Journal of the Faculty of Engineering and Architecture of Gazi University 37:1 (2022) 305-315

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

The limitations of signature-based and dynamic analysis methods in detecting malwares: A case study

Derviş Aygör*, Ertuğrul Aktan
Banking Regulation and Supervision Agency, Data and System Management Department, 34394, Istanbul, Turkey

Highlights:

- Evolution of malware
- Malware detection
- Anti-detection methods

Keywords:

- Malware
- Antivirus
- Signature-based detection
- Antivirus evasion techniques

Article Info:
Research Article
Received: 31.12.2019
Accepted: 04.06.2021

DOI:
10.17341/gazimmfd.668290

Correspondence:
Author: Derviş Aygör
e-mail: daygor@bddk.org.tr
phone: +90 212 214 5000

Graphical/Tabular Abstract

In this study, common methods used by antivirus engines for malware detection are examined and the limitations of existing antivirus solutions are explored through a case study. In accordance with this purpose, code injection, anti-dynamic modification and encryption operations have been performed on an innocent program respectively. As a result, the original flow of the program is manipulated and a remote shell is executed on the victim machine. This result shows that antivirus solutions based on signature and dynamic analysis are fragile in regard to manipulations with code injection, anti-dynamic modification and encryption.

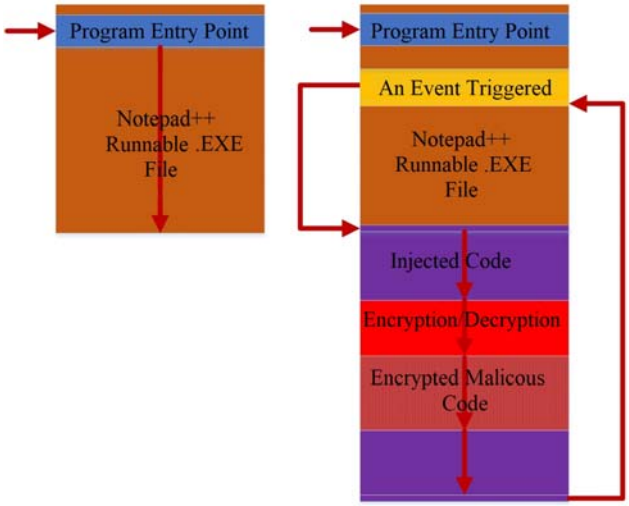


Figure A. The developed experimental method for manipulation of innocent program

Purpose: The aim of the study is to show the limitations of known antiviruses in detecting malwares and shed light on malware & anti-malware ecosystem for researchers in cyber security field.

Theory and Methods:
By using developed experimental method given in Fig. A, an innocent program can be easily changed into a malicious software which evades many well-known antiviruses. As an innocent program, Notepad++; to generate malicious code, Metasploit; for anti-dynamic modification, Ollydbg debugger; for malware analysis, Virustotal platform have been used.

Results:
Initially 30/67 different antivirus solutions detected the malicious code injected into an innocent program. After the manipulations with anti-dynamic modification and encryption, this result was dropped to 9/67 and 4/66 respectively.

Conclusion:
The likelihood of evading antivirus solutions based on signature and dynamic analysis reveals the need for integrated use of different advanced analysis and security approaches such as sandbox, endpoint detection and response, security information and event management, as well as existing antivirus solutions.

Journal of the Faculty of Engineering and Architecture of Gazi University 37:1 (2022) 305-315

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Kötücül yazılımların tespitinde imza temelli ve dinamik analiz yöntemlerinin zayıflıkları: Örnek olay çalışması

Derviş Aygör*, Ertuğrul Aktan
Bankacılık Düzenleme ve Denetleme Kurumu, Veri ve Sistem Yönetimi Dairesi, 34394, İstanbul, Türkiye

ÖNEÇIKANLAR

- Kötücül yazılımların gelişimi
- Kötücül yazılım tespiti
- Anti-tespit yöntemleri

Makale Bilgileri
Araştırma Makalesi
Geliş: 31.12.2019
Kabul: 04.06.2021

DOI:
10.17341/gazimmfd.668290

Anahtar Kelimeler:
Kötücül yazılım,
antivirüs,
imza temelli tespit,
antivirüs atlatma teknikleri

ÖZ

Hedef sistemleri manipüle etmek için siber saldırganlar tarafından en çok kullanılan araçlardan biri kötücül yazılımlardır. Mevcut kötücül yazılımlara her geçen gün yenilerinin eklenmesiyle kötücül yazılımların hızlı ve etkin bir şekilde tespit edilmesi zorlaşmaktadır. Kötücül yazılımları tespit etmek ve zararlı etkilerini gidermek amacıyla genel olarak antivirüs yazılımları kullanılmaktadır. Bu çalışmada, birçok antivirüs yazılımı tarafından kullanılan imza ve dinamik analiz temelli tespit yöntemlerinin başarımı incelenmiştir. Sırasıyla kod enjeksiyonu, anti-dinamik modifikasyon ve şifreleme adımlarından oluşan deneysel metotla elde edilmiş bir kötücül yazılımın antivirüs yazılımlarını büyük ölçüde atlatabildiği gösterilmiştir. Kötücül kod üretmek için Metasploit, anti-dinamik modifikasyon için Ollydbg ve antivirüs yazılımlarının başarımını karşılaştırmak için Virustotal platformu kullanılmıştır. Virustotal sonuçlarına bakıldığında; sadece kod enjeksiyonuyla oluşturulmuş kötücül bir yazılımın başlangıçta 30/67 adet antivirüs yazılımı tarafından tespit edilebildiği, fakat söz konusu kötücül koda anti-dinamik modifikasyon ve şifreleme uygulanması hâlinde bu sayının, sırasıyla 9/67 ve 4/66 adede düştüğü gözlenmiştir. Sonuç olarak, ticari birçok antivirüs yazılımının, farklı anti-tespit yöntemleri ile atlatılabildiği anlaşılmıştır. Bu sonuç, mevcut antivirüs çözümlerinin yanı sıra sandbox, uç nokta tehdit algılama ve yanıt, güvenlik bilgi ve olay yönetimi gibi farklı ileri analiz ve güvenlik yaklaşımlarının bütünleşik bir biçimde kullanılmasının gerekliliğini ortaya koymaktadır.

The limitations of signature-based and dynamic analysis methods in detecting malwares: A case study

HIGHLIGHTS

- Evolution of malware
- Malware detection
- Anti-detection methods

Article Info
Research Article
Received: 31.12.2019
Accepted: 04.06.2021

DOI:
10.17341/gazimmfd.668290

Keywords:
Malware,
antivirus,
signature-based detection,
antivirus evasion techniques

ABSTRACT

To manipulate target systems, malware is one of the most frequently used tool by cyber attackers. As new malware is added to existing malware every day, it becomes difficult to detect malware quickly and effectively. Antivirus software is generally used to detect malware and remove its harmful effects. In this study, the success of signature and dynamic analysis based detection methods used by many antivirus software, was examined. It has been shown that a malware created by experimental method consisting of code injection, anti-dynamic modification and encryption steps respectively, evades many antivirus software to a large extent. To generate malicious code, Metasploit; for anti-dynamic modification, Ollydbg; to compare the performance of antivirus software, Virustotal platform was used. Looking at the Virustotal results; it has been observed that a malware created by only code injection was initially detected by 30/67 antivirus software, but if anti-dynamic modification and encryption were applied to the malicious code, this value decreased to 9/67 and 4/66, respectively. As a result, it has been understood that many commercial antivirus software can be circumvented by different anti-detection methods. This result reveals the need for integrated use of different advanced analysis and security approaches such as sandbox, endpoint detection and response, security information and event management, as well as existing antivirus solutions.

*Sorumlu Yazar/Yazarlar / Corresponding Author/Authors : *daygor@bddk.org.tr, eaktan@bddk.org.tr / Tel: +90 212 214 5000

1.GİRİŞ (INTRODUCTION)

Programlanabilirlik, elektronik cihazları daha akıllı ve kullanışlı hâle getirmekle birlikte pek çok kötü amaç için de saldırganlara oldukça geniş bir saldırı yüzeyi sunmaktadır. Günümüzde, elektronik cihazlar farklı iletişim arayüzlerini kullanarak birbirleriyle kolaylıkla haberleşebilmekte ve bu durum, siber saldırganların yeni tip saldırıları çok hızlı bir şekilde uygulamaya koyabilmelerine ve daha geniş kitlelere yayabilmelerine sebep olmaktadır. Bir siber saldırıda asıl amaç hedef sistemin kötücül niyetlerle manipüle edilmesi olduğundan hedef sistemin ulaşılabilir olması bu noktada oldukça önemlidir. Bu bağlamda yakın gelecekte, tüm akıllı cihazların birbirleriyle haberleşebilmesini öngören IoT (Internet of Things) teknolojisi üzerinden siber saldırıların hem yoğunluğunda hem de çeşitliliğinde büyük artışlar beklenmektedir [1, 2]. Günümüzde hedef sistemleri manipüle etmek için siber saldırganlar tarafından yoğun biçimde tercih edilen yöntemlerden biri kötücül yazılımların kullanımıdır. Kötücül yazılımlar, süregelen evrimi nedeniyle her geçen gün birçok elektronik cihazı ve bilgisayarı etkileyebilmektedir. Bu çalışmada kötücül yazılımların doğasına ilişkin bir araştırma sunulacak ve kötücül yazılımlarla mücadele noktasında önerilen imza ve dinamik analiz temelli tespit yöntemlerinin başarımı incelenecektir. Çalışmanın ikinci bölümünde, kötücül yazılım tipleri ve imza temelli güvenlik çözümlerine değinilecektir. Üçüncü bölümde kötücül bir yazılımın gelişiminin deneysel metodu sunularak kötücül yazılımların mevcut tespit yöntemlerini nasıl aşabildiği gösterilecektir. Dördüncü bölümde ise kötücül yazılımların tespitinde kullanılan yöntemlere ilişkin öneriler sunulacak ve gelecekte yapılacak araştırmalarda dikkat edilmesi gereken hususlara değinilecektir.

2. KÖTÜCÜL YAZILIM TİPLERİ VE TESPİT YÖNTEMLERİ
(TYPES OF MALWARES AND DETECTION METHODS FOR THEM)

2.1. Kötücül Yazılımlar (Malwares)

Kötücül yazılımlar, elektronik ortamda tutulan bilginin gizliliğini, bütünlüğünü veya erişilebilirliğini tehlikeye atan ve hedef sistemlere genellikle gizlice yerleştirilen programlardır [3]. Kötücül yazılımlarla ilgili ilk teorik yaklaşım, Von Neumann tarafından geliştirilmiştir [4]. Von Neumann, yürüttüğü çalışmada otomata teorisini (automata theory) kullanarak herhangi bir programın kontrolünün farklı bir program vasıtasıyla ele geçirilebilmesinin mümkün olduğunu ilk kez teorik olarak göstermiştir. Her ne kadar ilk teorik yaklaşım Von Neumann'a ait olsa da ilk kötücül yazılım Bob Thomas tarafından 1971 yılında assembly dili kullanılarak geliştirilmiştir [5]. Creeper adı ile bilinen söz konusu bu kötücül yazılım, esasında zarar verme amacı gütmeyen deneysel olarak geliştirilmiş bir programdır. Bu programın kötücül olarak nitelendirilebilecek tek faaliyeti, kendisini sistemler arasında insan müdahalesine ihtiyaç duymadan kopyalayabiliyor olmasıdır. Bununla birlikte program, bulaştığı sistemlerde kullanıcıya kopyalama işleminin başarılı olduğunu gösterir metin temelli zararsız bir mesaj üretmektedir. Creeper, geliştirilirken her ne kadar kötücül bir amaç taşımamış olsa da daha sonraları solucan (worm) olarak adlandırılacak kötücül yazılım familyasının ilk örneğini teşkil etmiştir. Creeper'in davranışını taklit eden kötücül yazılımların günümüze kadar farklı birçok sisteme zarar verdiği bilinmektedir [6]. Kötücül yazılımların sahip olduğu özelliklere göre farklı familyalara ayrılması, diğer bir ifadeyle kötücül yazılımlarla ilgili sınıflandırmalar yapılması kötücül yazılımlarla etkin mücadele edilebilmesinde büyük önem arz etmektedir. Bu bağlamda kötücül yazılımların sahip olduğu özellikleri ortaya koymaya yönelik yürütülmüş birçok çalışmaya alanyazında yer verilmiş ve farklı yöntemler geliştirilmiştir [7]. Hedef sistemde gerçekleştirilmek istenen asıl amaçları ve yayılma biçimleri dikkate alındığında kötücül yazılımların üç farklı kategori altında toplanabileceği görülmektedir [8]. Kötücül yazılımlar ile ilgili bu kategorilerin içeriği ve sınırları genel olarak şu şekilde belirlenebilir:

- Virüs: Bulaştığı program veya dosyalar aracılığıyla kendisini bir ana bilgisayarda çoğaltabilen kötücül yazılım türüdür. Virüs tipi kötücül yazılımlar, genellikle sadece ilk bulaşma esnasında insan tarafından tetiklenmeye ihtiyaç duyar.
- Solucan: Ağ üzerinde yayılım göstererek kendisini çoğaltabilen kötücül yazılım türüdür. Solucanlar, bulaşma ve yayılımın tetiklenmesinde insan müdahalesine en az ihtiyaç duyan kötücül yazılımlardandır.
- Truva atı: Asıl kötücül amacını maskeleyerek kendisini faydalı bir program gibi gösteren kötücül yazılım türüdür. Truva atı yerleştirilen bir sistemde, yerel güvenlik önlemleri devre dışı bırakılarak sistem uzaktan erişime açılabilmekte ve bu sayede üçüncü kişiler tarafından sistemin kontrolü ele geçirilebilmektedir. Bu durum, truva atının farklı bir kötücül yazılım davranışı olan arka kapı (back door) özelliği taşıdığı anlamına gelmektedir. Truva atları, genellikle kendisini kopyalayarak yayılım gösterme özelliği taşımaz.

Yukarıda verilmiş olan sınıflandırma dışında kötücül yazılımların farklı özelliklerinin temel alındığı sınıflandırmalar da yapmak mümkündür [5, 9, 10]. Öte yandan tespit edilen kötücül bir yazılım, bilinen sınıflardan hiçbirine uymuyor veya farklı sınıflara ait özellikleri aynı anda taşıyor olabilir. Nitekim bulaştığı ortama özgü farklı davranışlar sergileyebilen yeni nesil kötücül yazılımların var olduğu da bilinmektedir [11]. Farklı sınıflara ait özellikleri aynı anda taşıyan kötücül yazılımlar melez kötücül yazılım olarak adlandırılmakta ve günümüzde kötücül yazılımların birçoğu bu sınıf altında değerlendirilmektedir [12].

2.2. Kötücül Yazılım Tespit Yöntemleri (Malware Detection Methods)

İlk kötücül yazılımın Bob Thomas tarafından yazılmasının ardından söz konusu yazılımı tespit etmek ve bulaştığı sistemlerden kaldırmak için yine benzer şekilde sistemler

arasında kendisini kopyalayabilen Reaper adında bir yazılım geliştirilmiştir [5]. Her ne kadar bir kötücül yazılım davranış özelliği gösterse de Reaper, antivirüs yazılımlarının atası olarak kabul edilmektedir. Kötücül yazılımlarla mücadele noktasında en başından beri önemli görevler üstlenen antivirüs yazılımları günümüzde de yaygın biçimde kullanılmakta ve uç nokta (end point) güvenliğinin vazgeçilmez bir ögesi olma özelliğini devam ettirmektedir [13, 14].

Günümüzde farklı yöntemler temelinde inşa edilen çeşitli antivirüs yazılımları bulunmakla birlikte mevcut antivirüs yazılımlarının en temel bileşenlerinden biri sahip olduğu imza veri tabanıdır [15]. İmza temelli güvenlik çözümlerinde her bir kötücül yazılıma karşılık gelen bir imza, veri tabanında tutulmakta ve tarama esnasında veri tabanında tutulan bu imzalar incelenen dosya ve programlar ile karşılaştırılmaktadır [16]. Bu açıdan bakıldığında imza temelli kötücül yazılım tespitinin esasında bir kayıt eşleştirme problemi (string matching problem) olduğu görülmektedir. Etkin bir eşleştirme için herhangi bir imzada asgari bulunması gereken özellikler şunlardır [17]:

- İmza, kötücül yazılımın zararsız veri barındıran bölümünden üretilmemiş olmalıdır. Aksi takdirde, zararsız veri bölümü kolaylıkla değiştirilebileceğinden imza etkisiz bırakılabilir.
- Depolama ve veri işleme için gerekli olan alanı ve zamanı en aza indirmek için imzanın mümkün olduğunca küçük boyutlu olması gerekmektedir.
- İmza, kötücül yazılımın içerisinde yer alan ve kötücül amaca hizmet eden ilgili kodu benzersiz şekilde temsil etmelidir.

İmza temelli kötücül yazılım tespiti birçok farklı güvenlik çözümünde kullanılmakla birlikte bu yöntemin etkinliği ve verimliliği konusundaki tartışma hâlen devam etmektedir. Alanyazında imza temelli güvenlik önlemlerine birçok eleştiri yöneltilmiş olmasına rağmen, bu yöntemden vazgeçilemediği ve günümüzde de birçok güvenlik çözümünde bu yöntemin önemli bir bileşen olmaya devam ettiği görülmektedir. İmza temelli yöntemlerden vazgeçilememesinin altında yatan başlıca sebepler; (i) imza temelli yöntemlerin basit, hızlı ve her zaman kesin sonuçlar üretmesi, (ii) sezgisel veya davranışsal yöntemlerin hatalı sonuçlar üretme ihtimali, (iii) imza temelli yöntemlerin diğer yöntemlere kıyasla daha az kaynağa ihtiyaç duymasıdır [17]. Bununla birlikte, antivirüs yazılımlarının çalıştırılacağı uç nokta cihazlarının kaynak kapasitesinin genellikle düşük olması, sezgisel veya davranışsal temelli dinamik analiz yöntemlerinin bu cihazlar üzerinde kullanılmasını imkânsız kılabilmektedir [18-20]. Bu bağlamda daha az kaynakla hızlı ve güvenilir sonuçlar üretmesi, imza temelli güvenlik önlemlerini diğer yöntemlerden ayrıştıran en önemli avantajıdır.

Tarihsel açıdan bakıldığında kötücül yazılımlarla mücadelede imza temelli yaklaşım başlangıçta her ne kadar başarılı olmuşsa da kötücül yazılımların zamanla geçirmiş olduğu değişim neticesinde elde edilen bu başarı düzeyinde ciddi bir düşüş yaşanmıştır. Kötücül yazılımların değişimi sürecinde antivirüs yazılımlarının katkısı şüphesiz çok büyüktür. Kötücül yazılımlarının amacı hedef sistemleri manipüle etmek olduğundan bu amacın önündeki en büyük engellerden biri konumunda olan antivirüs yazılımlarının atlatılması için birçok yöntem geliştirilmiş ve bunun neticesinde kötücül yazılımlar günümüze çok büyük bir evrim geçirerek ulaşmıştır. İlk yıllarda masum amaçlarla geliştirilen kötücül yazılımlar, zamanla daha sofistike yöntemlerin kullanılmasıyla günümüzde bilgisayar kullanıcılarını çok farklı şekillerde tehdit eder hâle gelmiştir [21]. Kötücül yazılımlarda güvenlik çözümlerini atlatmaya dönük iki temel yöntem bulunmaktadır: Şifreleme (encryption) ve karıştırma (obfuscation). Daha erken zamanlarda kullanılmaya başlanan ilk yöntemde, kötücül yazılımdaki zararlı kod şifrelenerek gizlenmeye çalışılırken; ikinci yöntemde, yürütülecek atomik işlemlerin sıraları değiştirilmek suretiyle kötücül faaliyet gözlerden saklanmaya çalışılmaktadır. Söz konusu bu iki yöntemi farklı şekillerde kullanan kötücül yazılımların başlangıçtan günümüze kadar göstermiş olduğu dört temel form bulunmaktadır. Söz konusu formlar, kronolojik sıralamaları dikkate alınarak aşağıda sunulmuştur:

- Şifreleme: Kötücül yazılımın imzaya konu zararlı kod bloğunun şifrelenerek kötücül yazılımın tespit edilmesinin zorlaştırılmaya çalışıldığı yöntemdir. Kötücül yazılımın çalışması esnasında şifresinin çözülmesinden sorumlu ilgili kod bloğu ise bu yöntemin en zayıf halkasını oluşturmaktadır. Nitekim bu kod bloğu, antivirüs yazılımları tarafından kötücül yazılımın imzası geliştirilirken kullanılmaktadır.
- Oligomorfizm: Şifreleme yöntemine göre daha gelişmiş bir yöntem olan ve yarı-polimorfizm olarak da adlandırılabilecek bu yöntemde, şifreleme yönteminin en zayıf halkası olan şifre çözmeden sorumlu kod bloğu belirli bir kümeden seçilmekte ve böylelikle kötücül yazılımın kolaylıkla tespit edilmesinin önüne geçilmeye çalışılmaktadır. Oligomorfik yöntemin en zayıf halkası ise kötücül yazılımın şifre çözmeden sorumlu kod bloğunun sınırlı bir kümeden seçiliyor olmasıdır. Antivirüs yazılımları, eleman sayısı sınırlı bu kümedeki tüm alternatifleri inceleyerek kötücül yazılımın tüm varyasyonları için imzaları oluşturabilmektedir. Bu şekilde kötücül yazılımın farklı tüm varyasyonlarının tespit edilebilmesi mümkün hâle gelmektedir.
- Polimorfizm: Bu yöntemde, oligomorfik kötücül yazılımların en zayıf halkasını oluşturan şifre çözmeden sorumlu kod bloğunun sınırlı bir kümeden seçilmesinin önüne geçilmeye çalışılmaktadır. Bu amaç doğrultusunda polimorfik kötücül yazılımın içerisinde yer alan mutasyon motoru adındaki kod bloğu, yazılımın her yeni bulaşmasında otomatik olarak çalışmakta ve böylelikle şifre çözmeden sorumlu kod bloğu sınırsız hâle getirilmektedir. Polimorfik kötücül yazılımların en zayıf halkası şifrelenmiş kod bloğudur. Bu noktada pek çok antivirüs yazılımı, şifrelemenin varlığını kötücül bir eylemin işareti olarak kabul etmektedir. Polimorfik

kötücül yazılımların tespiti için imza temelli yaklaşım sınırlı seviyede çözüm sunabilmekte, bu tür kötücül yazılımlar için daha gelişmiş tespit yöntemlerinin (sezgisel, davranışsal vs.) kullanılması gerekmektedir.

- Metamorfizm: Şifreleme prensibi temelinde inşa edilen diğer üç yöntemden farklı olarak bu tip kötücül yazılımlarda, şifreleme ve dolayısıyla şifre çözmeden sorumlu bir kod bloğu bulunmaz. Kod karıştırma yöntemini kullanan bu kötücül yazılım çeşidinde, polimorfik kötücül yazılımlarda görülen mutasyon motoruna benzer bir kod bloğu bulunmaktadır. Söz konusu bu kod bloğu tarafından her yeni bulaşmada kötücül yazılımın ana işlevi değiştirilmeden tüm görüntüsü baştan aşağı değiştirilmektedir. Bu açıdan bakıldığında metamorfik kötücül yazılımlar, tam polimorfik kötücül yazılım olarak da adlandırılmaktadır [22].

Kötücül yazılımlarda gözlemlenen bu kronolojik değişim, var olan güvenlik çözümlerini yeni arayışlara itmiş ve bunun neticesinde kötücül yazılımların tespitinde kullanılan araçların ve yöntemlerin günümüze kadar hem sayısı hem de karmaşıklığı artmıştır [23]. Nitekim kötücül yazılımlar ile güvenlik metodolojileri arasında süregelen bir savaşın varlığı ve karşılıklı silahlanma yarışının uzun zamandır devam ettiği bilinmektedir [24]. Günden güne hızla artan ve daha fazla cihaza bulaşmayı başaran kötücül yazılımlar, günümüzde bilgi güvenliğinin yetkin düzeyde sağlanmasının önündeki en büyük engellerden biridir [25]. Her yeni bulaşma öncesinde yapısını ve kodunu değiştirebilmesi, polimorfik ve metamorfik kötücül yazılımların otomatik yöntemlerle tespit edilebilmesini zorlaştırmaktadır [26]. İkinci nesil kötücül yazılımlara örnek olarak verilebilecek bu tip yeni kötücül yazılımlar, farklı karıştırma teknikleri kullanarak iyi bilinen temel güvenlik yöntemlerine karşı güçlü bir bağışıklık kazanmaktadır [27].

Geleneksel imza temelli güvenlik çözümleri, özellikle ikinci nesil kötücül yazılımları tespit etme noktasında yetersiz kalmakta ve imzası önceden bilinemeyen sıfır gün saldırılarını (zero day attack) önleyememektedir [28]. Bu nedenle günümüzde kötücül yazılımları daha yetkin düzeyde tespit etmek ve engellemek adına, farklı metot ve süreçlerin bir arada kullanılmasını ve elde edilen sonuçların birlikte değerlendirilmesini gerektiren bir paradigma değişimine ihtiyaç duyulmaktadır [29]. İmzası henüz bilinmeyen sıfır gün saldırılarının imza temelli yöntemler ile tespiti ve dolayısıyla engellenmesi mümkün değildir. Sıfır gün saldırılarının tespit edilmesi ve engellenmesi adına faydalanılan teknolojilerin genellikle dinamik analiz temelli inşa edilmesi ise bu teknolojileri hatalı sonuç üretmeye açık hâle getirmektedir. İmzası bulunmayan veya bilinmeyen kötücül yazılımların tespit edilmesi ve engellenmesi için farklı birçok yöntem önerilmiş olmasına rağmen söz konusu bu yöntemler, benimsenen analiz prensibine göre temelde iki farklı sınıfta ele alınabilir: Durağan (static) ve dinamik (dynamic) [29, 30]. Durağan analiz yönteminde kötücül yazılımın çalıştırılmadan önceki görüntüsü analiz edilirken, dinamik analiz yönteminde ise kötücül yazılımın çalıştırılması esnasında göstermiş olduğu davranış ve geride bıraktığı izler analiz edilmektedir [31]. Durağan ve dinamik analiz yöntemleri birbirlerine kıyasla farklı avantaj ve dezavantajlara sahiptir [32] (Tablo 1).

3. DENEYSEL METOT (EXPERIMENTAL METHOD)

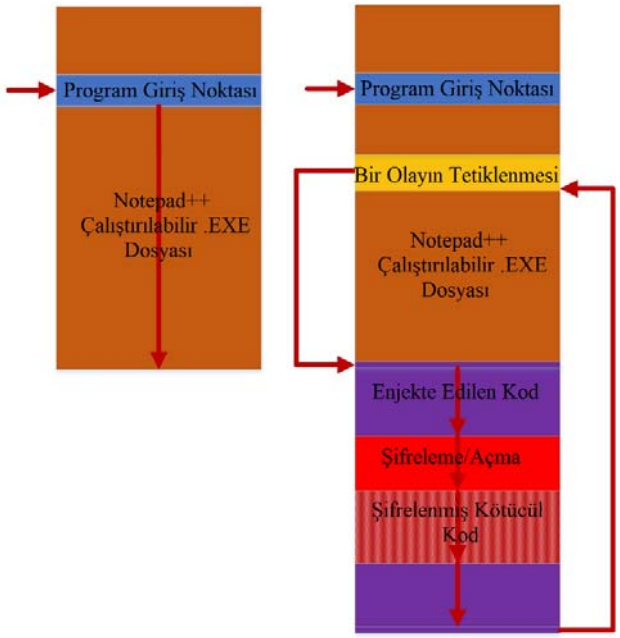
Kötücül yazılımların çok hızlı artış gösterdiği günümüzde, her gün yaklaşık 1 milyon yeni kötücül yazılım örneği ile karşılaşılmaktadır [33]. Gözlemlenen bu hızlı artışın altında yatan en önemli nedenler arasında, kötücül yazılımların otomatik araçlarla uzmanlık gerektirmeden kolaylıkla üretilebiliyor olması yer almaktadır [34, 35]. Ayrıca bilinen kötücül yazılım kodlarının yeni geliştirilecek kötücül yazılımlarda kullanılması veya var olan kötücül yazılımların önemli kod bloklarının yeni geliştirilen kötücül yazılımlara doğrudan kopyalanması, kötücül yazılımların geliştirilmesi için ihtiyaç duyulan zamanın oldukça kısalmasına neden

Tablo 1. Durağan ve dinamik analiz yöntemlerinin karşılaştırılması (Comparison of static and dynamic analysis methods)

Faktörler	Durağan Analiz	Dinamik Analiz
Zaman	Analiz süresi dinamik analize göre kısadır.	Analiz süresi durağan analize göre daha uzundur.
İncelemeye konu olan kısım	Uygulamanın derlenmiş makine kodunun incelenmesi yeterlidir.	Bellek görüntüleri ve çalışma zamanına ait diğer verilerin de incelenmesi gerekmektedir.
Kaynak kullanımı	Düşük bellek ve işlemci zamanı yeterli olabilmektedir.	Daha fazla bellek ve işlemci zamanına ihtiyaç duyulur.
Doğruluk	Analiz süreci, dinamik analize göre daha düşük doğruluk, fakat daha yüksek kesinlikte işletilir (düşük false-positive, yüksek false-negative)	Analiz süreci, durağan analize göre daha yüksek doğruluk, fakat daha düşük kesinlikte işletilir (düşük false-negative, yüksek false-positive)
Genel avantajlar	Kodda yer alan zayıflıklar, geliştirilmenin erken safhalarında daha hızlı ve düşük maliyetle tespit edilebilir.	Zayıflıklar, kaynak koda ihtiyaç duymadan uygulamanın çalıştırılması esnasında tespit edilebilir.
Genel dezavantajlar	Çeşitli karıştırma, şifreleme ve sıkıştırma teknikleriyle kaynak kodun gizlenmesi durumlarında analiz süreci uzayabilir.	Eş zamanlı olarak sadece tek bir kötücül yazılımın analizi yapılabilir.

olmaktadır [36]. Kötücül yazılımların kaynak kodlarının birtakım yöntemlerle paylaşılması ve kod bloklarındaki benzerlikler, kötücül yazılım geliştirmenin gün geçtikçe yeni bir endüstri dalı olmaya başladığı izlenimini uyandırmaktadır [37]. Tüm bu kolaylaştırıcı etmenler nedeniyle kötücül yazılım geliştirmek için gereken zaman ve iş gücü her geçen gün daha da azalmaktadır.

Bu çalışmada; durağan tespit yöntemlerinden olan imza temelli yaklaşım ile kolaylıkla tespit edilebilen bir kötücül yazılımın, Şekil 1'de görülen deneysel metot takip edilerek farklı birçok antivirüs yazılımını nasıl atlatabildiği gösterilmiştir. Bu amaçla masum bir program üzerinde sırasıyla kod enjeksiyonu, anti-dinamik modifikasyon ve şifreleme işlemleri gerçekleştirilmiş ve böylelikle programın normal akışı manipüle edilerek kurban makine üzerinde bir komut yorumlayıcısı çalıştırılmıştır. Diğer bir ifadeyle masum bir program, kötücül kod ile truva atı olarak nitelendirilebilecek kötücül bir yazılıma dönüştürülmüştür.



Şekil 1. Masum bir programın manipülasyonu
(The manipulation of an innocent program)

Deneysel süreçte, masum program olarak açık kaynak kodlu Notepad++ metin editörü [38] kullanılmışken, kötücül kod bloğu ise Metasploit [39] adlı yine açık kaynak kodlu zafiyet analiz ve test çatısı kullanılarak üretilmiştir. Elde edilen kötücül yazılımın bilişim pazarında yaygın olarak kullanılan antivirüs yazılımlarının kaçı tarafından başarıyla tespit edilebildiği sonucuna ise herkesin kullanımına açık kötücül yazılım analiz platformu Virustotal aracılığıyla ulaşılmıştır [40, 41]. Birden fazla antivirüs yazılımının eş zamanlı aynı dosya üzerinde tarama yapabilmesine imkân veren bu platform, tarama neticesinde elden edilen bulguları bir rapor hâlinde kullanıcıya sunma yeteneğine sahiptir.

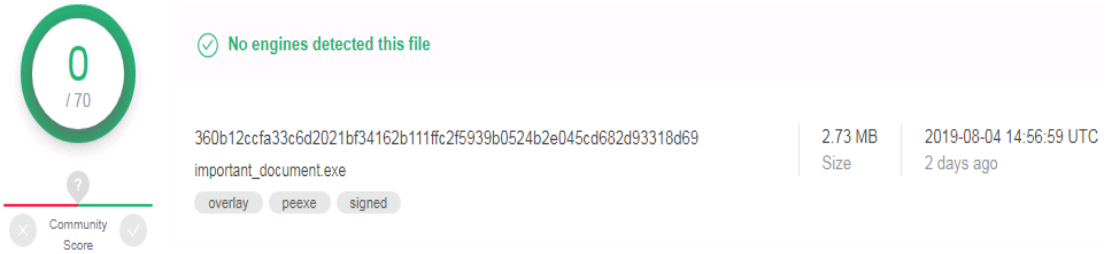
4. DENEYSEL SONUÇLAR VE TARTIŞMALAR (EXPERIMENTAL RESULTS AND DISCUSSIONS)

Üzerinde herhangi bir değişiklik yapılmamış Notepad++ adlı metin editörü programının Virustotal aracılığıyla taratılması neticesinde elde edilen rapor Şekil 2'de sunulmaktadır. Söz konusu rapordan da açıkça görüleceği üzere başlangıçta masum olan metin editörü programında beklendiği üzere herhangi bir kötücül faaliyet tespit edilmemiştir.

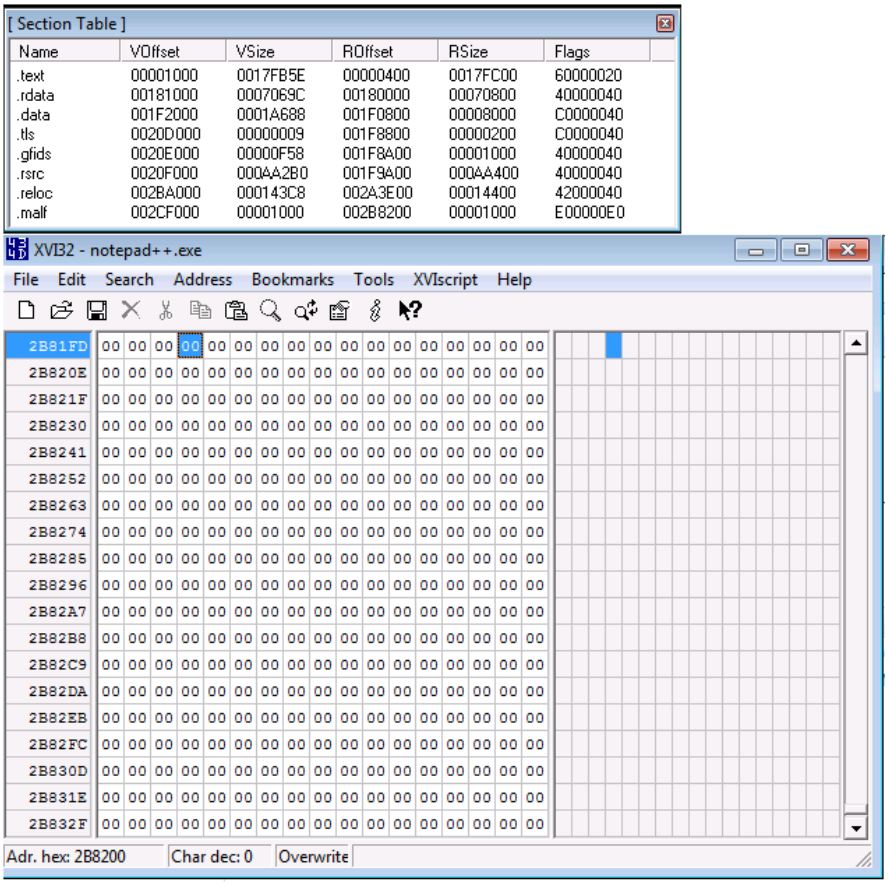
4.1. Kod Enjeksiyonu ile Manipülasyon
(Manipulation with Code Injection)

Metasploit aracı kullanılarak üretilen kötücül kodu enjekte etmek için öncelikli olarak Notepad++ metin editörü programı üzerinde ".malf" adında yeni bir bölüm oluşturulmuş ve söz konusu bölüm, varsayılan olarak zararsız özellikte "00" veri setiyle doldurulmuştur (Şekil 3). Masum program üzerinde kod enjeksiyonu için gerekli bölüm oluşturulduktan sonra elde edilen yeni çalıştırılabilir dosyanın Virustotal platformuna sokulmasıyla elde edilen rapor Şekil 4'te görülmektedir. Bu raporda görüldüğü üzere, kötücül kod henüz enjekte edilmemiş olmasına rağmen taratılan dosya, 5/66 adet antivirüs yazılımı tarafından kötücül yazılım sınıfında değerlendirilmiştir. Bu noktada söz konusu antivirüs yazılımlarının, masum olan bir programı kötücül yazılım sınıfında değerlendirdiği, diğer bir ifadeyle düşük doğrulukta (false-positive) tespit yaptığı anlaşılmaktadır.

Masum program üzerinde kod enjeksiyonu için gerekli ".malf" bölümü oluşturulduktan sonra bu bölüme Metasploit aracı kullanılarak üretilmiş olan kötücül kod enjekte edilmiştir. Enjekte edilen kötücül kodun ikili formattaki görüntüsü Şekil 5'te yer almaktadır. Kötücül kod enjekte edildikten sonra oluşturulan yeni çalıştırılabilir dosya Virustotal aracılığıyla taratıldığında elde edilen rapor Şekil 6'da görülmektedir. Bu rapora göre; 30/67 adet antivirüs



Şekil 2. Masum programa ait tarama raporu (The scanning report of the innocent program)



Şekil 3. Masum program üzerinde oluşturulmuş yeni bölüm (New section which was created on the innocent program)

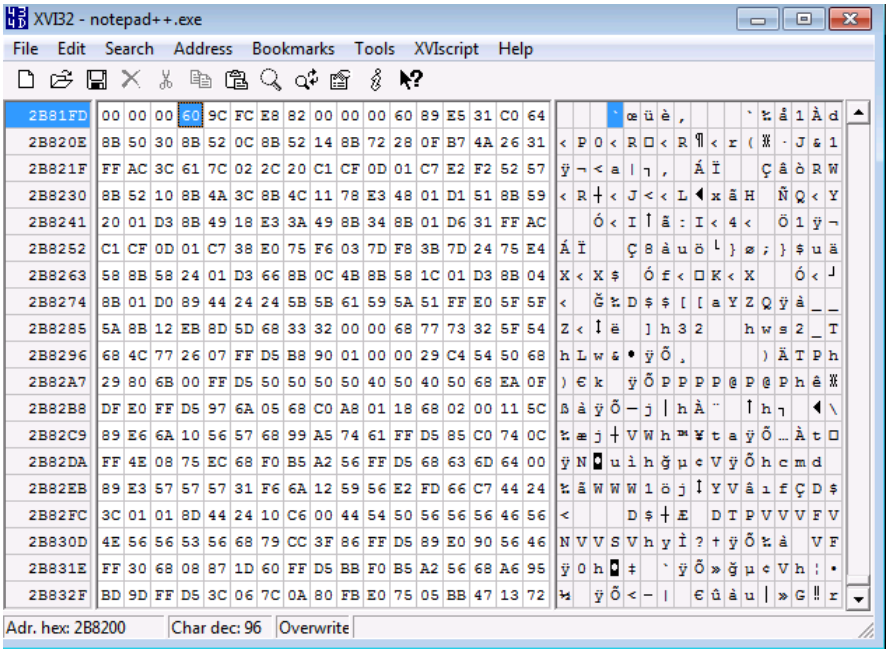
Şekil 4. Üzerinde yeni bölüm oluşturulmuş masum programın tarama raporu
(The scanning report of the innocent program which new section was created on)

yazılımı masum bir programa gizlenmiş olan kötücül kodu tespit edebilmiştir. Bu noktada kötücül kodun üretimi için kullanılan Metasploit çatısının ve bu çatı kullanılarak üretilen kötücül yazılımlara ait imzaların pek çok antivirüs yazılımınca bilinmesi, söz konusu kötücül yazılımın farklı antivirüs yazılımları tarafından kolaylıkla tespit edilebilmesini sağlamıştır.

4.2. Anti-Dinamik Modifikasyon ile Manipülasyon
(Manipulation with Anti-Dynamic Modification)

Günümüzde antivirüs yazılımlarının birçoğu kötücül yazılımları tespit edebilmek için durağan ve dinamik analiz yöntemlerini bir arada kullanmaktadır. Dolayısıyla Şekil 6'da paylaşılan rapora bakılarak her ne kadar 30/67 adet farklı antivirüs yazılımının kötücül yazılımı tespit edebildiği söylenebilse de söz konusu tespit için hangi analiz yöntemlerinin kullanıldığı bilinememektedir. Antivirüs yazılımları tarafından kullanılan dinamik analiz yöntemleri, siber saldırganlar tarafından anti-dinamik özellikli modifikasyonlar yapılarak etkisiz hâle getirilebilmektedir. Birçok anti-dinamik analiz yöntemi bulunmakla birlikte bu çalışmada, masum programın gerçek bir kullanıcı ile etkileşim hâlinde olup olmadığının kontrol edildiği bir yöntem tercih edilmiştir [30]. Bu amaçla kötücül kod bloğu enjekte edilmiş masum program, Ollydbg [42] ile manipüle edilerek anti-dinamik özellikli kötücül yazılıma dönüştürülmüştür. Oluşturulan yeni çalıştırılabilir dosya Virustotal aracılığıyla taratıldığında elde edilen yeni rapor Şekil 7'de yer almaktadır.

Elde edilen yeni rapora göre kötücül yazılımın sadece 9/67 adet antivirüs yazılımı tarafından tespit edildiği ve dinamik analiz yöntemleri temelinde daha fazla antivirüs yazılımını



Şekil 5. Masum programa enjekte edilmiş kötücül kodun ikili formattaki görüntüsü
(Binary form image of malicious code which was injected into the innocent program)

Şekil 6. Kötücül kod enjekte edilmiş masum programın tarama raporu
(The scanning report of the innocent program which the malicious code was injected into)

Şekil 7. Anti-dinamik modifikasyon ile manipüle edilmiş kötücül yazılımın tarama raporu
(The scanning report of the malware which was manipulated with anti-dynamic modification)

atlatabildiği görülmektedir. Anti-dinamik modifikasyon işlemine rağmen kötücül yazılımın hâlen 9 adet antivirüs yazılımı tarafından tespit edilebilmesinin altında yatan ana sebep ise kötücül yazılımın antivirüs yazılımlarının çalışma zamanlı kontrollerinden gizlenmiş olmasına karşın disk üzerinde yer alan açık görüntüsünün (şifrelenmemiş hâlinin) antivirüs yazılımları tarafından kötücül yazılım tespitinde kullanılabilir durumda olmasıdır.

4.3. Şifreleme ile Manipülasyon (Manipulation with Encryption)

Kötücül yazılımlar, bellek ve disk üzerinde yer alan şifresiz açık görüntüleri nedeniyle başta imza temelli olmak üzere durağan tespit yöntemleri tarafından kolaylıkla tespit edilebilmektedir. Bu duruma karşın siber saldırganlar tarafından uygulanan yöntemlerin başında kötücül kod bloğunun önceden belirlenmiş bir anahtar ile şifrelenmesi gelmektedir. Çalışmanın bu bölümünde de kötücül yazılımın imza temelli yöntemlerle tespit edilebilmesini güçleştirmek amacıyla masum programa enjekte edilen kötücül kod "0000 1000" bit dizisi kullanılarak şifrelenmiş ve şifreleme için en basit yöntemlerden biri olan XOR kullanılmıştır.

Enjekte edilen kötücül kodun şifrelenmesiyle oluşturulan yeni çalıştırılabilir dosya Virustotal aracılığıyla taratıldığında elde edilen rapor Şekil 8'de yer almaktadır. Rapordan da görüldüğü üzere kötücül yazılım sadece 4/66 adet farklı antivirüs yazılımı tarafından tespit edilebilmiştir.

Şekil 8. Şifreleme ve anti-dinamik modifikasyon ile manipüle edilmiş kötücül yazılımın tarama raporu
(The scanning report of the malware which was manipulated with encryption and anti-dynamic modification)

Diğer bir ifadeyle bu son tarama, kötücül kod bloğu şifrelenmiş bir kötücül yazılımın anti-dinamik manipülasyondakine göre fazladan 5 adet antivirüs yazılımını daha atlatabildiğini göstermektedir. Diğer taraftan kötücül yazılımın hâlen 4 adet antivirüs yazılımı tarafından tespit edilebiliyor olması, söz konusu kötücül yazılımın farklı imza temelli veya diğer dinamik analiz temelli yöntemleri aşamadığı anlamına gelmektedir.

5. SONUÇLAR (CONCLUSIONS)

Bu çalışma; kod enjeksiyonu, anti-dinamik modifikasyon ve şifreleme yöntemleriyle imza ve dinamik analiz temelli antivirüs yazılımlarının büyük ölçüde atlatılabildiğini ortaya koymuştur. Bununla birlikte atlatılamayan antivirüs yazılımları için bu çalışmaya konu olan atlatma yöntemleri dışındaki farklı anti-tespit yöntemlerinin kullanılması gerektiği anlaşılmıştır.

Virustotal platformunda olduğu gibi birçok antivirüs yazılımının hangi konfigürasyonda ve ne kadar hassas düzeyde tarama yaptığı tam olarak bilinemediğinden, kötücül yazılım üzerinde yapılacak farklı her yeni modifikasyonun, tarama sonuçlarında ne şekilde değişimlere neden olacağı öngörülememektedir [41]. Bu sebeple her yeni modifikasyonda kötücül yazılımın Virustotal aracılığıyla tekrardan taratılabiliyor olması, siber saldırganlara bir sonraki modifikasyon adımının belirlenmesi ve test edilmesinde yol gösterici olmaktadır.

İmza ve dinamik analiz temelli tespit yöntemlerini kullanan antivirüs yazılımlarının kod enjeksiyonu, anti-dinamik modifikasyon ve şifreleme yöntemleriyle atlatılabilmesi, bu çalışmanın kapsamı dışındaki antivirüs yazılımlarının da farklı ileri anti-tespit yöntemleri ile atlatılabilme ihtimalini ortaya koymaktadır. Hem durağan hem de dinamik tespit yöntemlerinin siber saldırganlar tarafından kolaylıkla atlatılabiliyor olması, siber uzayda farklı bir güvenlik perspektifine ihtiyaç olduğunu göstermektedir. Bu doğrultuda sandbox (kum havuzu) teknolojisi, mevcut antivirüs yazılımlarında yer alan zayıflıkların giderilmesi noktasında umut vaat eden yöntemlerden birisi olarak karşımıza çıkmaktadır [43]. Bu kapsamda, ileriki çalışmalarda sandbox teknolojisinin sunmuş olduğu avantajları ele alan ve siber saldırganlar tarafından kullanılan anti-sandbox yöntemlerini inceleyen araştırmaların yürütülmesinin alanyazına önemli katkılar sunacağı düşünülmektedir. Anti-sandbox yöntemlerine karşı kötücül yazılımların tespitinde de uç nokta tehdit algılama ve yanıt (endpoint detection and response - EDR) [44-46], güvenlik bilgi ve olay yönetimi (security information and event management - SIEM) [47, 48] gibi farklı ileri analiz ve güvenlik yaklaşımları tamamlayıcı rol oynamaktadır. Sonuç olarak mevcut antivirüs çözümlerinin sandbox, EDR, SIEM gibi farklı güvenlik yaklaşımlarıyla bütünleşik bir biçimde kullanılmasının, kötücül yazılımların tespit seviyesini artıracağı değerlendirilmektedir.

KAYNAKLAR (REFERENCES)

1. Abomhara M., Køien G.M., Cyber Security and the Internet of Things: Vulnerabilities, Threats, Intruders and Attacks, Journal of Cyber Security and Mobility, 4 (4), 65-88, 2015.
2. Görmüş S., Aydın H., Ulutaş G., Security for The Internet of Things: A Survey of Existing Mechanisms, Protocols and Open Research Issues, Journal of the Faculty of Engineering and Architecture of Gazi University, 33 (4), 1247-1272, 2018.
3. Ranveer S., Hiray S., Comparative Analysis of Feature Extraction Methods of Malware Detection, International Journal of Computer Applications, 120 (5), 1-7, 2015.
4. Neumann J.V., Theory of Self-Reproducing Automata, Editör: Burks A.W., University of Illinois Press, Illinois, USA, 1966.
5. Barria C., Cordero D., Cubillos C., Palma M., Proposed Classification of Malware, Based on Obfuscation, 6th International Conference on Computers Communications and Control (ICCCC), Oradea-Romania, 37-44, 10-14 May, 2016.
6. Fosnock C., Computer Worms: Past Present and Future, East Carolina University Technical Report, 2005.
7. Imran M., Afzal M.T., Qadir, M.A., A Comparison of Feature Extraction Techniques for Malware Analysis, Turkish Journal of Electrical Engineering and Computer Sciences, 25, 1173-1183, 2017.
8. Hughes L.A., DeLone, G.J., Viruses, Worms, and Trojan Horses: Serious Crimes, Nuisance, or Both?, Social Science Computer Review, 25 (1), 78-98, 2007.
9. Canbek G., Sağıroğlu Ş., Malware And Spyware: A Comprehensive Review, Journal of the Faculty of Engineering and Architecture of Gazi University, 22 (1), 121-136, 2007.
10. Kara I., A Basic Malware Analysis Method, Computer Fraud & Security, 2019 (6), 11-19, 2019.
11. Veerappan C.S., Keong P.L.K., Tang Z., Tan F., Taxonomy on Malware Evasion Countermeasures Techniques, IEEE 4th World Forum on Internet of

Things (WF-IoT), Singapore, 558-563, 5-8 February, 2018.
12. Ye Y., Li T., Adjeroh D., Iyengar S.S., A Survey on Malware Detection Using Data Mining Techniques, ACM Computing Surveys (CSUR), 50 (3), 41:1-40, 2017.
13. Nachenberg C., Computer Virus-Antivirus Coevolution, Communications of the ACM, 40 (1), 46-51, 1997.
14. **14.** Zarghoon A., Awan I., Disso J.P., Dennis R., Evaluation of AV Systems Against Modern Malware, 12th International Conference for Internet Technology and Secured Transactions (ICITST), Cambridge-UK, 269-273, 11-14 December, 2017.
15. Al-Asli M., Ghaleb T.A., Review of Signature-Based Techniques in Antivirus Products, 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka-Saudi Arabia, 1-6, 3-4 April, 2019.
16. Özgür A., Erdem H., Feature Selection and Multiple Classifier Fusion Using Genetic Algorithms in Intrusion Detection Systems, Journal of the Faculty of Engineering and Architecture of Gazi University, 33 (1), 75-87, 2018.
17. Li J., Li Q., Zhou S., Yao Y., Ou J., A Review on Signature-Based Detection for Network Threats, IEEE 9th International Conference on Communication Software and Networks (ICCSN), Guangzhou-China, 1117-1121, 6-8 May, 2017.
18. Lee J., Jo M.J., Shin J.S., LigeroAV: A Light-Weight, Signature-Based Antivirus for Mobile Environment, IEICE Transactions on Information and Systems, E99.D (12), 3185-3187, 2016.
19. Abbas M.F.B., Srikanthan T., Low-Complexity Signature-Based Malware Detection for IoT Devices, Applications and Techniques in Information Security, 719, 181-189, 2017.
20. Alzahrani A.J., Ghorbani A.A., Real-Time Signature-Based Detection Approach for SMS Botnet, 13th Annual Conference on Privacy, Security and Trust, İzmir-Türkiye, 157-164, 21-23 Temmuz, 2015.
21. Rad B.B., Masrom M., Ibrahim S., Camouflage in Malware: From Encryption to Metamorphism, International Journal of Computer Science And Network Security (IJCSNS), 12 (8), 74-83, 2012.
22. Szor P., The Art of Computer Virus Research and Defense, 1st Edition, Editörs: Gettman K., Goldstein J., Kanouse G., Hart K., Andry C., Addison-Wesley, Maryland, USA, 2005.
23. Namanya A.P., Cullen A., Awan I.U., Disso J.P., The World of Malware: An Overview, IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), Barcelona-Spain, 420-427, 6-8 August, 2018.
24. Courtney M., States of Cyber Warfare, Engineering & Technology, 12 (3), 22-25, 2017.
25. AV-TEST Independent IT-Security Institute. Malware. https://www.av-test.org/en/statistics/malware/. Yayın tarihi Ocak, 2010. Erişim tarihi Ağustos 11, 2019.
26. Sharma A., Sahay S.K., Evolution and Detection of Polymorphic and Metamorphic Malwares: A Survey, International Journal of Computer Applications, 90 (2), 7-11, 2014.
27. Singh J., Singh J., Challenges of Malware Analysis: Obfuscation Techniques, International Journal of Information Security Science, 7 (3), 100-110, 2018.
28. Rieck K., Trinius P., Willems C., Holz T., Automatic Analysis of Malware Behavior Using Machine Learning, Journal of Computer Security, 19 (4), 639-668, 2011.
29. Deka D., Sarma N., Panicker N.J., Malware Detection Vectors and Analysis Techniques: A Brief Survey, 2016 International Conference on Accessibility to Digital World (ICADW), Guwahati-India, 81-85, 16-18 December, 2016.
30. Jadhav A., Vidyarthi D., Hemavathy M., Evolution of Evasive Malwares: A Survey, 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), New Delhi-India, 641-646, 11-13 March, 2016.
31. Durmuş G., Soğukpınar İ., A Novel Approach for Analyzing Buffer Overflow Vulnerabilities in Binary Executables by Using Machine Learning Techniques, Journal of the Faculty of Engineering and Architecture of Gazi University, 34 (4), 1695-1704, 2019.
32. Soliman S.W., Sobh M.A., Bahaa-Eldin A.M., Taxonomy of Malware Analysis in the IoT, 12th International Conference on Computer Engineering and Systems (ICCES), Cairo-Egypt, 519-529, 19-20 December, 2017.
33. Upchurch J., Zhou X., Malware Provenance: Code Reuse Detection in Malicious Software at Scale, 11th International Conference on Malicious and Unwanted Software (MALWARE), Fajardo-Puerto Rico, 1-9, 18-21 October, 2016.
34. Cani A., Gaudesi M., Sanchez E., Squillero G., Tonda A., Towards Automated Malware Creation: Code Generation and Code Integration, Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC'14), Gyeongju-Korea, 157-160, 24-28 March, 2014.
35. Zarghoon A., Awan I., Disso J.P., Dennis R., Evaluation of AV Systems Against Modern Malware, 12th International Conference for Internet Technology and Secured Transactions (ICITST), Cambridge-UK, 269-273, 11-14 December, 2017.
36. Lim C., Suryadi K.R., Kotualubun Y.S., Mal-Flux: Rendering Hidden Code of Packed Binary Executable, Digital Investigation, 28, 83-95, 2019.
37. Calleja A., Tapiador J., Caballero J., The MalSource Dataset: Quantifying Complexity and Code Reuse in Malware Development, IEEE Transactions on Information Forensics and Security, 14 (2), 3175-3190, 2019.
38. Notepad++. https://notepad-plus- plus.org/repository/ repository/ 7.x/7.7.1/npp.7.7.1.bin.zip. Erişim tarihi Ağustos 11, 2019.
39. Stefinko Y., Piskozub A., Banakh R., Manual and Automated Penetration Testing. Benefits and

Drawbacks. Modern Tendency, 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET), Lviv-Ukraine, 488-491, 23-26 February, 2016.

40. Catak F.O., Ahmet F.Y., A Benchmark API Call Dataset for Windows PE Malware Classification, arXiv:1905.01999v1 [cs.CR], 1-8, 2019.

41. Quarta D., Salvioni F., Continella A., Zanero S., Extended Abstract: Toward Systematically Exploring Antivirus Engines, Detection of Intrusions and Malware, and Vulnerability Assessment, 15th International Conference DIMVA, Saclay-France, 393-403, 28-29 June, 2018.

42. Novkovic I., Groš S., Can Malware Analysts Be Assisted in Their Work Using Techniques from Machine Learning?, 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija-Croatia, 1408-1413, 30 May-3 June, 2016.

43. Vokorokos L., Baláž A., Madoš B., Application Security through Sandbox Virtualization, Acta Polytechnica Hungarica, 12 (1), 83-101, 2015.

44. Hassan W.U., Bates A., Marino D., Tactical Provenance Analysis for Endpoint Detection and Response Systems, 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA-USA, 1172-1189, 18-21 May, 2020.

45. Sjarif N.N.A., Chuprat S., Mahrin M.N., Ahmad N.A., Ariffin A., Senan F.M., Zamani N.A., Saupi A., Endpoint Detection and Response: Why Use Machine Learning?, 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island-Korea (South), 283-288, 16-18 October, 2019.

46. Campfield M., The Problem with (Most) Network Detection and Response, Network Security, 2020 (9), 6-9, 2020.

47. Majeed A., ur Rasool R., Ahmad F., Alam M., Javaid N., Near-Miss Situation Based Visual Analysis of SIEM Rules for Real Time Network Security Monitoring, Journal of Ambient Intelligence and Humanized Computing, 10, 1509-1526, 2019.

48. Bryant B.D., Saiedian H., A Novel Kill-Chain Framework for Remote Security Log Analysis with SIEM Software, Computers & Security, 67, 198-210, 2017.