Araştırma Makalesi

ARAYÜZ ÇEŞİTLENDİRMESİNİN KÖTÜ AMAÇLI YAZILIMLARDA KULLANIM DURUMU

Nasrullah Frotan[†], Rıfat YAZICI[††]

[†] İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Ana Bilim Dalı, İstanbul, Türkiye

[††] İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Ana Bilim Dalı, İstanbul, Türkiye

frotan.nt@gmail.com, ryazici@ticaret.edu.tr

0000-0002-3245-597X, 0000-0003-0670-8803,

Atıf/Citation: FROTAN, N., YAZICI, R., (2024). Arayüz Çeşitlendirmesinin Kötü Amaçlı Yazılımlarda Kullanım Durumu, Journal of Technology and Applied Sciences 7(1) s.35-53, DOI: 10.56809/icujtas.1410198

ÖZET

Bu makale, dahili arayüz çeşitlendirmesinin etkinliğini göstermek amacıyla çeşitlendirme ile yenilenebilen güvenlik açıkları ve suistimallerin gerçek dünya örneklerini sunmaktadır. Arayüz çeşitlendirmesi, kötü amaçlı yazılımların birçok istismarı ve güvenlik açığı önlemeye yardımcı olmuştur. Ayrıca, büyük ölçekli siber saldırı tehdidini azaltmaya yardımcı olmuş ve kaynakları sınırlı olan cihazlarda ek yük oluşturmadığı görülmüştür. Arayüz çeşitlendirmesi basit işlemlerle uygulanabilir ve performans üzerinde olumsuz bir etkisi yoktur. Dahili arayüz çeşitlendirmesi, güncellemeler alamayan cihazlarda bile zararlı programların çalışmasını zorlaştırmaktadır. Ayrıca, bot ağları gibi saldırıları engellemekte ve yayılmasını önlemektedir. Bununla birlikte, bazı saldırı türlerine karşı etkili olmayabilir ve uygulaması zor bir yöntem olabilir.

Anahtar Kelimeler: arayüz çeşitlendirmesi, kötü amaçlı yazılımlar, siber güvenlik, IoT, kötü amaçlı yazımların tespiti ve engellenmesi

THE USE OF INTERFACE DIVERSIFICATION IN MALICIOUS SOFTWARE

ABSTRACT

This article provides real-world examples of vulnerabilities and exploits that can be renewed with diversification to demonstrate the effectiveness of internal interface diversification. Interface diversification has helped prevent many exploits and vulnerabilities of malware. It has also helped reduce the threat of large-scale cyberattacks and has not been shown to create overhead on devices with limited resources. Interface diversification can be implemented with simple operations and does not have a negative impact on performance. The internal interface diversification makes it difficult for malicious programs to run even on devices that do not receive updates. It also blocks attacks such as bot networks and prevents them from spreading. However, it may not be effective against some types of attacks and may be a difficult method to implement.

Keywords: interface diversification, malware, cybersecurity, IoT, detecting and blocking malware

Geliş/Received	**:**	**26.12.2023**
Gözden Geçirme/Revised	**:**	**29.01.2024**
Kabul/Accepted	**:**	**09.05.2024**

1. GİRİŞ

Kötü amaçlı yazılımlar, dijitalleşen dünyamızda giderek artan bir tehdit oluşturmaktadır. Özellikle, Android platformunda, ücretsiz uygulamaların yaygınlığı, adware olarak bilinen ve ciddi güvenlik sorunlarına yol açabilen reklam destekli yazılımların ortaya çıkmasına neden olmuştur (Seraj vd., 2023). Bu tür yazılımlar, kullanıcı verilerini çalarak ve cihazlara ek kötü amaçlı yazılımlar yükleyerek zararlı hale gelebilmektedir. Bu bağlamda, kötü amaçlı yazılımların, özellikle de adware'lerin, daha sofistike kaçınma teknikleri geliştirmesi nedeniyle, daha iyi tespit tekniklerine duyulan ihtiyaç artmaktadır.

Kötü amaçlı yazılım tespiti ve önlenmesi alanında, arayüz çeşitlendirmesi stratejisi, sistemlerin güvenliğini artırmak adına ciddi bir çözüm önerisi vadetmektedir. Bu strateji, kötü amaçlı yazılımların hedef sistemlerin dahili arayüzlerini kullanmasını zorlaştırarak, yazılım satıcılarının güvenlik yamalarını yayımlamasından daha hızlı bir şekilde harekete geçen saldırganlara karşı proaktif bir koruma sağlar. Bu yaklaşım, özellikle IoT cihazları ve siber uzaya bağlı kritik altyapı gibi giderek daha fazla dijitalleşen ve internete bağlanan sistemler için hayati önem taşımaktadır (Uitto vd., 2016). Günümüzde, anti-virüs yazılımlarının yeni tehditlerle başa çıkma konusunda yaşadığı zorluklar göz önünde bulundurulduğunda, bu tür çok katmanlı güvenlik önlemleri daha da önem kazanmaktadır. Arayüz çeşitlendirmesi, yazılımın iç yapısını değiştirerek tersine mühendislik yapılmasını zorlaştırırken, çeşitlendirme sayesinde, aynı programın farklı versiyonlarını üreterek kod-yeniden kullanım saldırılarını azaltır (Gobbo, 2023). Bu yöntemler, kötü amaçlı yazılımların iç detaylarını gizlemekte ve tespit edilmesini zorlaştırmaktadır. Bu nedenle, yazılım güvenliği sağlayıcıları, bu tür saldırılara karşı etkili olabilecek çok katmanlı güvenlik önlemleri geliştirmek zorundadır.

Bu çalışmanın literatüre katkısı, mevcut tespit ve önleme stratejilerinin ötesine geçen, özellikle sofistike kaçınma teknikleri geliştiren adware ve diğer kötü amaçlı yazılımların etkili bir şekilde tespit edilip engellenmesine yönelik yenilikçi bir yaklaşım sunmasıdır. Bu kapsamda, çalışma, siber güvenlik alanında, akademik literatür adına önemli bir adım olmanın yanı sıra endüstriyel uygulamalarda kullanılabilecek stratejiler sunma potansiyeli de taşımaktadır.

Çalışmanın yapılandırılması, öncelikle giriş bölümünde kötü amaçlı yazılımların mevcut durumuna ve bu tehditlerle mücadelede karşılaşılan zorluklara dair genel bir bakış sunulmasıyla başlar. Ardından, yöntem bölümünde, önerilen sistemin nasıl tasarlandığı ve uygulandığı detaylandırılır. Deneysel sonuçlar ve tartışma bölümünde, sistem testlerinin sonuçları sunulur ve bu sonuçlar literatürdeki benzer çalışmalarla karşılaştırılır. Son olarak, çalışma, elde edilen bulguların özetlendiği ve gelecek araştırmalar için önerilerin sunulduğu sonuç bölümüyle tamamlanır. Bu kapsamlı yaklaşım, kötü amaçlı yazılımlara karşı mücadelede yeni ve etkili yöntemlerin geliştirilmesine katkıda bulunmayı amaçlamaktadır.

2. KÖTÜ AMAÇLI YAZILIMLAR

2.1. Kötü Amaçlı Yazılımların Tanımı ve Genel Özellikleri

Kötü amaçlı yazılımlar, bilgisayar sistemlerine, ağlara veya cihazlara zarar vermek, izinsiz erişim sağlamak veya kullanıcı bilgilerini çalmak amacıyla tasarlanmış zararlı programlar veya kodlardır. Bu geniş kapsamlı terim, virüsler, solucanlar, truva atları, casus yazılımlar, reklam yazılımları, fidye yazılımları gibi çeşitli zararlı yazılım türlerini içerir ve bu yazılımların ortak özelliği, kullanıcının bilgisi dışında veya izni olmadan çalıştırılmalarıdır (Cooper Jr. vd., 2023). Bu yazılımların temel amacı, bilgi hırsızlığı, sistem performansının bozulması, kullanıcı etkinliklerinin izlenmesi ve diğer cihazlara zarar verme yeteneğidir (Acharya vd., 2021).

Kötü amaçlı yazılım kavramı, bir sistemin işlevselliğini kasıtlı olarak bozan veya sistemlere zarar veren kodları ifade eder ve bu genel tanım altında, kötü amaçlı yazılımlar farklı davranışlar sergileyebilir ve çok çeşitli zararlı aktivitelerde bulunabilirler. İnternetin ve dijital teknolojilerin yaygınlaşmasıyla birlikte, kötü amaçlı yazılımların yayılma yöntemleri de çeşitlenmiştir. Kullanıcılar, zararlı yazılım içeren e-posta eklerini açarak, güvenli olmayan web sitelerini ziyaret ederek veya zararlı uygulamaları indirerek bulaşma riski altındadır. Bu durum, kötü amaçlı yazılımların kolayca yayılmasına ve geniş bir kullanıcı kitlesini etkilemesine olanak tanır (Maniriho vd., 2022).

Zararlı yazılımların kullanımının artması ve bu yazılımların profesyonel gruplar tarafından özel amaçlar için kullanılmaya başlanması, kötücül yazılım çeşitliliğini, sayısını ve karmaşıklığını artırmıştır. Bu durum, sistemlerin işlevlerini kasıtlı olarak değiştiren ve sistemlere zarar veren kodların ortaya çıkmasına yol açmıştır (McGraw ve Morrisett, 2000). Vasudevan ve Yerraballi (2006) kötü amaçlı yazılımları, virüsler, casus yazılımlar, truva atları

ve diğer müdahaleci yazılımları içeren genel bir terim olarak tanımlamaktadır. Kötü amaçlı yazılımlar, cihaz kullanıcıları için bir risk teşkil eden her türlü kod veya yazılım olarak kabul edilmekte olup, sistem performansını negatif yönde etkileyerek yavaşlamaya sebep olan, genellikle kötü niyetli amaçlar için tasarlanmış programlardır.

Kötü amaçlı yazılımların davranışları farklı olabilir; ancak genellikle, güvenlik ihlallerine sebep olma, cihaz kontrolünün kullanıcıdan alınması, kötü amaçlı uzaktan erişim yazılımlarıyla cihaz özelliklerinin etkinleştirilmesi, kişisel verilerin başka alanlara aktarılması, diğer cihazları ve ağları olumsuz etkileme veya kullanıcılarını dolandırma gibi en az bir maddeyi gerçekleştirmeye çalışır (Felt vd., 2011). Bu çerçevede, farklı kaynaklardan indirilen uygulamalar, kötü amaçlı yazılım davranışlarına sebep olabilecek riskler taşımaktadır; ancak bu genellikle uygulamanın çalışma şeklindeki değişikliklerden kaynaklanır. Kötü amaçlı yazılımların çeşitliliği, yayılma yöntemleri, etkileri ve bu tehditlerle mücadele stratejilerinin önemi, bilişim güvenliği alanındaki araştırmalarda sıkça değinilen bir konudur.

2.2. Kötü Amaçlı Yazılımların Tarihsel Gelişimi

Kötü amaçlı yazılımların tarihi, bilgisayar teknolojilerinin gelişim süreciyle paralel bir evrim göstermiştir. İlk virüsler ve solucanlar, teknolojik merakın bir ürünü olarak ortaya çıkmış, zaman içinde ise karmaşık ve yıkıcı faaliyetler için kullanılan araçlara dönüşmüştür. 1980'lerin başlarında, bilgisayar virüsleri sahneye çıkmıştır. Bu dönemdeki virüsler, genellikle programcıların becerilerini sergilemek ya da sistem zayıflıklarını göstermek amacıyla yazılmıştır. Örneğin, 1982 yılında ortaya çıkan Elk Cloner, Apple II sistemlerini hedef alan ve kendini disketler aracılığıyla yayabilen ilk bilinen virüstür (Miles, 2012).

Şekil 1. Geçmişten Günümüze DDoS Saldırılarındaki Temel Motivasyon Unsurları (Atasever vd., 2019)

1990'ların ortalarına gelindiğinde, İnternetin yaygınlaşması kötü amaçlı yazılımların da evrimleşmesine zemin hazırlamıştır. Özellikle e-posta yoluyla yayılan virüsler, ağ güvenliğinin zayıf olduğu dönemlerde hızla çoğalmıştır. Melissa ve ILOVEYOU gibi zararlılar, milyonlarca bilgisayarı etkileyerek büyük ölçekli hasarlara neden olmuştur (Kabay, 2012). Bu dönemde kötü amaçlı yazılım saldırıları, sadece bireysel kullanıcıları değil, kurumsal ağları da hedef almaya başlamıştır.

Bu tarihsel süreç, kötü amaçlı yazılımların sadece teknik bir sorun olmaktan çıkıp, geniş çaplı sosyal ve ekonomik etkiler yaratan küresel bir tehdit haline geldiğini göstermektedir. Dolayısıyla, bu tehditlere karşı geliştirilen savunma mekanizmalarının da sürekli olarak güncellenmesi gerekmektedir.

2.3. Kötü Amaçlı Yazılım Çeşitleri

Dijital çağın getirdiği kolaylıkların yanı sıra, kötü amaçlı yazılımlar gibi siber tehditler de her geçen gün artmakta ve çeşitlenmektedir. Kötü amaçlı yazılım türlerinin anlaşılması, bu tehditlerle mücadelede önemli bir adımdır. Özellikle virüs ve solucanlar, zarar verme kapasiteleri ve yayılma yöntemleriyle öne çıkmaktadır ve bu durum onları siber tehditlerle mücadelede öncelikli hedefler haline getirmiştir. Virüs ve solucanın yanı sıra truva atları gibi diğer kötü amaçlı yazılımlar, sahte yararlılık vaatleriyle kullanıcıları kandırarak zararlı faaliyetlerde bulunurken, reklam yazılımları ve casus yazılımlar kullanıcı deneyimini olumsuz yönde etkileyerek gizlilik ve performans sorunlarına neden olmaktadır. Bu çeşitlilik ve geniş tehdit yelpazesi, siber güvenlik önlemlerinin sürekli olarak güncellenmesini ve çeşitlenmesini gerektirmekte, böylece güvenlik stratejilerinin bu dinamik tehdit ortamına uyum sağlaması sağlanmaktadır.

Virüs: Bilgisayar virüsleri, sistemlerin temel yapıtaşlarını hedef alarak, çeşitli zararlar verebilen kötü amaçlı yazılımlardır. Bu zararlı yazılımlar, genellikle kullanıcı etkileşimi gerektiren aktivitelerle sisteme bulaşır ve kendilerini fark ettirmeden ciddi hasarlar verebilirler. Virüsler, 1984 yılında Cohen tarafından tanımlanmış olup, çalıştırılabilir dosyalara eklenerek, ilgili program çalıştırıldığında aktif hale gelir ve kendi kodlarını diğer programlara kopyalayarak çoğalır (Cohen, 1987). Bu öz çoğaltma mekanizması, virüslerin en temel özelliği olarak kabul edilir (Aycock, 2006). Virüslerin yayılma yöntemleri, e-postalar, flash sürücüler ve internet üzerinden indirilen dosyalar gibi çeşitli yollardır. Sisteme bulaştıktan sonra, virüsler dosyalara zarar verebilir, sistem performansını düşürebilir ve istenmeyen yazılımların çalışmasına sebep olabilir (Pektaş, 2012). Bu zararlı yazılımların tanımlanması ve silinmesi genellikle zordur çünkü kasıtlı olarak karmaşık ve silinmesi güç bir yapıda tasarlanmışlardır. Virüslerin çeşitleri arasında, şifre çözme algoritması ve ana gövdeden oluşan yapıları ile dikkat çeken şifreli virüsler yer alır. Bu virüsler, kendilerini anti-virüs yazılımlarından saklamak için şifreleme teknikleri kullanır. XOR şifrelemesi gibi basit yöntemlerle, virüsler kendilerini gizleyebilir ve tespit edilmelerini zorlaştırabilir (Wong, 2006). Virüslere karşı korunma yöntemleri arasında, güncel anti-virüs yazılımlarının kullanımı, düzenli sistem taramaları ve şüpheli e-posta eklerini açmaktan kaçınma gibi önlemler yer alır. Ayrıca, kullanıcıların zararlı yazılımlar konusunda eğitilmesi ve farkındalık yaratılması, virüs saldırılarının önlenmesinde kritik bir öneme sahiptir (Aziz vd., 2021).

Solucanlar (Worm): bilgisayar ağları ve sistemler arasında bağımsız bir şekilde yayılabilen zararlı yazılımlardır. İnsan müdahalesi olmaksızın hızla çoğalabilme yetenekleriyle, bu zararlı yazılımlar özellikle ağ güvenliği için ciddi bir tehdit oluşturur. Solucanlar, zarar verme kapasiteleri ile basit rahatsızlıklardan ciddi ağ kesintilerine ve veri ihlallerine kadar geniş bir yelpazede etki gösterebilirler. Bu zararlı yazılımlar, güvenlik açıklarını ve sosyal mühendislik taktiklerini kullanarak yayılarak, tespit ve mücadelelerini zorlaştırır (Schmidt vd., 2009). Solucanların zarar verme yöntemleri arasında, sistem kaynaklarının aşırı kullanımı, görev yöneticisinin ve diğer kritik sistem araçlarının devre dışı bırakılması yer alır. Bu tür etkiler, kullanıcıların sistemlerini normal şekilde kullanmalarını engeller ve bazen ciddi performans düşüşlerine veya sistem çökmelerine yol açabilir. Ancak, modern antivirüs yazılımları bu zararlıları tespit etme ve temizleme konusunda oldukça etkilidir, bu da enfekte olmuş sistemlerin düzeltilmesi için geniş bir araç yelpazesi sunar. Solucanların yayılma mekanizmaları, otomatik yürütme özelliklerini, özellikle de "autorun.inf" dosyalarını kullanarak, bir ağ üzerinde hızla çoğalmalarını sağlar. Bu özellik, özellikle eski sistemlerde yaygın bir risk oluşturur ve solucanların hafıza aygıtları aracılığıyla kolayca yayılmasına olanak tanır. Ayrıca, e-posta ekleri veya sahte indirme bağlantıları aracılığıyla da yayılabilirler, bu da kendilerini kullanıcıların kişi listelerine dağıtarak daha geniş bir alana erişim sağlamalarına imkan tanır (Yiğit vd., 2012). Solucan ve virüsler arasındaki temel fark, solucanların insan etkileşimi gerektirmeksizin bağımsız olarak yayılabilme yetenekleridir. Virüsler genellikle bir konak dosyaya veya programına bağlıyken, solucanlar kendi başlarına hareket edebilir ve ağ üzerinden kendilerini çoğaltabilirler.

Truva atı (Trojan horse): Yaygın rastlanan diğer kötü amaçlı yazılımlardan olan truva atları, adları antik Yunan mitolojisinden gelen, kullanıcıların güvenini kazanarak bilgisayar sistemlerine sızan kötü amaçlı yazılımlardır. Truva atlarının temel amacı, sahte yararlılık vaatleriyle kullanıcıları kandırarak gizlice sistem kaynaklarına erişim sağlamak ve bu kaynakları zararlı amaçlar için kullanmaktır. Bu zararlı yazılımlar, veri çalma, kopyalama, değiştirme veya engelleme gibi işlemler gerçekleştirerek ciddi zararlara yol açabilir. Truva atları, gerçek dosya veya programlar gibi görünebilir ancak içlerinde kötü amaçlı kodlar barındırırlar. Kullanıcı etkileşimi gerektiren bu zararlı yazılımlar, e-posta ekleri veya zararlı web siteleri üzerinden sisteme bulaşabilir ve hedef sistemde geniş çapta zararlar verebilir (Li vd., 2021).

Reklam yazılımları (Adware): kullanıcıların rızası olmadan reklam yayınlayan ve genellikle diğer kötü amaçlı yazılımlarla birlikte bulaşan yazılımlardır. Kullanıcı deneyimini bozan bu yazılımlar, İnternet tarayıcı ayarlarını değiştirebilir ve kullanıcının karşısına istenmeyen reklamlar çıkarabilir. Bunlar genellikle zararsız kabul edilse de, bazı durumlarda kullanıcıyı zararlı içeriğe yönlendirebilir ve daha ciddi güvenlik tehditlerine kapı açabilir (Zhou, 2012).

Casus yazılımlar (Spyware): kullanıcıların bilgisayar kullanım alışkanlıklarını izleyen ve bu verileri üçüncü taraflarla paylaşabilen zararlı yazılımlardır. Bu yazılımlar, kullanıcıların farkında olmadan kişisel ve hassas bilgileri toplayabilir. Casus yazılımlar, genellikle bilgisayar performansında düşüşe, İnternet bağlantısında yavaşlamalara ve kullanıcı gizliliğinin ihlaline neden olur. ABD'de yapılan bir araştırmada, birçok şirketin casus yazılımlar nedeniyle finansal kayıplara uğradığı belirlenmiştir, bu da casus yazılımların ne kadar ciddi tehditler oluşturduğunu göstermektedir (Quinn, 2020).

Fidye yazılımları (Ransomware): kullanıcıların dosyalarını şifreleyerek onlardan fidye talep eden zararlı yazılımlardır. Bu saldırılar sonucunda, kullanıcılar önemli verilere erişimini kaybedebilir ve çoğu zaman verilerini

geri alabilmek için saldırganlara ödeme yapmaya zorlanır. Fidye yazılımları, son yıllarda özellikle Android ve Windows işletim sistemlerini hedef alarak yayılmıştır ve şüpheli linkler veya e-posta ekleri aracılığıyla bulaşabilir (Çelik vd., 2021).

Tuş kaydediciler (Keyloggers), kullanıcıların klavye hareketlerini kaydederek şifreler gibi hassas bilgileri ele geçirebilen zararlı yazılımlardır. Bu tür, kullanıcıların farkında olmadan finansal bilgiler veya kişisel verileri sızdırabilir, son derece tehlikeli ve gizli bir tehdit oluşturur (Çelik vd., 2021).

2.3. Kötü Amaçlı Yazılımların Tespiti

Kötü amaçlı yazılımların tespiti, siber güvenlik alanında sürekli gelişen ve karmaşıklaşan bir zorluk oluşturmaktadır. Bu yazılımlar, zarar verici faaliyetlerde bulunmak, kişisel verileri çalmak veya sistemlere izinsiz erişim sağlamak amacıyla tasarlanmıştır. Son yıllarda, kötü amaçlı yazılım tespit tekniklerinde önemli gelişmeler kaydedilmiş olup, bu teknikler temel olarak üç kategoriye ayrılmaktadır: imza tabanlı yaklaşım, davranış tabanlı yaklaşım ve sezgisel tabanlı yaklaşım.

İmza tabanlı yaklaşım: Kötü amaçlı yazılımların benzersiz bayt dizileri veya "imzaları" kullanılarak tespit edilmesine dayanır. Bu yöntem, zararlı yazılımların tanımlanmış bir veritabanına karşı karşılaştırılmasını içerir ve bilinen kötü amaçlı kodların tespitinde oldukça etkilidir. Ancak, yeni veya değişikliğe uğramış kötü amaçlı yazılımların tespitinde yetersiz kalabilir, zira bu yazılımların imzaları henüz veritabanında yer almamaktadır (Griffin vd., 2009; Gutmann, 2007). İmza tabanlı yöntemlerin bir diğer dezavantajı, veritabanının sürekli güncellenmesi gereksinimidir, bu da zaman ve kaynak tüketimi açısından zorluklar yaratır (Tran, 2013).

Davranış tabanlı yaklaşım: Bu yöntem kötü amaçlı yazılımların sistem üzerindeki etkileşimlerini ve davranışlarını izleyerek tespit etmeye çalışır, Zararlı yazılımların tipik olarak sergilediği davranış kalıplarına dayanır ve bu sayede bilinmeyen ya da yeni kötü amaçlı yazılımları tespit etme potansiyeline sahiptir. Davranış tabanlı tespit yöntemleri, zararlı yazılımların sistem kaynaklarına olan erişimlerini ve bu erişimler sırasında gerçekleştirdikleri işlemleri izler, bu da onları imza tabanlı yöntemlere kıyasla daha esnek kılar (Jacob vd., 2008).

Sezgisel tabanlı yaklaşım: Kötü amaçlı yazılımların tespitinde en yenilikçi yöntemlerden biridir ve özellikle metamorfik ve polimorfik virüsler gibi kendilerini değiştirebilen zararlı yazılımlara karşı etkilidir. Sezgisel yöntemler, potansiyel olarak zararlı olabilecek davranışların ve kod yapılarının genel özelliklerine dayanarak tespit gerçekleştirir. Bu yaklaşım, bilinen kötü amaçlı yazılımların imzalarına veya önceden tanımlanmış davranış kalıplarına güvenmek yerine, yazılımların potansiyel olarak zararlı olabilecek özelliklerine odaklanır (Nair vd., 2010; Govindaraju, 2010).

Her bir tespit metodunun avantajları ve sınırlılıkları göz önüne alındığında, en etkili kötü amaçlı yazılım tespit stratejisinin bu farklı yaklaşımların bir kombinasyonunu kullanmak olduğu görülmektedir. Bu kapsamlı yaklaşım, hem bilinen hem de bilinmeyen kötü amaçlı yazılımlara karşı daha geniş bir koruma sağlar. Özellikle, imza tabanlı ve davranış tabanlı yöntemlerin birleştirilmesi, geniş bir tespit kapsamı sunarken, sezgisel yöntemler bilinmeyen tehditlere karşı ek bir koruma katmanı ekler. Kötü amaçlı yazılımların tespiti, siber güvenlikte sürekli bir mücadele alanıdır.

3. ARAYÜZ ÇEŞİTLENDİRMESİ

Zararlı yazılımlar ile mücadele kapsamında, geleneksel ve modern savunma mekanizmalarının yanı sıra, arayüz çeşitlendirmesi gibi yenilikçi yaklaşımlar da önem kazanmaktadır. Arayüz çeşitlendirmesinin temelleri, sistemlerin ve uygulamaların farklı versiyonlarını ve konfigürasyonlarını kullanarak oluşturulur. Bu çeşitlilik, saldırganların bir sisteme yönelik özelleştirilmiş saldırılar geliştirmesini zorlaştırır çünkü her bir sistem benzersiz bir yapıya sahip olur (Mäki vd., 2016). Örneğin, aynı ağ içinde farklı işletim sistemleri kullanmak veya web sunucularında farklı arayüz yazılımları tercih etmek bu stratejinin uygulamalarındandır.

Arayüz çeşitlendirmesi, siber savunma stratejilerinde önemli bir rol oynar ve modern tehditlere karşı proaktif bir savunma katmanı sağlar. Bu yaklaşımın etkin bir şekilde uygulanması, kuruluşların siber direncini arttırır ve güvenlik açıklarını azaltır (Rauti & Leppänen, 2017).

3.1. Nesnelerin İnterneti İşletim Sistemlerinde Çeşitlendirilebilir Arayüzler

Nesnelerin İnterneti (IoT) işletim sistemlerinde bir güvenlik önlemi olarak çeşitlendirmenin uygunluğunu ve fizibilitesini daha iyi anlamak için yaygın IoT işletim sistemlerinde bulunan çeşitlendirilebilir arayüzlere bakıldığında çeşitlendirme için potansiyel hedefler olarak çeşitli arayüzler söylenebilir. Bunlardan ilki bellek alanıdır ki çeşitlendirilebilir bir arayüz olarak da görülebilmektedir. ASLR (Address Space Layout Randomization) arabellek taşması saldırılarına karşı koruma sağlayan bir güvenlik yaklaşımıdır. ASLR'de, saldırganın belirli bir kod parçasının bellekte nerede olduğunu tahmin etmesini zorlaştırmak için bir işlemin temel bölümlerinin adres alanı konumları rastgele yeniden düzenlenmektedir. Rastgele dağıtılan bellek düzeniyle, saldırgan bellekteki belirli bir konuma (istismar edilen bir işlev gibi) güvenilir bir şekilde atlayamamaktadır.

Veri yapılarında ise bellek düzeni rastgeleleştirmeyle ilgili olarak bireysel veri yapılarının düzeni de çeşitlendirilebilmektedir. Mesela, yapılar içindeki veri öğelerinin sırası değiştirilebilmekte veya aralarına ek dolgu eklenebilmektedir. Bu tür bir koruma Linux çekirdeği ana satırında uygulanmıştır. Çekirdek derlendiğinde, çekirdek içinde kullanılan veri yapılarının düzenini değiştirmek için özel bir derleyici eklentisi çağrılabilmektedir (Cook vd., 2018). Protokollerde ise IoT cihazları bir ağda çalışırken iletişim kurmak için protokolleri kullanmaları gerekmektedir. Bu protokoller aynı zamanda çeşitlendirme hedefleri olarak da görülebilmektedir. Mesela, Kısıtlı Uygulama Protokolü 'CoAP' (Constrained Application Protocol) basit elektronik cihazların birbirleriyle iletişim kurmasını sağlayan bir yazılım protokolüdür. IoT'deki "nesneler" gibi kaynakları kısıtlı cihazlar için özel olarak tasarlanmış bir uygulama katmanı protokolüdür. Bir protokolü çeşitlendirerek, orijinal protokolden benzersiz şekilde çeşitlendirilmiş çok sayıda protokol meydana getirilebilmektedir. Protokolü bir durum makinesi şeklinde düşünüldüğünde bu, çeşitlendirmenin orijinal protokole keyfi olarak birçok yeni durum ve geçiş ekleyebileceği manasına gelmektedir (Shelby vd., 2014).

Diğer bir yandan Google tarafından IoT'ye yönelik bir işletim sistemi olan Android Things (eski adıyla Brillo), hâlâ geliştirici ön izleme aşamasındadır, fakat güvenlik ve çeşitlendirme açısından, mimari gelecek vadetmektedir (Amadeo vd., 2017). Ayrıca ARM (Advanced RISC Machines)'nin gömülü işletim sistemi, donanımda tam bir MMU (Bellek Yönetim Birimi) olmadan, bunun yerine birçok Gelişmiş RISC Makineleri (ARM) işlemcisinde bulunan hafif bir Bellek Koruma Birimi (MPU) desteği ile sistem bileşenlerini birbirinden izole etmek için bir süpervizör içerebilmektedir.

3.2. Dahili Arayüz Çeşitlendirme

Günümüzün birbirine bağlı bilgisayar sistemlerinde, kötü niyetli yazılımlar ciddi bir tehdit oluşturmaktadır. Yeni kötü amaçlı programlar hızla ortaya çıkmaya devam etmektedir ve her gün binlercesi keşfedilmektedir. Bir yazılım parçasında güvenlik açığı bulunduğunda, saldırganlar genellikle yazılım satıcılarının yamalarını beklemek yerine hızla bu açıklardan faydalanabilmektedir.
Anti-virüs programları da bu hızlı gelişmelere ayak uydurmakta zorlanmakta ve kaynakları sınırlı olan IoT gibi ortamlarda kullanılamamaktadır. Bu nedenle, kötü amaçlı yazılım saldırılarını azaltmak için yeni proaktif yöntemlere ihtiyaç duyulmaktadır. Bunun da uygulanması şu şekilde ele alınabilir:

Arayüz Çeşitlendirme Uygulanışı: Arayüz çeşitlendirmesi, her belirli sistemdeki arayüzleri benzersiz kılmaktadır. Başka bir deyişle, çeşitlendirme, işletim ortamını kötü amaçlı yazılım için öngörülemez hale getirmekle ilgilidir.

Dahili arayüzlerin çeşitlendirilmesi, kötü niyetli bir yazılımın çalıştığı sistemin özellikleri hakkında yapabileceği varsayımların sayısını azaltmaktadır. Pratikte, şaşırtma dönüşümleri kullanarak çeşitlendirme uygulanabilmektedir. Arayüz çeşitlendirmede, basit şaşırtma dönüşümleri bile kullanılarak fonksiyonların adlarının değiştirilmesi veya parametrelerin fonksiyon imzalarında görünme sırasının değiştirilmesi gibi yöntemler kullanılabilmektedir. Ayrıca, farklı sistemler için çeşitlendirilmiş yazılımların benzersiz sürümlerinin otomatik olarak oluşturulması da mümkündür. Dahili arayüz çeşitlendirmesi, saldırıları dahili arayüz bilgilerine dayalı saldırılara karşı koruma sağlamaktadır. Yaygın bir örnek olarak farklı enjeksiyon saldırıları gösterilebilmektedir. Bunlarla birlikte yönlendirme gibi birçok diğer saldırı, bozuk bir kimlik doğrulama mekanizmasını atlamayı içerdiğinden dahili arayüz çeşitlendirmesiyle engellenememektedir. Bu makalede, üç farklı saldırı yakından incelenmiş ve dahili arayüzlerin çeşitlendirilmesinin bu saldırıların hedef sistemde zarar vermesini nasıl engellediği açıklanmıştır.

İlk olarak "Mirai"ya bakıldığında, bot adı verilen çok sayıda virüslü cihazı birbirine bağlayarak büyük bir botnet oluşturmak için kullanılan bir kötü amaçlı yazılım parçasıdır. Mirai, yaygın olarak kullanılan varsayılan kullanıcı adlarını ve parolaları deneyerek kötü niyetli oturum açma girişimleri için SSH (Secure Shell) ve Telnet

protokollerini kullanmaktadır. Daha sonra, sahiplerinin izni olmadan, virüslü cihazlar Internet'teki bilgisayarlara yönelik saldırılar başlatmak için kullanılmaktadır. Bu saldırılar genellikle Dağıtılmış Hizmet Reddi (DDoS) saldırılarıdır ve çok sayıda (hatta on binlerce) bot bir sunucuya trafik gönderir ve sunucunun normal istemcilerden gelen isteklere yanıt vermesini engellemektedir. Mirai'nin bulaştığı cihazların, genellikle bot oluşturmak için toplanan virüslü cihazlardan farklı olduğu gözlenmektedir. Geçmişteki botnet saldırılarının çoğu virüs bulaşmış ev bilgisayarlarından yararlanırken, Mirai güvenlik kameraları gibi IoT cihazlarına bulaştırmıştır.

Mirai'nin kaynak kodu Ekim 2016'da ortaya çıkmıştır. Bu kötü amaçlı yazılım, botnet'teki cihaz sayısını artırmak için geniş bir IP adresi taraması yapmakta ve IoT cihazlarını tehlikeye atmaktadır. Uzaktan bir komut ve kontrol sunucusundan alınan yönergeler doğrultusunda, Mirai farklı türde DDoS saldırılarına, örneğin GRE IP flood ve SYN ve ACK flood gibi saldırılara katılabilmektedir. Ayrıca, Mirai bölgesel bir kötü amaçlı yazılım olup IoT cihazına uzaktan erişimi engellemek için SSH, Telnet ve HTTP bağlantı noktalarını kapatmaktadır. Bunun yanı sıra, arayüz saptırma ve komut ve kontrol sunucusuna bağlantı kurmak gibi kötü amaçlı yazılım tarafından gerçekleştirilen diğer birçok eylemi de doğal olarak engelleye bilmektedir. Arayüz çeşitlendirmesi, kaynakları sınırlı olan IoT cihazları için iyi bir seçenek olduğunu göstermektedir.

ShellShock ise, ilk olarak Eylül 2014'te keşfedilen Bash'teki (Unix komut kabuğu) bir güvenlik açığıdır. ShellShock ile kötü niyetli bir yazılım, Bash'in keyfi komutlar yürütmesini sağlayabilmektedir ve örneğin istekleri işlemek için Bash kabuğunu kullanan web sunucuları gibi halka açık hizmetlere yetkisiz erişim elde edebilmektedir. Güvenlik açığı, yeni oluşturulan bir kabuk örneği, işlev gibi görünen bir kod içeren bir ortam değişkeniyle karşılaştığında ve ardından onu değerlendirdiğinde ortaya çıkmaktadır. Bash kodunda, işlev tanımı sona erdiğinde değerlendirmenin durmamasına neden olan bir hata vardı. Kodda, x değişkenine ayarlanan değer, bir fonksiyon tanımına benzerlik göstermektedir. Burada işlev yalnızca tek bir iki nokta üst üste, hiçbir şey yapmayan basit bir komuttur. İşlev tanımını sonlandıran noktalı virgülden sonra bir yankı komutu vardır. Bu komutun burada olmaması gerekmesine rağmen, rakibin onu oraya koymasını hiçbir şey engelleyememektedir.

```
env x='() { :;}; echo TEST' bash -c :
```

Şekil 2. Güvenlik açığının kullanılması.

Son olarak, yine hiçbir şey yapmayan bir iki nokta üst üste komutuyla yeni bir kabuk örneği başlatılmaktadır. Ancak, yeni kabuk örneği başladığında ve sağladığımız ortamı okuduğunda, x değişkenini de okumaktadır. Bu değişkenin içeriği bir fonksiyon gibi göründüğü için değerlendirilmektedir. İşlev tanımı yüklendikten sonra, düşman tarafından oluşturulan kötü amaçlı yük de yürütülmektedir.

"Advanced power" botnet ve Yapılandırılmış Sorgu Dili (SQL) dilini çeşitlendirmede ise son zamanlarda web uygulamalarının karşılaştığı önemli tehditlerden biri SQL enjeksiyon saldırısıdır. Bir web sunucusunda çalışan sözde kodu gösteren aşağıdaki örneğe bakıldığında bir kullanıcı, bir kullanıcı adı ve parola ile kimlik doğrulaması yaptığında kullanılmaktadır. Veri tabanı, kullanıcı adı ve parola sütunlarına sahip tablo kullanıcıları içermektedir. Örneğin, bir saldırgan girişte kötü amaçlı SQL komutları kullanabilmekte, böylece veri tabanı sunucusunun yürüttüğü SQL deyimi değiştirilmektedir. Örneğin, parola alanı aşağıdaki gibi ayarlanmışsa sunucu SQL sorgusunu yürütmektedir.

```
password' OR 1=1
```

Şekil 3. Parola alanının ayarlanması

```
SELECT id FROM users WHERE username='username'
                       AND password='password' OR 1=1'
```

Şekil 4. Sunucu tarafından yürütülen SQL sorgusu

OR 1=1 ifadesi, WHERE yan tümcesinin users tablosunun ilk kimliğini döndürmesini sağlamaktadır. Kullanıcı adı ve şifre değerlerinin ne olduğu önemli değildir. Listelenen ilk kullanıcı bir yöneticiyse (ki bu genellikle böyledir), saldırgan yönetici ayrıcalıkları da kazanmaktadır. Bunlarla birlikte kötü niyetli SQL girişlerini icat etmek ve test etmek sıkıcı ve zaman alan bir süreç olabilmektedir. Bundan dolayı, "Advanced Power" botnet gibi birçok saldırıda SQL enjeksiyon saldırıları otomatikleştirilmiştir. Bu kötü amaçlı yazılım, meşru bir Firefox

eklentisi kılığına girdi ve virüslü makineleri, kullanıcı tarafından göz atılan web sitelerinde birkaç farklı SQL enjeksiyon saldırısını test etmek için kullanmıştır. Bash komut satırı yorumlayıcısında olduğu gibi, buradaki çözüm, saldırganın anahtar sözcükleri bilmemesi için SQL'i çeşitlendirmektir. Saldırgan, yukarıda açıklanan saldırıda başarılı olmak için veya anahtar kelimesinin çeşitlendirilmiş dilde nasıl sunulduğunu bilmelidir. Ek güvenlik için operatörler de çeşitlendirilebilir. Ayrıca, bir anahtar kelimenin beklenen çeşitlenmesi, zamana veya o anahtar kelimenin ifadedeki konumuna bağlı olarak bile değişebilmektedir.

4. METODOLOJİ

4.1. Tehdit Senaryoları ve Etki Alanlarının Kapsamlı Analizi

Günümüzde, server işletim sistemlerine yönelik tehditler, teknolojinin hızla gelişimiyle birlikte daha çeşitli ve sofistike bir yapıya bürünmüştür. Bu tehditler, bireysel kullanıcılardan büyük ölçekli kurumsal yapıları da içerecek şekilde geniş bir kapsama sahiptir. Sosyal mühendislikten ağ zafiyetlerinin kötüye kullanılmasına kadar uzanan yayılma yöntemleri, phishing (oltalama) saldırıları ve ransomware gibi zararlı yazılımlar aracılığıyla kişisel ve finansal bilgilerin ele geçirilmesi ve kamu sektörü üzerinde ciddi zararlara yol açılması gibi çeşitli riskleri beraberinde getirmektedir (Connolly vd., 2020). Server işletim sistemleri, özellikle Windows Server ve Linux tabanlı sistemler, EternalBlue gibi exploitler ve Apache, OpenSSL gibi yazılımlardaki güvenlik açıkları aracılığıyla gerçekleştirilen saldırılarla karşı karşıya kalmaktadır (Liu vd., 2022; Van Heerden vd., 2018). Bu zafiyetler, siber saldırganlara hassas verilere erişim imkânı tanıyarak kurumların ve bireylerin güvenlik savunmalarını tehdit etmektedir.

Bu bağlamda, tehditlerin ve etki alanlarının kapsamlı bir şekilde analiz edilmesi, etkili savunma mekanizmalarının tasarlanması ve uygulanması için hayati öneme sahiptir. Kötü amaçlı yazılımların yayılma stratejileri ve saldırı vektörleri, geniş bir yelpazede zarara yol açabilme potansiyeline sahiptir. Bu nedenle, güvenlik önlemlerinin sürekli olarak güncellenmesi ve yeni savunma stratejilerinin geliştirilmesi gerekmektedir. Geleneksel güvenlik yaklaşımları, antivirüs yazılımları ve güvenlik duvarları gibi araçlar, imza tabanlı tespit yöntemlerini kullanarak zararlı yazılımları tanımlamaya çalışırken, modern savunma teknikleri, davranışsal tespit sistemleri gibi araçlarla daha etkili bir koruma sağlayabilmektedir (Chakravarty vd., 2019).

Bu çalışmada geliştirilen uygulama, kötü amaçlı yazılımların server işletim sistemlerine erişimini engelleyerek güvenliği artırmayı amaçlamaktadır. uygulama, zararlı yazılımları gerçek servisler yerine izole edilmiş bir sahte servis katmanına yönlendirerek etkisiz hale getirir. Bu yaklaşım, zararlı yazılımların sisteme zarar vermesini önlerken aynı zamanda server işletim sistemlerinin güvenliğini artırır. Server işletim sistemlerine yönelik tehditlerin ve etki alanlarının derinlemesine analizi, etkili savunma stratejilerinin geliştirilmesi adına son derece yüksek öneme sahiptir. Gelişen siber tehditlere karşı proaktif bir yaklaşım benimseyerek, server sistemlerinin güvenliğini sağlamak ve potansiyel zararları minimize etmek mümkündür. Bu yaklaşım, siber güvenlik alanında sürekli bir adaptasyon ve yenilik gerektirmektedir.

4.2. Kötü Amaçlı Yazılımlara Karşı Geliştirilen Savunma Mekanizmaları

Kötü amaçlı yazılımlara karşı mücadele, siber güvenlik alanında sürekli gelişim ve yenilik gerektiren bir süreçtir. Bu sürecin temelinde, tehditleri erken aşamada tespit etmek, engellemek ve zararlarını minimize etmek için geliştirilen savunma mekanizmaları yer alır. Günümüzde, server işletim sistemlerine yönelik kötü amaçlı yazılımların artan çeşitliliği ve sofistikasyonu, savunma stratejilerinin de buna uygun olarak evrilmesini zorunlu kılmaktadır.

Geleneksel güvenlik çözümleri, kötü amaçlı yazılımları tespit etmek ve engellemek için önemli bir temel sağlar. Bu çözümler arasında, antivirüs yazılımları ve güvenlik duvarları bulunur. Antivirüs yazılımları, zararlı yazılımları tanımlamak için genellikle imza tabanlı tespit yöntemlerini kullanırken, güvenlik duvarları izinsiz erişimleri engelleyerek ağ güvenliğini artırmayı hedefler (Caruso, 2003). Ancak, sıfır gün saldırıları gibi yeni ve bilinmeyen tehditlere karşı, bu yöntemlerin yetersiz kalabileceği bilinmektedir. Bu nedenle, daha dinamik ve uyum kabiliyeti yüksek savunma mekanizmalarının geliştirilmesi gerekmektedir. Davranışsal tespit sistemleri, bu ihtiyacı karşılamak üzere tasarlanmıştır. Bu sistemler, normalden sapma gösteren davranışları izleyerek, kötü amaçlı yazılımların potansiyel tehditlerini saptar. Davranışsal tespit, özellikle yeni ve gelişmekte olan kötü amaçlı yazılımların tespit edilmesinde etkili bir yöntem olarak öne çıkmaktadır (Chakravarty vd., 2019).

Bu çalışmada geliştirilen uygulama, kötü amaçlı yazılımların server işletim sistemlerine erişimini engelleyen yenilikçi bir savunma mekanizması sunmaktadır. uygulama, kötü amaçlı yazılımları gerçek servisler yerine özel

olarak tasarlanmış bir sahte servis katmanına yönlendirerek etkisiz hale getirir. Bu katman, zararlı yazılımların sisteme zarar vermeden önce izole edilmesini ve etkisizleştirilmesini sağlar. Bu yaklaşım, siber güvenlikte proaktif bir savunma stratejisi olarak değerlendirilebilir. Programın çekirdeğinde, içeriği kurulmadan önce tarayabilme ve zararlı içerik tespit edildiğinde sahte servis katmanına yönlendirme yeteneği bulunmaktadır. Bu yetenek, kötü amaçlı yazılımların etkilerini başlamadan önce engelleyebilir. Programın bu özelliği, Python kodlaması ile oluşturulan JSON paketi aracılığıyla gerçekleştirilir. Bu kodlama, güvenlik önlemlerinin dinamik ve esnek bir şekilde uygulanmasını sağlar. Server işletim sistemlerine yönelik kötü amaçlı yazılımlara karşı geliştirilen bu savunma mekanizması, siber güvenlik alanında önemli bir yenilik sunmaktadır. Geleneksel ve modern güvenlik tekniklerinin birleştirilmesiyle oluşturulan bu yaklaşım, tehditlerin sürekli değişimine uyum sağlayabilen ve sistem güvenliğini proaktif bir şekilde koruyabilen bir savunma mekanizmasıdır.

4.3. Önerilen Yöntemin Detaylandırılması

Geliştirilen programın genel yapısı ve işlevselliği, özellikle sahte servis katmanının nasıl oluşturulduğu, kötü amaçlı yazılımların algılanma ve yönlendirilme süreci, programın Windows Installer ile olan entegrasyonu, ve içerik taraması ile güvenlik denetiminin JSON paketi ve Python kodlaması kullanılarak nasıl gerçekleştirildiği aşağıdaki başlıklarda özetlenebilir:

- Geliştirilen program, server işletim sistemlerini kötü amaçlı yazılımlara karşı korumak amacıyla tasarlanmıştır. Programın temelinde, kötü amaçlı yazılımların gerçek sistem servislerine erişimini engelleyen ve bunları sahte bir servis katmanına yönlendiren bir mekanizma bulunmaktadır. Bu yaklaşım, zararlı yazılımların sisteme zarar vermeden izole edilmesini ve etkisiz hale getirilmesini sağlar.

- Sahte servis katmanı, gerçek servislerin davranışlarını taklit eden, ancak aslında herhangi bir gerçek işlevi olmayan bir yapıdır. Bu katmanın temel amacı, kötü amaçlı yazılımları gerçek servislerle etkileşime girmeden önce yakalamak ve sahte bir ortama yönlendirmektir. Bu yöntem, zararlı yazılımların tespiti ve izolasyonu için etkili bir stratejidir.

- Program, kötü amaçlı yazılımların algılanması için dinamik tespit mekanizmalarını kullanır. Bu süreç, Windows Installer'ın aktivitelerini izleyerek, herhangi bir yükleme veya güncelleme işlemi sırasında kötü amaçlı yazılımların varlığını kontrol eder. Tespit edilen zararlı yazılımlar, gerçek sistem servislerine erişmeden önce sahte servis katmanına yönlendirilir.

- Programın Windows Installer ile entegrasyonu, kötü amaçlı yazılımların etkisiz hale getirilmesinde kritik bir rol oynar. Bu entegrasyon sayesinde, zararlı yazılımların yükleme veya güncelleme işlemleri sırasında tespit edilmesi ve yönlendirilmesi mümkün hale gelir. Program, Windows Installer'ın işlemlerini izleyerek, herhangi bir zararlı aktiviteyi algıladığında müdahale edebilir.

- Programın güvenlik denetimi, JSON paketi ve Python kodlaması kullanılarak gerçekleştirilir. Bu teknik, programın içeriği kurulmadan önce tarayabilmesini ve zararlı içerikleri güvenli olmayan olarak tanımlayabilmesini sağlar. Tarama sonucunda, içerik güvenli olarak değerlendirilirse kurulum devam eder; güvenli değilse, içerik bir malware olarak tanımlanır ve sahte servis katmanına yönlendirilir.

Bu program kötü amaçlı yazılım **(malware)** tespiti için temel bir örnek sunar. Kod için EK 1 e bakınız

Bu program bir **Windows Forms uygulaması** olup, kötü amaçlı yazılım **(malware)** tespiti için temel bir örnek sunar. Programın genel işleyişi aşağıdaki gibidir:

- using anahtar kelimeleri gerekli kütüphaneleri (System ve System.Windows.Forms) programa dahil eder.

- Main() metodu, STAThread özelliği ile tek bir iş parçacığında çalışır.

- CheckForMalware() metodunu çağırır ve bu metod malware olup olmadığını kontrol eder.

- Sonuç isMalwareDetected değişkenine atanır.

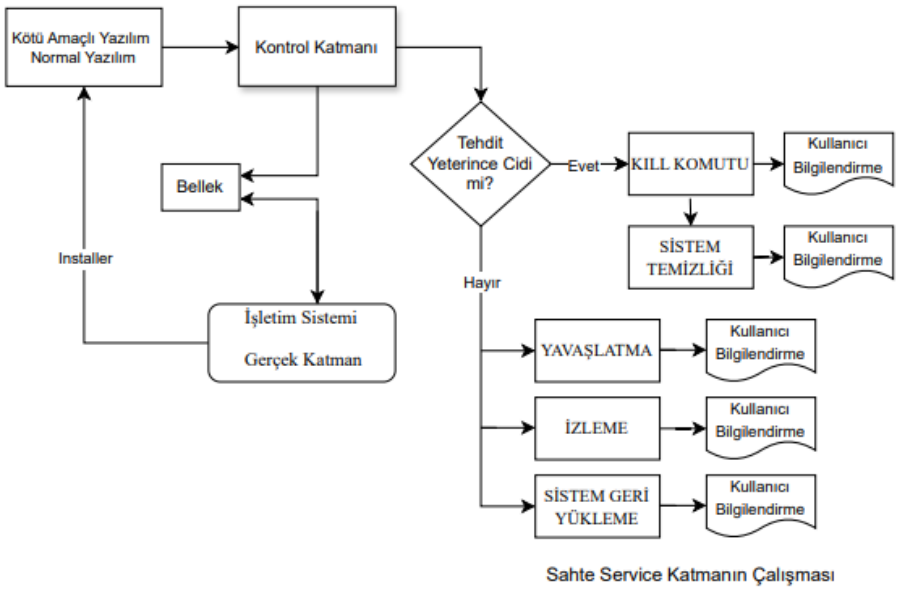
- Fonksiyon, işlem bulunursa true, bulunmazsa false değerini döndürür.

- Eğer true ise, ShowFakeScreen() metodu çağırılarak sahte bir ekran gösterilir.
- Eğer false ise, konsola "Herhangi bir malware tespit edilmedi." mesajı yazdırılır.

Bu program kötü amaçlı yazılım **(malware)** tespiti senaryosunu simüle etmektir. Kod için EK 2 ya bakınız.

Bu program, **MalwareDetectionSimulator** adlı bir Windows Forms uygulamasıdır. Programın amacı, kötü amaçlı yazılım **(malware)** tespiti senaryosunu simüle etmektir.

- MonitorProcess() adında bir thread başlatılır. Bu thread, arka planda sürekli olarak çalışan işlemleri izler.
- Process.GetProcesses() fonksiyonu kullanılarak tüm çalışan işlemler bir listeye alınır.
- Where() fonksiyonu kullanılarak bu listedeki işlemler, "notepad" adını içeren (büyük/küçük harf fark etmeksizin) işlemlerle filtrelenir.
- Any() fonksiyonu kullanılarak filtrelenmiş listede herhangi bir işlem olup olmadığı kontrol edilir.
- Eğer "notepad" adını içeren bir işlem tespit edilirse, bu şüpheli bir işlem olarak kabul edilir.
- Show Fake Warning() fonksiyonu çağrılarak kullanıcıya sahte bir uyarı mesajı ("Şüpheli bir işlem tespit edildi...") gösterilir.
- Kullanıcı uyarı mesajında "Evet" seçerse, sahte bir tarama işlemi başlatılır.
- Tarama işlemi tamamlandığında, kullanıcılara bilgisayarın güvenli olduğu bildirilir.
- Monitor Process () fonksiyonundan çıkılır ve program sonlandırılır.



Şekil 5. Sistemin Genel Akış Diyagramı.

4.4. Uygulamanın Güvenlik Etkinliği ve Proaktif Savunma Kapasitesi

Geliştirilen uygulama, server işletim sistemlerine yönelik tehditlere karşı kapsamlı bir güvenlik çözümü sunarak, zararlı yazılımların sızma girişimlerini etkin bir şekilde önlemekte ve siber güvenlik alanında proaktif bir savunma mekanizması olarak ön plana çıkmaktadır. Programın server işletim sistemlerinin güvenliğine katkısı, özellikle sahte servis katmanı aracılığıyla kötü amaçlı yazılımların gerçek sistem servislerine erişimini engelleyerek, bu tehditleri başlangıç aşamasında etkisiz hale getirme yeteneğinden kaynaklanmaktadır. Bu yaklaşım, server sistemlerinin bütünlüğünü ve veri güvenliğini koruma açısından hayati öneme sahip olup, programın sahte servis katmanı, zararlı yazılımların tespiti ve izolasyonu için kilit bir bileşen olarak işlev görür.

Program, zararlı yazılımların server işletim sistemlerine sızma girişimlerini, Windows Installer üzerinden gerçekleşen yükleme ve güncelleme işlemlerini dinamik olarak izleyerek ve potansiyel tehditleri analiz ederek proaktif bir şekilde önlemektedir. Tespit edilen zararlı yazılımlar, gerçek sistem servislerine erişim sağlayamadan önce sahte servis katmanına yönlendirilir ve izole edilerek etkisiz hale getirilir. Bu süreç, zararlı yazılımların sisteme zarar vermesini önler ve server sistemlerinin güvenliğini korur. Geliştirilen programın savunma mekanizması, siber güvenlikteki yeniliği ve önemini, sadece mevcut tehditlere karşı koruma sağlamakla kalmayıp, aynı zamanda yeni ve bilinmeyen tehditlere karşı da etkili bir savunma sunabilmektedir. Bu sayede, server işletim sistemlerinin daha dinamik ve esnek bir şekilde korunmasını sağlayarak, siber güvenlik stratejilerinde önleyici yaklaşımların artmasına katkıda bulunur. Proaktif savunma, siber saldırıları erken aşamalarda tespit edip etkisiz hale getirme imkânı sunarak, geleneksel reaktif yaklaşımlara kıyasla önemli avantajlar sağlar.

5. DENEYSEL SONUÇLAR VE TARTIŞMA

5.1. Deneysel Kurulum, Sonuçlar ve Analiz

Geliştirilen savunma mekanizmasının server işletim sistemlerine yönelik kötü amaçlı yazılımlara karşı etkinliğini değerlendirmek amacıyla yapılan deneysel çalışma, özenle tasarlanmış test senaryoları ve kapsamlı bir değerlendirme metodolojisi kullanılarak gerçekleştirilmiştir. Bu çalışma, Windows ve Linux tabanlı server sistemlerinde gerçekleştirilen testlerle, geniş bir kötü amaçlı yazılım yelpazesine karşı programın savunma kabiliyetini ölçmüş ve sonuçları detaylı bir şekilde analiz etmiştir. Test süreci, ransomware, rootkit ve trojan gibi farklı kötü amaçlı yazılım türlerini içeren, gerçek dünya senaryolarına dayalı olarak düzenlenmiştir. Bu senaryolar, programın kötü amaçlı yazılımların sisteme sızma girişimlerini nasıl engellediğini, sahte servis katmanının zararlı yazılımları nasıl etkisiz hale getirdiğini ve programın sistem kaynakları üzerindeki etkisini ölçmeyi amaçlamıştır. Testler sonucunda, programın kötü amaçlı yazılımları yüksek oranda tespit edip izole edebildiği ve özellikle ransomware ve rootkit türündeki tehditlere karşı etkili bir koruma sağladığı görülmüştür.

Elde edilen sonuçlar, programın server işletim sistemlerindeki güvenliği önemli ölçüde artırabileceğini ve zararlı yazılımların sızma girişimlerini proaktif bir şekilde önleyebileceğini göstermektedir. Sahte servis katmanının kötü amaçlı yazılımların tespiti ve izolasyonunda ne denli etkili bir rol oynadığı ve bu yaklaşımın siber güvenlikte önemli bir yenilik olduğu ortaya konmuştur. Elde edilen bulgular, mevcut akademik literatür ve benzer güvenlik çözümleriyle karşılaştırarak programın siber tehditlere karşı sunduğu savunma stratejisinin etkinliği ele alınmıştır. Programın güçlü ve zayıf yönleri açıkça tartışılmış, beklenen ve beklenmeyen sonuçlar analiz edilmiştir. Bu analiz, programın performansını etkileyen faktörleri belirlemek adına yol göstericidir.

5.2. Literatürle Karşılaştırma

Geliştirilen programın kötü amaçlı yazılımlara karşı sunduğu savunma mekanizmalarının etkinliği, literatürdeki benzer çalışmalarla karşılaştırıldığında, özellikle proaktif savunma stratejilerinin ve sahte servis katmanı kullanımının önemini vurgulamaktadır. Rehman vd. (2024) tarafından geliştirilen IoT güvenliği için proaktif savunma çerçevesi ve bu çalışmada önerilen sahte servis katmanı arasındaki temel benzerlik, her iki yaklaşımın da dinamik savunma stratejilerine ve siber aldatmacaya odaklanmasıdır. Ancak, bu çalışma, kötü amaçlı yazılımları etkisiz hale getirme sürecinde Windows Installer ve Python kodlaması kullanımı gibi özgün entegrasyonlar sunarak kendini ayırmaktadır.

Shi vd. (2023) tarafından sunulan pasif ve proaktif savunma stratejileri üzerine yapılan tartışma, bu çalışmanın kötü amaçlı yazılımların algılanması ve yönlendirilmesi sürecinde kullandığı tekniklerin, özellikle proaktif savunma yönündeki katkısını ortaya koymaktadır. Bu çalışma, sahte servis katmanı ve içerik taraması gibi

yenilikçi yöntemlerle, Shi vd.'in vurguladığı proaktif savunma stratejilerinin pratikte nasıl uygulanabileceğini göstermektedir.

Lin vd. (2013) tarafından sunulan proaktif ve reaktif savunma stratejileri, bu çalışmanın kötü amaçlı yazılımlara karşı kullandığı çift yönlü savunma yaklaşımıyla paralellik göstermektedir. Lin vd.'in vurguladığı proaktif savunma stratejileri, bu çalışmada sahte servis katmanı aracılığıyla etkin bir şekilde uygulanmaktadır. Bu çalışma, Lin vd.'in önerdiği stratejileri daha da genişleterek, kötü amaçlı yazılımların izolasyonu ve etkisiz hale getirilmesi konusunda somut bir metodoloji sunmaktadır.

Rodriguez vd. (2016) tarafından geliştirilen analiz-bilinçli kötü amaçlı yazılım tespit aracıyla bu çalışmanın sunduğu sahte servis katmanı arasındaki karşılaştırma, her iki yaklaşımın da kötü amaçlı yazılımların gelişmiş kaçınma tekniklerine karşı etkili çözümler sunduğunu göstermektedir. Ancak, bu çalışma, kötü amaçlı yazılımların tespiti ve yönlendirilmesinde daha kapsamlı bir yaklaşım sunarak, sahte servis katmanının kullanımıyla kötü amaçlı yazılımların sadece tespit edilmesini değil, aynı zamanda etkisiz hale getirilmesini sağlamaktadır.

Alsmadi & Alqudah (2021) tarafından yapılan kötü amaçlı yazılım tespit teknikleri üzerine yapılan araştırma ve bu çalışmanın kullanımı arasındaki karşılaştırma, bu çalışmanın kötü amaçlı yazılımları tespit etmek için kullandığı benzersiz tekniklerin ve stratejilerin literatürdeki mevcut yaklaşımlardan nasıl ayrıldığını ortaya koymaktadır. Bu çalışma, Alsmadi & Alqudah'ın belirttiği imza tabanlı ve heuristik yaklaşımların ötesine geçerek, kötü amaçlı yazılımların tespiti ve etkisiz hale getirilmesinde yenilikçi bir yol sunmaktadır.

Netice itibariyle, literatürdeki benzer çalışmalarla karşılaştırıldığında, bu çalışmanın kötü amaçlı yazılımlara karşı sunduğu savunma mekanizmaları, özellikle sahte servis katmanı kullanımı ve proaktif savunma stratejileri açısından önemli yenilikler sunmaktadır. Bu çalışma, siber güvenlik alanında mevcut zorluklara karşı etkili çözümler geliştirme konusunda önemli bir katkı sağlamakta ve gelecekteki araştırmalar için yeni yollar açmaktadır.

6. SONUÇ VE ÖNERİLER

Bu çalışma, kötü amaçlı yazılımlara karşı etkili bir savunma stratejisi olarak arayüz çeşitlendirmesinin önemini ve uygulanabilirliğini vurgulamaktadır. Geleneksel güvenlik önlemleri yetersiz kaldığında, arayüz çeşitlendirmesi, saldırganların hedef sistemlerin iç yapılarını tahmin etmesini zorlaştırarak, güvenlik açıklarını istismar etme çabalarını boşa çıkarmaktadır. Nesnelerin İnterneti (IoT) işletim sistemleri üzerinde yapılan analiz, çeşitlendirmenin bellek düzeni, veri yapıları ve protokoller gibi farklı alanlarda nasıl uygulanabileceğini göstermiştir. Bu çeşitlendirme yöntemleri, sistemleri daha az tahmin edilebilir ve dolayısıyla daha güvenli hale getirmektedir. Arayüz çeşitlendirmesi yaklaşımı, güvenlik duvarları ve antivirüs programları gibi geleneksel güvenlik önlemlerinin yanı sıra bir ek katman olarak düşünülebilir. Arayüz çeşitlendirmesinin, zararlı yazılımların yayılmasını engellemede ve bot ağları gibi saldırıları zorlaştırmada önemli bir rol oynadığı kanıtlanmıştır.

Ancak, arayüz çeşitlendirmesinin uygulanabilirliği ve etkinliği bağlamında bazı önemli noktalar göz önünde bulundurulmalıdır:

- *Sürekli Güncelleme ve Bakım Gereksinimi:* Arayüz çeşitlendirmesi stratejileri, etkili olabilmeleri için sürekli güncellenmeli ve bakımı yapılmalıdır. Yeni güvenlik tehditleri ortaya çıktıkça, çeşitlendirme tekniklerinin de buna uygun şekilde adapte edilmesi gerekmektedir.

- *Performans ve Uyumluluk:* Arayüz çeşitlendirmesi uygulamalarının sistemin performansı üzerinde olumsuz bir etki yapmadığından ve mevcut yazılımlarla uyumlu olduğundan emin olunmalıdır. Çeşitlendirme, sistem kaynaklarını aşırı yüklememeli veya kullanıcı deneyimini olumsuz yönde etkilememelidir.

- *Kapsamlı Güvenlik Stratejisi:* Arayüz çeşitlendirmesi, bir güvenlik stratejisinin parçası olarak ele alınmalıdır. Tek başına bir güvenlik çözümü yerine, diğer güvenlik önlemleriyle birlikte düşünülmelidir.

- *Araştırma ve Geliştirme:* Arayüz çeşitlendirmesinin potansiyelini tam olarak ortaya çıkarmak için sürekli araştırma ve geliştirme çalışmalarına ihtiyaç vardır. Yeni çeşitlendirme tekniklerinin keşfi ve mevcut yöntemlerin iyileştirilmesi, bu alanın gelişimine katkıda bulunacaktır.

Sonsöz olarak, arayüz çeşitlendirmesi, modern siber tehditlere karşı etkili bir savunma mekanizması sunmaktadır. Bu yaklaşım, siber güvenlik stratejilerinde önemli bir yer tutmalı ve gelişen teknolojiyle birlikte sürekli evrim geçirmelidir. Gelecekteki çalışmalar, çeşitlendirmenin yeni alanlarda nasıl uygulanabileceğini ve mevcut yöntemlerin nasıl optimize edilebileceğini keşfetmeye devam etmelidir.

KAYNAKLAR

Acharya, J., Chaudhary, A., Chhabria, A., & Jangale, S. (2021, May). Detecting malware, malicious URLs and virus using machine learning and signature matching. In *2021 2nd International Conference for Emerging Technology (INCET)* (pp. 1-5). IEEE.

Alsmadi, T., & Alqudah, N. (2021, July). A Survey on malware detection techniques. In *2021 International Conference on Information Technology (ICIT)* (pp. 371-376). IEEE.

Amadeo, R. (2017). *Google'snew "Android Things" OS hopestosolveawfulIoTsecurity. ArsTechnica 2016.* Available online: https://arstechnica.com/gadgets/2016/12/google-brillo-rebrands-as-android-thingsgoogles-internet-of-things-os/ (accessed on 15 December 2017)

Atasever, S., Özçelik, İ., & SAĞIROĞLU, Ş. (2019). Siber Terör ve DDoS. *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, *23*(1), 238-244.

Aycock, J. (2006). *Computer viruses and malware* (Vol. 22). Springer Science & Business Media.

Aziz, A., Jawaid, A. B., & Khan, H. R. (2021). Inclination of Computer Virus and Anti-Virus Techniques A Short Survey. *ILMA Journal of Technology & Software Management (IJTSM)*, *2*(1).

Bazrafshan, Z., Hashemi, H., Fard, S. M. H., & Hamzeh, A. (2013, May). A survey on heuristic malware detection techniques. In *The 5th Conference on Information and Knowledge Technology* (pp. 113-120). IEEE.

Caruso, R. D. (2003). Personal computer security: Part 1. Firewalls, antivirus software, and Internet security suites. *Radiographics*, 23(5), 1329-1337.

Chakravarty, A. K., Raj, A., Paul, S., & Apoorva, S. (2019). A study of signature-based and behaviour-based malware detection approaches. *Int. J. Adv. Res. Ideas Innov. Technol*., 5(3), 1509-1511.

Cohen, F. (1987). Computer viruses: theory and experiments. *Computers & security*, *6*(1), 22-35.

Cooper Jr, A. B., Hall, J., Mundhenk, D., & Rothke, B. (2023). Protect All Systems and Networks from Malicious Software. In *The Definitive Guide to PCI DSS Version 4: Documentation, Compliance, and Management* (pp. 73-80). Berkeley, CA: Apress.

Çalık Bayazıt, E. (2023). Android sistemlerde derin öğrenme tabanlı kötü amaçlı yazılım tespit sistemi.

Çelik, S., & Çeliktaş, B. (2021). Güncel Siber Güvenlik Tehditleri: Fidye Yazılımlar. *CyberPolitik Journal*, *3*(5), 105-132.

Felt, A. P., Finifter, M., Chin, E., Hanna, S., & Wagner, D. (2011, October). A survey of mobile malware in the wild. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices* (pp. 3-14).

Gnatyuk, S., Satybaldiyeva, F., Sydorenko, V., Zhyharevych, O., & Polozhentsev, A. (2023) Model of Information Technology for Efficient Data Processing in Cloud-based Malware Detection Systems of Critical Information Infrastructure.

Gobbo, J. (2023). Analysis and implementation of Software Similarity metrics.

Gordon, J. (2004, May). Lessons from virus developers: The Beagle worm history through April 24, 2004. In *SecurityFocus Guest Feature Forum*.

Govindaraju, A. (2010). Exhaustive statistical analysis for detection of metamorphic malware.

Griffin, K., Schneider, S., Hu, X., & Chiueh, T. C. (2009, September). Automatic Generation of String Signatures for Malware Detection. In *RAID* (Vol. 5758, pp. 101-120).

Gutmann, P. (2007). The commercial malware industry. In *DEFCON conference*.

Jacob, G., Debar, H., & Filiol, E. (2008). Behavioral detection of malware: from a survey towards an established taxonomy. *Journal in computer Virology*, *4*, 251-266.

Kabay, M. E. (2012). History of computer crime. *Computer security handbook*, 2-1.

Kerrisk, M. (2010). *The Linux programming interface: a Linux and UNIX system programming handbook*. No Starch Press.

Konstantinou, E., & Wolthusen, S. (2008). Metamorphic virus: Analysis and detection. *Royal Holloway University of London*, *15*, 15.

Kuriyal, V., Bordoloi, D., Singh, D. P., & Tripathi, V. (2022, November). Metamorphic and polymorphic malware detection and classification using dynamic analysis of API calls. In *AIP Conference Proceedings* (Vol. 2481, No. 1). AIP Publishing.

Li, C., Chen, X., Wang, D., Wen, S., Ahmed, M. E., Camtepe, S., & Xiang, Y. (2021). Backdoor attack on machine learning based android malware detectors. *IEEE Transactions on Dependable and Secure Computing*, *19*(5), 3357-3370.

Li, Z., Rios, A. L. G., & Trajković, L. (2020, October). Detecting internet worms, ransomware, and blackouts using recurrent neural networks. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 2165-2172). IEEE.

Liao, Q. (2008). Ransomware: a growing threat to SMEs. In *Conference Southwest Decision Science Institutes* (pp. 1-7). USA: Southwest Decision Science Institutes.

Liu, Z., Chen, C., Zhang, L. Y., & Gao, S. (2022, September). Working mechanism of Eternalblue and its application in ransomworm. In International Symposium on Cyberspace Safety and Security (pp. 178-191). Cham: Springer International Publishing.

Mäki, P., Rauti, S., Hosseinzadeh, S., Koivunen, L., & Leppänen, V. (2016, December). Interface diversification in IoT operating systems. In Proceedings of the 9th International Conference on Utility and Cloud Computing (pp. 304-309).

Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. (2022). A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges. *Future Generation Computer Systems*, *130*, 1-18.

McGraw, G., & Morrisett, G. (2000). Attacking malicious code: A report to the infosec research council. *IEEE software*, *17*(5), 33-41.

Miles, C. (2012). Early History of the Computer Virus. *Prof. Dasgupta's History of Computer Science The Center for Advanced Computer Studies University of Louisiana*, 1-8.

Nair, V. P., Jain, H., Golecha, Y. K., Gaur, M. S., & Laxmi, V. (2010, September). Medusa: Metamorphic malware dynamic analysis usingsignature from api. In *Proceedings of the 3rd International Conference on Security of Information and Networks* (pp. 263-269).

Pektaş, A. (2012). *Behavior based malicious software detection and classification* (Master's thesis, Fen Bilimleri Enstitüsü).

Quinn, K. (2020). *Cybersecurity a simple beginner's guide to cybersecurity, computer networks and protecting oneself from hacking in the form of phishing*, Malware, Ransomware, and Social Engineering, New York.

Rad, B. B., Masrom, M., & Ibrahim, S. (2012, September). Opcodes histogram for classifying metamorphic portable executables malware. In *2012 International Conference on e-Learning and e-Technologies in Education (ICEEE)* (pp. 209-213). IEEE.

Rauti, S., & Leppänen, V. (2017, October). Internal interface diversification with multiple fake interfaces. In Proceedings of the 10th International Conference on Security of Information and Networks (pp. 245-250).

Rauti, S., Koivunen, L., Mäki, P., Hosseinzadeh, S., Laurén, S., Holvitie, J., & Leppänen, V. (2018). Internal interface diversification as a security measure in sensor networks. *Journal of Sensor and Actuator Networks*, *7*(1), 12.

Rehman, Z., Gondal, I., Ge, M., Dong, H., Gregory, M. ve Tari, Z. (2024). Proaktif savunma mekanizması: Çeşitliliğe dayalı hareketli hedef savunması ve siber aldatma yoluyla IoT güvenliğini artırma. *Bilgisayarlar ve Güvenlik* , *139* , 103685.

Rodriguez, RJ, Gaston, IR ve Alonso, J. (2016). İzolasyona duyarlı kötü amaçlı yazılımların tespitine yönelik. *IEEE Latin Amerika İşlemleri* , *14* (2), 1024-1036.

Schmidt, A. D., Bye, R., Schmidt, H. G., Clausen, J., Kiraz, O., Yuksel, K. A., ... & Albayrak, S. (2009, June). Static analysis of executables for collaborative malware detection on android. In *2009 IEEE International Conference on Communications* (pp. 1-5). IEEE.

Seraj, S., Pavlidis, M., Trovati, M., & Polatidis, N. (2023). MadDroid: malicious adware detection in Android using deep learning. *Journal of Cyber Security Technology*, 1-28.

Sharma, A., & Sahay, S. K. (2014). Evolution and detection of polymorphic and metamorphic malwares: A survey. *arXiv preprint arXiv:1406.7061*.

Shelby, Z., & Bormann, C. (2011). *6LoWPAN: The wireless embedded Internet*. John Wiley & Sons.

Shelby, Z.; Hartke, K.; Bormann, C. (2014). *TheConstrained Application Protocol (CoAP*); Internet EngineeringTask Force (IETF): Fremont, CA, USA.

Shi, C., Peng, J., Zhu, S. ve Ren, X. (2023, Aralık). Pasif Savunmadan Proaktif Savunmaya: Stratejiler ve Teknolojiler. *Uluslararası Yapay Zeka Güvenliği ve Gizliliği Konferansında* (s. 190-205). Singapur: Springer Nature Singapur.

Song, S., Kim, B., & Lee, S. (2016). The effective ransomware prevention technique using process monitoring on android platform. *Mobile Information Systems*.

Talukder, S. (2020). Kötü amaçlı yazılım tespiti ve analizine yönelik araçlar ve teknikler. *arXiv ön baskı arXiv:2002.06819* .

Tran, N. P., & Lee, M. (2013, June). High performance string matching for security applications. In *International Conference on ICT for Smart Society* (pp. 1-5). IEEE.

Uitto, J., Rauti, S., & Leppänen, V. (2016, April). Practical implications and requirements of diversifying interpreted languages. In *Proceedings of the 11th Annual Cyber and Information Security Research Conference* (pp. 1-4).

Van Heerden, R., Von Solms, S., & Vorster, J. (2018, May). Major security incidents since 2014: An African perspective. In 2018 IST-Africa Week Conference (IST-Africa) (pp. Page-1). IEEE.

Vasudevan, A., & Yerraballi, R. (2006, January). Spike: engineering malware analysis tools using unobtrusive binary-instrumentation. In *Proceedings of the 29th Australasian Computer Science Conference-Volume 48* (pp. 311-320).

Wong, W. (2006). Analysis and detection of metamorphic computer viruses.

Yiğit, T., & Akyıldız, M. (2014). Sızma Testleri İçin Bir Model Ağ Üzerinde Siber Saldırı Senaryolarının Değerlendirilmesi. *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, *18*(1), 14-21.

Yuryna Connolly, L., Wall, D. S., Lang, M., & Oddson, B. (2020). An empirical study of ransomware attacks on organizations: an assessment of severity and salient factors affecting vulnerability. Journal of Cybersecurity, 6(1), tyaa023.

Zhou, Y., & Jiang, X. (2012, May). Dissecting android malware: Characterization and evolution. In *2012 IEEE symposium on security and privacy* (pp. 95-109). IEEE.

Zuo, Z. H., Zhu, Q. X., & Zhou, M. T. (2005). On the time complexity of computer viruses. *IEEE Transactions on information theory*, *51*(8), 2962-2966.

EK 1

```
using System;

using System.Diagnostics;

using System.Windows.Forms;          // Windows Forms uygulaması için

   namespace MalwareDetectionExample

   {

   class Program

   {

   [STAThread]

   static void Main()

   {

// Burada, örneğin bir dosya veya işlem varlığına göre kontrol      yapabilirsiniz.

// Bu örnek için basit bir koşul kullanıyoruz.

          bool isMalwareDetected = CheckForMalware();

               if (isMalwareDetected)

      {

     // Malware tespit edildiyse, kullanıcıyı bir sahte ekrana yönlendir

            ShowFakeScreen();

            }

            else

            {

            Console.WriteLine("Herhangi bir malware tespit edilmedi.");

            }

        }
```

```
        static bool CheckForMalware()
        {
// Bu metod, sistemde belirli bir dosya veya işlem varlığını kontrol eder
// Gerçek bir senaryoda, burada daha karmaşık tespit mekanizmaları kullanılabilir
      return Process.GetProcessesByName("notepad").Length > 0;  // Örnek koşul
      }
      static void ShowFakeScreen()
      {
                          // Kullanıcıya gösterilecek sahte ekranı başlat
     Application.EnableVisualStyles();
     Application.SetCompatibleTextRenderingDefault(false);
                          // Sahte bir form veya uyarı mesajı göster
MessageBox.Show("Sistem hatası algılandı! Lütfen bilgisayarınızı kontrol
edin.", "Sistem Uyarısı", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

EK 2

```
using System;
using System.Diagnostics;
using System.Linq;
using System.Threading;
using System.Windows.Forms;
namespace MalwareDetectionSimulator
{
    class Program
    {
        [STAThread]
        static void Main(string[] args)
        {
// Arka planda sürekli olarak belirli bir işlemi denetleyen bir Thread başlatılıyor
     Thread monitoringThread = new Thread(new ThreadStart(MonitorProcess));
            monitoringThread.Start();
     Application.Run();  // Bu, uygulamanın arka planda çalışmaya devam etmesini sağlar
```

```
        }
        static void MonitorProcess()
        {
            while (true)
            {
      var suspiciousProcesses = Process.GetProcesses().Where(p =>
      p.ProcessName.ToLower().Contains("notepad"));
           if (suspiciousProcesses.Any())
            {
  // Eğer şüpheli bir işlem bulunursa, kullanıcıyı sahte bir ekran ile uyar
           ShowFakeWarning();
                break;      // Döngüyü kır ve uygulamayı sonlandır
                }
          // Belirli bir zaman aralığında kontrol etmek için bekle
    Thread.Sleep(5000); // 5 saniyede bir kontrol et
            }
        }
        static void ShowFakeWarning()
        // GUI elemanları ana thread dışında çalıştırıldığı için güvenli bir şekilde çalıştırılması gerekir.
 if (MessageBox.Show("Şüpheli bir işlem tespit edildi. Bu bir virüs olabilir.
Şimdi tarama başlatılsın mı?", "Güvenlik Uyarısı", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning) == DialogResult.Yes)
  {
       // Kullanıcı 'Evet' dediyse, sahte bir tarama işlemi başlatılabilir.
      // Bu örnekte, basitçe bir mesaj gösteriliyor.
MessageBox.Show("Sahte tarama başlatıldı. Bilgisayarınız güvende.", "Tarama
Tamamlandı", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }
    }
}
```

TEŞEKKÜR ve BEYANLAR

Yazarlar çalışmaya eşit oranda katkı sağlamıştır. Bu çalışmada herhangi bir potansiyel çıkar çatışması bulunmamaktadır. Yapılan çalışmada araştırma ve yayın etiğine uyulmuştur.

Not: *Bu makale, İstanbul Ticaret Üniversitesi Fen Bilimleri Enstitüsü, Siber Güvenlik Tezli Yüksek Lisans Programı'nda, Prof. Dr. Rıfat YAZICI danışmanlığında, Nasrullah FROTAN tarafından yürütülecek olan, "Kötü amaçlı Yazılımlara Karşı Bir Yöntem Olarak Dahili Arayüz Çeşitlendirmesi" başlıklı yüksek lisans tezinin ön çalışmalarından yararlanılarak hazırlanmıştır.*