ÖHÜ Müh. Bilim. Derg. / OHU J. Eng. Sci.

ISSN: 2564-6605
doi: 10.28948/ngumuh.443149
Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi, Cilt 7, Sayı 2, (2018), 478-489
Omer Halisdemir University Journal of Engineering Sciences, Volume 7, Issue 2, (2018), 478-489

Araştırma / Research

YAYGIN GÖRÜLEN DOSYA ENJEKSİYON ZARARLILARININ ANALİZİ VE SİSTEMATİK OLARAK TESPİTİ

Alper ECEMİŞ[1] (ORCID ID: 0000-0001-5455-0006) *
Ecir Uğur KÜÇÜKSİLLE[1] (ORCID ID: 0000-0002-3293-9878)
Mehmet Ali YALÇINKAYA[2] (ORCID: 0000-0002-7320-5643)

[1] *Bilgisayar Mühendisliği, Süleyman Demirel Üniversitesi, Isparta, Türkiye*
[2] *Bilgisayar Mühendisliği, Ahi Evran Üniversitesi, Kırşehir, Türkiye*

Geliş / Received: *25.04.2018*
Kabul / Accepted: *26.06.2018*

ÖZ

Zararlı yazılım, işletim sistemlerinin işleyişini bozmak, veri hırsızlığı yapmak, dosyaları kullanılmaz hale getirmek, kişisel bilgisayar sistemlerine erişmek ve istenmeyen reklamları göstermek amacı ile kullanılan yazılımdır. AV-TEST Enstitüsü, her gün dünya üzerindeki farklı bölgelerden gelen 250,000'in üzerinde ki yeni zararlı yazılımı kayıt altına almaktadır. Zararlı yazılım türlerinden birisi de dosya enjeksiyon zararlılarıdır. Dosya enjeksiyon zararlıları işletim sistemleri üzerindeki çalıştırılabilir veya çalışmakta olan dosyalara enjeksiyon yaparak kullanıcıların istismarına neden olan zararlı yazılımlardır. Kurban bilgisayardaki dosyaları şifrelemek, zararlı sunucu bağlantısı kurmak, çalışmakta olan dosyalara bulaşıp sistemin yavaşlamasına sebebiyet vermek, çıkarılabilir sürücülere bulaşmak, bilgisayar donanımını kullanarak fiziksel hasarlara yol açmak etkileri arasındadır. Yapılan çalışmada, dünya üzerinde yaygın olarak görülen ve önem arz eden 10 farklı türde dosya enjeksiyon zararlısının Windows ve Linux tabanlı sanal laboratuvarlar üzerinde statik ve dinamik analizi gerçekleştirilmiş, elde edilen ortak özelliklere göre dosya enjeksiyon zararlılarını tespit etmek için bir algoritma önerilmiştir. Bu algoritma aracılığıyla statik ve dinamik analiz yapılarak girdi olarak verilen dosyanın zararlı olup olmadığı tespit edilebilmiştir.

Anahtar Kelimeler: Dosya enjeksiyon zararlıları, zararlı yazılım, zararlı yazılım analizi, siber güvenlik

WIDELY SEEN FILE INJECTION MALWARE ANALYSIS AND SYSTEMATIC DETECTION

ABSTRACT

Malware is used as a software that aims to break the processing of operating system, doing data theft, making the folders useless, reaching the private computer system and showing the unwanted advertisement. Above 250,000 new malware is recorded everyday by the Institute of AV-TEST from a different region of the world. File injection malware is considered one of the harmful malware in the world. File injection malware is defined as a malware that is used to abuse the users by injecting a virus into the executable or already executed folders on the operating system. The effects of this kind of malware can be stated as putting a password to the folders in the victim computer, installing harmful server connection, spreading the effect on running folders to slow down the operating system, effecting removable drives, causing physical damage using computer hardware. In this case, 10 different types of file injection malwares which are widely seen and important in the world determined. The structural and behavioural analysis of the file injection malware was then carried out in virtual laboratories established on Windows and Linux operating systems. A new algorithm is proposed to detect the file injection

* Corresponding author / Sorumlu yazar. Tel.:+90 506 644 1692 ; e-mail / e-posta: ecemisalper@gmail.com

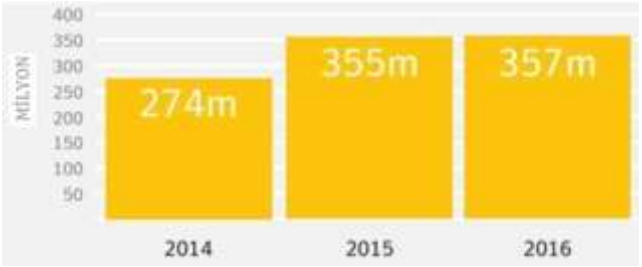
malwares by capturing the common behavioural properties. Through this algorithm, it is possible to determine whether the input file is harmful by performing static and dynamic analysis.

Keywords: File injection malwares, malware, malware analysis, cyber security

1. GİRİŞ

Teknolojinin baş döndüren bir hızla ilerlediği günümüzde, her geçen gün yeni bilgisayar programları pazara sürülmekte ve mevcut programlar ise güncellemeler ile yeni sürümlerine geçmektedir. Bu güncellemeler her ne kadar zararlı yazılımlardan korunmak için güvenlik açıklarını önlese de yeni güvenlik zafiyetlerine de neden olabilmektedir. Bilgisayar korsanları yeni güncellemeler ile gelen bu güvenlik zafiyetlerini keşfederek kullanıcı istismarına neden olmaktadırlar. Programcıların güvenlik zafiyetlerini önlemek için üretmiş oldukları yazılım güncellemeleri ve bilgisayar korsanlarının bu yazılımların getirmiş olduğu güvenlik açıklarını kullanarak yeni zararlı yazılım üretme amacı günümüzde kısır bir döngü oluşturmuştur. Şekil 1' de yıl bazlı olarak tespit edilen yeni zararlı yazılım varyasyon sayısı gösterilmektedir.

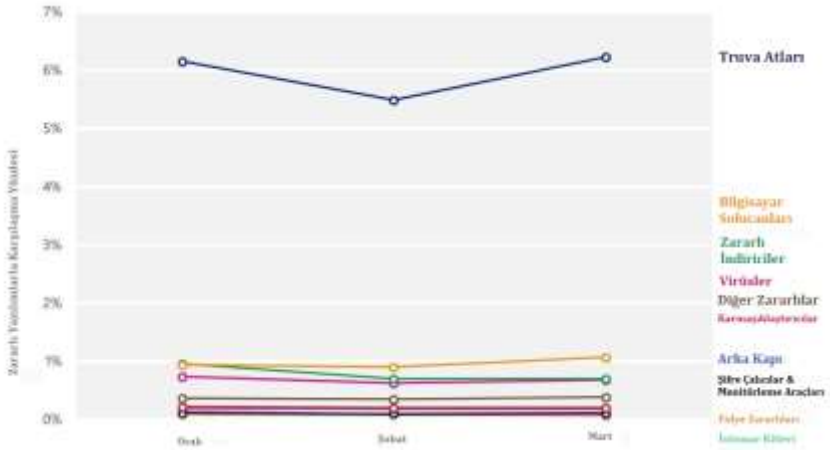
Sosyal mühendislik saldırıları, işletim sistemleri, bilgisayar programları ve zararlı yazılım önleyici programlar üzerinde yer alan güvenlik zafiyetleri kullanılarak sistemlere bulaşan zararlı yazılımlar, pek çok farklı amaca hizmet etmektedir. Önemli dosyaları şifreleyerek fidye karşılığı para kazanmak, geri döndürülemeyecek türde zarar vermek, bilgi ve veri hırsızlığı yapmak verilebilecek örneklerden bazılarıdır.



Şekil 1. Tespit edilen yeni zararlı yazılım varyasyon sayısı [1]

Dosya enjeksiyon zararlıları, işletim sistemi üzerindeki çalıştırılabilir dosyalara enjekte olup; veri çalma, dosyaları şifreleme, zararlı network bağlantıları kurma gibi kötücül eylemlere sahip olan zararlı yapılardır[2].

Dosya enjeksiyonu gerçekleştiren zararlılar, çoğunlukla truva atı, fidye virüsü (ransomware), arka kapı virüsü (backdoor) ve bilgisayar virüsleri gibi kötücül yazılım türlerinde gözlemlenmektedir. Yüksek miktarda finansal getirisi olduğu için dosya enjeksiyon zararlılarının popülaritesi yıldan yıla artmaktadır.



Şekil 2. Ocak, Şubat, Mart aylarına ait zararlı yazılım türleri ve bilgisayarların karşılaşma oranı [3]

Siber güvenlik sektörünün önde gelen kuruluşlarından Microsoft'un yayınladığı Şekil 2'de ki veriler, 2017 yılının Ocak, Şubat ve Mart aylarındaki zararlı yazılım türleri ve bu zararlı yazılım türlerine bilgisayarlarda rastlanma oranına aittir. Bu da dosya enjeksiyon zararlılarının siber güvenlik sektöründe ne kadar önem taşıdığını kanıtlamaktadır. Dosya enjeksiyon zararlıları içerisinde bulunan fidye ve "bitcoinminer" zararlıları geliştiricilerine en büyük finansal getiriyi sağlamaktadır. Bu sebeple fidye ve bitcoin zararlıları, yaygınlık oranı diğer dosya enjeksiyon zararlılardan daha az olsa bile meydana getirdiği maddi zarar yönünden büyük tehlike arz etmektedir.

Dosya enjeksiyon zararlılarının dünya üzerinde görülme oranları incelendiğinde, çoğunlukla belirli bölgelerin hedef alındığı ya da zararlıların ilgili bölgelere yönelik olarak geliştirildiği görülmektedir. Yaygınlığı belirlenen dosya enjeksiyon zararlıları; Sality Türkmenistan' da %8.29[4], Ramnit Bangladeş' de %6.76[5], Bundpil Afganistan'da %12.76[6], ZeusBot İtalya' da %0.54[7], Dridex İngiltere' de %0.17[8], Spatet Fildişi Sahili' nde %0.33[9], Kryptik Pakistan' da %0.04[10], Virut Afganistan' da %7.84[11] ve WannaCry Venezuela' da %5.5[12] oranla görülmektedir. Petya zararlısının yaygınlık oranına bakıldığında ise WannaCry ile eş zamanda yükseliş gösterdiği görülmektedir[13].

Literatürde gerçekleştirilen çalışmalar incelendiğinde; bazı dosya enjeksiyonu zararlılarının analiz edilerek yapılarının incelendiği, zararlı analizinde kullanılan statik ve dinamik analiz tekniklerinin yorumlandığı, çeşitli savunma önerilerinin sunulduğu farklı çalışmalar bulunmaktadır. Rieck ve arkadaşları, [14] çalışmalarında makine öğrenmesini kullanarak zararlı yazılım davranışlarının otomatik analizi için bir çerçeve önermişlerdir. Ehrenfeld, [15] çalışmasında 12 Mayıs 2017 Cuma günü WannaCry zararlısı kullanılarak gerçekleştirilen siber saldırının, farklı sektörler üzerindeki etkisini incelemiştir. Moser ve arkadaşları gerçekleştirmiş oldukları [16] çalışmalarında, statik analiz tekniğini kullanan, semantik imza tabanlı zararlı yazılım analiz araçlarının yeterliliğini incelemişlerdir. Yazarlar program kontrol akışını değiştirme, yerel ve global değişkenlere erişimi kısıtlama, işlemci kayıtçılarında tutulan değerleri gizleme gibi çeşitli teknikler ile statik analiz tekniklerinin zararlı yazılımları tespit etmede yeterli olmadığını belirtmişlerdir. Snow vd. [17] çalışmalarında, kod enjeksiyon zararlılarının tespiti için yazılım tabanlı tespit tekniklerini kullanmak yerine, donanım sanallaştırmayı kullanmanın daha verimli olacağını belirtmişlerdir. Bu çalışmada içerisinde zararlı yazılım bulunduran "PDF" formatındaki dosyalar ve ağ tabanlı saldırılar bu bakış açısı kullanılarak analiz edilip elde edilen sonuçlar raporlanmıştır. Davide vd. gerçekleştirmiş oldukları [18] çalışmalarında, GPU kullanarak çalışmakta olan zararlı yazılımların bellek üzerinde oluşturdukları etkilerini incelemişlerdir. Öncelikle zararlı yazılımların varlığını gizlemek için kullanabilecekleri dört farklı tekniği tanıtmışlar sonrasında Intel marka GPU üzerinde çalışmak için geliştirilmiş bir zararlı yazılımın analizini gerçekleştirmişlerdir. Gazet gerçekleştirmiş olduğu [19] çalışmasında fidye zararlılarını, kod kalitesi, işlevi, şifreleme teknikleri bakımından analiz etmiştir. Gerçekleştirilen analiz sonrasında fidye zararlılarının, güçlü ve zayıf yönleri ortaya konulmuştur. Cabaj ve arkadaşları [20] çalışmalarında, CryptoWall fidye zararlısı için ağ davranış analizi gerçekleştirmişlerdir. HoneyPot olarak bilinen tuzak sunucu yapısı kullanılarak zararlı yazılımın ağ üzerindeki davranışlarını incelemişlerdir.

Bu çalışmada; son 2 yılda gerçekleştirilen siber saldırılarda en yaygın olarak kullanılan 10 farklı dosya enjeksiyon zararlısı belirlenmiş, söz konusu zararlı yazılımların yayılma yöntemleri, dünya üzerinde yoğunlukla görüldüğü bölgeler, meydana getirdikleri maddi hasarlar ve bu zararlılara karşı alınabilecek önlemler hakkında bilgiler verilmiştir. Ayrıca belirlenen 10 farklı dosya enjeksiyon zararlısı, Windows ve Linux işletim sistemleri kullanılarak oluşturulmuş sanal laboratuvar ortamlarında, ayrı ayrı statik ve dinamik analiz işlemlerine tabi tutulmuş, yapısal ve davranışsal karakteristikleri elde edilmiştir. Gerçekleştirilen analiz işlemleri sonrasında elde edilen veriler karşılaştırılmış, söz konusu zararlıların ortak davranışsal özellikler belirlenmiş ve elde edilen veriler ışığında dosya enjeksiyon zararlılarının tespiti için yeni bir algoritma önerilmiştir.

Gerçekleştirilen çalışma; analiz işlemlerinde son iki yılda en yaygın olarak kullanılan dosya enjeksiyon zararlılarının kullanılması, söz konusu zararlıların Windows ve Linux tabanlı sanal laboratuvarlar üzerinde hem statik hem dinamik analiz işlemine tabi tutulması ve analiz işlemlerinden elde edilen veriler ışığında bir savunma algoritmasının önerilmesi yönleri ile literatürde yer alan diğer çalışmalardan farklılık göstermektedir.

2. DOSYA ENJEKSİYON ZARARLISI

Dosya enjeksiyon zararlıları işletim sistemleri üzerindeki çalıştırılabilir dosyalara ya da işlemlere etki ederek kullanıcıların istismarına neden olan zararlı yazılımlardır[2].

Sisteme bulaştıktan sonra oluşturdukları kötücül eylemlere göre sınıflandırılan dosya enjeksiyon zararlıları; truva atı, arka kapı virüsleri, bilgisayar virüsleri, bilgisayar solucanları, fidye virüsleri gibi pek çok farklı zararlı yazılım türünü de içerisinde barındırmaktadır.

Dosya enjeksiyon zararlısının temel amacı, kullanıcının haberi ya da izni olmadan kurban sistem üzerinde bilgisayar korsanlarının kötücül eylemlerini gerçekleştirmektir. Dosya enjeksiyon zararlısı hem virüs önleyici programlar tarafından tespit edilmemek hem de analiz edilmesini güçleştirmek için zararlı yapısını şifreli ya da karmaşıklaştırılmış kod olarak, paket halinde tutmaktadır. Kurban sistem üzerinde çalışmaya başladıktan sonra, paketlenmiş yapı çözülerek içerisinde bulunan zararlı kod enjekte edilmektedir.

Zararlı IP adresleri ile bağlantı kurmak, kurban bilgisayarı zombi ağına eklemek, veri hırsızlığı yapmak, donanımsal ve yazılımsal hasarlara sebebiyet vermek dosya enjeksiyon zararlılarının yapabileceği kötücül faaliyetlere örnek gösterilebilir. Ayrıca bilgisayar korsanları, veri hırsızlığı yapan zararlı yazılımları ve fidye virüslerini kullanarak, kurban kullanıcılara maddi ve manevi hasarlar da vermektedir[21].

Dosya enjeksiyon zararlıları yaptıkları kötücül eylemlere göre sınıflara ayrılsa da günümüzde geliştirilen yeni zararlı yazılımlar, farklı türdeki zararlı yazılımın kötücül eylemlerine de sahip olabilmektedir. Örneğin; Androm isimli arka kapı virüsü sadece sisteme arka kapı açmakla kalmayıp, truva atlarının kullanmış olduğu veri hırsızlığı kötücül eylemini de gerçekleştirmektedir.

Bu zararlıların başlıca üretilme nedeni para kazanmak olsa da; veri hırsızlığı ve şantaj yapmak, sistemleri yavaşlatmak veya kullanılmaz duruma getirmek, istenilen siteyi arama motorlarında ön sıraya getirmek(SEO) gibi üretilme amaçları da vardır. Günümüz siber suç ortamında, farklı ülkelere mensup bilgisayar korsanlarının kendi içerisinde gruplaştıkları ve diğer ülkelerde bulunan şirketlere saldırılar düzenleyerek maddi ve sosyal zararlar verdikleri bilinmektedir.

Dosya enjeksiyon zararlılarının yayılmasında ise genellikle sosyal mühendislik saldırıları kullanılmaktadır. Bilgisayar korsanları bu işlemi gerçekleştirirken ya zararlı mailler ve istismar kitlerini kullanarak ya da kullanıcının merak ve korku gibi duygularını sömürerek zararlı yazılımı kurban kullanıcıya bulaştırır. Ayrıca, dosya enjeksiyon zararlılarının yayılmasında; kendi kendini kopyalayarak yayılma, reklamlar aracılığı ile yayılma, kaba kuvvet saldırıları kullanılarak yayılma, sunuculardaki güvenlik açıkları kullanarak yayılma gibi yayılma yöntemleri de kullanmaktadırlar[13].

3. MATERYAL VE METOT

Bu bölüm içerisinde gerçekleştirilen deneysel çalışmada analizi yapılan zararlı yazılımların hash değerleri ve analiz yöntemleri hakkında bilgi verilmiştir.

3.1. Dosya Enjeksiyon Zararlısının Analizi

Dosya enjeksiyon zararlısının analizi yapılırken öncelikle zararlının yapısal analizi yapılmalıdır. Yapısal analizden kasıt; zararlı yazılımın sistemde çalıştırılmadan önce içinde barındırdığı metinsel ifadeler, çağırılan Windows fonksiyonları, dosya bölüm entropileri, yapı içeriğinin paket halinde olup olmaması ve hash değerleridir. Yapısal analiz sonucunda elde edilen verilerde anormallik olup olmadığı incelenerek zararlının tespitine katkıda bulunulabilir.

Bilgisayar korsanları zararlıları oluştururken antivirüs programları tarafından tespit edilmemesi için verileri paketler. Zararlı, kurban sistemde çalıştırılırken önce paketli yapı açılır sonrasında karmaşıklaştırılmış ya da şifrelenmiş veri varsa çözülerek işleme devam edilir. Bu sebeple zararlı yazılım analizinde zararlının işleyişini anlamak için adım adım çalıştırılarak incelemek önem arzetmektedir.

Zararlı yapının yapısal analizi gerçekleştirildikten sonra, sandbox yapısı ya da sanal bilgisayarlar kullanılarak zararlı adım adım çalıştırılır. Böylece zararlı tarafından oluşturulan dosya ve kayıt defteri aktiviteleri incelenip, ağ analizi yapılabilir. Daha sonra elde edilen veriler kullanılarak zararlı hakkında davranışsal ve karakteristik bilgiler elde edilir.

Zararlı yazılım analizi, statik ve dinamik analiz olmak üzere 2 alt başlıkta incelenmektedir.

3.1.1. Çalışmada Analizi Gerçekleştirilen Zararlıların Hash Değerleri

Sırasıyla statik analizi gerçekleştirilen Sality, Ramnit, Bundpil, ZeusBot, Dridex, Spatet, Kryptik, Virut, WannaCry ve Petya zararlılarına ait “SHA256” hash değerleri aşağıda verilmiştir.

- c321ef762853a2f6e9ebbc6e09c5f258e902f11efc7b5819259923c85e2bb6e4
- 988706aad4817c347b4e1b4cd224726efdf1047227ca4f42378f41de73265b75
- f18266034e0ee4d529cdbee040c489f0db6a82f67ade86544dbfb350738d6bcd
- 167b612f63a554798f879d8052c8ad5554d434e6b7e72bc3fff652d5cf6bfc3b
- e30b76f9454a5fd3d11b5792ff93e56c52bf5dfba6ab375c3b96e17af562f5fc
- 4ccf0ac051ea60a9c6ae7e2001ef79a547dd9e376fcbdeea4814d62e69b709c
- 638554093bfbd55f65b42eb86a9d11ecf53b678cb4e9e5ec058c0e4712189f0e
- 0264066ef28d1ab422e98c4ebc6738fb6f8676447aa27768d991bb8d8cad0a7f
- 39c7bdb2d0fc8f2b11de1e781d88296ac233b0b6a8a9cea08f230661af75fdec
- c85d6c55012215efaf6f2e87a18d18a4c7f73999c827fcdd5eea898866924d53

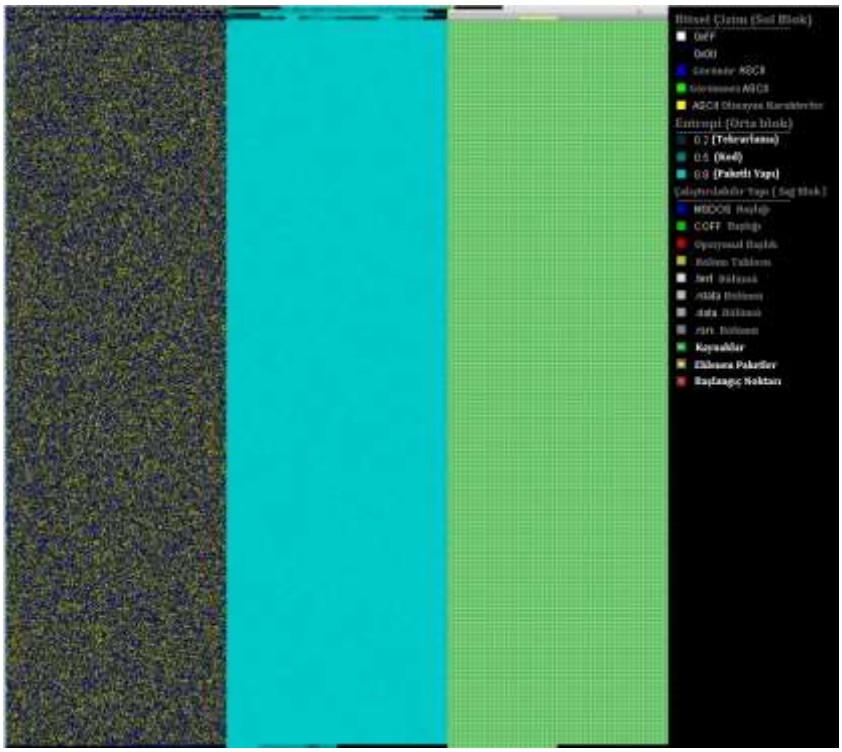
3.1.2. Statik Analiz

Statik analiz, zararlının sistem üzerinde çalıştırılmadan önceki yapısal analizini içerir. Statik analiz sayesinde zararlı yapının içerdiği metinsel ifadeler, çağırılan Windows fonksiyonları, dosya bölüm entropileri, yapı içeriğinin paket halinde olup olmaması, hash değerleri, üretim tarihi gibi veriler elde edilerek anormallikler tespit edilir. Paketli yapıdaki dosya bölüm entropilerinin 6.677' den yüksek olması anormallik olarak kabul edilir[22].

Çalışmada Windows işletim sistemi üzerinde zararlıların statik analizi gerçekleştirilirken kullanılan programlar şunlardır;

- PortexAnalyzer[23]
- Interactive Disassembler Pro[24]
- Strings[25]

Şekil 3' de "PortexAnalyzer" programı kullanılarak elde edilen, dosya enjeksiyon zararlısının yapısal gösterimi verilmiştir. Şekil 3' deki verilere bakılarak zararlı yapının paketli halde tutulduğu açıkca görülmektedir. Ayrıca zararlı yazılım içerisindeki kaynaklar bölümünün paketli yapı ile aynı oranda kullanılması, zararlı yazılımın bu bölüm içerisinde kodlandığına işarettir.



Şekil 3. PortexAnaylzer programı kullanılarak elde edilen, zararlı yapının şekilsel gösterimi

3.1.3. Dinamik Analiz

Dinamik analiz, zararlı yapının sandbox ya da sanal bilgisayarlar üzerinde çalıştırılmasıyla elde edilen dosya, kayıt defteri, işlem ve ağ aktivitelerini içermektedir.

Bilgisayar korsanları zararlı dosyanın çalıştırılmadan önce antivirüs programları tarafından tespit edilmemesi için birçok gizleme işlemi uygulamaktadır. Bu sebeple, zararlı yapı çalıştırıldıktan sonra elde edilen veriler statik analize kıyasla daha önemlidir. Bazı zararlı yapılar anti-sandbox, anti-vm ve anti-debug özelliği taşımaktadır. Anti-debug özelliği taşıyan zararlı sistem üzerinde adım adım çalıştırılıp analiz edilememektedir. Anti-vm ve anti-sandbox özelliğine sahip zararlı yapı ise sanal bir makine üzerinde veya sandbox içerisinde çalıştırılacağını tespit ettiği zaman davranışsal özelliklerini değiştirip araştırmacıyı yanıltmaktadır.

Çalışmada Windows işletim sistemi üzerinde dinamik analiz için kullanılan programlar şunlardır;

- ProcessExplorer[25]
- ProcMon[25]
- RegShot[26]
- ProcessHacker[27]
- API Monitor[28]
- X64dbg[29]
- WireShark[30]
- GlobalFlags[31]

Linux işletim sistemi üzerinde gerçekleştirilen analizlerde ise Cuckoo Sandbox kullanılmıştır[32].

Zararlı yazılımlar, işletim sistemi üzerindeki mevcut ve çalışmakta olan dosyalara etki ederken, birçok Windows fonksiyonu kullanmaktadır.

4. DOSYA ENJEKSİYON ZARARLILARININ TESPİT ALGORİTMASI

Dosya enjeksiyon zararlısının tespiti yapılırken, araştırma bulguları kullanılmıştır. Elde edilen verilerde belirlenen sıra dışı, şüpheli ve zararlı davranışlar gözönüne alınarak zararlı yazılım tespit algoritması oluşturulmuş ve tehlike sevyesi yüzdesel olarak belirlenmiştir. Oluşturulan algoritma 3 ana temel üzerine kurulmuştur.

DEZT (Dosya Enjeksiyon Zararlısı Tespit) algoritmasının kaba kodu Algoritma 1' de verilmiştir.

Algoritma 1. DEZT Tespit Algoritması Kaba Kodu

DEZT TESPİT ALGORİTMASI KABA KODU

Girdiler :
int suphe_sayaci=0, **File** taranacak_dosya, **Object** suphe_zararli[]

Çıktı :
String analizSonucu

```
1. suphe_sayaci = statik_analiz_yap(taranacak_dosya)
2. suphe_zararli[] =  dinamik_analiz_yap(taranacak_dosya)
3. suphe_sayaci = suphe_sayaci + suphe_zararli[0]
4. if  (suphe_zararli[1] == true)
        analizSonucu = ("Taranan veri dosya enjeksiyon zararlısıdır ve tehdit
                        seviyesi % "+ suphe_sayaci*100/7 + " 'dir. ")
   else
      analizSonucu= ("Taranan veri dosya enjeksiyonu yapmamaktadır.
                      Tehdit seviyesi % " + suphe_sayaci*100/7 + " 'dir. ")
5. return analizSonucu
```

Algoritma 1.' in 1. adımında, "suphe_sayaci" isimli değer zararlı yapının farklı şüpheli davranışlarının birikimini göstermektedir. Bu veri sayesinde analiz sonucu elde edilecek şüphe değeri yüzdesel ifadeye dönüştürülüp tehdit seviyesi belirlenmiştir. Bu sebeple 1. adımda "statik_analiz_yap" metodu çağırılarak statik analiz sonucu şüphe sayısı elde edilmiştir. 2. adımda, "suphe_zararli" isimli dizi 1. sütununda şüpheli sayaç verisi, 2. sütununda ise taranan dosyanın zararlı olup olmadığını belirten boolean bir değer taşır. "dinamik_analiz_yap" isimli metot çağırılarak, dinamik analiz esnasında yapının dosya enjeksiyonu yapıp yapmadığı ve şüpheli aktivitelerin sayısını içeren dizi elde edilir. 3. adımda "suphe_sayaci" değişkenine dinamik analiz sonucu elde edilen şüphe sayısı eklenir. 4. adımda zararlının fidye zararlısı olup olmadığı kontrol edilir ve buna göre analiz sonuçları "analizSonucu" değişkenine aktarılır. Son adımda ise analizSonucu değeri geri döndürülür.

Şüphe sayacının 100/7 işleminin sonucu ile çarpılmasının sebebi ise, şüphe sayacının maksimum 7 minimum 0 değeri aralığında olmasıdır.

DEZT algoritması içerisindeki "statik_analiz_yap" metodunun kaba kodu Algoritma 2. 'de verilmiştir.

Algoritma 2. Statik Analiz Fazı

STATİK ANALİZ FAZI KABA KODU

Girdiler:

int suphe_sayaci = 0, **float** entropi = 6.677, **File** taranacak_dosya

Çıktı :

int suphe_sayaci

```
1.  for  int i=0; i < taranacak_dosya.bolumsayisi ; i++
    if   taranacak_dosya.entropi[i] > entropi
         suphe_sayaci++;
         i++;
         break;
    end
2.  if   taranacak_dosya.stringler contains supheli_stringler
         suphe_sayaci++;
    end
3.  if   taranacak_dosya.call  contains supheli_cagrilar
         suphe_sayaci++;
    end
return suphe_sayaci;
```

Statik analiz fazının 1. adımında, taranacak dosyanın dosya bölüm entropileri teker teker belirlenen maksimum normal entropi değeri ile kıyaslanır. Eğer dosya içerisindeki herhangi bir dosya bölümünde bu değerden daha yüksek entropiye rastlanırsa, şüphe sayaç değeri 1 arttırılır ve döngüden çıkılır. 2. adımda, taranacak dosya içerisinde şüpheli metinsel ifadelerin olup olmadığı kontrol edilir. Eğer şüpheli metinsel ifadeler varsa şüphe sayacı 1 arttırılır. Şüpheli metinsel ifadeler gerçekleştirilen analizler sonucu belirlenmiş ve içeriği ilgili referansda verilmiştir[33]. 3. adımda, taranacak dosya içerisinde dosya enjeksiyonunda kullanılan şüpheli Windows çağrılarına ait metinsel ifadelerin olup olmadığı kontrol edilir. Şüpheli Windows çağrılarına ait metinsel ifadeler gerçekleştirilen analizler sonucu belirlenmiş ve içeriği ilgili referansda verilmiştir[33]. 4. adımda ise, toplanan şüpheli durum sayısı geri döndürülür. Bu değer maksimum 3 minimum 0 olabilir.

DEZT algoritması içerisindeki "dinamik_analiz_yap" metodunun kaba kodu Algoritma 3.' de verilmiştir.

Dinamik analiz fazının 1. adımında, "process_enjeksiyon_denetle" metodu kullanılarak dinamik analizi yapılan dosyanın sırasıyla kullandığı Windows çağrıları analiz edilir. Fonksiyon işlem enjeksiyonu tespit ederse "true" edemezse "false" değerini geri dönderir. İşlem enjeksiyonun belirlenmesinde kullanılan Windows çağrıları sırasıyla ilgili referans içerisinde verilmiştir[34]. 2. adımda, eğer işlem enjeksiyonu varsa dosyanın zararlı olduğunu belirtmek üzere "suphe_zararli" dizisinin ikinci sütununa "true" değeri yazılır. Sonrasında şüphe sayacı 1 arttırılır. 3. adımda, "sifreleyici_denetle" metodu kullanılarak yapının dosyaları şifreleyip şifrelemediği kontrol edilir. Elde edilen değer "boolean sifreleyici" içerisine true ya da false olarak yazılır.

Dosya şifrelemede kullanılan Windows çağrıları sırasıyla ilgili referansda verilmiştir[34]. 4. adımda, eğer işletim sistemi üzerindeki dosyalara şifreleme işlemi yapılıyorsa "suphel_zararli" dizisinin 2. sütununa "true" değeri yazılır. Sonrasında şüphe sayacı 1 arttırılır. 5. adımda, "supheli_kayit_defteri_denetle" metodu kullanılarak, yapı çalışıtırıldıktan sonra elde edilen kayıt defteri aktiviteleri incelenir. Yapı şüpheli veya zararlı kayıt defteri aktiviteleri gösteriyorsa "kayit_defteri_aktivitesi" değerine "true" değeri yazılır. Şüpheli kayıt defteri aktiviteleri gerçekleştirilen analizler sonucu belirlenmiş ve içeriği ilgili referansda verilmiştir[34]. 6. adımda, "kayit_defteri_aktivitesi" değeri "true" ise yani yapı şüpheli veya zararlı kayıt defteri aktiviteleri gerçekleştiriyorsa, şüphe sayacı 1 arttırılır. 7. adımda, "supheli_ag_aktivitesi_denetle" metodu kullanılarak, yapı çalıştırıldıktan sonra elde edilen ağ aktiviteleri incelenir. Yapı şüpheli veya zararlı ağ aktiviteleri gösteriyorsa "ag_aktivitesi" isimli "boolean" değere "true" değeri yazılır. 8. adımda, "ag_aktivitesi" değeri "true" ise yani yapı zararlı veya şüpheli ağ akitivitesi gerçekleştiriyorsa, şüphe sayacı 1 arttırılır. 9. adımda, "suphe_zararli" dizisinin 1. sütununa "suphe_sayaci" değeri yazılır. Dinamik analiz fazının son adımında ise "suphe_zararli" dizisi geri döndürülür.

Algoritma 3. Dinamik Analiz Fazı

DİNAMİK ANALİZ FAZI KABA KODU

Girdiler :
int suphe_sayaci = 0, **File** taranacak_dosya, **Object** suphe_zararli[]
Çıktı :
suphe_zararli

```
1.  boolean process_enjeksiyon = process_enjeksiyon_denetle()
2.  if (process_enjeksiyon)
        suphe_zararli[1] = true;
        suphe_sayaci++;
    end
3.  boolean sifreleyici = sifreleyici_denetle()
4.  if (sifreleyici)
        suphe_zararli[1] = true;
        suphe_sayaci++;
    end
5.  boolean kayit_defteri_aktivitesi = supheli_kayit_defteri_denetle()
6.  if (kayit_defteri_aktivitesi)
      suphe_sayaci++;
    end
7.  boolean ag_aktivitesi = supheli_ag_aktivitesi_denetle(taranacak_dos
                                                          ya)
8.  if (ag_aktivitesi)
       suphe_sayaci++;
    end
9.  suphe_zararli[0] = suphe_sayaci;
10. return suphe_zararli
```

DEZT algoritması içerisindeki "supheli_ag_aktivitesi_denetle" metodunun kaba kodu Algoritma 5' de verilmiştir.

Şüpheli ağ aktivitesi denetlenirken taranan dosyanın kurduğu bağlantılar incelenmektedir.

1. adımda, taranan dosya tarafından oluşturulan her bağlantı "virustotal.com" üzerinde analiz edilir. Eğer bağlantı zararlıysa "boolean zararli" değerine "true" değeri yazılır ve döngüden çıkılır. Değilse döngü çalışmaya devam eder.

2. adımda, taranan dosya tarafından oluşturulan bağlantıların zararlı olup olmadığı verisini içeren "boolean zararli" değeri geri döndürülür.

Algoritma 4. Şüpheli Ağ Aktivitesi Denetleme

ŞÜPHELİ AĞ AKTİVİTESİ DENETLE KABA KODU

```
Girdiler :
File taranacak_dosya Boolean zararli=false

Çıktı :
Boolean zararli
    [1]  for  int i=0; i < taranacak_dosya.baglantisayisi ; i++
                if (taranacak_dosya.baglanti[i] tara virustotal.com)
                      zararli=true
                      break;
                    end
              end
    [2] return zararli;
```

5. BULGULAR VE TARTIŞMA

5.1. Statik Analiz Bulguları

Analizi gerçekleştirilen dosya enjeksiyon zararlılarının, statik analiz sonucu elde edilen şüpheli dosya bölüm entropileri Tablo 1.' de verilmiştir.

Tablo 1. Dosya Enjeksiyon Zararlılarının Şüpheli Dosya Bölüm Entropileri

Zararli İsmi	.text	.rdata	.data	.gfids	.tls	.rsrc	.reloc	UPX0	UPX1
Sality	7.99								
Ramnit						7.63			
Bundpil									
ZeusBot									
Dridex						7.94			
Spatet						7.96			
Kryptik									
Virut						7.17			7.80
WannaCry						8.00			
Petya		6.99				8.00			

Tablo 1.'deki veriler göz önüne alındığında 10 zararlı türünden 7 tanesinde şüpheli dosya bölüm entropisi görülmektedir.

"PortexAnalyzer" programı kullanılarak statik analizi gerçekleştirilen zararlı yapıların, "Kryptik" zararlısı hariç 9 tanesinde, zararlı içeriğin paketli halde tutulduğu gözlemlenmiştir. Ayrıca "strings.exe" programı kullanılarak yapının içerisindeki sözel ifadeler incelendiğinde "Petya" ve "Kryptik" zararlıları hariç 8 zararlıda; ip adresleri, karmaşıklaştırılmış ifadeler, enjekte edilen zararlı başlığı vb. şüpheli metinsel ifadeler gözlemlenmiştir.

Tüm bu verilerden yola çıkılarak; şüpheli dosya bölüm entropisi, yapının paketli olup olmaması ve yapının şüpheli metinsel ifadeler içermesi gibi hususların zararlı yazılım tespitinde büyük önem taşıdığını söylemek mümkündür.

5.2. Dinamik Analiz Bulguları

Analizi gerçekleştirilen zararlıların dosya enjeksiyonu gerçekleştirirken kullanmış olduğu Windows fonksiyonları Tablo 2.' de verilmiştir. Tablo 2.' deki sütunlarda verilen sayıların içerdiği fonksiyonlar çizelgenin hemen altında verilmiştir.

Tablo 2. Zararlıların Dosya Enjeksiyonu Gerçekleştirirken Kullanmış Oldukları Windows Fonksiyonları

Zararli İsmi	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Sality	x	x	x	x	x															
Ramnit		x	x	x	x	X														
Bundpil		x	x	x	x															
ZeusBot	x		x	x	x	X														
Dridex			x	x	x	X														
Spatet		x	x	x	x															
Kryptik		x	x	x	x															
Virut	x	x			x		x	x	x											
WannaCry										x	x	x	x				x			
Petya						X					x	x	x	x	x	x	x	x	x	x

1. Process32Next, CreateToolhelp32Snapshot, Process32First
2. OpenProcess
3. VirtualAllocEx
4. WriteProcessMemory
5. CreateRemoteThread, ResumeThread, SetThreadContex
6. CreateProcessW, CreateProcessA
7. ZwCreateSection
8. ZwOpenSection
9. ZwMapViewOfSection
10. FindFirstFileExW, FindFirstFileW, FindFirstFileA
11. GetFileAttributesW
12. NtReadFile
13. NtWriteFile
14. GetLogicalDrives
15. GetDriveTypeW
16. CreateThread
17. FindNextFileW
18. wsprintfW
19. StrStrIW
20. CryptEncrypt

Algoritma 2.' de kullandığı Windows fonksiyonları verilen zararlılar tarafından gerçekleştirilen enjeksiyon aşağıda verilmiştir.

Sality zararlısı, çalıştığı anda içerisinde barındırdığı zararlı yapıyı işletim sistemi üzerindeki tüm processlere enjekte eder ve çalıştırır. Öncelikle Process32Next ile hedef process belirleyip, OpenProcess fonksiyonunu kullanarak handle değeri elde eder. Sonrasında hedef process üzerinde VirtualAllocEx fonksiyonunu kullanarak yer tahsis edip, WriteProcessMemory fonksiyonu ile zararlı yapıyı hedef belleğe yazar. Son olarak CreateRemoteThread fonksiyonunu kullanarak hedef proccess üzerine enjekte ettiği zararlı kodu çalıştırır.

Ramnit, Bundpil, Spatet ve Kryptik zararlıları, OpenProcess fonksiyonunu kullanarak hedef process ile bağlantı kurup handle değeri elde eder. Sonrasında hedef process üzerinde VirtualAllocEx fonksiyonu kullanarak yer tahsis edip, WriteProcessMemory fonksiyonu ile zararlı yapıyı hedef belleğe yazar. Son olarak SetThreadContext fonksiyonunu kullanarak hedef proccess üzerinde zararlı kodu çalıştırır.

ZeusBot ve Dridex zararlısı, çalıştırıldığında içerisinde barındırdığı zararlı yapıyı kendi oluşturduğu process üzerine enjekte eder ve çalıştırır. Bunu yaparken işlem maskeleme yöntemini kullanır. CreateProcessW fonksiyonunu kullanarak askı modda(Suspended) process oluşturur. Sonrasında hedef process üzerinde NtUnmapViewOfSection ve VirtualAllocEx fonksiyonu kullanarak yer tahsis edip, WriteProcessMemory fonksiyonu ile zararlı yapıyı hedef belleğe yazar. Son olarak SetThreadContext fonksiyonunu kullanarak hedef proccess üzerinde zararlı kodun çalıştırılacağı işaretçileri ayarlar.

Virut zararlısı, Process32First ve Process32Next fonksiyonlarını kullanarak hedef processlere erişim sağlar. Ardından OpenProcess fonksiyonunu kullanarak hedef process ile arasında bir handle değeri elde eder. ZwCreateSection kullanarak hedef process üzerinde yeni bir dosya bölümü oluşturur. Rastgele isimle oluşturulan bölüm nesnesine ZwOpenSection fonksiyonu kullanarak erişir. Sonrasında ZwMapViewOfSection fonksiyonu kullanarak oluşturulan bölümü biçimlendirir ve zararlı yazılımı içerisine kopyalar. Virut bu işlemden sonra hedef process üzerinde NtCreateProcessEx fonksiyonunu kullanarak, oluşturulan her yeni processe enfekte olmayı hedefler. NtSetInformationThread fonksiyonunu kullanarak dosyanın başlangıç noktasını virüsün kopyalandığı son sekmeye ayarlar. Son olarak CreateRemoteThread fonksiyonunu kullanarak zararlı yazılımı çalıştırır.

WannaCry zararlısı, ilk olarak FindFirstFileExW fonksiyonunu kullanarak belirli bir dizin altındaki tüm dosyaları bulur. GetFileAttributesW ve NtQueryDirectoryFile fonksiyonları ile belirlenen dosyanın özelliklerini alır. Sonrasında NtReadFile kullanarak dosya içerisindeki verileri okur. Son olarak NtWriteFile fonksiyonu kullanarak şifrelediği veriyi ".WNCRYT" uzantısıyla dosyanın üzerine yazar.

Petya zararlısı, öncelikle PathFileFindName fonksiyonunu kullanarak dosya veri yolunu elde eder. Daha sonra CreateFileW kullanarak diğer processlerle iletişim sağlayabileceği bir "Pipe" değeri oluşturur. Sonrasında CreateProcessW fonksiyonu ile komut istemini "C:\Windows\system32\cmd.exe /c schtasks/Create /SC once /TN /TR "C:\Windows\system32\shutdown.exe /r /f /ST HH:MM" kodu ile çalıştırır. Bu sayede sistemi, belirlenen saatde yeniden başlatılmaya ayarlar. Sonrasında şifrelemek üzere GetLogicalDrives fonksiyonunu kullanır ve sabit sürücüleri bulur. Son olarak FindNextFileW fonksiyonunu kullanarak şifrelenecek dosyaları bulur ve belirlenen dosya formatındaki dosyaları CryptEncrypt fonksiyonu kullanarak şifreler. Ayrıca Petya işletim sisteminin boot sektörüne kullanıcıya dosyalarının şifrelendiğine dair bilgi veren bir ekran görmesini sağlayacak veriyi WriteFile fonksiyonunu kullanarak yazar. Sürücülerin geri kalan sektörlerini "7" ile xor ederek üzerine yazar.

Analiz sonucu elde edilen bulgular önerilen algoritmanın test ve doğruluğunu kanıtlar niteliktedir. Örneğin Sality zararlısı DEZT algoritması ile işletildiğinde, "Taranan veri dosya enjeksiyonu yapmaktadır. Tehdit seviyesi %71.42' dir." olarak metinsel bir ifade ile geri döndürülmektedir.

6. SONUÇ VE ÖNERİLER

Bu çalışmada, 2017 ve 2018 yıllarında gerçekleştirilen siber saldırılarda en yaygın görülen dosya enjeksiyon zararlı türleri belirlenmiş ve dünya üzerinde etkili oldukları bölgeler oransal olarak verilmiştir. Ayrıca belirlenen dosya enjeksiyon zararlılarının statik ve dinamik analizi yapılarak, işlemlere (process) ve Windows işletim sistemi üzerinde bulunan çalıştırılabilir dosyalara etki etme ve yayılma şekilleri detaylı olarak irdelenmiştir. Belirlenen zararlılarının statik ve dinamik analizi sonucu elde edilen veriler kullanılarak, dosya enjeksiyon zararlısının tespiti için DEZT algoritması önerilmiştir.

Önerilen DEZT algoritması; zararlı dosya içerisinde taranan şüpheli metinsel ifadeler, şüpheli ip adresleri, şüpheli Windows fonksiyon çağrıları ve şüpheli kayıt defteri aktivite sayıları arttırılarak geliştirilmeye açık bir yapıdadır. Bu sayede gelecekte DEZT algoritması geliştirilip, dosya enjeksiyon zararlılarının kullanıldığı siber saldırılara karşı geniş çaplı bir tespit veya savunma sistemi kurulabilir.

KAYNAKLAR

[1] SYMANTEC, "Internet Security Threat Report Volume 22", Amerika, 2017.
[2] CANBEK, G., SAĞIROĞLU, Ş. "Kötücül ve Casus Yazılımlar : Kapsamlı Bir Araştırma", Gazi Üniv. Müh. Mim. Fak. Dergisi, 22, 1, 121-136, 2007.
[3] MICROSOFT, "Security Intelligence Report Volume 22", Amerika, 2017.
[4] http://www.virusradar.com/en/Win32_Sality/map, (erişim tarihi 10.02.2018).

[5] http://www.virusradar.com/en/Win32_Ramnit/map, (erişim tarihi 11.02.2018).
[6] http://www.virusradar.com/en/Win32_Bundpil/map, (erişim tarihi 14.02.2018).
[7] http://www.virusradar.com/en/Win32_Spy.Zbot/map (erişim tarihi 14.02.2018).
[8] http://www.virusradar.com/en/Win32_Dridex/map (erişim tarihi 16.02.2018).
[9] http://www.virusradar.com/en/Win32_Ramnit/map (erişim tarihi 19.02.2018).
[10] http://www.virusradar.com/en/Win32_Kryptik/map (erişim tarihi 19.02.2018).
[11] http://www.virusradar.com/en/Win32_Virut/map (erişim tarihi 19.02.2018).
[12] http://www.virusradar.com/en/Exploit.CVE-2017-0147/map (erişim tarihi 20.02.2018).
[13] SYMANTEC, "Internet Security Threat Report Ransomware 2017, Amerika, 2017.
[14] RİECK, K.,TRİNİUS, P., WİLLEMS, C., HOLZ, T., "Automatic Analysis Of Malware Behavior Using Machine Learning". Journal of Computer Security, 19(4), 639-668, 2011.
[15] EHRENFELD, J. M., "Wannacry, Cybersecurity And Health İnformation Technology: A Time To Act". Journal of medical systems, 41(7), 104, 2017.
[16] MOSER, A., KRUEGEL, C., KİRDA, E., "Limits Of Static Analysis For Malware Detection". Computer security applications conference ACSAC 2007. Miami, USA, 104,10-14 Aralık 2007.
[17] SNOW, K. Z., KRİSHNAN, S., MONROSE, F., PROVOS, N., "SHELLOS: Enabling Fast Detection and Forensic Analysis of Code Injection Attacks". USENIX Security Symposium, San Francisco, USA, 183-200, 8-12 Ağustos 2011.
[18] VİLLANİ, A., BALZAROTTİ, D., Dİ PİETRO, R., "The İmpact Of Gpu-Assisted Malware On Memory Forensics: A Case Study". Annual Digital Forensics Research Conference, Philadelphia, USA. 9-13 Ağustos 2015.
[19] GAZET, A., "Comparative Analysis Of Various Ransomware Virii". Journal in computer virology, 6(1), 77-90, 2010.
[20] CABAJ, K., GAWKOWSKİ, P., GROCHOWSKİ, K., OSOJCA, D.,"Network Activity Analysis Of Cryptowall Ransomware". Przeglad Elektrotechniczny, 91(11), 201-204, 2015.
[21] https://tr.wikipedia.org/wiki/Malware (erişim tarihi 19.02.2018).
[22] LYDA, R., HAMROCK, J., Using Entropy Analysis To Find Encrypted And Packed Malware. IEEE Security & Privacy, 5(2), 2007.
[23] https://github.com/katjahahn/PortEx (erişim tarihi 18.02.2018)
[24] https://www.hex-rays.com/products/ida (erişim tarihi 18.02.2018)
[25] https://docs.microsoft.com/en-us/sysinternals (erişim tarihi 16.02.2018)
[26] https://sourceforge.net/projects/regshot (erişim tarihi 15.01.2018)
[27] https://processhacker.sourceforge.io (erişim tarihi 13.01.2018)
[28] https://www.apimonitor.com (erişim tarihi 12.01.2018)
[29] https://x64dbg.com (erişim tarihi 14.01.2018)
[30] https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/gflags (erişim tarihi 14.01.2018)
[31] https://cuckoosandbox.org (erişim tarihi 12.01.2018)
[32] https://github.com/zararliyazilimanalizi/supheliaktiviteler2/blob/master/suphelistring.txt (erişim tarihi 23.02.2018).
[33] https://github.com/zararliyazilimanalizi/supheliaktiviteler2/blob/master/supheliwindowscagrilari.txt (erişim tarihi 25.02.2018).
[34] https://github.com/zararliyazilimanalizi/supheliaktiviteler2/blob/master/suphelikayitdefterleri.txt (erişim tarihi 25.02.2018).