TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ZARARLI YAZILIMLARIN KULLANDIĞI SANALLAŞTIRMA KARŞITI YÖNTEMLER VE ALINABİLECEK ÖNLEMLER

YÜKSEK LİSANS TEZİ

Osman Tufan TEKİN

Bilgisayar Mühendisliği Anabilim Dalı
Bilgi Güvenliği

Tez Danışmanı: Prof. Dr. Ali Aydın SELÇUK

ARALIK 2019

Fen Bilimleri Enstitüsü Onayı

……..
Prof. Dr. Osman EROĞUL
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

……
Prof. Dr. Oğuz ERGİN
Anabilim Dalı Başkanı

TOBB ETÜ, Fen Bilimleri Enstitüsü'nün 171111021 numaralı Yüksek Lisans öğrencisi **OSMAN TUFAN TEKİN**'in ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"ZARARLI YAZILIMLARIN KULLANDIĞI SANALLAŞTIRMA KARŞITI YÖNTEMLER VE ALINABİLECEK ÖNLEMLER"** başlıklı tezi **10 Aralık 2019** tarihinde aşağıda imzaları olan jüri tarafından kabul edilmiştir.

Tez Danışmanı :	**Prof. Dr. Ali Aydın SELÇUK** TOBB Ekonomive Teknoloji Üniversitesi	.....
Jüri Üyeleri :	**Doç. Dr. Osman ABUL (Başkan)** TOBB Ekonomi ve Teknoloji Üniversitesi	.....
	Doç. Dr. Sevil ŞEN Hacettepe Üniversitesi	.....

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

Osman Tufan Tekin

ÖZET

Yüksek Lisans

ZARARLI YAZILIMLARIN KULLANDIĞI SANALLAŞTIRMA KARŞITI TEKNİKLER VE ALINABİLECEK ÖNLEMLER

Osman Tufan Tekin

TOBB Ekonomi ve Teknoloji Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Bilgi Güvenliği

Danışman: Prof. Dr. Ali Aydın Selçuk

Tarih: Aralık 2019

Zararlı yazılım analizi yıllar içerisinde çok fazla yol kat etmiştir. Dinamik ve statik olarak ikiye ayırabileceğimiz analiz sürecinde, statik analizin davranışsal sonuçları verememesi sebebiyle dinamik analiz süreçleri gelişmiştir. Kum havuzları, otomatik dinamik analizi çok kısa sürelerde yapıp, analiste raporu sunabilmektedir. Bu sebeple günümüz zararlı yazılımları, tespit edilmemek veya tespit edilse de davranışlarının analiz edilme sürelerini uzatmak için analiz karşıtı yöntemler kullanmaya başlamıştır. Zararlı yazılımların bunu yaparken motivasyonu, her türlü kazançlarını arttırmaktır. Bu sebeple bütün analiz tekniklerinden kaçmak için her geçen gün yeni teknikler geliştirilmekte ve bu geliştirilen tekniklerin zararlı yazılımlar tarafından kullanımı da artmaktadır.

Sanal ortamların kullanımı, zararlı yazılım analizinde aktif bir yer almaktadır. İsteğe bağlı ortam oluşturulabilmesi, kişisel veri barındırmaması ve analiz adımları arasında kolay geçişler sağlanabilmesinden dolayı manuel analizler için vazgeçilmez bir hal almaktadır. Ayrıca kum havuzları için de aynı kolaylıklar sağlandığı için, onlar tarafından da tercih edilmektedir.

Zararlı yazılımlar, her iki analiz sürecinde de zararlı olduğunu belli etmemek için birçok yöntem kullanmaktadır. Bu yöntemlerden birisi olan sanallaştırma karşıtı yöntemi, kum havuzları içerisinde de sanal ortamlar kullanıldığı için, bütün analiz karşıtı yöntemler arasında çok geniş bir alan kaplamaktadır. Sanallaştırma için kullanılan ürünlerinin kullandığı sanallaştırma tekniklerinin farklı olması da zararlı yazılım geliştiricileri tarafında bu konu üzerine daha fazla çalışmak için motivasyon olmaktadır. Dolayısıyla birçok sanallaştırma karşıtı teknik şu anda zararlı yazılımlar tarafından kullanılmaktadır.

Sanal ortam tespiti için donanımsal ve yazılımsal olarak kontroller yapılabilmektedir. Donanımsal olarak sistem kaynakları veya donanım bilgileri içerisinde sanallaştırma platformunun imzaları kontrol edilebilir. Yazılımsal olarak da işletim sistemi içerisinden ulaşılabilen sanallaştırma platformlarına ait bilgiler bulunmaktadır. Özellikle sanal makine içerisinde Windows işletim sistemi varsa, Windows'a ait Kayıt Defteri, Windows Management Instrumantation gibi yapılar, içerisinde çok fazla sanallaştırma platformlarına ait imza barındırdıkları için sanal makine tespitini kolaylaştırmaktadır.

Bu çalışmada, sanallaştırma ortamı olarak en çok tercih edilen VMware, işletim sistemi olarak en çok tercih edilen Windows 10 üzerinde zararlı yazılımlar tarafından kullanılan sanallaştırma karşıtı yöntemler incelenmiştir. Bu sanallaştırma karşıtı yöntemlere karşı alınabilecek önlemler anlatılmıştır. Sanallaştırma karşıtı kullanılan bir başka yöntem ise Windows Management Instrumantation'dır. Çoğu kum havuzu tarafından önlem alınmamış bu yöntem incelenmiş ve kancalama tekniği kullanılarak bu yöntemi engelleme yolları anlatılmıştır. Windows Management Instrumantation için kullanılabilecek bir kancalama uygulaması geliştirilmiş ve bu uygulama ile sanallaştırma karşıtı tekniklere önlem alınabileceği gösterilmiştir.

Anahtar Kelimeler: Zararlı yazılım, Kum havuzu, Zararlı yazılım analizi, Sanallaştırma karşıtı teknikler, Windows Management Instrumantation

ABSTRACT

Master of Science

ANTI-VIRTUALIZATION TECHNIQUE USED BY MALWARE AND MEASURES THAT CAN BE TAKEN

Osman Tufan Tekin

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Department of Computer Engineering
Information Security

Supervisor: Prof. Dr. Ali Aydın Selçuk

Date: December 2019

Malware analysis has been improved over the years. In the analysis process, which can be divided into dynamic and static, dynamic analysis processes have developed due to the fact that static analysis cannot give behavioral results. Sandboxes can perform automatic dynamic analysis in very short time and present the report to the analyst. Therefore, today's malware has begun to use anti-analysis methods to be not detected or even if they detected they prolong the analysis times of their behavior. The motivation of malware to do this is to increase all kinds of returns. Therefore, in order to avoid all analysis techniques, new techniques are being developed and the use of these techniques by malware is increasing.

The use of virtual environments is actively involved in malware analysis. It is essential for manual analysis due to the ability to create an optional environment, does not contain any personal data and transition between analysis steps is simple. Also, they preferred by sandboxes since they provide same facilities.

Malware uses many methods to ensure that it is not malicious in both analysis processes. The anti-virtualization method, which is one of these methods, covers a

wide area among all anti-analysis methods since virtual environments are also used in sandboxes. The difference in the virtualization techniques used by the products used for virtualization is the motivation of malicious software developers to work more on this issue. Therefore, many anti-virtualization techniques are currently being used by malware.

Hardware and software controls can be performed for the detection of virtual environment. System resources or the signatures of virtualization platform can be checked by hardware. There is also information about virtualization platforms that can be accessed from within the operating system by software. Structures such as the Windows Registry, Windows Management Instrumentation facilitate the detection of virtual machines since they contain signatures belonging to many virtualization platforms, especially when there is a Windows operating system in the virtual machine.

In this study, anti-virtualization methods used by malware are examined on VMware the most preferred virtualization environment and on Windows 10 the most preferred operating system. Measures that can be taken against these virtualization methods are explained. Another method used against virtualization is Windows Management Instrumentation. This method, which was not taken precaution by most sandboxes, was examined and the ways to prevent this method by using hooking technique were explained. A hooking application has been developed that can be used for Windows Management Instrumentation and it has been shown that anti-virtualization techniques can be taken with this application.

Keywords: Malware, Sandbox, Malware analysis, anti-virtualization, Windows Management Instrumentation

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren danışman hocam Prof. Dr. Ali Aydın Selçuk'a, kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine, okul dışından gelip ders veren öğretim üyelerine, değerli katkılarından dolayı arkadaşım Atanur Serkan Elmasoğlu'na, destekleriyle her zaman yanımda olan aileme ve arkadaşlarıma çok teşekkür ederim. Ayrıca sağladığı burs için TOBB Ekonomi ve Teknoloji Üniversitesi'ne teşekkür ederim.

İÇİNDEKİLER

KISALTMALAR

API **:** Application Programing Interface
AV **:** Anti-Virüs
RAM **:** Random Access Memory
VM **:** Virtual Machine
WIN **:** Windows
WMI **:** Windows Management Instrumentation

RESİM LİSTESİ

Sayfa

1. GİRİŞ

Zararlı yazılımlar, bilgisayarlar üzerinden bilgi çalmak, sistemi çalışamaz hale getirmek, verileri şifrelemek, istihbarat toplamak gibi zararlı aktiviteleri gerçekleştiren yazılımlardır. Zararlı yazılımlar ile maddi kazançlar elde edilebildiği gibi devletler tarafından kullanılarak da istihbarat toplanabilmektedir. Bilgisayar sistemlerinin vazgeçilmez bir ihtiyaç olarak öne çıkması, zararlı yazılım sayısında da büyük bir artışa sebep olmuştur. AV-TEST raporuna göre 2019 yılı itibari ile toplamda 975 milyondan fazla zararlı yazılım bilgisayar sistemlerine zarar vermektedir [1]. Amaçlarına göre farklı çalışan zararlı yazılımlar geliştirilmekte ve bu durum çeşitliliği de arttırmaktadır. Trojan, spyware, adware gibi klasik zararlı yazılım çeşitleri dışında yeni nesil fidye zararlıları ve kripto madencilik zararlıları gibi geliştiricisine maddi kazanç sağlayan zararlı yazılımların sayıları her geçen gün artmaktadır [2] [3].

Zararlı yazılımlar, çoğunlukla Windows işletim sistemli bilgisayarları hedef almaktadır. Windows işletim sistemleri dünya üzerinde kullanım oranı en yüksek olan işletim sistem olarak öne çıkmaktadır [4]. Zararlı yazılım geliştiricileri de bu kullanım oranından dolayı Windows işletim sistemlerini tercih etmektedirler.

Zararlı yazılımların etkilerinin bu kadar fazla olması nedeniyle bunlara karşı önlem alma ihtiyacı doğmuştur. Zararlı yazılım tespiti için statik ve dinamik olmak üzere iki ana başlıkta çeşitli analiz yöntemleri geliştirilmektedir [5] [6]. Statik analiz süreçleri, zararlı yazılım çalıştırılmadan yapılan analiz türüdür. Yazılımın içerisinde yer alan ASCII karakterlerden oluşan kelimeleri, içerisinde yer alan Windows kütüphane isimleri gibi bilgilere bakılarak ve yazılımın işlemci üzerinde çalıştırdığı makine (assembly) kodları incelenerek yazılım hakkında bilgi toplanabilmektedir [5]. Ancak yazılım çalıştırıldığı zaman bunlara ek olarak birçok yeni bilgi eklenebilmektedir. Çalışma anında yeni kütüphaneler eklenebildiği gibi, statik analizde görülmemesi için kelimeler, parolalar, C&C bağlantı bilgileri (IP, domain adı, vb.) gibi veriler çalışma anında ortaya çıkabilmektedir. Dinamik analiz ise zararlı yazılım çalıştırılması ile ortaya çıkan davranışları, verileri inceleyerek gerçekleştirilmektedir.

Bu süreçte zararlı yazılım, güvenli bir ortam üzerinde çalıştırılarak bilgisayar üzerinde yaptığı işlemler çeşitli yöntemlerle ve araçlarla incelenebilmektedir. Dinamik analiz manuel olarak analistler tarafından yapılabildiği gibi otomasyon olarak kum havuzu (sandbox) ürünleri içerisinde de gerçekleştirilebilmektedir [6]. Manuel analiz sırasında çeşitli araçlar kullanılarak bilgisayar üzerindeki değişimler takip edilir. Otomasyon olarak ise yapılan sistem çağrıları takip edilerek kayıt altına alınır ve sergilenen davranışlar zararlı bir aktiviteye işaret ediyor ise zararlı yazılım teşhisi konulabilmektedir.

Statik analizin önüne, uygulanabilecek kod gizleme (obusfication) yöntemi ile geçilebilmektedir [7]. Bu yöntemler, yazılımın makine kodunun gizlenerek kod analizi yapılmasını engellemesi ile yazılım hakkında yapılan incelemeden bir sonuç alınamamasını sağlamaktadır. Bu tekniklerden dolayı dinamik analizin önemi artmaktadır. Dinamik analizde yazılımın davranışları incelenebildiği için yapılan kod gizleme gibi teknikler önemini kaybetmektedir. Bu sebeplerden dolayı dinamik analiz çalışmalarına daha fazla ağırlık verilmeye başlanmıştır [7].

Dinamik analiz manuel olarak yapılabileceği gibi otomasyon şeklinde kum havuzu denilen ürünler ile de yapılabilmektedir. Kum havuzları, zararlı yazılımın davranışlarını güvenilir bir ortam içerisinde takip ederek, sistem içeresinde yaptığı çağrıları yakalayabilmektedirler [8]. Yakalanan bu çağrılar, sonrasında analiste rapor olarak sunulur.

Analiz yeteneklerinin bu kadar artmasıyla zararlı yazılım yazarları tarafından anti-analiz teknikleri geliştirilmeye başlandı. Anti-analiz teknikleri ile saldırganlar, zararlı yazılımın analiz edilmesini engellemek ve elde edilen kazancı arttırmayı amaçlamaktadır. Statik analize karşı kullanılan anti-disassembly tekniği, yazılımın makine kodunu karıştırarak yapılan işi gizlemeye çalışan bir yöntemdir [9]. “Packer” olarak bilinen araçlar ile yine makine kodu şifrelenerek, statik analizin önüne geçilmeye çalışılmaktadır [10]. Manuel dinamik analiz ortamlarına karşı geliştirilen teknik ise ayıklayıcı karşıtı (anti -debug) olarak adlandırılmaktadır. Ayıklayıcı karşıtı yöntemi, analiz esnasında çalışan zararlı yazılımın, ayıklayıcı uygulamaları ile incelenmesi sırasında bıraktığı izleri kontrol ederek analiz edildiğini anlamasına yardımcı olmaktadır [11]. Son olarak sanallaştırma karşıtı teknikler hem manuel dinamik analizleri hem de kum havuzları üzerinde otomatik dinamik analizlerine karşı zararlı yazılımların kendilerini saklamak için kullandıkları yöntemdir [12].

Dinamik analizlerin tamamı sanal ortamlar üzerinde gerçekleştirilmektedir. Sanal ortamlar hem zararlı yazılımın verebileceği zararlardan etkilenmemek hem de analiz sürecini kolaylaştırmak için kullanılmaktadır [13]. Ayrıca kum havuzlarının da içerisinde sanal ortamlar kullanılmaktadır [8]. Sanallaştırma karşıtı yöntemler kum havuzları için de kullanılabilmektedir.

Zararlı yazılım analizinde en etkili yöntem dinamik analizdir. Dinamik analiz yönteminde kullanılan sanal ortamlar ise saldırganların zararlı yazılımlar içerisinde en çok önlem almaya çalıştığı mekanizma olarak ortaya çıkmaktadır. Bu sebeple çoğu zararlı yazılım, sanallaştırma karşıtı teknikleri kullanılarak ortam tespiti yapmaya çalışmaktadır [12]. Analistler ve kum havuzları ise bu analiz karşıtı yönteme karşı önlem almaya çalışmaktadır. Analiz ortamındaki sanal platform imzaları, sanal ortamın kaynak ayarlamaları gibi zararlı yazılımlar tarafından kontrol edilen özellikler değiştirilerek sanallaştırma karşıtı tekniklere karşı önlemler alınmaya çalışılmaktadır. Ancak son zamanlarda fazlasıyla kullanılmaya başlanan Windows Management Instrumentation (WMI) kullanılarak yapılan sanallaştırma karşıtı tekniğini çoğu kum havuzu yakalayamamaktadır [14].

Manuel dinamik analize karşı uygulanan yöntemler dışında, kum havuzlarının otomatik yaptığı analizlere karşı kum havuzu karşıtı (anti-sandbox) teknikler de uygulanmaktadır. Bu teknikler içerisinde insan etkileşimleri, zaman atakları ve kancalama tabanlı çalışan kum havuzlarına karşı da sistem çağrılarında araya girilip girilmediğinin kontrolünü yapan teknikler uygulanmaktadır [15].

1.1 Tezin Katkıları

Bu yüksek lisans tezinde, zararlı yazılımlarda kullanılan, VMware sanallaştırma platformuna karşı kullanılabilecek sanallaştırma karşıtı teknikler incelenmiş, bunlara karşı nasıl önlemler alınabileceği hakkında bilgiler verilmiştir. Vmware'ın konfigürasyon ayarları kullanılarak donanımsal kontrollere karşı sıkılaştırmaların nasıl yapılması gerektiği, yazılımsal olarak VMware ve Windows işletim sistemi bileşenleri içerisindeki sanal ortam belirteçlerinin nasıl yok edilebileceği üzerine yapılan çalışmalar gösterilmiştir.

Zararlı yazlımlar tarafından sanallaştırma karşıtı yöntemler için kullanılan ve bir Windows bileşeni olan WMI ile sanal ortamın nasıl tespit edilebildiği aktarılmıştır.

WMI kullanılarak uygulanan sanallaştırma karşıtı tekniklerin, dört tanesi yüksek maliyetli, iki tanesi ücretsiz toplamda altı kum havuzu incelendiğinde, kontrol edilmediği tespit edilmiştir. Bu tespitleri kontrol edip engellemek için kancalama yöntemi kullanan bir araç geliştirilmiştir. Geliştirilen araç, kancalama tabanlı çalışan kum havuzları içerisinde kullanılabilir bir teknikle geliştirilmiştir.

1.2 Literatür Özeti

Cohen (1984) virüsün ilk tanımını "zararlı aktivitesini içerisinde bulunduğu sistem üzerindeki başka programlara da bulaştırarak kendi kopyasını oluşturan program" olarak yapmıştır. Cohen, bir virüsün amacını, "ağ üzerindeki diğer bilgisayarlara da bulaşmak için başka uygulamaların içerisine kendi kopyasını ekleyerek sistem içerisinde yayılmak" olarak anlatmaktadır. Bu davranışsal modeli de örnek kodlar ile göstermektedir [16].

Kurzban (1989) bilgisayar virüslerine ve solucanların tanımlarını birbirinden ayırmış, virüslerin dosyaları değiştirerek yayıldığın solucanların ise ağdan yayıldığını anlatmıştır. Ayrıca çalışmasında günümüzdeki anti-virüs programlarının kullandığı dosya bütünlüğü kontrolünden bahsederek virüslere karşı önlem alınabileceğinden bahsetmiştir [17].

Chess ve diğ. (1994) yazdıkları patentte, otomatik virüs taramasının sistemde yer alan dosya içerikleri karşılaştırılarak yapılabileceğini anlatmışlardır. Yaptıkları sistem, dosyaların içerisindeki parçaların, zararlı olarak gördükleri yazılım ile benzerlik gösterip göstermediğini kontrol ederek tespit yapabilmektedir. Bu yöntem güncel kullanılan anti-virüs programlarının kullandıkları yöntemin temelini oluşturmaktadır [18].

Moser ve diğ. (2007) yaptıkları kod gizleme çalışmaları ile statik analiz sonucunda zararlı yazılım tespitinin yapılamayacağını anlatmışlardır. Kodun içerisinde yer alan verilerin gizlenmesi ve kod takibinin yapılamaması için makine kodunun gizlenmesi teknikleri ile statik analize karşı kullanılabilecek yöntemleri göstermişlerdir [7].

Gadhiya ve Bhavsar (2013) yaptıkları çalışmada statik analiz için kullanılabilen metotları sıralamışlardır. Statik analizin, zararlı yazılımın özet (hash) bilgisi ile incelemesi ile, string kontrolü ile, dosya formatı belirlemede, zararlı yazılımın şifreli olup olmadığının kontrol edilmesinde ve tersine çevrilerek makine kodu incelemesi

ile yapılabileceğini aktarmışlardır. Kaynak kod incelemesi yapılamayacağı için statik analizin limitleri olduğundan bahsetmişlerdir [5].

Willems ve diğ. (2007) zararlı yazılım sayısındaki artıştan dolayı analiz süreçlerinin bütün zararlı yazılımlara yetemeyeceğini söylemiş ve otomatik bir sistem gerekliliğinden bahsetmişlerdir. Bunun sonucunda geliştirdikleri kancalama tabanlı (hooking-based) bir kum havuzu olan CWSandbox ile sistem çağrılarında araya girip Windows üzerindeki davranışları yakaladıktan sonra aralarında eleme yaparak bir rapor oluşturduklarını çalışmalarında göstermişlerdir. Ayrıca CWSandbox, Kancalama tabanlı çalışan bir kum havuzu olduğu için sanallaştırma karşıtı yöntemleri de manipüle ederek kendisini saklamaya çalışan bir sistem kullanmaktadır [19].

Dinaburg ve diğ (2008) zararlı yazılım analiz ortamının şeffaflığı için beş gereklilik olduğunu söylemiştir. Bunlar; analiz ortamının, zararlı yazılımın çalıştığı ortamdan daha yüksek yetkide olması, analiz ortamının varlığından kaynaklanan yan etkiler (side-efect) zararlı yazılımın çalıştığı ortamın yetkisinden daha yüksek bir yetkide gerçekleştirilmesi, zararlı yazılım ile aynı buyruk (instruction) kümesine sahip olmak, sistem içerisinde hata giderme ve zaman yapısının zararlı yazılım ile aynı olmasıdır [20].

Boris Lau ve Vanja Svajcer (2008) DSD-TRACER adında geliştirdikleri uygulama ile zararlı yazılımın yaptığı sistem çağrılarını bir yapı ile tutarak içerisinde VMware üzerinde uygulandığı bilinen sanallaştırma karşıtı (anti-VM) tekniklerin olup olmadığını kontrol edebildiklerini çalışmalarında göstermişlerdir. Ayrıca çalışmalarında yaklaşık 400 tane zararlı yazılımın %2.13'ü içerisinde sanallaştırma karşıtı tekniğin kullanıldığını açıklamışlardır [21].

Dinaburg ve diğ. (2008) yapılan manuel dinamik analizinde kullanılan araçların zararlı yazılımlara yakalanmalarından dolayı, donanım sanallaştırılması ile yapılacak bir ortamın analiz anında yakalanmayı en aza indireceğinden bahsetmişlerdir. Çalışmalarında, Xen hypervisor'ü üzerinde kullanılabilen ve kendi geliştirdikleri Ether uygulamasını önermişlerdir. Ether, Intel işlemcileri üzerinde çalışan buyrukları izleyerek, hafızaya yazılan verileri takip ederek ve sistem çağrılarını gözlemleyerek uzaktan zararlı yazılım analizi için gerekli bilgileri toplayabilmektedir. Ayrıca Ether,

bütün bu işlemler çok fazla kalabalık yaratacağı için kısıtlamalar ile daha net bir sonuç elde edebilmektedir [22].

Kruegel ve diğ. (2009) yaptıkları çalışmada yaklaşık bir milyon zararlı yazılım örneğinde ANUBIS ortamına karşı geliştirilen anti-ANUBIS tekniklerinin %0,3 oranında kullanıldığını söylemişlerdir [23].

Paleari ve diğ. (2009) "red pill" tekniği olarak bilinen ortam tespit yöntemini kullanarak oluşturulan programlar ile işemci buyruklarının çalıştıktan sonra eax ve ebx yazmaçlarına (register) döndükleri değerler doğrultusunda gerçek bir işlemci mi yoksa emüle edilmiş bir işlemci mi olduğu tespiti yapabildiklerini çalışmalarında göstermişlerdir [24].

Pék ve diğ. (2011) gerçek bilgisayar kaynaklarını kullanarak yapılan sanallaştırma tekniğine "hosted virtual machines" tanımını vermişlerdir. Bu yöntem ile "guest" olarak tanımlanan sanal makineler, gerçek makinenin donanımlarını paylaşarak bir işletim sistemi çalıştırmaktadır. "Hosted" sanallaştırma tekniği, VMware'ın sanallaştırma için kullandığı sistemdir [25]. Ayrıca açık kaynak kodlu Cuckoo Sandbox da aynı yapıyı kullanarak analizleri ana bilgisayar üzerinden takip edebilmektedir [26] [27].

Lindorfer ve diğ. (2011) bir analiz ortamı olarak kullanılan ANUBIS sisteminin kullanıcı adının USER olmasından analiz ortamı tespiti yapıldığından bahsetmişlerdir. Ayrıca "C:\exec\exec.exe" klasörü içerisinin kontrol edilerek ve çalışan işlemler (process) içerisinde yer alan "popupkiller.exe" kontrolleri ile zararlı yazılımların ortam analizi yapabildiğini göstermişlerdir [28].

Yoshioka ve diğ. (2011) zararlı yazılım analiz sürecinde analiz ortamının internetsiz olmasının, zararlı yazılım analiz edildiğini direkt olarak fark edilmesine sebep vereceğini söylemişlerdir. İnternetli bir analiz ortamı olması gerektiği ve sonrasında dosyalar üzerindeki değişikliklerin, kayıt defteri (registry) değişikliklerinin ve internet üzerinden yapılan haberleşmenin kontrol edilmesinin daha doğru olacağından bahsetmişlerdir [29].

Chen ve diğ. (2016) sanallaştırma karşıtı (anti-VM) ve ayıklayıcı karşıtı (anti-debuging) yönteminin kullanım oranını görmek için 2009-2014 yılları arasında çıkan 17,283 zararlı yazılım üzerinde yaptıkları çalışmalarında, erken süreçte geliştirilen yazılımlarda anti teknik kullanımın çok olmadığından ancak zaman ilerledikçe analiz

karşıtı tekniklerin çoğu zararlı yazılım içerisinde kullanılmaya başlamasına dikkat çekmişlerdir [30].

Yokayama ve diğ. (2016) zararlı yazılımlar tarafından insan etkileşiminin kontrol edilerek analiz ortamı içerisinde olup olmadığını kontrol ettiklerini, bunu yaparken bir dosyanın ya da verinin kopyalın tekrar yapıştırılma süresine, bir dosyanın erişim süresine, fare hareketlerine ve klavyede yazma hızlarına bakıldığını ve bunların kontrolü için geliştirdikleri SANDPRINT üzerine çalışmalarını anlatmışlardır [31].

Carpenter ve diğ. (2007) çalışmalarında, kullanılan in buyruğu ile VMware üzerinde çalışan analiz ortamını tespit edebileceklerini göstermişlerdir. Bu tespiti önlemek için VMware'ın konfigürasyon dosyasına belgelenmemiş ayarlarını kullanmış ve başarılı olduklarını göstermişlerdir [32].

Sun ve diğ. (2008) yaptıkları çalışmada, zararlı yazılımların kullandığı VMware kontrollerine karşı uygulanan yöntemleri IDA uygulaması üzerinde çalışabilen bir plug-in olan REFORM ile yakalayıp, çalışma zamanında yaptıkları değişimler ile atlatabildiklerini göstermişlerdir [13].

Egele ve diğ. (2012) yaptıkları araştırmada, zararlı yazılım analizinin otomatikleştirmek için Windows'ta kullanılan Windows API ile yapılan servis çağrılarının kancalanarak (hooking) bir uygulamanın davranışlarının izlenebileceğinden bahsetmişlerdir. Bu işlemi, bir emulatör üzerinde yaparken dışardan takip edilebileceğini, sanal makine içerisinde yaparken içerden takip edilip dışarıya aktarılabileceğini anlatmışlardır [6].

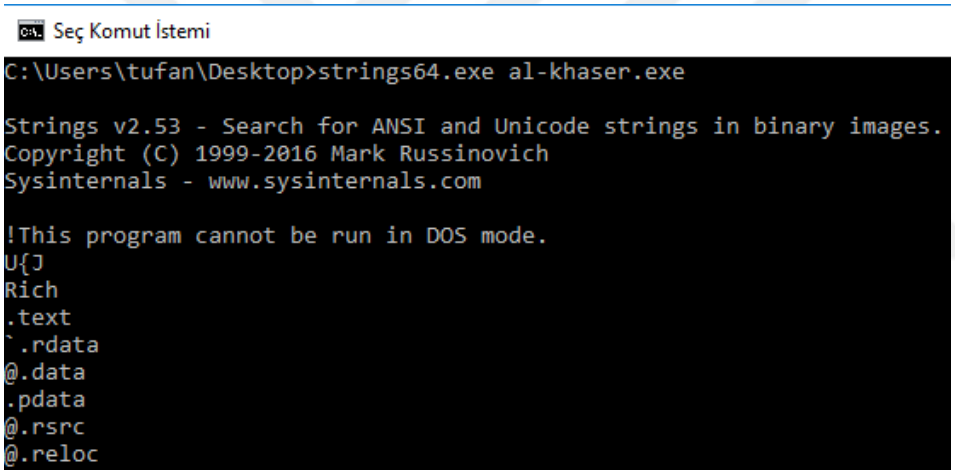
Yapılan literatür taramasında, zararlı yazılımların ilk ortaya çıkışlarıyla birlikte analiz edilme süreçlerinin başladığı görülmektedir. Analiz süreçlerine karşılık, farklı türlerde analiz karşıtı teknikler türemiştir. Bu analiz karşıtı tekniklere önlem almanın, analiz süreçlerinin daha verimli hale getirilmesi için önemli hale geldiği anlaşılmaktadır.

Ayrıca, yapılan çalışmalar incelendiğinde görülmektedir ki, tez içerisinde özellikle vurgulanacak olan Windows Management Instrumentation kullanımı ile uygulanan sanallaştırma karşıtı tekniklerin kullanımı ile ilgili hem de bu teknikleri önlemek için yapılmış bir çalışma yer almamaktadır.

1.3 Zararlı Yazılım Analiz Süreçleri

Zararlı yazılım analiz süreci genel olarak, statik analiz ve dinamik analiz, olarak ikiye ayrılmaktadır. Ancak yapılan işlemin zorluğuna göre süreç dört parçada incelenebilir. Bu dört parça temel statik analiz, temel dinamik analiz, gelişmiş statik analiz ve gelişmiş dinamik analiz olarak isimlendirilmektedir.

Statik analizde amaç, yazılım çalıştırılmadan elde edilen bilgiler ile zararlı bir yazılım olup olmadığını anlayabilmektir. Bu analiz türünde kullanılan araçlar ile yazılım içerisinde yer alan bilgiler kontrol edilebilir. Ayrıca yazılımın özet değerleri internet üzerinde araştırılarak, daha önceden tespit edilmiş bir yazılım olup olmadığı kontrol edilebilir. Temel statik analizde kullanılan araçlardan "strings" aracı, yazılım içerisinde yer alan ASCII karakterli değerleri göstermektedir. Strings aracının çıktısı Resim 1.1'de görülmektedir.



Resim 1.1: Strings aracı çıktısı

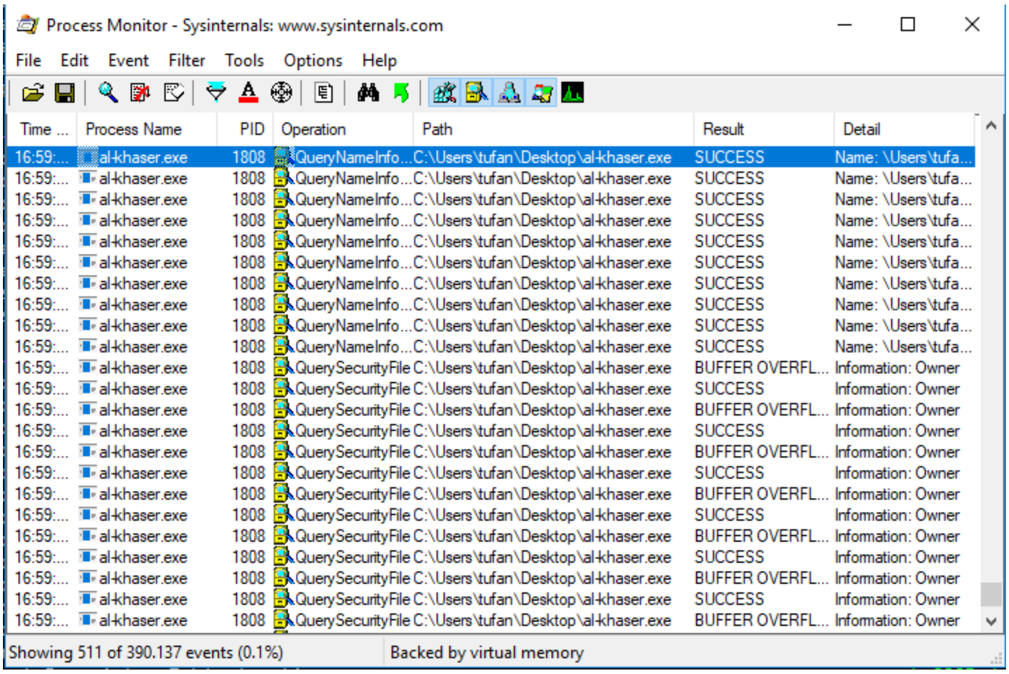
Statik analizde kullanılan ve Windows kütüphane bilgileri, içerisinde yer alan kaynaklar gibi özelliklerin kontrol edilebildiği uygulamalar ile statik analiz süreci işletilebilir. Bu araçlara örnek olarak "CFF Explorer" gösterilebilir [33].

Yazılımın makine kodunun (assembly) incelenmesi için kullanılan "IDA Pro" da gelişmiş statik analiz sürecinde kullanılan bir araçtır [34]. Bu araç ile işlemci seviyesinde çalışan kodlar analiz edilebilmektedir. Bu araç ile makine kodu incelenerek zararlı kod parçası barındırıp barındırmadığı kontrol edilebilmektedir.

Statik analiz için kullanılan bu araçlar zararlı yazılımın davranışları hakkında bilgi veremediği için yetersiz kalmaktadır [7]. Ayrıca anti-dissassembly yöntemi ve

kullanılan packer araçları ile statik analiz sonucunda hiçbir çıktı elde edilmeyebilmektedir [9].

Dinamik analizde amaç, zararlının güvenli bir ortam üzerinde çalıştırılarak, davranışlarının takip edilmesi ile zararlı bir aktivite yapıp yapmadığının kontrol edilmesine dayanmaktadır. Bu analiz türünde açılan process'ler, registry değişiklikleri ve yapılan sistem çağrıları çeşitli araçlar ile takip edilebilmektedir. "Procmon" aracı ile yapılan sistem çağrılarının görüntülenmesi Resim 1.2'de görülmektedir [35].



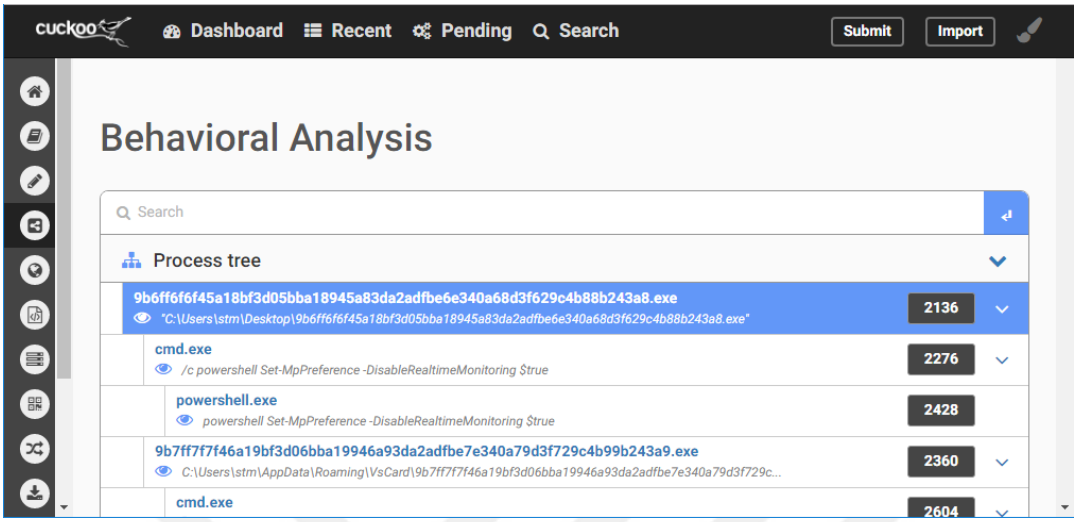
Resim 1.2: Procmon uygulaması

Resim 1.2'de görülen çıktıda çalışan bir uygulamanın "Procmon" aracı ile adım adım yaptığı sistem çağrıları görülmektedir. Bu sistem çağrıların sonuçları, zamanı ve ayrıntıları bu uygulama ile görüntülenebilir. Dinamik analiz sürecinde işlemlerin incelenmesi için kullanılan önemli bir araçtır.

Ayrıca manuel dinamik analiz ile ayıklayıcı (debugger) araçları kullanılarak makine kodları tane tane takip edilip davranış analizi yapılabilmektedir. "OllyDbg", "x64dbg", "windbg" gibi araçlar ile bu işlem gerçekleştirilebilir. Bu araçlar ile yapılan işlemler gelişmiş dinamik analiz süreci içerisinde yer almaktadır.

Davranış analizleri ayrıca otomasyon şekilde kum havuzları (sandbox) ile de yapılabilmektedir. Bu analizler için farklı yöntemler kullanan kum havuzlarının ortak noktası, sanal ortamlar üzerinde analiz yapmalarıdır. Zararlı yazılımın davranışları

yakalayıp kendi oluşturdukları davranış setleriyle kıyaslayan kum havuzları zararlı yazılım tespitini yapabilmektedir. Resim 1.3'te açık kaynak kodlu Cuckoo Sandbox'ın rapor örneği görülmektedir [26].



Resim 1.3: Cuckoo Sandbox rapor çıktısı

Resim 1.3'te görülen rapor örneğinde, çalıştırılan bir uygulamanın, "cmd.exe"yi çalıştırdığı, oradan da "powershell.exe"yi çalıştırdığı görülmektedir. Her bir adım ayrıntılı bir şekilde incelenebilmektedir. Analiz otomatik gerçekleştirildiği için çok kısa zaman içerisinde bütün işlemler içerisinde gerçekleştirilen davranışlara erişmek mümkündür.

Dinamik analiz yöntemleri ile zararlı yazılım analizinde daha doğru sonuçlara ulaşmak mümkündür. Zararlı yazılımlar bu başarı yüzdesinden dolayı dinamik analize karşı daha fazla önlem almaktadır. Manuel dinamik analize karşı ayıklayıcı tespiti yapabilen ayıklayıcı karşıtı (anti-debug) yöntemleri bu önlemlere bir örnektir [11]. Ancak zararlı yazılımlar en büyük savunmayı bütün dinamik analizin gerçekleştirildiği sanal ortamlara karşı yapmaktadır [12].

Dinamik analiz için kullanılacak sanal ortamın oluşturulması ve zararlı yazılıma yakalanmayacak şekilde ayarlanması çok önemlidir [36]. Sandbox'lar genel olarak sanal ortam tespitine karşı tedbirler almaya çalışmaktadır. Bu tedbirler içerisinde üzerinde inceleme yapılan kum havuzlarının da yapmadığı görülen, WMI kontrolleri kullanılarak uygulanan sanallaştırma karşıtı yöntemler bu tez içerisinde, üzerinde özellikle durulan konu olacaktır.

2. ZARARLI YAZILIMLAR TARAFINDAN KULLANILAN ANALİZ KARŞITI YÖNTEMLER

Analiz karşıtı yöntemler, güncel zararlı yazılımlarda aktif olarak kullanılmaktadır. Amaç, analizi engellemek ve sistemdeki kalma sürelerini arttırmaktır. Dört ana başlıkta analiz karşıtı yöntemler incelenebilir. Bunlar, tersine çevirici karşıtı (anti-disassembly) yöntemler, ayıklayıcı karşıtı (anti-debugging) yöntemler, sanallaştırma karşıtı (anti-VM) yöntemler, kum havuzu karşıtı (anti-sandbox) yöntemlerdir.

2.1 Tersine Çevirici Karşıtı Yöntemler

Tersine çevirici karşıtı yöntemler, zararlı yazılımların makine diline yanlış çevrilmesine yol açarak, statik analizi engellemeyi amaçlamaktadır. Bu yöntem kullanıldığında, makine diline çevrilmiş zararlı yazılım kodu, yanlış ve anlamsız bir hale bürünmektedir.

Tersine çevirici karşıtı yöntemlerde genel olarak amaç, makine kodunu gizlemektir. Bu yöntemlerden bir tanesi, "packer" denilen, makine dilini gizleme uygulamaları ile bütün kodu şifreleyerek makine kodu tamamen anlamsız hale getirilebilir. Ancak bu yöntem bilinen "packer" uygulamaları ile yapıldıysa kodlar geri açılabilmektedir. "Packer" uygulamaları genel olarak, lisanslı ürünlerin uygulamalarında kullanılmaktadır.

Diğer bir yöntem, zararlı yazılım makine diline çevrildikten sonra, araya konulan ve tersine çevirme(disassembly) akışını değiştiren kod parçaları ile yapılmaktadır. Tersine çevirici olarak kullanılan yazılımlar, kod parçalarını akışa göre tersine çevirmektedir. Yani kullanılan "jump" buyrukları (instruction) ile gidilebilen yerler tersine çevrildikten sonra kalan yerlerde kullanılmayan kod parçaları ve yazılım içindeki veriler yalnızca adres olarak tutulmaktadır. Bu sistem zararlı yazılımlar tarafından bilindiği için, akışı bozan kod parçası eklendiği zaman, işlem kodları (opcode) tamamen kayarak anlamsız bir hale gelmektedir [9].

Bunlardan en çok kullanılan yöntem, araya bir yanlış byte eklenerek uygulanan yöntemdir. Bu yöntemde yanlış byte öncesi bir jump işlem kodu ile eklenen byte'tan bir sonra gelen byte'a zıplanmaktadır. Ancak yanlış byte bir işlem kodu gibi göründüğü için asıl çalıştırılması gereken işlem bir veri gibi görünür ve o adımdan sonra gelen bütün kod tersine çevirici uygulama tarafından yanlış oluşturulmaya başlanır ve böylece uygulamayı makine kodlarıyla incelemek olanaksız hale gelir. Bu yöntem ile statik analiz ve ayıklama (debug) yöntemi kullanılarak yapılan dinamik analiz süreçleri engellenmeye çalışılmaktadır.

2.2 Ayıklayıcı Karşıtı Yöntemler

Ayıklayıcı karşıtı (anti-debugging) yöntemler, zararlı yazılımların makine diline çevrilmiş halleri adım adım incelenirken, bu incelemeyi yakalamak için kullanılmaktadır. Zararlı yazılımlar, incelendiğini tespit ederlerse ya kendilerini kapatmaktadır ya da zararlı davranışlarını gizlemektedirler [11].

Uygulanan en temel ayıklayıcı karşıtı yöntem, WIN32 kütüphanesinde yer alan, "IsDebuggerPresent" fonksiyonudur [37]. Bu fonksiyon çağrıldığı zaman, işletim sistemine ait Process Envirement Block (PEB) içerisinde yer "BeingDebugged" değişkeni kontrol edilerek, ayıklayı uygulama varlığı tespit edilebilmektedir. Tersine çevrilen bir zararlı yazılım içerisinde bu fonksiyon çağrısı görüldüğünde bu çağrıyı atlayarak, bu yöntemi atlatmak mümkündür.

Bir başka yöntem, "IsDebuggerPresent" fonksiyonunu zararlı yazılımın ana fonksiyonu başlatılmadan kontrol edilmesidir. Windows uygulamaları yer alan Thread Local Storage (TLS) "callback" içerisinde, ana fonksiyon başlamadan ayıklayıcı kontrolü yapılabilmektedir.

Windows içerisinde tutulan "NTGlobalFlag" yapısı ile de ayıklayıcı uygulama kontrolü yapılabilmektedir. Bir uygulamanın ayıklayıcı uygulama ile açılıp açılmadığını bu yapı ile kontrol etmek mümkündür.

Windows üzerinde kullanılan ayıklayıcı uygulamalar ile koyulan durak noktası (breakpoint), makine koduna da yansımaktadır. Makine kodu içerisinde durak noktası koyulan buyruklar (instrcution) bir yere kopyalanarak yerine bazı işlem kodları (opcode) koyulur ve ayıklayıcı o işlem kodu üzerine geldiği zaman durması gerektiğini anlar. Ardından kopyalanan buyruk çalıştırılarak devam eder.

Ayıklayıcıya ait durak noktası işlem kodları makine kodu içerisinde görülmesi ile ayıklayıcı tespiti de yapılabilmektedir.

Anlatılan yöntemler dışında birçok ayıklayıcı uygula karşıyı yöntemler mevcuttur. Genel yapısı ile ayıklayıcı kontrolü yapan zararlı yazılımların bu kontrolü, ilk açılış anında, zararlı davranışını yapmadan önce ya da çalışma anının herhangi bir zamanında yapması mümkün olduğu için manuel analiz zorlaştıran bir teknik olarak öne çıkmaktadır.

2.3 Kum Havuzu Karşıtı Yöntemler

Kum havuzu karşıtı yöntemler (anti-sandbox), zararlı yazılımların analizden kaçmak için kullandığı bir başka yöntemdir. Zararlı yazılımlar bu yöntem ile içerisinde bulunduğu ortamın bir kum havuzu mu yoksa gerçek bir bilgisayar ortamı mı olduğunu tespit edebilmektedir. Sanal ortam kullanan kum havuzlarından dolayı kullanılan bazı teknikler, sanallaştırma karşıtı yöntemlerle de benzerlik göstermektedir.

Kum havuzlarında kullanılan sanallaştırma ortamının tespitini yapmak için zararlı yazılımlar, öncelikli olarak donanımsal kontroller yapmaktadırlar. Bunlarla beraber Bölüm3. yer alan tekniklerin hepsi kum havuzları için de uygulanmaktadır [15].

Sanallaştırma ortamı tespiti dışında, piyasada yer alan ve bilinen kum havuzlarının imzalarının bulunduğu alanlar kontrol edilir. Kancalama tabanlı (hooking-base) bir kum havuzunun sanal ortamda yer alan ajan programı kontrolü bu tekniklerden bir tanesidir. Zararlı yazılımlar, ajan yazılımı ismiyle veya yaptığı işle tespit edebilmektedirler.

Bu yöntemler dışında kullanılan insan davranışı tespiti ile zararlı yazılımlar, ortam içerisinde insan hareketlerini takip etmektedirler. Fare, klavye hareketleri, dosyaların açılıp kapanması, internet bağlantısı gibi normal bir insanın bilgisayarı kullanırken yaptığı davranışların varlığı kontrol edilir. Bu davranışların olmaması durumunda zararlı yazılımlar, kum havuzu tespitini yapabilmektedirler.

En çok kullanılan yöntem ise zaman ataklarıdır. Bu yöntemin amacı, kısa sürede analiz yapmaya çalışan kum havuzlarının, sistem zamanında yaptığı oynamaların takibi ile yapılabilmektedir. Bu yöntem için WIN32 kütüphanesinde yer alan “GetTickCount” fonksiyonu ile uygulanmaktadır. “GetTickCount” fonksiyonu

sistem başlatıldığı andan itibaren geçen süreyi dönmektedir [38]. İlk başta fonksiyon çağrılır ve sayı değeri kaydedilir ve ardından zararlı yazılım uyku moduna geçer. Uyku süresi kum havuzlarının analiz süresine göre belirlenmektedir. Genel olarak kum havuzları, standart analiz süresi olarak 180 saniye belirlemektedir. Zararlı yazılım, bu süreden çok daha fazla bir süre uyku modunda kalır. Eğer kum havuzu uyku modundaki bir zararlı yazılıma karşı önlem almadıysa, analiz süresi aşar ve analiz süresince bir zararlı davranış gözlemlenemez. Uyku modu için önlem alındıysa ve uyku süresi üzerinde bir oynama yapıldıysa, zararlı yazılımlar "GetTickCount" fonksiyonun tekrar çağırır ve ilk aldığı değer ile kıyaslama yapar. Uyku süresinden daha kısaysa, yine kum havuzları içerisinde olduğunu tespit edebilir.

Kum havuzu karşıtı yöntemler ile zararlı yazılımlar, kum havuzu ortamlarını, kum havuzlarında kullanılan zaman değiştirme yöntemlerini ve insan davranışlarını kontrol ederek analiz ortamı içerisinde olup olmadığını tespit edebilmektedir. Bu yöntemlere karşı kum havuzu ürünleri, çok yoğun bir çalışma gerçekleştirmektedir.

2.4 Sanallaştırma Karşıtı Yöntemler

Sanallaştırma karşıtı yöntemler (anti-VM) ile zararlı yazılımlar analiz ortamı için kullanılan sanal ortamları tespit etmeye çalışmaktadır. Ayrıca kum havuzu ortamlarını tespit etmek için de kullanılan sanallaştırma karşıtı yöntemler bulunmaktadır. Bu yöntemlerin ayrıntıları, Bölüm 3. anlatılmaktadır.

3. ZARARLI YAZILIMLARIN KULLANDIĞI SANALLAŞTIRMA KARŞITI YÖNTEMLER

Son çıkan zararlı yazılım örneklerinde, dinamik analiz süreçlerinde yakalanmamak için kullanılan sanallaştırma karşıtı yöntemlerin (anti-VM) sayısı arttığı görülmektedir. Sanallaştırma ortamları, hem analistlerin manuel olarak dinamik analiz yaptığı ortamlar için hem de kum havuzları (sandbox) için vazgeçilmez bir analiz ortamıdır. Zararlı yazılımlar, sistem içerisindeki geçirdikleri süreyi arttırabilmek için analiz sistemlerinden kendileri korumaya çalışmaktadır. Korunma yöntemlerinin en kompleksi ise sanal ortamlara karşı uygulanan sanallaştırma karşıtı tekniklerdir.

Sanallaştırma karşıtı yöntemler, donanımsal ve yazılımsal olarak ortam kontrolleri içermektedir. Donanımsal olarak işlemci, RAM, disk, sürücü kontrolleri yapılarak gerçek bir bilgisayar ile sanal bir bilgisayarı birbirinden ayırmak mümkündür. Bu kontroller çeşitli şekillerde yapılabilmektedir.

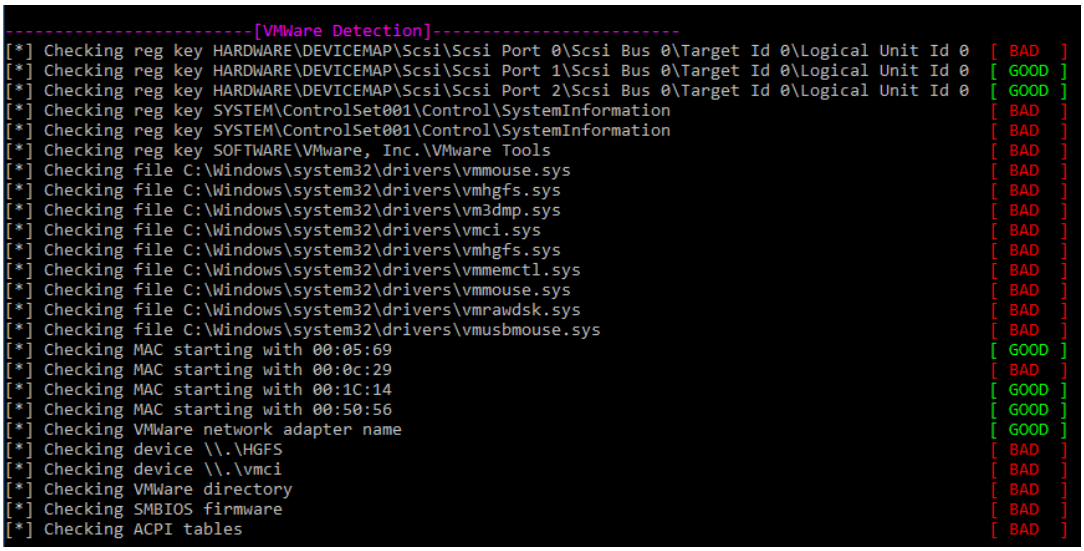
Yazılımsal olarak sanallaştırma ortamına ait birçok bilgiye işletim sistemi içerisinden ulaşılabilmektedir. İşletim sistemi içerisinde yer alan dosya isimleri, çalışan işlemler (process) isimleri, servis isimleri ve kayıt defterinde (registry) yer alan bilgiler ile çalışılan ortam hakkında bilgi edinmek mümkündür.

Kayıt defteri, Windows işletim sistemlerinde yapılan bütün işlemlerin tutulduğu bir yapıdır. Kayıt defterinde işletim sistemine ait verileri tutan, işletim sistemi üzerinde çalışan uygulama ve hizmetleri depolayan hiyerarşik bir veritabanıdır. İşletim sistemi çalıştığı her an kayıt defteri de çalışabilir durumda olduğundan, işletim sistemi ve diğer programlar kalıcı verilerini burada saklamaktadırlar [39]. Sanallaştırma platformları da çoğu verisini kayıt defteri içerisinde sakladığı için sanallaştırma karşıtı yöntemi olarak kayıt defteri kayıtlarını kontrol etmek aktif olarak kullanılmaktadır.

Dinamik zararlı yazılım analizinde en çok tercih edilen sanallaştırma ortamı VMware'dır. Stabil çalışıyor olması, kolay konfigüre edilmesi ve kullanım

kolaylığından dolayı VMware çoğu analistin ve bazı kum havuzlarının tercihidir [13]. VMware, çalıştığı gerçek makine ile üzerinde yer alan fiziki kaynakları ortak kullanarak çalışmaktadır. Ayrıca sadece zararlı yazılım analizi için kullanılmayan VMware, içerisinde kendisine ait birçok iz bulundurmaktadır. Bu durum zararlı yazılımların ortam tespiti yapmak için işini kolaylaştırmaktadır.

Github üzerinde yer alan al-khaser uygulaması ile birçok sanallaştırma ortamı ve kum havuzu ortamı, analiz karşıtı tekniklere karşı ne kadar hazırlıklı olduğunu test edebilmektedir [40]. Al-khaser uygulamasının VMware testleri Resim 3.1'de görülmektedir. Al-khaser uygulaması içerisinde yer alan bazı kontrol parametreleri, bu tez kapsamında kullanılmıştır.



Resim 3.1: al-khaser uygulaması

Resim 3.1'de görülen al-khaser uygulamasının yaptığı testlerde, VMware ortamını tespit ettiği değerler "BAD" olarak belirtilmiş, yakalanmayan değerler "GOOD" olarak belirtilmiştir. Al-khaser, kayıt defteri, sürücü kontrolleri, klasör kontrolü, MAC adresi kontrolü gibi birçok testi yapabilmektedir.

Bu bölümde zararlı yazımlar tarafından uygulanan sanallaştırma karşıtı yöntemler detaylı bir şekilde anlatılacaktır. Bu tez içerisinde yapılan çalışmaların hepsi VMware içerisinde yer alan Windows 10 sanal makine ile gerçekleştirilmiştir.

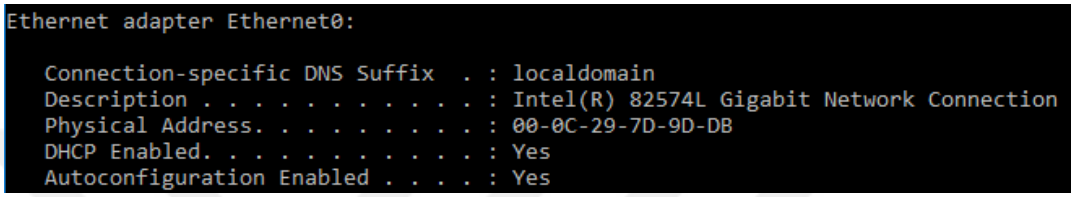
3.1 MAC Adresi Kontrolü

VMware, kontrol kolaylığı açısından oluşturduğu bütün sanal makinelere verdiği MAC adreslerinin ilk üç byte'ını yalnızca dört farklı şekilde vermektedir. Bu durum

zararlı yazılımlar tarafından analiz ortamı kontrolü olarak kullanılmaktadır. Bu dört MAC adresi aşağıdaki gibidir.

1. 00:0C:29* (\x00\x0C\x29)
2. 00:1C:14* (\x00\x1C\x14)
3. 00:50:56* (\x00\x50\x56)
4. 00:05:69* (\x00\x05\x69)

Resim 3.2'de MAC adresi kontrolü için cmd üzerinde "ipconfig \all" komutu çalıştırıldığında görülen değer, listede yer alan "00:0C:29" değeri gibi başlamıştır.



Resim 3.2: "ipconfig \all" komutu çıktısı

"00:0C:29" ile başlayan değer, listede yer alan bir başlangıç olarak görülmektedir. BU kontrol ile VMware tespiti yapılabilmektedir.

3.2 cpuid Buyruğu Kontrolü

x86 işlemci mimarisinde yer alan cpuid buyruğu (instruction) işlemci hakkında bilgiler dönmektedir. cpuid buyruğu EAX yazmacı (register) "1" olarak ayarlanıp çalıştırıldıktan sonra ECX yazmacı kontrol edilir. ECX yazmacının 31. biti "1" ise sanal ortam tespiti yapılmış olur. Bu değer gerçek makinelerde "0"dır. Resim 3.3'de örnek kod görülmektedir.

```
bool test1() {
    bool IsUnderVM = false;
    __asm {
        xor     eax, eax
        inc     eax
        cpuid
        bt      ecx, 0x1f
        jc      UnderVM
        NotUnderVM :
        jmp      NopInstr
            UnderVM :
        mov     IsUnderVM, 0x1
            NopInstr :
            nop
    }

    return IsUnderVM;
}
```

Resim 3.3 cpuid buyruk testi (1) kod parçası

Bir diğer yöntemde ise, EAX yazmacına “40000000” değeri ayarlandıktan sonra çağrılan cpuid sonrası ECX ve EDX yazmaçlarına yazılan değerler kontrol edilir. ECX değeri “0x4D566572” ve EDX değeri “0x65726177” ise VMware tespiti yapılmış olur. Resim 3.4’te örnek kod parçası Resim 3.5’te iki cpuid buyruk testinin de çıktısı görülmektedir.

```
bool test2() {

    bool IsUnderVM = false;

    __asm {
        xor     eax, eax
        mov     eax, 0x40000000
        cpuid
        cmp ecx, 0x4D566572
        jne NopInstr
        cmp edx, 0x65726177
        jne NopInstr
        mov IsUnderVM, 0x1
        NopInstr:
        nop
    }
    return IsUnderVM;

}
```

Resim 3.4: cpuid buyruk testi (2) kod parçası

```
C:\Users\tufan\Desktop>CPUID_test.exe
CPUID Test 1
CPUID Test1 Sonuc : VMware testip edildi
CPUID Test 2
CPUID Test2 Sonuc : VMware testip edildi
```

Resim 3.5: cpuid testleri çıktıları

3.3 in Buyruğu Kontrolü

VMware, sanal makine ile üzerinde yer aldığı gerçek makine arasında iletişim kuran bir mekanizmaya sahiptir. Bu mekanizmayı x86 mimarisinde yer alan in buyruğu (instruction) ile kontrol etmek mümkündür. Bu yöntem “VMware magic number” olarak da adlandırılmaktadır [13]. Öncelikle, EAX yazmacına (register) “VMware magic number”ı olarak bilinen “VMXh” değeri yazılır. EBX yazmacına da bu değere eşit olmayacak şekilde bir değer yazılır. EDX yazmacına da VMware’e ait port değerine denk gelen “VXh” değeri yazılır. Sonrasında IN buyruğu çağrılır. Gerçek bilgisayarlar üzerinde, klavye, fare gibi donanımlardan gelen girdiler ile kullanılan IN buyruğu çağrıldığında yetki hatası verilir ve çıktıya ulaşılamaz. Ancak VMware üzerinde çalıştırıldığında hata vermez ve EBX değerine “VMXh” değeri yazılırsa,

yazılım VMware üzerinde olduğunu anlar. in buyruğu ile ilgili kod Resim 3.6'da ve kodun çıktısı Resim 3.7'de görülmektedir.

```
unsigned int    a = 0;
__try {
    __asm {
        push eax
        push ebx
        push ecx
        push edx
        mov eax, 'VMXh'
        mov ecx, 14h
        mov dx, 'VX'
        in eax, dx
        mov a, eax
        pop edx
        pop ecx
        pop ebx
        pop eax
    }
}
__except (EXCEPTION_EXECUTE_HANDLER) {}
```

Resim 3.6: in buyruk test kodu

```
C:\Users\tufan\Desktop>Vmware_ins_test.exe

[+] Test 1: VMware "in instruction" test
Sonuc : VMware icerisindeyim!

[+] Test 2 : VMware "in instruction" test
Sonuc : VMware icerisindeyim!
```

Resim 3.7: in buyruk kod çıktısı

3.4 VMware Klasör İsmi Kontrolü

VMware, Windows işletim sistemlerinde, VMware Tools adını verdiği yardımcı uygulamalar kullanmaktadır. Bu uygulamaları ise Windows üzerinde "Program Files" altında "VMware" klasörü içerisinde tutmaktadır. Zararlı yazılımlar bu klasörün varlığını kontrol ederek sanal ortam içerisinde olup olmadığını kontrol edebilmektedirler. WIN32 kütüphanesinde yer alan "GetFileAtrributes" fonksiyonu ile "VMware" dosyasının bilgileri alınmaya çalışılır. Eğer dosya bilgileri boş gelir ise fonksiyon, "INVALID_FILE_ATTRIBUTES" hatası döner. Bu hatanın dönüp dönmediği kontrol edilerek sanal makine tespiti yapılabilmektedir. Resim 3.8'de bu klasörün varlığını kontrol eden örnek bir kod parçası ve Resim 3.9'da kodun çıktısı görülmektedir.

```
int main()
{
    DWORD dwAttrib = GetFileAttributes(L"C:\\Program Files\\VMware");

    if (dwAttrib != INVALID_FILE_ATTRIBUTES ) {
        printf("\"C:\\Program Files\\VMware\" Boyle bir dosya var\n");
    }
    else {

        printf("\"C:\\Program Files\\VMware\" Boyle bir dosya yok");
    }
}
```

Resim 3.8: VMware klasörü kontrolü örnek kod parçası

```
C:\Users\tufan\Desktop>VMWareDirTest.exe
"C:\Program Files\VMware" Boyle bir dosya var
```

Resim 3.9: VMware klasörü kontrol kodu çıktısı

Bu yöntem dışında yine WIN32 kütüphanesinde yer alan "CreateFile" fonksiyonu kullanılarak bu kontrol yapılabilmektedir. "CreateFile" fonksiyonu, "CREATE_ALWAYS" bayrağı (flag) ile çağrıldığı zaman eğer dosya varsa "ERROR_FILE_EXISTS" hatası döner ve böyle klasörün varlığı görülmüş olur. Bu yöntem ile de VMware klasörü kontrolü Sanallaştırma karşıtı bir yöntem olarak uygulanabilmektedir.

3.5 Kayıt Defteri Değerleri Kontrolü

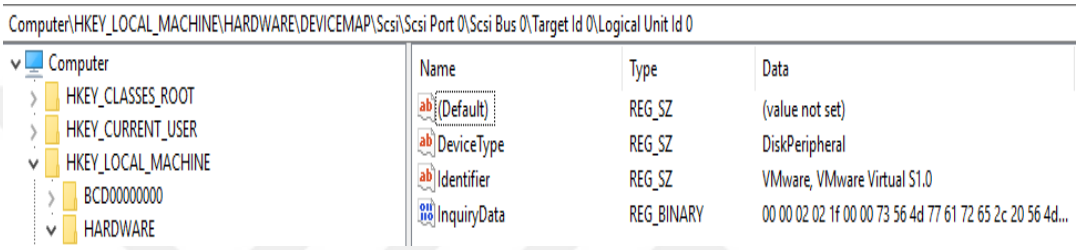
VMware, Windows içerisinde kullandığı uygulamalardan ve emüle ettiği donanımlardan dolayı Kayıt Defteri (registry) üzerinde birçok iz bırakmaktadır. Bu izleri kontrol ederek zararlı yazılımlar, sanal bir ortam içerisinde olup olmadığını tespit edebilmektedir. Kayıt Defterinde sistem üreticisi bilgisine, sistem ismine, yüklü programlara, donanım isimlerine bakılarak VMware imzası aranabilir.

WIN32 kütüphanesinde yer alan "RegOpenKey" fonksiyonu ile öncelikle adresteki kayıt defteri objesine erişilir. Ardında "RegQueryValueEx" fonksiyonu ile istenilen parametreye ait değere bakılır [41].

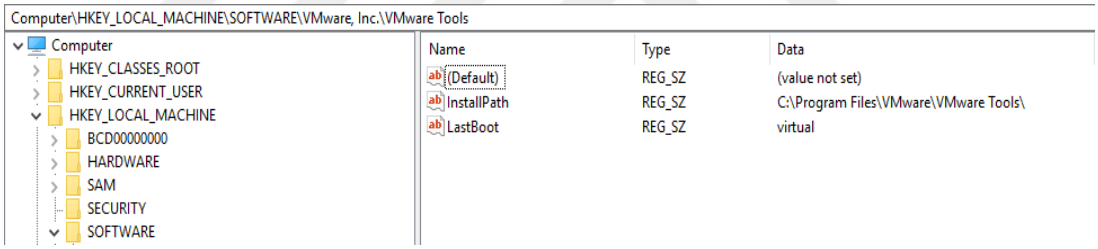
Bakılabilecek donanım ile alakalı Kayıt Defteri değerlerinden bir tanesi, "HKLM\HARDWARE\DEVICEMAP\Scsi\Scsi Port 0\Scsi Bus 0\ Target Id 0\ Logical Unit Id 0" adresi altında yer alan "Identifier" değeridir. Sanallaştırma ortamında bu değer içerisinde VMware imzası bulunmaktadır. Resim 3.10'da regedit uygulaması ile bu adres üzerindeki değer görülebilmektedir.

Kayıt Defteri içerisinde, VMware Tools kontrolü için bakılacak adres değeri, "HKLM\SOFTWARE\VMware, Inc.\VMware Tools" değerinin kendisidir. Böyle bir adresin varlığı kontrol edilerek, VMware imzası bulunabilir. Resim 3.11'de yer alan regedit uygulaması ile bu değer görülebilmektedir.

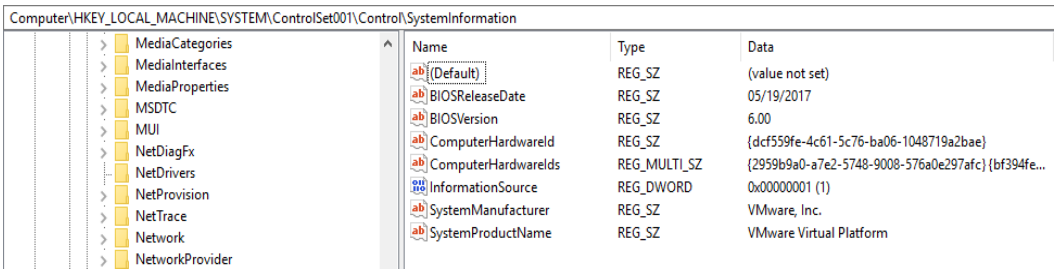
Kayıt Defteri ile üretici ve sistem bilgileri de kontrol edilebilmektedir. Kayıt defteri içerisinde yer alan "HKLM\SYSTEM\ControlSet001\Control\SystemInformation" adresinde yer alan "SystemManifacturer" ve "SystemProductName" parametreleri içerisinde yer alan değerlerde VMware imzası bulunmaktadır. Resim 3.12'de bu değerle ilgili regedit uygulamasının görüntüsü görülmektedir.



Resim 3.10: regedit uygulaması ile donanım bilgisi kontrolü



Resim 3.11: regedit uygulaması ile yüklü yazılım kontrolü



Resim 3.12: regedit uygulaması ile üretici ve sistem bilgileri kontrolü

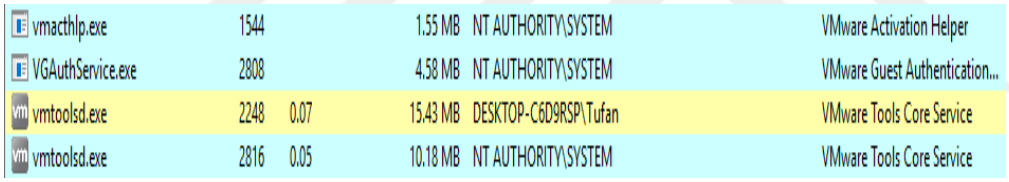
3.6 İşlem İsimleri Kontrolü

VMware içerisinde çalışan sanal işletim sistemlerinde, üzerinde çalıştığı bilgisayar ile iletişimi daha rahat sağlamak ve fiziksel ortamı daha efektif kullanabilmek için arkada tarafta birçok uygulama çalışmaktadır. Bu uygulamaların kontrol edilmesi, sanal ortam tespiti için kullanılan yöntemlerden bir tanesidir. Bu uygulamaların

listesi aşağıda yer almaktadır. Listede yer alan işlemler VMware tarafından başlatılmaktadır.

- VMwareService.exe
- VMaretrat.exe
- VMwareuser.exe
- Vmacthlp.exe
- vmwaretoolsd.exe
- VGAuthService.exe

WIN32 API ile sistemde çalışan işlemlere ulaşılabilmektedir. Öncelikle uzun bir dizi oluşturulur. Ardından "EnumProcesses" fonksiyonu ile, oluşturulan diziye sırasıyla işlem ID'leri yazılmaktadır. İşlem adını öğrenmek için ise öncelikle bir HANDLE oluşturulur ve içerisine "process ID" ile işleme bağlanır. Ardından "GetModuleBaseName" fonksiyonu ile işlemin adına ulaşılabilmektedir. Listede yer alan işlemler de böyle kontrol edilerek bir sanallaştırma karşıtı tekniği olarak kullanılabilmektedir [42].



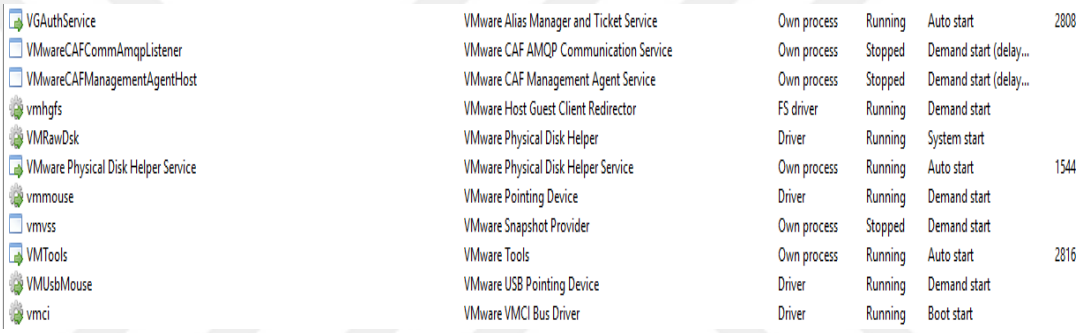
Resim 3.13: Process Hacker ile VMware işlemleri kontrolü

Resim 3.13'te Process Hacker uygulamasıyla sanal makine üzerinde tespit edilen VMware uygulamaları görülmektedir. Listede verilen işlemlerden bazılarının sanal makine üzerinde çalıştığı görülmektedir.

3.7 Servis İsimleri Kontrolü

Windows, başlangıç anında çalıştırmaya başladığı servisler ve ihtiyaç dahilinde kullanılan servisler ile kullanıcıya ara yüz üzerinde bir şey göstermeden çoğu sistemsel işleri yapabilmektedir. Ayrıca başka uygulamalar da Windows üzerinde servis çalıştırıp açık bırakabilmektedir. Bunlardan bir tanesi de VMware'dır. VMware hem sanal makine üzerinde hem de sanal makinenin çalıştığı ana bilgisayar üzerinde çalıştırdığı servislerle sanal ortam kontrolü yapabilmektedir. Bu servisler sayesinde zararlı yazılımlar sanal ortam tespiti yapabilmektedir. VMware imzası

taşıyan servis listesi şöyledir; VMTools, Vmhgfs, VMMEMCTL, Vmmouse, Vmrawdsk, Vmusbmouse, Vmvss, Vmscsi, Vmxnet, vmx_svga, Vmware Tools, Vmware Physical Disk Helper Service, Vmci, vm3dmp, vm3dmp-debug, vm3dmp-stats, vm3dmp-loader, Vmware CAF Management Agent Host, Vmware CAF Comm Amqp Listener, VMware Physical Disk Helper Service. Bu servisleri yakalamak için zararlı yazılımlar, WIN API kütüphanesi içerisinde yer alan "EnumServiceStatus" fonksiyonu ile servis listesini ENUM_SERVICE_STATUS yapısı ile bir dizi içerisinde tutar. Ardından o dizi üzerinde yer alan her bir elemanın "ServiceName" değişkeni kontrol edilir ve VMware'e ait bir servis olup olmadığı kontrol edilir. Resim 3.14'te VMware üzerinde çalıştırılan Process Hacker uygulaması ile görülen VMware servisleri görülmektedir.



VGAuthService	VMware Alias Manager and Ticket Service	Own process	Running	Auto start	2808
VMwareCAFCommAmqpListener	VMware CAF AMQP Communication Service	Own process	Stopped	Demand start (delay...	
VMwareCAFManagementAgentHost	VMware CAF Management Agent Service	Own process	Stopped	Demand start (delay...	
vmhgfs	VMware Host Guest Client Redirector	FS driver	Running	Demand start	
VMRawDsk	VMware Physical Disk Helper	Driver	Running	System start	
VMware Physical Disk Helper Service	VMware Physical Disk Helper Service	Own process	Running	Auto start	1544
vmmouse	VMware Pointing Device	Driver	Running	Demand start	
vmvss	VMware Snapshot Provider	Own process	Stopped	Demand start	
VMTools	VMware Tools	Own process	Running	Auto start	2816
VMUsbMouse	VMware USB Pointing Device	Driver	Running	Demand start	
vmci	VMware VMCI Bus Driver	Driver	Running	Boot start	

Resim 3.14: Process Hacker ile VMware servis kontrolü

3.8 Sürücü Kontrolü

VMware, ana makinenin donanımsal bileşenlerini sanal makinelere yansıttığı için, ara geçişlerde birçok sürücü (driver) kullanmaktadır. Bu sürücüler Windows üzerinde sistem dosyalarında yer aldığı için dosya kontrolleri yapılarak bulunabilir. Dosya kontrolü yapılırken, WIN API kütüphanesinde yer alan "CreateFile" fonksiyonu kullanılmaktadır. "CreateFile" fonksiyonu çağrılırken kullanılan "CREATE_NEW" bayrağı (flag) kullanılır. Bu bayrak, eğer dosya zaten var ise "ERROR_FILE_EXIST" hatası döner. Zararlı yazılım içerisinde bu fonksiyon çağrılarak bu hatanın dönüp dönmediği kontrol edilir. Bu yöntem ile listede yer alan sürücüler zararlı yazılımlar tarafından taranarak sanal ortam tespiti yapılabilmektedir.

- system32\drivers\vmmouse.sys
- system32\drivers\vmhgfs.sys
- system32\drivers\vm3dmp.sys

- system32\drivers\vmci.sys
- system32\drivers\vmhgfs.sys
- system32\drivers\vmmemctl.sys
- system32\drivers\vmmouse.sys
- system32\drivers\vmrawdsk.sys
- system32\drivers\vmusbmouse.sys

3.9 WMI Kullanılarak Yapılan Kontroller

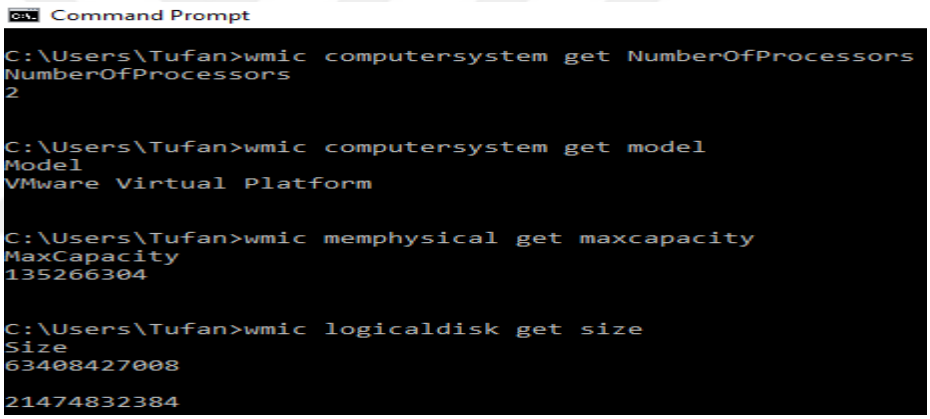
Sanallaştırma karşıtı yöntemi olarak kullanılan bir diğer teknik ise Windows Management Instrumentation'dır (WMI). WMI, Windows tabanlı işletim sistemlerinde yönetim verileri ve bu verilerin işlemleri için kullanılan bir alt yapıdır. Aynı zamanda uzak bilgisayarlardaki yönetim görevlerini otomatikleştirmek için WMI komut dosyaları veya uygulamaları yazılabilir. Üzerinde yapılabilen sorgularla, Windows işletim sistemlerine ait bilgilere erişim sağlanabilmektedir [14]. Erişim sağlanan bilgiler arasında Sanallaştırma karşıtı teknikler için kullanılabilecek veriler de bulunmaktadır. Disk boyutu, RAM kapasitesi gibi donanımsal bilgilerin yanı sıra, işletim sisteminin üzerinde çalıştığı sistem bilgileri de sorgulana-bilmektedir.

WMI kullanılarak disk kapasitesi sorgulanabilmektedir. Günümüz bilgisayarlarında disk kapasitesi en az 500 GB'dır. Ancak oluşturulan sanal makinelerde gerçek kaynak kullanımı çok olmaması için analistler ve kum havuzları kapasiteyi daha az belirlemektedir. Al-khaser uygulamasında kapasite 80 GB olarak belirlenmiştir. 80 GB disk kapasitesinin altında bir makine üzerinde sanal makine tespiti yapılabilmektedir. Resim 3.15'te disk boyutu, wmic uygulamasının çıktısı olarak görülmektedir.

WMI ile sorgulanıp sanal ortam tespiti için kullanılan bir diğer özellik ise bilgisayar hafızasıdır. Günümüz bilgisayarlarında efektif bir kullanım için minimum hafıza 8 GB olmalıdır. Disk kullanımıyla aynı olarak sanal makinelere, fiziksel kaynaktan fazla kaybetmemek için analistler ve kum havuzları tarafından 1 veya 2 GB hafıza verilmektedir. Bu durum zararlı yazılımlar tarafından WMI ile kontrol edilerek sanallaştırma karşıtı yöntem olarak kullanılabilmektedir. Al-khaser uygulamasında bu değer 1 GB olarak ayarlanmıştır. Resim 3.15'te hafıza kapasitesi, wmic uygulamasının çıktısı olarak görülmektedir.

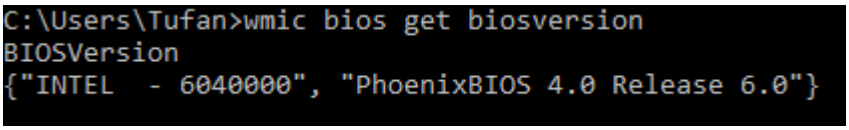
İşlemci çekirdek sayısı da WMI ile sorgulanıp sanal ortam tespiti için kontrol edilen bir değerdir. Analiz ortamı için çok fazla çekirdek ihtiyacı duyulmadığı için analistler ve kum havuzları tarafından bir işlemcili, bir çekirdekli sanal makineler kurulup bu makineler üzerinde analiz yapılmaktadır. Bu durum sanal makineler tarafından kontrol edilerek sanal ortam tespiti yapılabilmektedir. Al-khaser, minimum çekirdek sayısını iki olarak belirleyerek sanal ortam kontrolünü bu değere göre yapmaktadır. Resim 3.15'te işlemci sayısı, wmic uygulamasının çıktısı olarak görülmektedir.

WMI ile donanımsal kapasiteler dışında, bilgisayar modeli hakkında da bilgi alınmasını sağlamaktadır. VMware üzerinde çalışan Windows makinelerde bu sorgu yapıldığında cevap olarak "VMware Virtual Platform" dönmektedir. Bu sorgulama ile zararlı yazılımlar, sanal ortam tespiti yapabilmektedir. Resim 3.15'te bilgisayar modeli wmic uygulamasının çıktısı olarak görülmektedir.



Resim 3.15: wmic işlemci sayısı, disk kapasitesi, hafıza kapasitesi ve bilgisayar modeli çıktıları

WMI ile ayrıca BIOS sorguları da yapılabilmektedir. VMware, BIOS sistemi olarak kendi kendisine ait olan "PhoneixBIOS" kullanmaktadır. WMI ile BIOS versiyonu sorgulanarak bu bilgiye ulaşılabilir. Bu bilgi ile zararlı yazılımlar sanal ortamda çalıştığını tespit edebilir. BIOS versiyon sorgulaması, wmic uygulamasının çıktısı olarak Resim 3.16'da görülmektedir.



Resim 3.16: wmic BIOS versiyon sorgulama çıktısı

WMI ile sorgulanabilen bir diğer bileşen ise ekran kartlarıdır. Ekran kartı bilgileri içerisinde toplamda beş yerde VMWare imzası bulunmaktadır. Bu değerler, ekran sanallaştırılmasında kullanılan sanal sürücüler için ayarlanmış değerlerdir.

```
C:\Users\Tufan>wmic PATH Win32_videocontroller get description
Description
VMware SVGA 3D


C:\Users\Tufan>wmic PATH Win32_videocontroller get caption
Caption
VMware SVGA 3D


C:\Users\Tufan>wmic PATH Win32_videocontroller get adaptercompatibility
AdapterCompatibility
VMware, Inc.


C:\Users\Tufan>wmic PATH Win32_videocontroller get name
Name
VMware SVGA 3D


C:\Users\Tufan>wmic PATH Win32_videocontroller get videoprocessor
VideoProcessor
VMware Virtual SVGA 3D Graphics Adapter
```

Resim 3.17: wmic ile ekran kartı bilgileri sorgulama çıktısı

Resim 3.17'de görülen wmic uygulaması çıktılarında, "Win32_videocontroller" bileşeni içerisinde "caption", "Description", "AdapterCompatibility", "Name" ve "VideoProcessor" parametreleri içerisinde VMware imzası olduğu görülmektedir.

Bir başka değer ise disk sürücüsü bileşeni içerisinde yer alan "Model" değeri, içerisinde VMware imzasını barındırmaktadır.

```
C:\Users\Tufan>wmic diskdrive get model
Model
VMware, VMware Virtual S SCSI Disk Device
```

Resim 3.18: wmic ile disk sürücüsü model billgisi sorgulama

Resim 3.18'de wmic ile sorgulanan disk sürücüsü model ismi, içerisinde VMware imzası taşımaktadır ve zararlı yazılımlar tarafından sanallaştırma karşıtı tekniği olarak kullanılabilmektedir.

4. SANALLAŞTIRMA KARŞITI TEKNİKLERE KARŞI ALINABİLECEK ÖNLEMLER

Sanallaştırma karşıtı (anti-VM) teknikler kullanan zararlı yazılımların analizleri uzun sürmektedir ya da analiz hatalı sonuç vermektedir. Bu sayede zararlı yazılımlar piyasada daha fazla barınabilmek için sanallaştırma karşıtı teknikler kullanmaktadır. Analiz ortamlarını bu tekniklere karşı hazırlıklı tutmak hem analiz süresini kısaltmak için hem de analizlerde doğru sonuçlar elde edebilmek için önem arz etmektedir.

Güvenli ve kontrollü analiz ortamı oluşturmak, hem manuel analiz yapan analistler için hem de kum havuzlarının (sandbox) yaptığı dinamik analizler için çok büyük önem arz etmektedir. Zararlı yazılımın davranışları, çalıştırılmadan tespit edilemeyeceği için nasıl bir zararlı etkisi olabileceği bilinmemektedir. Zararlı yazılımın çalışma anının bir sanallaştırma içerisinde olması bu yüzden önemlidir.

Kum havuzları, sanallaştırma ortamlarını saklamak için çok fazla efor harcamaktadırlar. Kancalama tabanlı (hooking-based) çalışan bir kum havuzu, sanallaştırma karşıtı tekniklerde kullanılan sistem çağrılarında araya girerek, zararlı yazılımın yaptığı isteği manipüle edebilmektedir [19]. Emülatör tabanlı çalışan kum havuzları ise emülatör ortamını gerçek ortama benzetmek için çalışmalar yapmaktadır.

Sanallaştırma karşıtı yöntemler, sanal ortamlarda uygulanan sıkılaştırmalar ile ve yapılan sistem çağrılarında araya girilerek yapılabilmektedir. Sistem sıkılaştırmasında, sanallaştırma ortamının izlerinin silinmesi, donanımsal ve kaynak kontrollerinin engellenmesi gibi teknikler uygulanmaktadır. Sistem çağrılarında araya girme ile, yapılan yazılım kontrollerinin, işletim sistemi üzerinde yer alan sanallaştırma ortamı imzalarının kontrol edilmesi engellenebilir.

VMware üzerinde yapılan sıkılaştırmalar, sanallaştırma yapılan ana makine üzerinden, VMware'in konfigürasyon dosyası olan vmx içerisinde değişiklik yapılarak uygulanabilir. Vmx dosyası içerisinde bilgisayarın hangi ayarlarla

çalıştırılacağı bilgileri yer almaktadır. Ayrıca işletim sistemi içerisinden, uygulamalar üzerinde değişiklik yapılarak VMware izleri silinebilir [25].

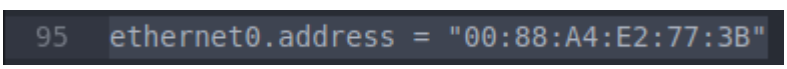
Sistem çağrılarında araya girme işleminde ise yazılan bir kancalama (hooking) aracı ile Windows kütüphanesinden sanallaştırma karşıtı teknikler için kullanılacak fonksiyonlarda araya girilerek dönülen değer üzerinde oynama yapılabilmektedir [43]. Bu yöntem kancalama tabanlı kum havuzlarında, davranışları izlemek ve davranışlar bütününü anlamlandırmak için kullanılmaktadır. Araya girme işleminde çağrılan fonksiyonun parametreleri de görülmektedir. Yapılan sistem çağrısı ve çağrıldığı parametrelerin sanallaştırma karşıtı teknikler için kullanıldığı biliniyorsa sistem çağrısında araya girilerek döndürdüğü değerler üzerinde oynamalar yapılabilmektedir.

Bu bölümde, Bölüm 3. anlatılan, zararlı yazılımların kullandığı sanallaştırma karşıtı tekniklere karşı alınabilecek önlemler anlatılacaktır.

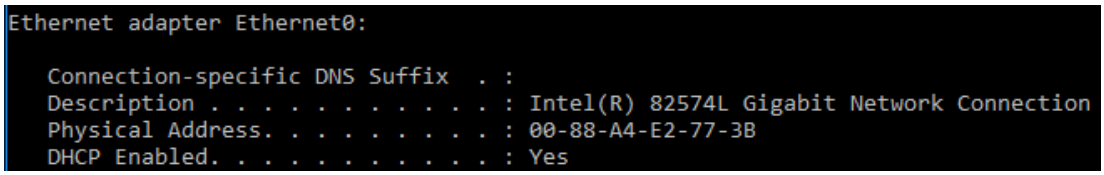
4.1 MAC Adresi Kontrolünü Engelleme

Zararlı yazılımların yaptığı VMware kontrolü olan MAC adresi kontrolü, VMware sıkılaştırması ile önüne geçilebilecek bir yöntemdir. VMware, sanal bilgisayarlara MAC adreslerinin ilk üç byte'ını dört farklı şekilde vermesinden dolayı MAC adresinin manuel bir şekilde değiştirilmesi gerekmektedir.

Sanal makinenin vmx dosyası içerisinde yer alan "ethernet0.adressTpye" parametresinin "generated" değerine eşit olması, VMware'ın kendi ürettiği MAC adresinin kullanılacağını göstermektedir. Bu değer, yeni bir MAC adresi manuel olarak eklense bile yine eski MAC adresini ağ kartı üzerine yazılmasına sebep olmaktadır. Bu sebeple vmx dosyası içerisinde bu değer silinmelidir. Bu değer silindikten sonra, yeni bir MAC adresinin sanal ağ kartının üzerine yazılması için, "erhernet0.adress" değişkeni "XX:XX:XX:XX" (X, 16'lık formatta rakamlar) şeklinde vmx dosyasına eklenmelidir. Resim 4.1'de eklenen MAC değeri görülmektedir.



Resim 4.1: vmx dosyası üzerinde MAC adresi değişimi



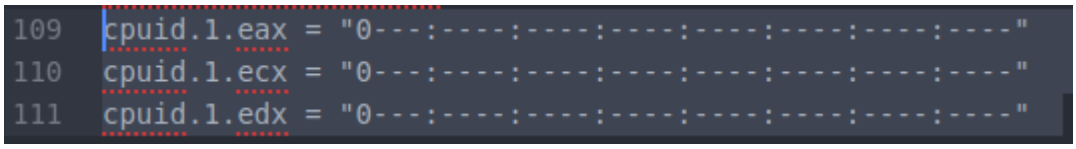
Resim 4.2: MAC değişimi sonrası ipconfig çıktısı

Resim 4.2'de sanal Windows makine üzerinde MAC adresi kontrolü yapmak için çağrılan "ipconfig /all" komutunun çıktısı görülmektedir. "Physical Adress" değerinin Resim 4.1'de vmx dosyasına eklenen değerler aynı olduğu görülmektedir.

4.2 Cpuid Buyruğu Kontrolünü Engelleme

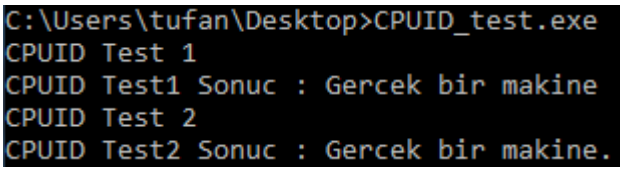
VMware, oluşturduğu sanal işlemci üzerinde kendi izini bırakmaktadır. İntel x86 buyruk kümesinde yer alan cpuid buyruğu çağrıldığı zaman, işlemci hakkında bilgiler eax, ebx ve ecx yazmaçlarına (register) değerler dönmektedir. Bu değerler içerisinde "VMware" imzası bulunmaktadır.

VMware tarafından oluşturulan sanal makinenin vmx dosyası içerisine cpuid buyruğu çağrıldıktan sonra eax, ecx ve edx yazmaçlarına yazılan değerler üzerinde oynamalar yapılabilmektedir. Vmx dosyasına "cpuid.1.eax", "cpuid.1.ecx" ve "cpuid.1.edx" değişkenleri, "0---:----:----:----:----:----:----:----" değerine eşitlenerek vmx dosyasına eklenir.



Resim 4.3: cpuid için vmx parametreleri

Vmx dosyasına Resim 4.3'te görülen parametreler eklendikten sonra, cpuid buyruğu çağrıldığı zaman eax, ecx ve edx yazmaçlarına ayarlanan değerler döner. Böylelikle VMware imzası silinmiş olur. Bölüm 3.2'de cpuid testi için kullanılan uygulama, bu işlem uygulandıktan sonra aynı makine üzerinde tekrar çalıştırılmıştır.



Resim 4.4: cpuid test uygulaması çıktısı

Test için yazılan uygulamanın gerçek makine çıktısı verdiği, Resim 4.4'te görülmektedir.

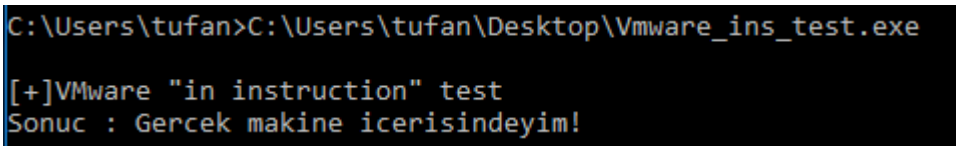
4.3 in Buyruğu Kontrolünü Engelleme

VMWare üzerinde oluşturulan sanal makine ile ana makine arasındaki iletişim için açılan port haberleşmesinde kullanılan in buyruğu (instruction) ile zararlı yazılımlar, sanal makine tespiti yapabilmektedir. Bu tespiti önlemek için ana makine ile sanal makine arasında yer alan haberleşmenin kısıtlanması gerekmektedir. Bu kısıtlamanın işleyişi aksatmadan yapılması için VMware'ın vmx dosyasına eklenebilecek ayarlar bulunmaktadır. Bu yöntemi engellemek için kullanılan parametreler, VMware tarafından paylaşılmamaktadır. Ancak belgelenmemiş ayarlar içerisinde yer alan "isolation.tools" ve "monitor_control" ile başlayan bu parametrelerle ana bilgisayar, sanal bilgisayar arasında iletişim kısıtlanabilmektedir [13]. "isolation.tools" parametreleri, ana makine ile haberleşme kontrolünü sağlamak için, "monitor_control" parametreleri de sanallaştırma ortamında kullanılan yazılımsal kontrol mekanizmaları için kullanılmaktadır. Bu parametreler Resim 4.5 üzerinde görülmektedir. Bu özellikler kapatıldığı zaman in buyruğu testi ile sanal makine tespiti yapılamamaktadır.

```
isolation.tools.getPtrLocation.disable = "TRUE"
isolation.tools.setPtrLocation.disable = "TRUE"
isolation.tools.setVersion.disable = "TRUE"
isolation.tools.getVersion.disable = "TRUE"
monitor_control.disable_directexec = "TRUE"
monitor_control.disable_chksimd = "TRUE"
monitor_control.disable_ntreloc = "TRUE"
monitor_control.disable_selfmod = "TRUE"
monitor_control.disable_reloc = "TRUE"
monitor_control.disable_btinout = "TRUE"
monitor_control.disable_btmemspace = "TRUE"
monitor_control.disable_btpriv = "TRUE"
monitor_control.disable_btseg = "TRUE"
```

Resim 4.5: in buyruğu engellemek için kullanılan vmx parametreleri

Bu yöntemler uygulandıktan sonra Bölüm 3.3'te yer alan in buyruğu test uygulaması tekrar çalıştırıldığında, uygulama sanal ortamı yakalayamamaktadır.



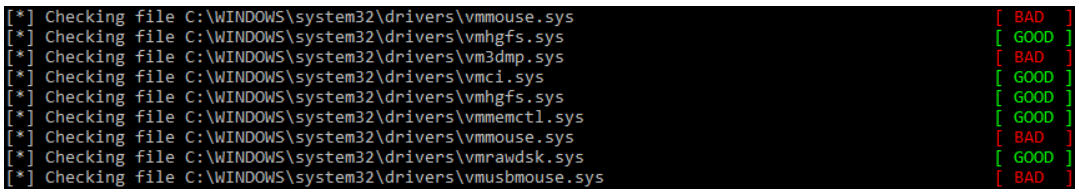
Resim 4.6: VMware in buyruğu test uygulaması çıktısı

Resim 4.6'da görüldüğü üzere, uygulama çalıştırıldıktan sonra test fonksiyonu in buyruğu kullanarak VMware imzasını aramıştır ancak bulamadığı için, "Gerçek makine içerisindeyim" cevabını dönmüştür.

4.4 VMware Klasörü ve Sürücü Kontrollerini Engelleme

VMware, sanallaştırma işlemlerinin performansını arttırmak için birçok uygulama ve sürücü (driver) kullanmaktadır. VMware Tools olarak geçen uygulamalar, sanal makine kurulumu sırasında yüklenmektedir. Yüklenen dosyalar da "Program Files" klasörü altında VMware klasörüne yüklenir. Bu klasör kontrolünü engellemek için uygulanan yöntemlerden bir tanesi, VMware Tools olarak geçen araçların kaldırılması ile klasörün otomatik olarak silinmesidir. Bu araçların neler yaptığını ve olmamaları durumunda oluşacak kayıpları Bölüm 4.6'da anlatılacaktır.

VMware Tools uygulamaları sanal makine üzerinden kaldırıldığı zaman çoğu sürücü, sanal makinenin kullanılabilirliğini bozmamak için varlığını sürdürmeye devam etmektedir. Resim 4.7'de görülen al-khaser çıktısında yanında "BAD" yazan sürücüler VMware Tools kaldırılması sonrasında da varlığını devam ettiren sürücüler olarak görülmektedir. Zararlı yazılımlar, yaptıkları sistem çağrısı ile sürücülerin de varlıklarını kontrol edebilmektedirler.



Resim 4.7: VMware Tools kaldırıldıktan sonra varlığını sürdüren sürücüler

Resim 4.7'de görüldüğü gibi yanında "BAD" yazan fare ve görüntü almak için kullanılan sürücüler varlığını sürdürmektedir. Bu sürücülerin varlığı hala kontrol edilebilir durumdadır.

VMware klasörünün ve sürücülerin kontrolünü yapabilecek fonksiyonların kancalanması ile bu sanallaştırma karşıtı teknik engellenebilir. "CreateFile"

fonksiyonu ile klasör varlığı kontrol edilebilmektedir. Kancalama ile, zararlı yazılımın yaptığı fonksiyon çağrısında araya girilerek, var olan bir dosya ise döndüğü hata kodu değiştirilerek zararlı yazılım manipüle edilebilmektedir.

"GetFileAtribute" fonksiyonu ile de dosya hakkında bilgi alınabilmektedir. Zararlı yazılım bu fonksiyonu çağırdığı zaman, hata mesajı olarak böyle bir dosyanın olmadığını belirten "INVALID_FILE_ATTRIBUTE" hatası dönülerek, zararlı yazılım manipüle edilebilmektedir.

Bunlara ek olarak sanal makine içerisinde yer alan analiz araçlarının da varlığını kontrol eden zararlı yazılımlar mevcuttur. Aynı fonksiyonlara uygulanan kancalama yöntemi ile eğer sanal makine içerisinde kullanılan bir analiz aracı varsa, onlar da zararlı yazılımlardan gizlenebilir.

4.5 Kayıt Defteri Değerleri Kontrolünü Engelleme

Kayıt defteri (Registry) değerleri kontrolleri, zararlı yazılımların VMware imzası aradığı bir başka sanallaştırma karşıtı tekniktir. Bu tekniği atlatabilmek için birden fazla yöntem uygulanmaktadır.

VMware'in imzasının yer aldığı "HKLM\SOFTWARE\VMware, Inc.\VMware Tools" adresindeki kayıt defteri kaydı, VMware Tools uygulamaları kaldırıldığında otomatik olarak silinmektedir. Eğer VMware Tools uygulamaları kullanılmaya devam edilmek istenirse, kayıt defteri değerlerinin sorgulandığı "RegOpenKey" WIN API fonksiyonu bir önceki sanallaştırma karşıtı tekniklerde olduğu gibi kancalanarak, bu kayıt defteri kaydı gizlenebilmektedir.

Diğer kayıt defteri bileşenleri, bir kayıt defteri adresi içerisinde yer alan değer kontrolü yapıldığı için "RegQueryKey" WIN API fonksiyonu kancalanarak gizlenebilmektedir. Bu fonksiyon kancalandığı zaman, "HKLM\HARDWARE\DEVICEMAP\Scsi\Scsi Port 0\Scsi Bus 0\ Target Id 0\ Logical Unit Id 0" adresi içerisinde yer alan "Identifier" değişkeni içerisindeki değer değiştirilerek zararlı yazılım manipüle edilebilmektedir. Aynı şekilde "HKLM\SYSTEM\ControlSet001\Control\SystemInformation" adresi içerisinde yer alan "SystemManufacturer" ve "SystemProductName" değerleri değiştirilerek, zararlı yazılım manipüle edilebilmektedir.

Kancalama yöntemi dışında bir başka yöntem ise, sanal makine her başlatıldığında otomatik çalıştırılan bir betik dosyası ile bu değerlerin değiştirilmesidir. Bu değerler sistem çalışmasını etkilemediği, sadece etiketleme için kullanılan değerler olduğundan, değiştirilmesi işletim sistemini etkilememektedir.

4.6 İşlem ve Servis İsimleri Kontrolünü Engelleme

VMware'ın oluşturduğu işlemler (process) ve servisler, zararlı yazılımlar tarafından kontrol edilerek sanallaştırma karşıtı yöntemi olarak kullanılmaktadır. Bu işlemler ve servislerin tamamı, VMware Tools uygulamaları ile yüklenip çalıştırılmaktadır.

VMware Tools uygulamaları, sanal makinenin daha optimize çalışması, donanım kontrollerini daha iyi yapabilmesi, I/O aygıtlarının sanal makineye aktarımının kolaylaştırılması, görüntünün ana bilgisayar ile aynı çözünürlükte çalışması gibi işlemlerde kullanılmaktadır.

Çalışan işlemler ve servislerinin tam anlamıyla yaptığı işlemler VMware tarafından belgelenmemiştir. Bu işlemler için kullanılan işlemler ve servislerin yokluğunda, sanal makine kullanımını kalitesinde belirli ölçüde azalmalar meydana gelmektedir. Oluşan olumsuzluklardan bazıları; ekran çözünürlüğünün düşük olması, ana bilgisayar ile dosya aktarımının "sürükle-bırak" gibi kolay bir yöntemden mahrum kalınması, I/O aygıtlarında çok az sıklıkla rastlanan gecikmeler, hata anında ana bilgisayar tarafından müdahale edilememesi, hafıza ve disk bağlantılarında gecikmelerden dolayı yavaşlamalar gibi olumsuzluklar ortaya çıkmaktadır. Bu olumsuzluklar düşünüldüğünde, VMware Tools'un sanallaştırmaya etkisinin büyük olduğu görülmektedir. Ancak bu uygulamaların açtığı işlem ve servisler, VMware'ın sanal ortam üzerinde bıraktığı en büyük izlerdir.

VMWare Tools'u Windows'tan kaldırmak, sanallaştırma karşıtı tekniği olarak kullanılan işlem ve servis isimleri kontrolleri tekniğinin tamamen önüne geçebilmektedir. VMware Tools'un kaldırılması sonucu yapılan kontrollerde, sanal makine üzerinde hiçbir işlem ve serviste VMware imzası kalmadığı görülmüştür. Zararlı yazılım analizi için kullanılan sanal makineler, üzerinde sürekli çalışılmadığı ve gündelik işlerin gerçekleştirildiği bir ortam olmamasından dolayı, kullanım kolaylıklarından vazgeçilmesi daha kolaydır.

VMware Tools'un tamamen kaldırmanın yanında, diğer tekniklerde uygulandığı gibi WIN API fonksiyonlarının kancalanması sonucu, bu işlemler ve servisler zararlı yazılımlardan gizlenebilir. WIN API içerisinde yer alan "GetModuleBaseName" fonksiyonu kancalanarak, işlem ID'sine karşılık gelen işlem adı manipüle edilir ve zararlı yazılımın işlem adına ulaşması engellenebilir. Bununla birlikte servisler için kullanılan "EnumServiceStatus" WIN API fonksiyonu kancalanarak dönülen dizi üzerinde yer alan VMware'a ait servisler diziden kaldırılarak zararlı yazılımın servis ismine ulaşması engellenebilir.

4.7 WMI Kontrollerini Engelleme

Windows Management Instrumentation (WMI), uzak bilgisayarlar üzerinde kontroller yapmak ve işlemler uygulamak için kullanılan bir Windows sistemidir. Bu sistem ile uygulanabilen işlemler, zararlı yazılımlar tarafından sanallaştırma karşıtı (anti-VM) uygulanan bir teknik olarak kullanılmaktadır. WMI ile donanım kontrolleri ve işletim sistemi kontrolleri gerçekleştirilebilmektedir.

WMI yapısında, bilgisayar üzerinde yer alan her bir bileşene ait veriler, veri tabanı gibi bir sistem içerisinde tutulmaktadır. WMI'a gönderilen sorgular ile bu bileşenlere ait verilere erişim sağlanabilmektedir. Bu veriler içerisinde işlemci bilgileri, ekran kartı bilgileri, RAM ve disk bilgileri, işletim sistemine ait bilgiler ve BIOS için tutulan bilgilere ulaşım sağlanabilmektedir. VMware, sanallaştırma işleminde bu veriler içerisine birçok imza bırakmakta ve bu imzalar da sanallaştırma karşıtı bir teknik olarak ortaya çıkmaktadır. Sayılan verilere birden fazla erişim mümkündür ancak çoğu kum havuzu (sandbox) WMI sorguları ile bu bilgilere erişimi henüz engellemiş değildir. Onun için zararlı yazılımlar bu verileri WMI üzerinde sorgulayarak sanal ortam tespitini bir engele takılmadan yapabilmektedir.

WMI kontrollerini engellemek için sistem sıkılaştırılması düşünülebilir, ancak WMI üzerinden yapılan sorgulara karşılık çok yeterli olamayacaktır. Bu sebeple WMI için bir kancalama aracı geliştirilme ihtiyacı ortaya çıkmıştır. Kancalama aracı ile WMI objeleri üzerinden yapılan sorgular üzerinde oynamalar yapılabildiği gibi, sorgu kontrolü sonrası amacın sanallaştırma karşıtı olduğu tespiti de mümkündür. Ancak WMI kancalama işlemi standart bir kancalamadan farklı gerçekleştirilebilmektedir. İlerleyen başlıklarda, standart bir kancalama işleminin nasıl yapıldığı anlatıldıktan

sonra WMI için uygulanan kancalama tekniğinin neden farklı olması gerektiği ve nasıl yapılacağı anlatılmıştır.

4.7.1 Kancalama yöntemi

Bilgisayar programlamasında kancalama (hooking) terimi, bir işletim sisteminin veya diğer yazılım bileşenlerinin davranışlarını işlev çağrılarını veya yazılım bileşenleri arasında geçen olayları ele alarak değiştirmek veya geliştirmek için kullanılan bir dizi tekniği kapsar. Bu tür ele geçirilmiş işlev çağrılarını, olayları veya mesajları yöneten kod kanca (hook) olarak adlandırılır [43].

Kancalama yöntemi en genel tanımıyla, yapılan sistem çağrılarında veya uygulama kütüphanelerinde yer alan fonksiyon çağrılarında araya girmek için, sistem çağrılan fonksiyonun içeriği değiştirilerek kendi fonksiyonumuza yönlendirme yapılmak için uygulanan yöntemdir. Genel olarak, bir fonksiyonun ne kadar hızlı çalıştığı, çalışma hızından dolayı kaynaklanan bir problem olup olmadığını kontrol etmek için ve gönderilen parametreleri kontrol etmek ve doğrulamak için kullanılır. Bu kullanımları dışında, kum havuzları tarafından, yapılan çağrılarda araya girilerek, davranış takibi için de kullanılmaktadır.

Kancalama için kullanılan yöntemlerin en temel davranışı, asıl çağrılmak istenen fonksiyonun başına yazılan "jump" buyruğu (instrcution) ile hafıza üzerinde önceden adresini bildiğimiz kendi fonksiyonumuza zıplatmaktır. "jump" buyruğu bir byte işlem kodu (opcode) ve dört byte adresten oluşmaktadır. Toplamda beş byte'lık bir alana ihtiyaç vardır. Bu beş byte'lık alan için kancalanmak istenen fonksiyonun ilk beş byte'lık alanı kullanılır. Kancalanacak fonksiyonun ilk beş byte'ı jump buyruğu ile doldurulduktan sonra eğer fonksiyonun 6. byte'ında yarım kalan bir işlem kodu olursa, geri dönüş anında bozukluk olmaması için onlar da nop buyruğu ile değiştirilir ve üzerine yazılan buyruklar jump ile gittiğimiz kendi fonksiyonumuzun sonuna eklenir. Ayrıca kancalanan fonksiyonda kalınan adres de tutularak geri dönüş için kullanılmak üzere saklanır. Kendi fonksiyonumuza zıpladıktan sonra istenilen işlemler yapılır. Ardından, kancalanan fonksiyondan alınan işlem kodları kendi fonksiyonumuz sonunda çağrılarak, kancalanan fonksiyona bir başka jump buyruğu ile geri dönülür. Böylelikle çağrılan sistem fonksiyonunda araya girilerek istenilen işlemler yapılmış olur. Bu yönteme, trambolin yöntemi denmektedir [43].

Kancalamada dikkat edilmesi gereken birkaç durum vardır. Bunlardan ilki, kancalanmak istenen fonksiyonun ilk beş byte'ının üzerine yazılmadan önce o adreslerde tutulan işlem kodlarının başka bir yere kaydedilmesidir. Kaydetme işlemi yapılırken, işlem kodunun bütünüyle alınmasına dikkat edilmesi gerekir. Yoksa geri dönüş anında bozuk işlem kodları yüzünden uygulama hata vermektedir. Beş byte'lık alan her zaman aynı büyüklükte olmayabilir. Bir diğer dikkat edilmesi gereken durum ise geri dönüşte gidilecek adres, üzerine kendi jump buyruğumuzu yazdığımız adreslerden bir sonraki adres olması gerekmektedir. Burada yapılacak bir hata, geri dönüşten sonra aynı işlem maddesinin tekrarlanması ya da bir işlem maddesinin çağrılmaması sonucu uygulama hata vermektedir.

Uygulanan kancalama işlemleri standart WIN API fonksiyonlarında ve uygulamaların kullandığı başka kütüphanelerde yer alan fonksiyonlarda uygulanabilmektedir. Ancak WMI için bu uygulama farklılık göstermektedir. Kancalama sistemi aynı olmasına karşın, çağrılan fonksiyonların adreslerinin hafıza üzerine yazılmamış olmasından dolayı sıkıntılar çıkmaktadır.

4.7.2 WMI çağrılarının kancalanması

WMI sistem çağrılarının kancalanması (hooking) işlemi, standart bir WIN API kütüphanesinde yer alan fonksiyonlara göre farklılık göstermektedir. WIN API kütüphanelerinde yer alan fonksiyonların adresleri, uygulama çalışma anında hafızaya yazıldığı için yer tespiti yapmak kolaydır. Ancak WMI çağrılarında kullanılan her adım için yeni bir obje oluşturulması gerekmektedir. Bu sebeple kancalanması gereken fonksiyonlar, obje oluşturulmadan hafızadaki adresine erişim mümkün değildir. Bu sebeple standart bir kancalama işlemi gerçekleştirmek adresler belli olmadığı için mümkün olmamaktadır.

WMI çağrıları için "wbemidl" kütüphanesi kullanılmaktadır. Bir WMI sorgusu gönderebilmek için "wbemidl" kütüphanesi üzerinden dört farklı obje oluşturulması gerekmektedir. Bu objeler, WbemLocater, WbemServices, EnumWbemClassObject, WbemClassObject şeklinde sıralanmaktadır.

WMI uzaktan da bağlantı kurulabilen bir yapı olduğu için öncelikli olarak bağlantı oluşturulması gerekmektedir. Bu bağlantı Resim 4.8'de görülen "CoCreateInstance" fonksiyonu ile gerçekleştirilmektedir.

```
IWbemLocator* WbemLocator = NULL;
hr = CoCreateInstance(&CLSID_WbemLocator,
                      0,
                      CLSCTX_INPROC_SERVER,
                      &IID_IWbemLocator,
                      (LPVOID *)&WbemLocator);
```

Resim 4.8: WMI için alan oluşturma fonksiyonu

"CoCreateInstance" fonksiyonu ile gerekli parametreler kütüphaneden alınır ve WbemLocater objesine bu bağlantı bilgiler ile yetki verilir. Böylelikle WbemLocater objesi üzerinden artık WbemService objesi oluşturulabilecek ve yerel makine üzerinde yer alan WMI tablolarına bağlantı kurulabilecek aşamaya gelinmiştir.

Bu işlemden sonra, WbemLocater objesi üzerinde yer alan ve Resim 4.9'da görülen "ConnectServer" fonksiyonu çağrılarak WbemServices objesi oluşturulur.

```
IWbemServices* WbemServices = NULL;
hr = WbemLocator->lpVtbl->ConnectServer(WbemLocator,
                                        L"ROOT\\CIMV2",
                                        NULL,
                                        NULL,
                                        NULL,
                                        WBEM_FLAG_CONNECT_USE_MAX_WAIT,
                                        NULL,
                                        NULL,
                                        &WbemServices);
```

Resim 4.9: WMI veri tabanına bağlantı fonksiyonu

"ConnectServer" fonksiyonu, WMI veri tabanında yer alan "ROOT\CIMV2" tablosuna bağlantı kurar. Ardından tablo içeriğini oluşturulan WBemSerives objesine iletir. VMware imzalarının bulunduğu verilerin hepsi "ROOT\CIMV2" tablosu üzerinde yer almaktadır. Bu bağlantı aşamasında müdahale edilecek bir durum olmadığı için kancalama işleminin uygulanmasına gerek yoktur.WbemServices objesi içerisinde yer alan tablo üzerinde sorgulama yapmak için "ExecQuery" fonksiyonu çağrılır. Bu fonksiyonun adresine erişim için WbemService objesinin oluşturulması gerekmektedir. "ConnectServer" sonrası oluşturulan WbemService objesi elimizde olduğu için bu fonksiyon kancalanabilir duruma gelmiştir. Resim 4.10'da WMI sorgusu gönderilmek için kullanılan "ExecQuery" fonksiyonu görülmektedir.

"ExecQuery" fonksiyonu ile WbemService içerisinde yer alan veri tabanına sorgu gönderilebilir. Resim 4.10'da görülen örnek bir sorgu olan "Select * FROM

Win32_VideoController" sorgusu sonucu veri listesi şeklinde EnumWbem objesine dönülerek listede dolaşma işlemi EnumWbem objesine bırakılmaktadır. "ExecQuery", içerisinde sorgunun metnini barındırdığı için kancalanması gerekmektedir. Böylelikle sorgulamanın nasıl yapıldığı elde edilmiş olur. Bölüm 4.7.1 anlatıldığı gibi trambolin kancalama işlemi gerçekleştirilir. "ExecQuery" fonksiyonu her çağrıldığında, kancalama sırasında gidilecek fonksiyona yönlendirilir ve gerekli işlemleri gerçekleştirdikten sonra orijinal fonksiyonu tekrar çağrılıp uygulamanın akışını bozmadan devam edilir.

```
IEnumWbemClassObject* EnumWbem = NULL;
hr = WbemServices->lpVtbl->ExecQuery(WbemServices,
                                     L"WQL",
                                     L"SELECT * FROM Win32_VideoController",
                                     WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
                                     NULL,
                                     &EnumWbem);
```

Resim 4.10: WMI veri tabanına gönderilen sorgu

Sonrasında EnumWbem objesi üzerinde yer alan listede dolaşmak ve sırasıyla bütün verileri alıp kontrol etmek için WbemClassObject objesi oluşturulur. Oluşturulan obje üzerinden adresi artık belli olan "Next" fonksiyonu çağrılarak dönülen değerler WbemClassObject içerisine gönderilir. "Next" fonksiyonu doğru parametreyi bulana kadar dönmesi gerektiği için bir döngü içerisinde sürekli çağrılmaktadır.

```
ULONG returnedCount = 0;
IWbemClassObject *result = NULL;
hr = EnumWbem->lpVtbl->Next(EnumWbem,
                            WBEM_INFINITE,
                            1,
                            &result,
                            &returnedCount);
```

Resim 4.11: WMI sorugusu sonucu dönen listeyi sıralı gezme

Resim 4.11'de "Next" fonksiyonun, listede bulunan her bir değeri WbemClassObject objesine gönderdiği görülmektedir. İçerisinde bir veri barındırmadığı için kancalama işlemine gerek yoktur.

WbemClassObject objesi de oluştuktan sonra "Get" fonksiyonu bu obje üzerinden çağrılabilir duruma gelir. "Get" fonksiyonu, yapılan sorgu sonucunda istenilen parametrenin değerinin kontrol edildiği yerdir. Bu yüzden zararlı yazılımlar için önemli olan fonksiyon "Get" fonksiyonudur. "Get" fonksiyonu ile WMI'a atılan sorgunun cevabı alınır.

```
VARIANT VideoProcessor;
hr = result->lpVtbl->Get(result,
                         L"VideoProcessor",
                         0,
                         &videoProcessor,
                         0,
                         0);
```

Resim 4.12: WMI sorgusu sonucunun alınması

Resim 4.12'de görülen "Get" fonksiyonu ile Resim 4.10'da yer alan "Select * FROM Win32_VideoController" sorgusu sonucu dönen objede yer alan "VideoProcessor" parametresinin değerini almaktadır. Bu değer alındıktan sonra "VARIANT" olarak tanımlanan veri yapısına gönderilir. Aranılan değer veri yapısı içerisinden elde edilebilir.

Bu sürecin en temel amacı, "Get" fonksiyonun kancalama aşamasına gelmesidir. Son yapılan işlem ile "Get" fonksiyonun obje üzerindeki adresi de bulanabileceğinden kancalama işlemi gerçekleştirilir. Trambolin kancalama yöntemi ile fonksiyon kancalandıktan sonra, verilerin üzerinde her türlü işlem gerçekleştirilebilir duruma gelir.

Bu aşamaların hiçbirinin yapılmaması durumunda sadece "Get" fonksiyonuna kancalama tekniği, hiçbir objenin adresi belli olmadığı için objeler boş kalacağından uygulanamaz. Zararlı yazılımların yaptığı sorgulamaların sonuçları "Get" fonksiyonu çağrıldığı anda ortaya çıkmaktadır. "Get" fonksiyonu çağrılana kadarki süreçte de dört farklı obje türetilip fonksiyon adreslerinin belirlenmesi gerekmektedir. WMI çağrılarının kancalanabilmesi için, Standart bir WIN API fonksiyonunda farklı olarak anlatılan işlemlerin uygulanması gerekmektedir.

4.7.3 WMI kancalama aracı ile sanallaştırma karşıtı tespitleri engelleme

WMI sorgularını kancalama (hooking) işlemi gerçekleştirildikten sonra, istenilen değerler üzerinde oynamalar yapılarak sonuç manipüle edilebilmekte ve zararlı yazılımların sanal ortam tespitinin önüne geçilebilmektedir. Bu tez kapsamında yazılan kancalama aracı ile WMI sorguları kancalanmakta ve sanallaştırma karşıtı kullanabilecek değerler üzerinde oynamalar yapılarak bunun önüne geçilebilmektedir. Bu bölümde anlatılacak kancalama çalışmaları, EK-1'de yer alan ve kancalama aracı içerisinde, "Get" fonksiyonu kancalandıktan sonra yönlendirilen

"hooked_get" fonksiyonu içerisindeki kontrollere göre kancalama aracı çıktıları ile anlatılmıştır.

WMI ile "Win32_BIOS" bileşeni içerisinde yer alan "SerialNumber" değeri içerisinde VMware imzası bulunmaktadır.

```
C:\Users\Tufan\Desktop\x64_hooking_for_WMI\x64\Release>x64_hooking_for_WMI.exe
QUERY = SELECT * FROM Win32_BIOS
Sorgulanan parametre: SerialNumber
Get fonksiyonunun dondugu parametre: VMware-56 4d 28 cf 92 3a c8 95-79 c7 f2 99 c2 7d 9d db
Fonksiyon kancalandiktan sonra donen deger: ASUS
```

Resim 4.13: WMI ile BIOS seri numarası sorgusunun kancalanma sonucu değiştirilmesi

Resim 4.13'te görüldüğü üzere, "SerialNumber" değerinin sorgulaması geldiğinde WMI kancalama aracımız, değeri "ASUS" ile değiştirmektedir. Böylece VMware imzası ortadan kaldırılmış olur.

WMI ile yapılan ve kancalanan bir diğer sorgu ise, "Win32_ComputerSystem" içerisinde yer alan "Manufacturer" değeridir.

```
C:\Users\Tufan\Desktop\x64_hooking_for_WMI\x64\Release>x64_hooking_for_WMI.exe
QUERY = SELECT * FROM Win32_ComputerSystem
Sorgulanan parametre: Manufacturer
Get fonksiyonunun dondugu parametre: VMware, Inc.
Fonksiyon kancalandiktan sonra donen deger: Monster
```

Resim 4.14: WMI ile bilgisayar üreticisi sorgusunun kancalanma sonucu değiştirilmesi

Bu değer WMI kancala aracımız tarafından yakalandığında, değeri "Monster" ile değiştirilerek, VMware imzası temizlenmiş olur.

Başka bir sorgu "Win32_ComputerSystem" içerisinde yer alan "Model" değeri sorgulandığı zaman uygulanmaktadır.

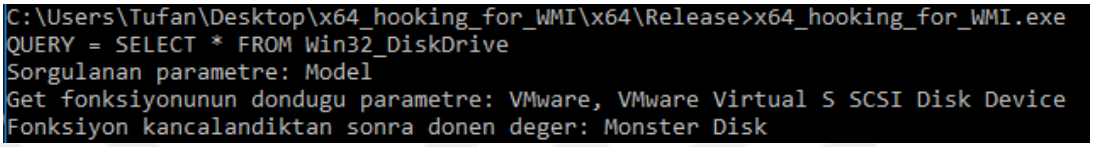
```
C:\Users\Tufan\Desktop\x64_hooking_for_WMI\x64\Release>x64_hooking_for_WMI.exe
QUERY = SELECT * FROM Win32_ComputerSystem
Sorgulanan parametre: Model
Get fonksiyonunun dondugu parametre: VMware Virtual Platform
Fonksiyon kancalandiktan sonra donen deger: Real PC
```

Resim 4.15: WMI ile bilgisayar modeli sorgusunun kancalanma sonucu değiştirilmesi

Kancalama aracımız tarafından yakalanan "Model" değeri sorgusu sonucu, içerisinde VMware imzası barındıran değer değiştirilerek yerine "Real PC" değeri

döndürülmektedir. Böylece VMware imzasını zararlı yazılım yakalayamamış olmaktadır.

Sanallaştırma karşıtı kullanılan bir diğer WMI sorgusu “Win32_DiskDrive” bileşeni içerisinde yer alan “Model” değeridir. Bu değer ile “Win32_ComputerSystem” içerisinde yer alan “Model” değerinin karışmaması için, “ExecQuery” fonksiyonu çağrıldığında kancalandığından, sorgunun hangisi için olduğu yakalanarak kaydedilir. “Get” fonksiyonu içinde de o kayıta bakılarak dönülmesi gereken değer buna göre karar verilir.



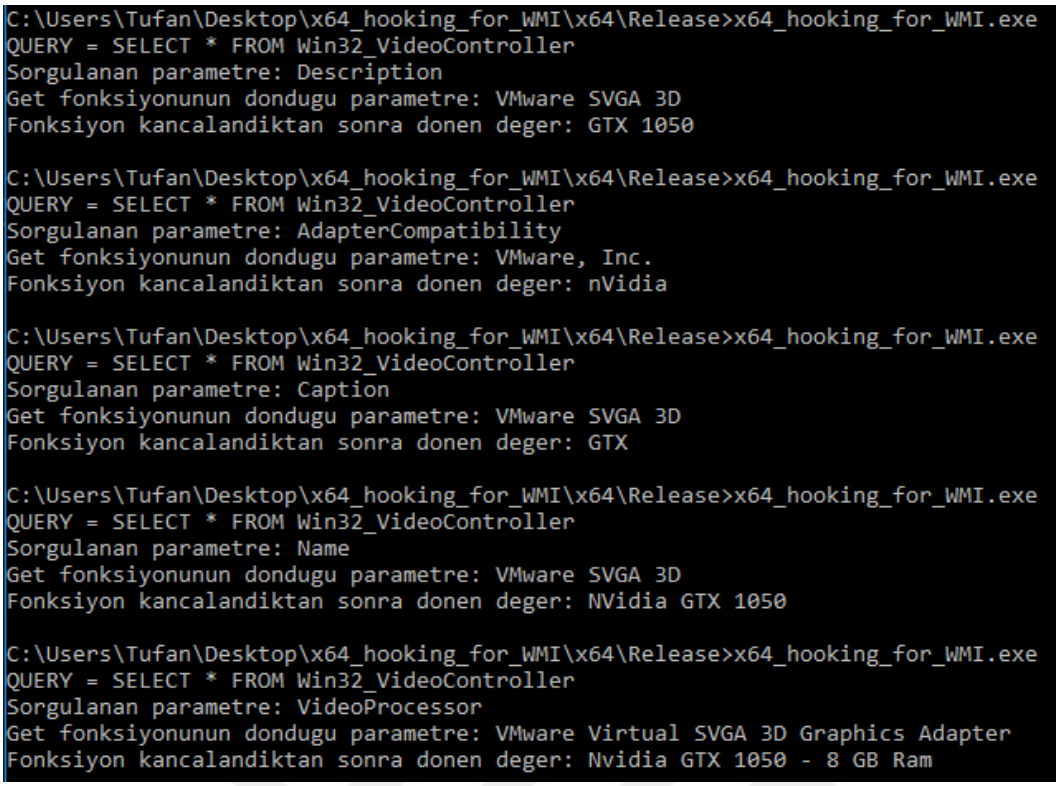
Resim 4.16: WMI ile disk modeli sorgusunun kancalanma sonucu değiştirilmesi

Resim 4.16’da görüldüğü gibi yapılan disk modeli sorgulaması sonucu dönmesi gereken asıl değer içerisinde VMware imzası bulunmaktadır. Ancak kancalama işlemi sonrasında değer “Monster Disk” olarak güncellenmiştir. Böylece bu sorgunun da sanallaştırma karşıtı kullanımının engellenebilir olduğu görülmüştür.

İçerisinde VMware imzası geçen bir başka bileşen de “Win32_VideoController” bileşenidir. İçerisinde beş tane VMware imzası barındıran değişken bulunmaktadır. Al-khaser uygulaması, “Win32_VideoController” içerisindeki çoğu değeri kontrol etmemektedir. Kancalama uygulaması içerisinde “Caption”, “Description”, “AdapterCompatibility”, “Name”, “VideoProcessor” değerleri, içerisinde VMware imzası barındırdığı için, kancalanarak değiştirilmektedir.

Resim 4.17’de görüldüğü üzere, kancalama aracı içerisinde ekran kartı için sorgulanan beş farklı değer kontrol edilmekte ve gerçek bilgisayarlarda olabilecek değerler ile değiştirilmektedir.

Bu bileşenler dışında RAM, disk boyutları, işlemci çekirdek sayısı gibi değerlerler sanal ortamlar içerisinde farklılıklar gösterebileceği için kesin bir kontrol belirtilmemiştir. Ancak bu değerler de WMI sorgu sonucunu alabildiğimiz “Get” fonksiyonun kancalanması ile kontrol edilebilir ve değiştirilebilirdir.



Resim 4.17: WMI ile ekran kartı sorgularının kancalama sonucu değiştirilmesi

Yazılan bu kancalama aracı, kancalama tabanlı (hooking-based) çalışan kum havuzları içerisinde kullanılabilir ve WMI sorgusu kullanılarak yapılan sanallaştırma karşıtı yöntemlerin önüne geçebilir.

5. SONUÇLAR

Zararlı yazımlar, analiz karşıtı yöntemleri aktif olarak kullanmaktadırlar. Bütün analiz süreçleri için farklı analiz teknikleri geliştirilmiştir. Statik analiz için tersine çevirici karşıtı yöntemler (anti-disassembly), dinamik analize karşı ayıklayıcı karşıtı yöntemler (anti-debugging) ve sanallaştırma karşıtı yöntemler (anti-VM), otomatik dinamik analize karşı da kum havuzu karşıtı yöntemler (anti-sandbox) kullanılmaktadır.

Analiz süreçlerinin vazgeçilmezi sanallaştırma ortamları, donanımsal ve yazılımsal olarak kendilerini belli edecek birçok yöntem kullanmaktadır. VMware sanallaştırma ortamı üzerinde kurulmuş bir Windows 10 işletim sistemi ile yapılan incelemelerde VMware'ın birçok yerde imza bıraktığı görülmüştür. Donanımsal olarak incelendiğinde, MAC adreslerinin sabit tutulması, çalıştırılan cpuid ve in buyrukları (instruction) içerisinde VMware geçen sonuçlar dönmesi ile zararlı yazılımlar tarafından kullanılan sanallaştırma karşıtı tekniklere yakalanmaktadır. Yazılımsal olarak bakıldığında VMware, Windows üzerinde yer alan Kayıt Defteri (Registry), klasör isimleri, işlem (process) ve servis isimleri, sürücü (driver) isimleri kendi imzasını barındırdığı için zararlı yazılımlara takılmaktadır.

MAC kontrolü, in ve cpuid buyruklarıyla yapılan kontroller, VMware'ın konfigürasyon dosyası olan vmx dosyasına yazılan yeni parametrelerle ve değiştirilen parametrelerle engellenebilir. VMware ait klasör kontrolü, sürücü kontrolü, işlem ve servis isimleri kontrolü Kayıt Defteri kontrolü VMware Tools uygulamaları silinerek engellenebilir. VMware Tools uygulamaları silinmek istenmediği durumda WIN API üzerinden çağrılan fonksiyonlar trambolin kancalama (hooking) ile fonksiyon parametreleri değiştirilerek sanallaştırma tespitinin önüne geçilebilmektedir.

Bu çok bilinen yöntemler dışında, Windows Management Instrumentation (WMI) ile uygulanan sanallaştırma karşıtı teknikler güncel zararlı yazılımlar içerisinde çokça denk gelmektedir. WMI ile işlemci, RAM, disk, BIOS, ekran kartı bilgileri içerisinde yer alan VMware imzaları bulunduğu yapılan incelemelerde ortaya. VMware dışında

birçok sanallaştırma ortamı yine WMI sorguları ile tespit edilebilmektedir. Bu durum, kum havuzlarının içerisinde de hala düzeltilmediği için zararlı yazılımlar WMI tekniğini aktif bir şekilde kullanmaya devam etmektedir.

WMI fonksiyonları kancalanırken, standart bir WIN API kütüphanesi fonksiyonu gibi kancalanamayacağı görülmüştür. Standart fonksiyonların adresleri uygulama başladığı zaman belli olduğu için üzerine yazılacak yerler de bulunabilir. Ancak WMI için objelerin oluşturulması gerektiği ve bu objeler oluşturulduktan sonra fonksiyonların adreslerinin belirlenmesiyle kancalama işlemi yapılabileceği görülmüştür. WMI fonksiyonlarından iki tanesi sanallaştırma karşıtı yöntemler için önemlidir. Bir tanesi ile yapılan sorgu çekilebilirken, diğeriyle de WMI içerisinde yer alan değere ulaşılabilmektedir. Bu fonksiyonların kancalanması sonucu WMI ile yapılan sanallaştırma karşıtı yöntemin önüne geçilebileceği görülmüştür.

Yazılan bir kancalama aracı ile WMI için önemli olan iki fonksiyon da kancalanmış, gelen değerlere göre görülmek istenen değerin sanallaştırma karşıtı olup olmadığına karar verilmiştir. Bu karardan sonra uygun şekilde değerler değiştirilmiş ve sanallaştırma karşıtı olarak kullanılan yöntem engellenebilmiştir. Geliştirilen bu kancalama aracı, kancalama tabanlı çalışan kum havuzları içinde zararlı yazılımlara karşı kullanılabilir şekilde tasarlanmıştır.

KAYNAKLAR

[1] https://www.av-test.org/en/statistics/malware/, Alındığı tarih: 25.11.2019

[2] **A. K. Maurya, N. Kumar, A. Agrawal ve P. R. A. Khan** (2019). Ransomware Evolution, Target and Safety Measures, *International Journal of Computer Sciences and Engineering*, cilt 6, no. 1, pp. 80-85

[3] **S. Pastrana ve G. Suarez-Tangil** (2019). A First Look at the Crypto-Mining MalwareEcosystem: A Decade of Unrestricted Wealth, *ACM Internet Measurement Conference*, Amsterdam

[4] **A. K. Maurya, N. Kumar, A. Agrawal ve P. R. A. Khan** (2019). Ransomware Evolution, Target and Safety Measures, *International Journal of Computer Sciences and Engineering*, cilt 6, no. 1, pp. 80-85

[5] **S. Gadhiya ve K. Bhavsar** (2013). Techniques for Malware Analysis, *International Journal of Advanced Research in Computer Science and Software Engineering*, cilt 3, no. 4, pp. 972-975

[6] **M. Egele, T. Scholte, E. Kirda ve C. Kruegel** (2012). A Survey on automated dynamic malware-analysis techniques and tools, *ACM Computing Surveys*, cilt 44, no. 2, p. 6

[7] **A. Moser, C. Kruegel ve E. Kirda** (2007). Limits of Static Analysis for Malware Detection, *Twenty-Third Annual Computer Security Applications Conference*, Miami Beach

[8] **C. Willems, T. Holz ve F. Freiling** (2007). Toward Automated Dynamic Malware Analysis Using CWSandbox, *IEEE Security and Privacy Magazine*, cilt 5, no. 2

[9] **C. Linn ve S. Debray** (2003) Obfuscation of executable code to improve resistance to static disassembly, *10th ACM conference on Computer and communications security*, New York

[10] **W. Yan, Z. Zhang ve N. Ansari** (2008). Revealing Packed Malware, *IEEE Security & Privacy*, cilt 6, no. 5, pp. 65-69

[11]https://www.apriorit.com/dev-blog/367-anti-reverse-engineering-protection-techniques-to-use-before-releasing-software, Alıntığı tarih: 14.11.2019

[12] **M.K. Sun, M. J. Lin, M. Chang, C.S. Laih ve H.T. Lin** (2011). Malware Virtualization-Resistant Behavior Detection, *2011 IEEE 17th International Conference on Parallel and Distributed Systems*, Tainan

[13] **L. Sun, T. Ebringer ve S. Boztas** (2008). An automatic anti-anti-VMware technique applicable for multi-stage packed malware. *2008 3rd International Conference on Malicious and Unwanted Software (MALWARE)*, Fairfax

[14]https://docs.microsoft.com/enus/windows/win32/wmisdk/wmi-start-page, Alındığı tarih: 24.09.2019

[15]https://www.exploit-db.com/docs/english/34591-breaking-the-sandbox.pdf, Alındığı tarih: 5.09.2019

[16] **F. Cohen** (1987). Computer viruses: Theory and experiments, *Computers & Security*, cilt 6, no. 1, pp. 22-35.

[17] **S. Kurzban** (1989)**.** Defending against viruses and worms, *ACM SIGUCCS Newsletter*, cilt 19, no. 3, pp. 11-23.

[18] **D. M. Chess, J. O. Kephart ve G. B. Sorkin** (1994). Automatic analysis of a computer virus structure and means of attachment to its hosts, Amerika Birleşik Devletleri Patent: US5485575A.

[19] **C. Willems, T. Holz ve F. Freiling** (2007). Toward Automated Dynamic Malware Analysis Using CWSandbox, *IEEE Security and Privacy Magazine*, cilt 2, p. 5.

[20] **A. Dinaburg, P. Royal, M. Sharif ve W. Lee** (2008). Ether: Malware Analysis via Hardware Virtualization Extensions, *CCS '08 Proceedings of the 15th ACM Conference on Computer and Communications Security*, Alexandria.

[21] **B. Lau ve V. Svajcer** (2008). Measuring virtual machine detection inmalware using DSD tracer, *Journal in Computer Virology*, cilt 6, no. 3.

[22] **A. Dinaburg, P. Royal, M. Sharif ve W. Lee** (2008). Ether: Malware Analysis via Hardware Virtualization Extensions, *ACM Conference on Computer and Communications Security (CCS)*, Alexandria.

[23] **C. Kruegel, E. Kirda, P. M. Comparetti, U. Bayer ve C. Hlauschek** (2009)**.** Scalable, Behavior-Based Malware Clustering, *the Symposium on Network And Distributed System Security*, San Diego.

[24] **R. Paleari, L. Martignoni, G. F. Roglia ve D. Bruschi** (2009). A fistful of red-pills: How to automatically generate procedures todetect CPU emulators, *WOOT'09 Proceedings of the 3rd USENIX Workshop on Offensive Technologies*, Montreal.

[25] https://vmware.com, Alındığı tarih: 15 Ağustos 2019.

[26] https://cuckoosandbox.org, Alındığı tarih: 29 Ocak 2019**.**

[27] **G. Pék, B. Bencsáth ve L. Buttyán** (2011). nEther: In-guest Detection of Out-of-the-guest Malware Analyzers, *EUROSEC '11 Proceedings of the Fourth European Workshop on System Security*, Salzburg.

[28] **M. Lindorfer, C. Kolbitsch ve P. M. Comparetti** (2011). «Detecting Environment-Sensitive Malware, *RAID'11 Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection*, Berlin.

[29] **K. Yoshioka, Y. Hosobuchi, T. Orii ve T. Matsumoto** (2011). Your Sandbox is Blinded: Impact of Decoy Injection to PublicMalware Analysis Systems, *Information and Media Technologies*, cilt 6, no. 2, pp. 633-648.

[30] **P. Chen, C. Huygens, L. Desmet ve W. Joosen** (2016). Advanced or Not? A Comparative Study of the Use of Anti-debugging and Anti-VM Techniques in Generic and Targeted Malware, *IFIP International InformationSecurity and Privacy Conference*, Gent.

[31] **A. Yokoyama, K. Ishii, R. Tanabe, Y. Papa, K. Yoshioka, T. Matsumoto, T. Kasama, D. Inoue, M. Brengel, MichaelBackes ve C. Rossow** (2016). SandPrint: Fingerprinting Malware Sand-boxes to Provide Intelligence for Sandbox Evasion, *RAID'16 Proceedings of the19th International Symposium on Recent Advances in Intrusion Detection*, Evry.

[32] **M. Carpenter, T. Liston ve E. Sloudis** (2007). Hiding virtualization from attackers and malware, *IEEE Security & Privacy*, cilt 5, no. 3, pp. 62-65.

[33] https://ntcore.com/?page_id=388, Alındığı tarih: 28 Ekim 2019.

[34] https://www.hex-rays.com/products/ida/, Alındığı tarih: 28 Ekim 2019.

[35]https://docs.microsoft.com/en-us/sysinternals/downloads/procmon, Alındığı tarih: 28 Ekim 2019.

[36] https://www.blackhat.com/presentations/bh-dc-07/Kendall_McMillan/Paper/bh-dc-07-Kendall_McMillan-WP.pdf, Alındığı tarih: 1 Kasım 2019.

[37]https://docs.microsoft.com/en-us/windows/win32/api/debugapi/nf-debugapi-isdebuggerpresent, Alındığı tarih: 1 Kasım 2019.

[38]https://docs.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-gettickcount, Alındığı tarih: 1 Kasım 2019.

[39]https://docs.microsoft.com/en-us/windows/win32/sysinfo/registry, Alındığı tarih: 1 Kasım 2019.

[40] https://github.com/LordNoteworthy/al-khaser, Alındığı tarih: 15 Ekim 2019.

[41]https://docs.microsoft.com/en-us/windows/win32/api/winreg/nf-winreg-regqueryvalueexa, Alındığı tarih: 2 Kasım 2019.

[42]https://docs.microsoft.com/en-us/windows/win32/api/psapi/nf-psapi-getmodulebasenamea, Alındığı tarih: 5 Kasım 2019.

[43] http://jbremer.org/x86-api-hooking-demystified/, Alındığı tarih: 18 Ekim 2019.

EKLER

EK 1: WMI “Get” kancalama fonksiyonu

EK 1

```
HRESULT WINAPI hooked_Get(IWbemClassObject * This, LPCWSTR wszName,
                    long lFlags, VARIANT *pVal,CIMTYPE *pType,
       long *plFlavor) {
       printf("Sorgulanan parametre: %S\n", wszName);
       LPCWSTR serialNumber = L"SerialNumber", model = L"Model";
       LPCWSTR manufacturer = L"Manufacturer", description = L"Description";
       LPCWSTR adapterCom = L"AdapterCompatibility";
       LPCWSTR caption = L"Caption", name = L"Name";
       LPCWSTR videoProcessor = L"VideoProcessor";
       HRESULT hExec;
       if (lstrcmpW(wszName, serialNumber) == 0) {
          hExec = originalGet(This, wszName, lFlags, pVal, pType, plFlavor);
          printf("Get fonksiyonunun dondugu parametre: %S\n", pVal->bstrVal);
          pVal->bstrVal = L"ASUS";
       }
       else if (lstrcmpW(wszName, model) == 0 && disk_query_count != 1) {
          hExec = originalGet(This, wszName, lFlags, pVal, pType, plFlavor);
          printf("Get fonksiyonunun dondugu parametre: %S\n", pVal->bstrVal);
          pVal->bstrVal = L"Real PC";
       }
       else if (lstrcmpW(wszName, manufacturer) == 0) {
          hExec = originalGet(This, wszName, lFlags, pVal, pType, plFlavor);
          printf("Get fonksiyonunun dondugu parametre: %S\n", pVal->bstrVal);
          pVal->bstrVal = L"Monster";
       }
       else if (lstrcmpW(wszName, model) == 0 && disk_query_count == 1) {
          hExec = originalGet(This, wszName, lFlags, pVal, pType, plFlavor);
          printf("Get fonksiyonunun dondugu parametre: %S\n", pVal->bstrVal);
          pVal->bstrVal = L"Monster Disk";
       }
       else if (lstrcmpW(wszName, description) == 0) {
          hExec = originalGet(This, wszName, lFlags, pVal, pType, plFlavor);
          printf("Get fonksiyonunun dondugu parametre: %S\n", pVal->bstrVal);
          pVal->bstrVal = L"GTX 1050";
       }
       else if (lstrcmpW(wszName, adapterCom) == 0) {
          hExec = originalGet(This, wszName, lFlags, pVal, pType, plFlavor);
          printf("Get fonksiyonunun dondugu parametre: %S\n", pVal->bstrVal);
          pVal->bstrVal = L"nVidia";
       }
       else if (lstrcmpW(wszName, caption) == 0) {
          hExec = originalGet(This, wszName, lFlags, pVal, pType, plFlavor);
          printf("Get fonksiyonunun dondugu parametre: %S\n", pVal->bstrVal);
              pVal->bstrVal = L"GTX";
       }
       else if (lstrcmpW(wszName, name) == 0) {
          hExec = originalGet(This, wszName, lFlags, pVal, pType, plFlavor);
          printf("Get fonksiyonunun dondugu parametre: %S\n", pVal->bstrVal);
          pVal->bstrVal = L"NVidia GTX 1050";
       }
       else if (lstrcmpW(wszName, videoProcessor) == 0) {
          hExec = originalGet(This, wszName, lFlags, pVal, pType, plFlavor);
          printf("Get fonksiyonunun dondugu parametre: %S\n", pVal->bstrVal);
          pVal->bstrVal = L"Nvidia GTX 1050 - 8 GB Ram";
       }

HRESULT hExec = originalGet(This, wszName, lFlags, pVal, pType, plFlavor);
       return hExec;
}
```

ÖZGEÇMİŞ

Ad-Soyad **: Osman Tufan TEKİN**

Uyruğu **: TC**

Doğum Tarihi ve Yeri **: 1994 ÇANKAYA**

E-posta **: osmantufantekin@gmail.com**

ÖĞRENİM DURUMU:

- **Lisans** **:** 2017, TOBB Ekonomi ve Teknoloji Üniversitesi, Bilgisayar Mühendisliği
- **Yüksek Lisans** **:** 2019, TOBB Ekonomi ve Teknoloji Üniversitesi, Bilgisayar Mühendisliği, Bilgi Güvenliği Programı

MESLEKİ DENEYİM VE ÖDÜLLER:

Yıl	**Yer**	**Görev**
2017-2018	TOBB ETÜ	Tam Burslu YL Öğrencisi
2018-Halen	STM	Zararlı Yazılım Analisti

YABANCI DİL: İngilizce

TEZDEN TÜRETİLEN SUNUMLAR:

- **Tekin, O. T.,** Elmasoğlu A. S., 2019. WMI sorguları kullanılarak uygulanan sanal makine tespit yönteminin hooking metodu ile atlatılması, a2cs'19 International Conferance on All Aspects of Cyber Security, 25 Ekim, Adana, Türkiye.