.

Docker İmajı	Dönem-1	Dönem-2	**Dönem-3**
atlassian/default-image	1621	1266	**4373**
circleci/python	4834	4834	**3987**
redash/redash	2526	2526	**3878**
library/hipache	3577	3577	**3578**
circleci/openjdk	3199	3199	**3367**
library/owncloud	2295	2295	**2313**
library/haskell	3433	2266	**1866**
library/sl	1190	1190	**1721**
library/pypy	2532	2022	**1445**
cimg/python	2479	2039	**1357**

Tablo 5.6. Dönem-3'deki en zafiyetli docker imajları

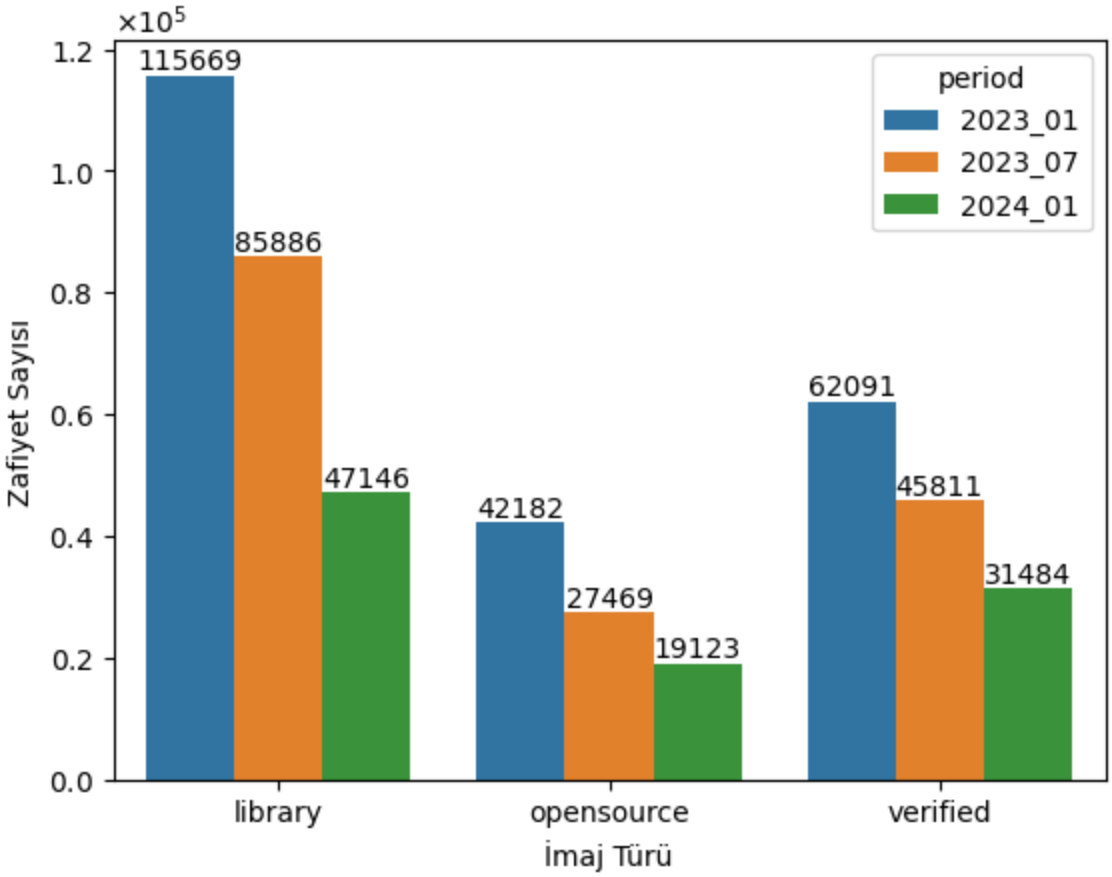
Tablo 5.6'de görüldüğü gibi Dönem-3 içinde "atlassian/default-image:latest" en zafiyetli imajdır, ardından "circleci/python:latest" ve üçüncü olarak "redash/redash:latest" gelmektedir.

Üç tablo göz önünde bulundurulduğunda, bu docker imajlarını kullanmaktan kaçınmanızı öneririz. Mümkünse sıkılaştırılmış (hardened) imajları kullanmayı düşünün.

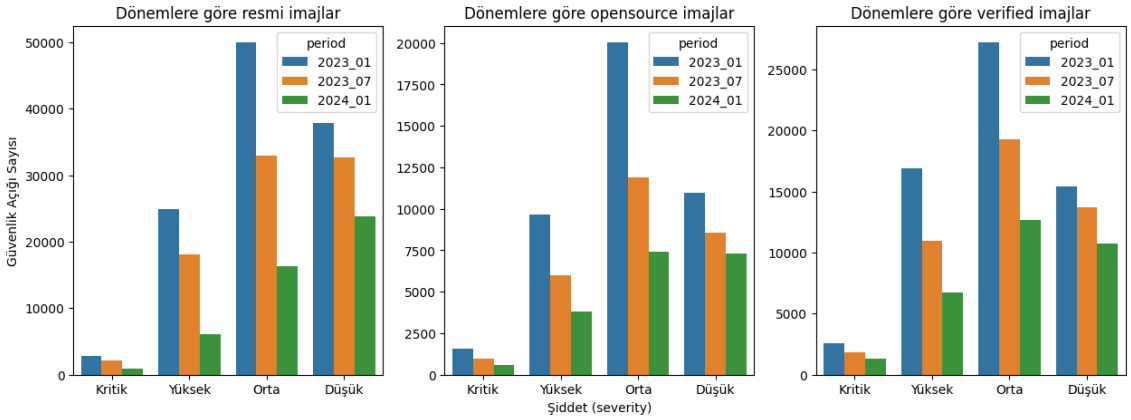
5.2.4. Dönemlerdeki İmaj Kategorileri

Şekil 5.5'de imaj kategorileri ve tespit edilen güvenlik açıkları görülmektedir. Güncellenmemiş docker imajları güncellenmiş olanlara kıyasla iki kat daha fazla güvenlik açığı içermektedir.

Güncel olmayan yazılım imajlarının, güncel olanlara kıyasla önemli ölçüde daha fazla güvenlik zaafiyeti içerdiği tespit adildi. Özellikle, bir yıldır güncellenmemiş library imajları, güncel olanlara kıyasla %145 daha fazla güvenlik açığı taşımaktadır. Bu güvenlik açığı artışı, yalnızca altı ay güncellenmemiş olan library imajları için %82'ye düşmektedir. Aynı eğilim açık kaynaklı imajlar için de geçerlidir. Bir yıldır güncellenmemiş açık kaynak Docker imajları, güncel sürümlere kıyasla %120 daha fazla güvenlik açığına maruz kalmaktadır. Bu risk, güncellemelerin yalnızca altı ay gerisinde kalan açık kaynaklı imajlar için %43 daha fazla güvenlik açığı ile hala önemlidir. Güncellemeleri bir yıl gecikmiş olan verified Docker imajları, güncellenmiş olanlara kıyasla %97 daha fazla güvenlik açığı taşımaktadır. Sadece altı ay güncellenmemiş verified imajlar için risk daha düşük olsa da %45'lık bir artış vardır.



Şekil 5.5. Dönemlere göre imaj türü güvenlik açıkları



Şekil 5.6. Zafiyetlerin imaj kategorilerine ve şiddete göre dağılımı.

Şekil 5.6'da gösterildiği gibi, veriler açık bir eğilimi doğrulamaktadır: güncel olmayan yazılım imajları, güncellenmiş benzerlerine kıyasla sürekli olarak daha fazla sayıda güvenlik açığı taşımaktadır. Bu güvenlik açığı farkı, çeşitli imaj türlerinde ve tüm güvenlik açığı şiddetlerinde belirgindir.

Bu eğilim, güncel yazılım imajlarını korumanın kritik önemini vurgulamaktadır. Kuruluşlar library, açık kaynaklı ve verified Docker imajlarını güncel tutarak güvenlik açıklarına maruz kalma risklerini önemli ölçüde azaltabilirler.

5.2.5. Tespit Edilen En Eski CVE'ler

Tablo 5.7 bölümünde tüm imajlar içerisinde tespit edilen en eski CVE'ler ve bunların tespit sıklıkları listelenmiştir. İlginç bir şekilde, çok eski CVE'ler hala Docker imajlarında bulunabilmektedir.

5.2.6. İşletim Sistemi İmajları

İşletim Sistemi (OS) imajları, Docker imajlarının büyük çoğunluğu için temel oluşturmaktadır. Python, OpenJDK ve Ruby gibi popüler imajlar, temel olarak OS imajlarından yararlanır. Bu katmanlı yaklaşım, birçok geliştiricinin uygulamaları için temel imajlar üzerine inşa etmesiyle daha da genişlemektedir. Sonuç olarak, bir işletim sistemi imajında bulunan herhangi bir güvenlik açığı, onu miras alan sonraki tüm imajlara yayılabilir. Bu miras zinciri, temel olarak kullanılan işletim sistemi imajlarını korumanın kritik öneminin altını çizmektedir. Zafiyetli bir temel (base) işletim sistemi imajının etkisi geniş ve kapsamlı olabilir. Yaygın olarak kullanılan bir Debian imajında bulunabilecek kritik bir güvenlik açığının bulunduğu bir senaryo düşünün. Bu güvenlik

Zafiyet (CVE)	Tespit Sayısı
CVE-1999-0236	14
CVE-1999-0678	12
CVE-1999-1237	14
CVE-1999-1412	14
CVE-2001-1534	143
CVE-2002-1976	6
CVE-2002-2439	7
CVE-2003-1307	143
CVE-2003-1580	143
CVE-2003-1581	143
CVE-2004-0230	118
CVE-2004-0971	10
CVE-2005-0406	560
CVE-2005-1119	16
CVE-2005-2541	387
CVE-2005-3660	118
CVE-2006-20001	77
CVE-2007-0086	157
CVE-2007-0450	14
CVE-2007-1667	3

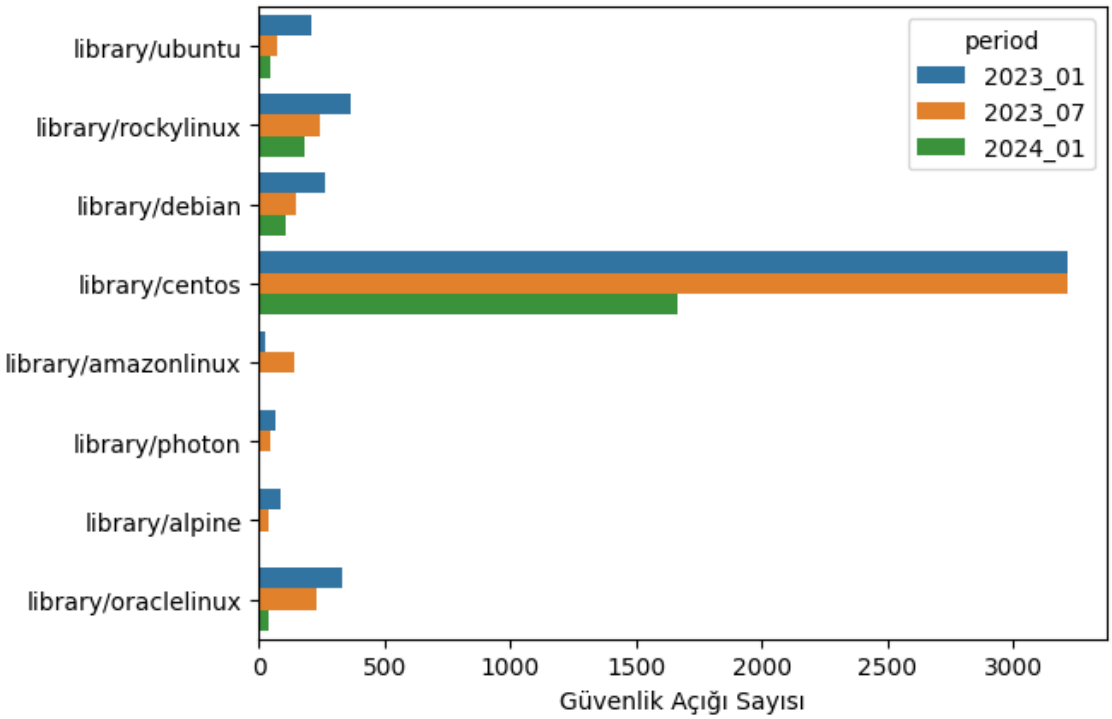
Tablo 5.7. Tespit edilen en eski CVE'ler ve tespit sayıları.

açığı Debian imajının kendisiyle sınırlı kalmayacaktır; aynı zamanda üzerine inşa edilen tüm Python, OpenJDK ve Ruby imajlarında ve hatta potansiyel olarak bu imajlar üzerine inşa edilen uygulamalarda da mevcut olacaktır. Bu zincirleme etki, tek bir güvenlik açığının neden olduğu potansiyel hasarı büyütür.

Şekil 5.7'da görüldüğü gibi RHEL tabanlı dağıtımlarda en zafiyetli işletim sistemi docker imajı CentOS'tur. Bunun nedeni, Docker Hub'daki CentOS imajlarının bir yıl içinde herhangi bir güncelleme almamış olmasıdır. Oracle Linux ve Rocky Linux gibi

İmaj İsmi	Dönem-1	Dönem-2	Dönem-3
library/alpine	87	40	4
library/amazonlinux	26	141	6
library/centos	3213	3213	1667
library/debian	262	149	104
library/oraclelinux	332	228	42
library/photon	64	46	0
library/rockylinux	363	245	182
library/ubuntu	206	72	46

Tablo 5.8. İşletim sistemi imajlarında periyotlara göre tespit edilen güvenlik açıkları.



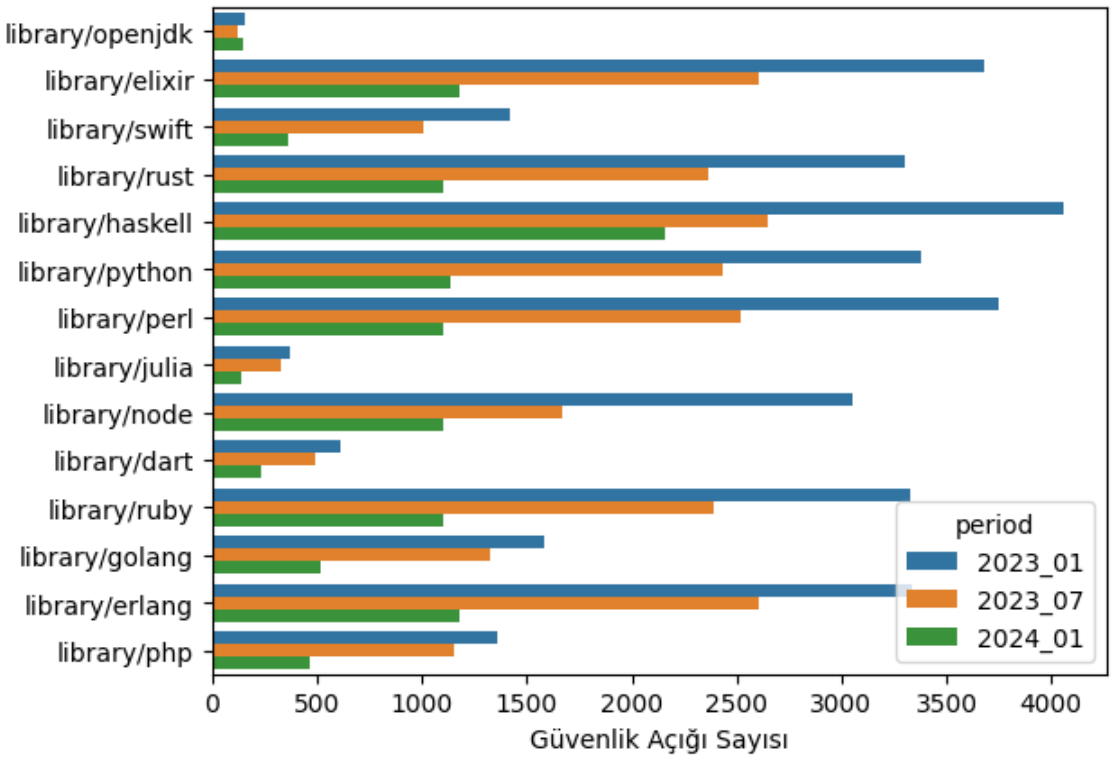
Şekil 5.7. OS imajlarına ait zafiyet sayıları.

diğer RHEL dağıtımları çok daha az güvenlik açığına sahiptir. Herhangi bir kullanım durumu için, Oracle Linux ve Rocky Linux Docker imajlarını CentOS imajlarına göre tavsiye edilebilir.

Tüm işletim sistemi imajlarında "library/photon" ve "library/alpine" en güvenli işletim sistemi imajlarından bazılarıdır çünkü Dönem-3'te bu imajların neredeyse sıfır veya birkaç güvenlik açığı vardır. Uygun olan yerlerde Alpine ve Photon docker imajlarını öneririz. Ubuntu Linux çok popüler bir Debian Linux dağıtımıdır, fakat ilginç bir şekilde Ubuntu Debian'dan daha az güvenlik açığına sahiptir. Bunun en temel sebebi olarak Ubuntu imajının Debian'a kıyasla daha az paket ile gelmesi ve daha küçük boyuta sahip olması (Debian 117MB, Ubuntu 76MB) gösterilebilir.

5.2.7. Programlama Dili İmajları

Programlama dili imajları birçok imaj tarafından temel imaj olarak kullanılır ve benzer şekilde birçok geliştirici bu imajları kendi uygulamalarında temel imaj olarak kullanırlar. Örneğin, Django, Flask ve FastAPI web çerçeveleri Python imajını temel imaj olarak kullanır, React, Vue, Nextjs, Express uygulamaları Nodejs'i temel imaj olarak kullanır, Springboot, JSF uygulamaları OpenJDK'yı temel olarak kullanır, Ruby on Rails



Şekil 5.8. Programlama dili imajlarının zafiyet sayısı.

uygulamaları Ruby imajlarını kullanır, Rust ve Golang uygulamaları uygulamaları oluşturmak/derlemek için Rust ve Golang imajlarını kullanır. Programlama dili imajlarında bulunan herhangi bir güvenlik açığı, bu imajları temel olarak kullanılan bütün Docker imajlarında da ortaya çıkar. Bu da programlama dili imajlarını çok önemli hale getirmektedir.

Şekil 5.8 ve tablo 5.9'da programlama dillerinin farklı dönemlerdeki güvenlik açıkları listelenmiştir. Güncellenmiş imajların eski imajlara göre çok daha az güvenlik açığı içerdiği gözlemlenebilir.

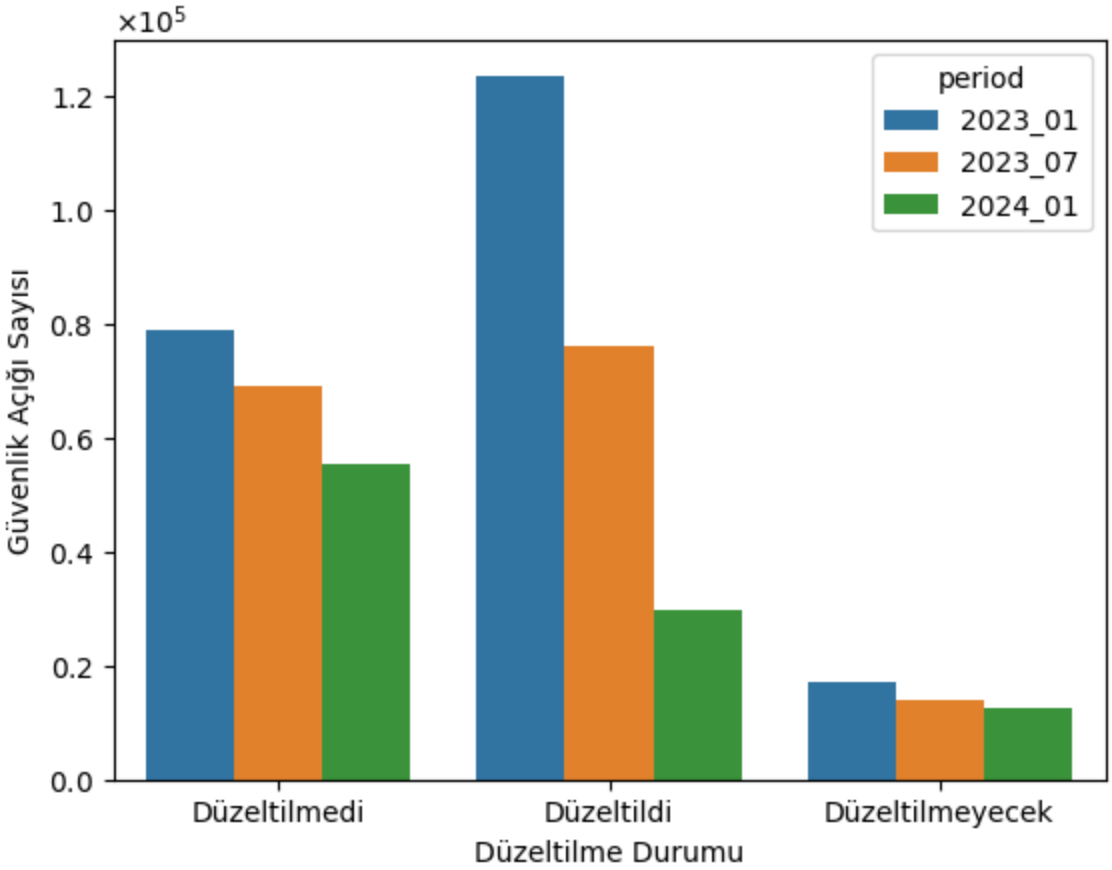
5.2.8. Zafiyet Düzeltilme Durumlarına Göre Dağılım

Her gün yeni güvenlik açıkları keşfedilmekte ve bu güvenlik açıklarını yamamak için sürekli olarak güncellemeler yayınlanmaktadır. Şekil 5.9'da güvenlik açığı düzeltilme durumu bulunabilir. Dönem-1'deki güvenlik açıklarının çoğunun "fixed" statüsüne sahip olduğuna dikkat edin, yani güncellemeler uygulanarak bu zafiyetler tamamiyle giderilebilir. Aynı şey Dönem-2 için de söylenebilir. Ayrıca, "not-fixed" ve "wont-fix" zafiyet değişikliklerinin "fixed" durumuna kıyasla önemli ölçüde değişmediği

Docker İmajı	Dönem-1	Dönem-2	Dönem-3
library/golang	159	117	145
library/ruby	3678	2605	1178
library/perl	1420	1006	363
library/openjdk	3304	2366	1102
library/rust	4061	2645	2155
library/python	3376	2434	1136
library/node	3751	2520	1102
library/erlang	372	324	141
library/haskell	3051	1664	1103
library/swift	613	487	236
library/julia	3324	2394	1102
library/php	1583	1321	520
library/dart	3335	2605	1178
library/elixir	1357	1153	461

Tablo 5.9. Dönemlere göre programlama dili imajlarındaki zafiyetler

söylenebilir.



Şekil 5.9. Güvenlik Açığı Düzeltme Durumu

6. KAPANIŞ

Docker imajlarının kullanımı, bilgi teknolojileri alanında uygulama ve yazılım geliştirmeyi kolaylaştırmıştır. Aynı zamanda yeni güvenlik problemlerini de beraberinde getirmiştir. Bu tez çalışmasında Docker imajlarındaki güvenlik açıklarının zaman içerisindeki değişiminin incelenmesi ve araştırılmasının yanı sıra mevcut Docker imaj tarama araçlarının etkinliği de ele alınmaktadır.

Çalışmanın ilk aşamasında Docker Hub'da resmi, doğrulanmış ve açık kaynak olarak kategorize edilen en popüler 439 Docker imajı üzerinde bir tarama gerçekleştirilmiştir. Tarama sonuçlarını değerlendirmek için güvenlik açığı tespit isabet oranı metriği kullanılmış ve CVSS güvenlik açığı puanlama sistemine dayalı yeni bir metrik önerilmiştir. Bu metrikler, imaj tarama araçlarının etkinliği hakkında bilgi sağlamış ve en son Docker imajlarındaki güvenlik açıklarını ortaya çıkarmıştır. Kapsamı artırmak için bu iki metrikten elde edilen sonuçların karşılaştırmalı bir analizi yapılmıştır. Bulgularımıza göre, en iyi tarama aracının bile Docker imajlarındaki önemli sayıda güvenlik açığını tespit edemediği ve diğer araçların daha da az etkili performans gösterdiği gözlemlenmiştir.

Çalışmanın ikinci aşamasında Docker Hub'dan alınan resmi, doğrulanmış ve açık kaynak olarak kategorize edilen en popüler 364 Docker imajı ve bu imajların 3 farklı sürümleri üzerinde bir tarama yapılmıştır. Tarama sonuçları, işletim sistemi docker imajlarının güvenlik açığı değişiklikleri, programlama dili docker imajları, kategoriye göre imajlar, dönemler arası zafiyet sayısı değişikliği vb. gibi farklı perspektiflerden değerlendirmek için kullanılmıştır. Bu metrikler Docker imajların güvenlik durumu hakkında fikir vermiş ve en güncel Docker imajlarındaki güvenlik risklerini ortaya çıkarmıştır.

7. VERİ ERİŞİLEBİLİRLİĞİ

Bu tez kapsamında oluşturulan veriseti, bu alanda daha farklı çalışmaları teşvik etmek için paylaşılmıştır. Verisetimiz, çalışmada kullanılan Bash ve Python betik dosyaları, Ipynb dosyaları, tarama sonuçlarının JSON hali ile birlikte GitHub depomuzda paylaşılmıştır. Bu kaynaklara aşağıdaki bağlantıdan erişilebilir:

```
https://github.com/hatsat32/yltez
```

KAYNAKLAR

Avi (2023). *11 Container Security Scanners to find Vulnerabilities*, https://geekflare.com /container-security-scanners/ (Ziyaret Tarihi: 16 Aralık 2023).

Berkovich, S., Kam, J., Wurster, G. (Ağustos 2020). UBCIS: Ultimate Benchmark for Container Image Scanning,

Bhat, S. (2023). *5 open source tools for container security*, https://opensource.com/articl e/18/8/tools-container-security (Ziyaret Tarihi: 16 Aralık 2023).

ChainGuard (2023). *ChainGuard Images*, https : / / images . chainguard . dev/ (Ziyaret Tarihi: 16 Aralık 2023).

CNCF (2022). *CNCF 2022 Annual Survey*, https://www.cncf.io/reports/cncf-annual-surv ey-2022/ (Ziyaret Tarihi: 10 Nisan 2024).

URL-1: *Common Vulnerability Scoring System SIG*, https://www.first.org/cvss/ (Ziyaret Tarihi: 16 Aralık 2023).

URL-2: *CVE-2019-5736*, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-5 736 (Ziyaret Tarihi: 16 Aralık 2023).

URL-3: *CVSS v3.1*, https://www.first.org/cvss/v3.1/specification-document (Ziyaret Tarihi: 16 Aralık 2023).

Dahlmanns, M., Sander, C., Decker, R., Wehrle, K. (2023). Secrets Revealed in Container Images: An Internet-Wide Study on Occurrence and Impact, 797–811. DOI: https: //doi.org/10.1145/3579856.3590329

Docker (2023). *Docker Scout*, https://docs.docker.com/scout/ (Ziyaret Tarihi: 16 Aralık 2023).

Doerrfeld, B. (2023). *10+ top open-source tools for Docker security*, https://techbeacon .com/security/10-top-open-source-tools-docker-security (Ziyaret Tarihi: 16 Aralık 2023).

Goodin, D. (2018). *Backdoored images downloaded 5 million times finally removed from Docker Hub*, https://arstechnica.com/information-technology/2018/06/backdoored -images-downloaded-5-million-times-finally-removed-from-docker-hub/ (Ziyaret Tarihi: 9 Nisan 2024).

Grande, E. (2023). *Dagda Github*, https://github.com/eliasgranderubio/dagda (Ziyaret Tarihi: 16 Aralık 2023).

Haque, M. U., Iwaya, L. H., Babar, M. A. (2020). Challenges in Docker Development: A Large-Scale Study Using Stack Overflow. DOI: https://doi.org/10.1145/3382494.3410693

Hykes, S. (2013). *The future of Linux Containers*, https://survey.stackoverflow.co/2023/ (Ziyaret Tarihi: 16 Aralık 2023).

Javed, O., Toor, S. (2021). An Evaluation of Container Security Vulnerability Detection Tools,

Khan, A. (2017). Key Characteristics of a Container Orchestration Platform to Enable a Modern Application. *IEEE Cloud Computing*, *4* (5), 42–48. DOI: https://doi.org/10.1109/MCC.2017.4250933

URL-4: *Malicious Docker Hub Container Images Used for Cryptocurrency Mining*, https://www.trendmicro.com/vinfo/us/security/news/virtualization-and-cloud/malicious-docker-hub-container-images-cryptocurrency-mining (Ziyaret Tarihi: 16 Aralık 2023).

URL-5: *MASSCAN*, https://github.com/robertdavidgraham/masscan (Ziyaret Tarihi: 16 Aralık 2023).

Mikkonen, T., Pautasso, C., Systä, K., Taivalsaari, A. (2022). Cargo-cult containerization: a critical view of containers in modern software development. *2022 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. DOI: https://doi.org/10.1109/sose55356.2022.00017

Mills, A., White, J., Legg, P. (Aralık 2023). Longitudinal Risk-based Security Assessment of Docker Software Container Images. *Computers & Security*, 103478. DOI: https://doi.org/10.1016/j.cose.2023.103478

Redhat (2023), https://quay.io/ (Ziyaret Tarihi: 16 Aralık 2023).

StackOverflow (2023). *StackOverflow 2023 Survey*, https://survey.stackoverflow.co/2023/ (Ziyaret Tarihi: 10 Nisan 2024).

Sultan, S., Ahmad, I., Dimitriou, T. (2019). Container security: issues, challenges, and the road ahead. *Ieee Access*, *7* 52976–52996. DOI: https://doi.org/10.1109/access.2019.2911732

URL-6: *The State of Kubernetes Security in 2023*, https://www.redhat.com/rhdc/managed-files/cl-state-kubernetes-security-report-262667-202304-en.pdf (Ziyaret Tarihi: 9 Nisan 2024).

Varun, M., Kalaiselvi, R., Ravisankar, S., Ravindran, D. (2023). Enhancing the Container Image Scanning Tool - GRYPE. *2nd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation, ICAECA 2023*. DOI: https://doi.org/10.1109/ICAECA56562.2023.10200828

Wist, K., Helsem, M., Gligoroski, D. (Haziran 2020). Vulnerability Analysis of 2500 Docker Hub Images. *CoRR*, *abs/2006.02932*

Wong, A. Y., Chekole, E. G., Ochoa, M., Zhou, J. (2023). On the Security of Containers: Threat Modeling, Attack Analysis, and Mitigation Strategies. *Computers & Security*, *128* 103140. DOI: https://doi.org/https://doi.org/10.1016/j.cose.2023.103140

Zhang, Q., Liu, L., Pu, C., Dou, Q., Wu, L., Zhou, W. (2018). *A Comparative Study of Containers and Virtual Machines in Big Data Environment* ()

ÖZGEÇMİŞ

2021 İstanbul Üniversitesi Bilgisayar Mühendisliği bölümünü bitirdi. 2022-2024 yılları arasında özel sektörde görev yapmıştır ve halen özel sektörde görev yapmaya devam etmektedir. İlgi alanları yazılım güvenliği, siber güvenliktir, uygulama güvenliğidir.