İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

VIRMON: SANALLAŞTIRMA TABANLI OTOMATİK BİR DİNAMİK ZARARLI YAZILIM ANALİZ SİSTEMİ

YÜKSEK LİSANS TEZİ

Hüseyin TİRLİ

Bilgisayar Bilimleri Anabilim Dalı

Bilgisayar Bilimleri Programı

OCAK 2014

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

VIRMON: SANALLAŞTIRMA TABANLI OTOMATİK BİR DİNAMİK ZARARLI YAZILIM ANALİZ SİSTEMİ

YÜKSEK LİSANS TEZİ

Hüseyin TİRLİ
(704101005)

Bilgisayar Bilimleri Anabilim Dalı

Bilgisayar Bilimleri Programı

Tez Danışmanı: Prof. Dr. Nadia ERDOĞAN

OCAK 2014

İTÜ, Bilişim Enstitüsü'nün 704101005 numaralı Yüksek Lisans Öğrencisi **Hüseyin TİRLİ**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"VIRMON: SANALLAŞTIRMA TABANLI OTOMATİK BİR DİNAMİK ZARARLI YAZILIM ANALİZ SİSTEMİ"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Nadia ERDOĞAN** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Bülent ÖRENCİK** Beykent Üniversitesi	.....
	Doç. Dr. D. Turgay ALTILAR İstanbul Teknik Üniversitesi	.....

Teslim Tarihi : **16 Aralık 2013**
Savunma Tarihi : **29 Ocak 2014**

Sevgili nişanlıma ve aileme,

ÖNSÖZ

Tez çalışmam süresince yardımlarını ve desteğini benden esirgemeyen saygıdeğer danışmanım sayın Prof. Dr. Nadia Erdoğan'a çok teşekkür ederim. Ayrıca, öncelikle çalışmam boyunca bana destek olan aileme, tez konusu seçimimde tavsiyeleriyle beni yönlendiren TÜBİTAK BİLGEM Siber Güvenlik Enstitüsü Müdürü sayın Dr. Hayretdin Bahşi'ye, alt yapı çalışmalarında bana yardımcı olan proje yöneticim sayın Necati Ersen Şişeci ve çalışma arkadaşım sayın Abdurrahman Pektaş'a teşekkür ederim.

Ocak 2014

Hüseyin Tirli
(Bilgisayar Mühendisi)

İÇİNDEKİLER

Sayfa

KISALTMALAR

API : Application Programming Interface
BSOD : Blue Screen of Death
C&C : Command and Control
DDOS : Distributed Denial of Service
DNS : Domain Name System
HTTP : Hypertext Transfer Protocol
IDS : Intrusion Detection System
IRP : Input/Output Request Packet
IPS : Intrusion Prevention System
NDIS : Network Driver Interface Specification
PE : Portable Executable
PCAP : Packet Capture
PLC : Programmable Logic Controller
RDP : Remote Desktop Protocol
VLAN : Virtual Local Area Network
VPN : Virtual Private Network

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

VIRMON: SANALLAŞTIRMA TABANLI OTOMATİK BİR DİNAMİK ZARARLI YAZILIM ANALİZ SİSTEMİ

ÖZET

Günümüzde, güvenlik firmaları her gün onbinlerce yeni zararlı yazılım örnekleriyle karşı karşıya kalmaktadırlar. Ortaya çıkan bu kadar çok sayıda tehdit içeren uygulamaların tek tek incelenmesi mümkün olmamaktadır. Bu sebeple, bilgisayar sistemlerinin güvenliğini tehdit eden zararlı yazılımların otomatik çalışan kontrollü ortamlarda analiz edilmesi tekniği etkin bir çözüm olarak geçerliliğini korumaktadır. Hali hazırda son kullanıcıların kullanımına açık çok yetenekli dinamik zararlı yazılım analiz sistemleri bulunmasına rağmen, bu sistemler yeni işletim sistemlerini hedefleyen zararlı yazılımlara karşı zayıf kalmaktadırlar.

Bu çalışmada, Windows işletim sistemlerini hedefleyen zararlı yazılımların davranış tabanlı analizi için geliştirilmiş güçlü bir sistem olan Virmon *(Virus Monitor)* tanıtılmıştır. Virmon zararlı yazılım analiz sistemi farklı bileşenlerden oluşan büyük bir ağ topolojisine sahip, dağıtık bir sistemdir. Analiz edilmek istenen şüpheli yazılımların sayısının artarak devam etmesine karşı, sistem kapasitesi yeni donanımlar eklenerek doğrusal bir şekilde artırılabilmektedir.

Virmon, analizini gerçekleştirdiği yazılımların proses, registry ve dosya sistemi aktivitelerini çekirdek düzeyinde toplamaktadır. Bu aktiviteler, geliştirilen bir dosya sistemi filtreleme sürücüsü aracılığıyla izlenmektedir. Windows işletim sistemi tarafından sunulan çekirdek geri bildirim mekanizması, sürücülerin çalışma zamanında ihtiyaç duydukları bilgilere erişibilmelerine imkan tanımaktadır. Analiz makinesi bileşenleri olan Proses, Registry ve Dosya Sistemi İzleyici'leri, sürücünün sisteme yüklenme zamanında kendilerini çekirdek geri bildirim mekanizmasına kayıt ettirmeleriyle ilgilendikleri olaylardan haberdar olabilmektedirler.Virmon, analiz edilen dosyaların davranış bilgilerini legal bir yöntem olan geri bildirim mekanizması aracılığıyla topladığı için tüm Windows işletim sistemlerini analiz ortamı olarak kullanabilmektedir. Bu durum Virmon'un ileride çıkması muhtemel olan işletim sistemlerine de kolay bir şekilde uyum sağlanmasına imkan tanımaktadır.

Analizi gerçekleştirilen uygulamaların ağ aktiviteleri sistem genelinde, IPS/IDS, HTTP, DNS, NetFlow, VPN gibi farklı ağ çözümleri kullanılarak toplanmaktadır. Gerçeklenen dağıtık sensör ağı, analiz makinelerinin ağ trafiğini internet üzerinde farklı lokasyonlara dağıtarak IP tabanlı analiz ortamı tespit etme tekniğini kullanan zararlı yazılımlara karşı sistemin gizlenmesine yardımcı olmaktadır.

Analiz işlemlerinin otomatik gerçekleştirilmesi aynı anda birden fazla şüpheli dosyanın incelenmesine olanak sağlamaktadır. Tüm analiz sürecini yöneten uygulama sunucusu ve analiz makineleri üzerindeki süreci yöneten analiz uygulamaları ağ üzerinden Thrift kütüphanesinin yardımıyla haberleşmektedir. Thrift, farklı programlama dilleri ile geliştirilmiş olan uygulamaların güvenli bir

şekilde haberleşmesine olanak sağlamaktadır. Geliştirilmiş olan web uygulaması aracılığıyla da analiz sonuçları takip edilebilmektedir.

Virmon analiz sisteminin etkinliğini göstermek için yakın zamanda tespit edilmiş bir zararlı yazılım incelenmiştir. Analiz sonuçlarına göz atıldığında hedef uygulamanın gerçekleştirmiş olduğu tüm proses, registry ve dosya sistemi aktivitelerinin başarılı bir şekilde kayıt altına alınabildiği görülmüştür. Bunların dışında analiz edilen yazılımın yapmış olduğu ağ trafiğinin neticesinde ortaya çıkan saldırı alarmları, HTTP ve DNS isteklerinin de izlenebildiği belirlenmiştir.

VIRMON: A VIRTUALIZATION BASED AUTOMATED DYNAMIC MALWARE ANALYSIS SYSTEM

SUMMARY

Nowadays, the technology has been developing rapidly and the computer systems have become indispensable of our daily life. More and more people are making use of technology such as smartphones, smart TVs, tablets etc. As the people use technological devices in all aspects of life, this situation attracts attention of some people having malicious intents. These bad guys use distinct methods to gain access to computer systems selected as a target and to steal private information from them. Due to the fact that the size of things which are threatening the computer systems is too high, it is a must to provide the security of these systems.

One of the most important threats to the computer systems are malwares. Malwares (malicious softwares) is commonly referred to as executables used or created by an attacker to attain access to or steal sensitive information from a computer system. There are some terms used to classify malicious softwares which have similar behaviors such as virus, backdoor, rootkit, worm or bot. As the time being passed, the motivation of the malware authors has changed. Previously, one could write malware for fun and prominence, now for money, espionage and ideological purposes. Furthermore, malicious softwares and the zero day vulnerabilities used by malwares are becoming a commercial industry. Some companies such as VUPEN provide zero day vulnerabilities to the governments that prefer to use these vulnerabilities in cyber operations. In addition, some people and firms offer deal to the security researchers in order to collect zero day vulnerabilities. Besides the commercial industry, a cyber war between countries is observed. As an example, the Stuxnet malicious software which was detected in June 2010 can be given. It is claimed that the Stuxnet, one of the most known sophisticated malware (used for advanced persistent threat – APT attacks), is developed by USA and Israel so as to slow down or prevent the nuclear activities of Iran. This malicious software is regarded as the first malware targeting the Siemens industrial control systems and including PLC rootkit. Also, it has been asserted that the Stuxnet caused to two year delay in Iran's nuclear enrichment program and is cost an estimated US$1 million to create. These information help us to understand how big and vast the malware development research is.

To protect the end users from cyber threats, security companies develops solutions which aim to identify malwares and cyber attacks. Typically, these solutions (anti-viruses) employs signature-based detection technique in order to identify known malwares or threats.This method requires the anti-viruses to have a database of signatures. As it is reported by the security vendors, they are faced tens of thousands new malware samples in every single day. It is impossible to analyze manually these samples and create signatures so as to identify them since it takes too long time. Due to the ineffectiveness of anti-viruses on previously unknown malwares (whose signature have not been created yet), many researchers have introduced several

techniques to overcome limitation of anti-virus solutions. These techniques can be divided into two categories: *static* and *dynamic* analysis. Static analysis means examining a file without running on the system. Dynamic analysis means observing activities of a suspicious file by executing it in a controlled environment in order to understand its purpose. Because of the obfuscation techniques such as polymorphism, metamorphism, compression and encryption, it can be difficult and time consuming to statically analyze malwares. Therefore, analysis of softwares threatening the computer system's security in an automated fashion and in a controlled environment are still the valid choice.

Up to now, the security researchers developed several useful systems in order to detect these threats. To the best of our knowledge, today's dynamic analysis systems generally employ old versions of windows OS as their analysis environment. Nevertheless, the majority of the computer end users prefer to use the latest ones. It seems that they cannot analyze malwares targeting latest Windows OS appropriately. Therefore, it is a necessity that next generation malware analysis solutions should cover latest 64-bit OSs and be adapted easily to future OS versions. Even though there are highly skilled dynamic malware analysis systems, they are weak against malwares targeting new operating systems.

In this work, we present Virmon (Virus Monitor), a powerful system designed for behavior based analysis of malwares which target Windows operating systems. Virmon dynamic malware analysis system consists of different components. It is designed as a distributed system, that means, Virmon works on several servers connected by distinct networks. The system capacity can be increased linearly by adding new hardware against the increasing number of new malware samples.

Virmon collects the process, registry and file system activities of analyzed sofwares in low kernel level. These activities are monitored through a file system filter driver. Kernel callback mechanism presented by Windows operating system enables the drivers to access the necessary information which is needed at runtime. Process, Registry and File System Monitors which are components of analysis machines can get feedback about events that they are interested in via registering themselves at driver load time.Virmon can use all the Windows operating systems as an analysis environment due to the fact that it collects the behavior information of analysed files through Windows callback mechanism which is a legal method. This situation allows us to easily adapt the Virmon to Windows operating system versions which will be released in the next time.

The network activities of analyzed samples are collected system-wide by using different network solutions such as IPS/IDS, HTTP, DNS, NetFlow and VPN. The distributed sensor network can help to system to hide itself from the malicious softwares which use analysis environment detection technique based on public network addresses by distributing the network traffic of analysis machines to different locations on the internet.

Since the Virmon is an automated system, it is possible to analyze several malwares at the same time. The application server managing the whole analysis processes and the analysis applications running on machines can communicates with each other by the help of Thrift library. Thrift allows the applications which are developed with different programming languages to communicate securely. In addition, a web application is designed to monitor the activities of analyzed malwares.

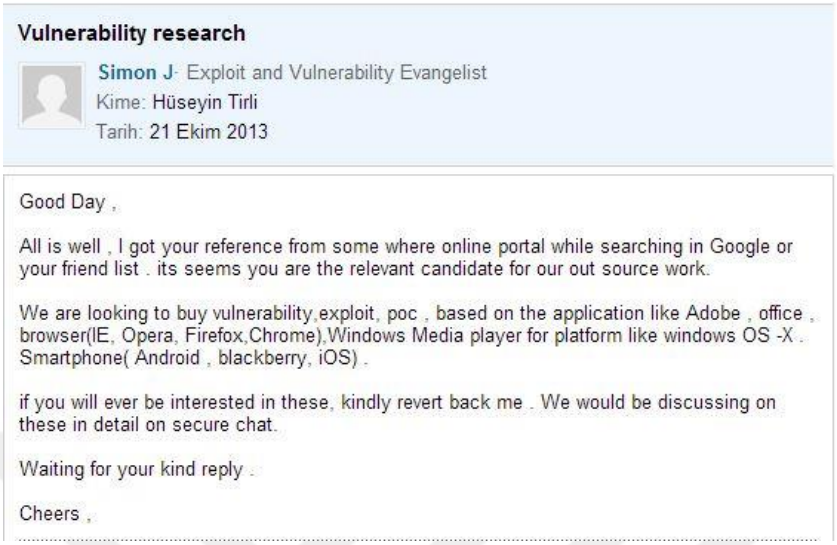
In order to demonstrate the effectiveness of Virmon, a recently observed malware sample is examined. When we look at the analysis result, it is clear that Virmon records successfully the all process, registry and file system events initiated by the analyzed sample. Additionally, it is determined that the intrusion alerts created by the result of network activities, the downloaded files, HTTP requests and DNS queries made by analysis machine are collected successfully.

1. GİRİŞ

Günümüzde teknoloji çok hızlı ilerlemekte, bilgisayar sistemleri son kullanıcılar da dahil olmak üzere hayatın her alanında vazgeçilmez olmaya başlamaktadır. Bilgisayar sistemlerinin yoğun bir şekilde kullanılır olması, kötü niyetli kişilerin de dikkatini çekmektedir. Bu kişiler hedef olarak belirledikleri bilgisayar sistemlerine erişim yetkisi kazanmak, kişesel bilgileri elde edebilmek için çok farklı yöntemler kullanabilmektedir. Bilgisayar sistemlerine yönelik tehditlerin boyutunun tahmin edilenden büyük olması siber güvenliğin önemini ortaya çıkarmakta ve bu sistemlerin güvenliğinin sağlanmasını zorunlu kılmaktadır.

Bilgisayar sistemlerinin güvenliğini tehdit eden unsurların başında zararlı yazılımlar gelmektedir. Geçmişte zararlı yazılımlar sadece gençlerin kendilerini ispatlamak ve eğlenmek amacıyla geliştirdikleri uygulamalar iken, günümüzde bu yazılımlar dolandırıcılık, casusluk ve ideolojik amaçların gerçeklenmesi için kullanılmaktadır. Zararlı yazılımlar ve kullanmış oldukları sıfırıncı gün açıklıkları, ticari birer endüstri ortaya çıkarmıştır. VUPEN gibi firmalar devletlere siber güvenlikteki amaçlarını gerçekleştirebilmeleri için sıfırıncı gün açıklıklarını kullanan exploitler temin etmektedirler [1]. Özellikle bazı firmalar sosyal medya üzerinden siber güvenlikle ilgilenen araştırmacılara tespit ettikleri uygulama zaafiyetlerini satmaları için anlaşma önermektedirler (Şekil 1.1). Ticari endüstrinin ötesinde, olaylar siber savaş boyutuna kadar uzanmış durumdadır. Bunun örneği de 2010 yılının Haziran ayında tespit edilen Stuxnet isimli zararlı yazılımda görülmektedir. Gelmiş geçmiş en kompleks zararlı yazılımlardan biri olarak kabul edilen Stuxnet'in İsrail ve Amerika Birleşik Devletleri tarafından, İran'ın nükleer faaliyetlerini yavaşlatmak ve mümkünse engellemek için geliştirildiği iddia edilmektedir [2]. Bu zararlı yazılım, Siemens endüstriyel kontrol sistemlerini hedef alan ve PLC rootkit içeren ilk yazılım olma özelliğine sahiptir. Stuxnet'in, İran'ın uranyum zenginleştirme programında tahmini olarak iki yıllık bir gecikmeye sebebiyet verdiği ve böyle kompleks bir zararlı yazılımın da ancak devlet tarafından finansal destek alan büyük bir takım tarafından geliştirilebileceği sıralanan diğer iddialar arasında bulunmaktadır. Bu tür

saldırıların bir ülkenin can damarını oluşturan elektrik, su doğalgaz gibi kritik altyapı sistemlerine yapılması durumunda ortaya çıkabilecek zorlukların düşünülmesi de konunun önemini vurgulamaktadır.



Şekil 1.1 : Sosyal medya üzerinden gelen örnek bir mesaj.

Zararlı yazılımların verebileceği zararın boyutu ile birlikte önem arz eden bir diğer konu da yapılan saldırıların miktarıdır. Kaspersky firmasının açıklamış olduğu verilere göre 2012 yılında günlük ortalama 200000 yeni zararlı yazılımın tespit edildiği anlaşılmaktadır [3]. Bu kadar zararlı yazılımın tek tek incelenerek tespit edilmesi mümkün olmamaktadır. Yeni zararlı yazılımların tespiti analiz işlemini otomatikleştiren, ölçeklenebilir sistemlerin geliştirilmesi ile sağlanabilmektedir.

Günümüzde şüpheli dosyaların zararlı yazılım olup olmadığını anlamak için geliştirilmiş çok sayıda kamuya açık ya da özel sistemler bulunmaktadır. Bu sistemler başarılı bir şekilde çalışıp dosyalar hakkında önemli bilgiler sunabilmektedir. Bununla birlikte bu sistemler analiz ortamı olarak Windows XP 32 bit gibi eski işletim sistemi versiyonları kullanmakta, bu durum yeni işletim sistemlerini hedefleyen zararlı yazılımların analizi konusunda sıkıntılar doğurmaktadır. Bu sistemlerin kullanmış olduğu teknikler çekirdek kodunu değiştirme gibi legal olmayan yöntemler olduğu için artık işletim sistemleri tarafından engellenmektedir. Bu nedenle işletim sisteminin sunmuş olduğu özellikleri kullanarak zararlı yazılım analizi yapabilen yeni bir sisteme ihtiyaç duyulmaktadır. Legal yollardan analiz yapabilen bir sistemin yeni işletim sistemi versiyonlarına kolayca adapte olabileceği de düşünülmektedir.

Bu çalışmada zararlı yazılım olduğundan şüphelenilen dosyaların analizini gerçekleştirebilecek sanallaştırma tabanlı, otomatikleştirilmiş ve ölçeklenebilir bir sistem tasarlanmış ve gerçeklenmiştir. Geliştirilen sistem tüm Windows işletim sistemlerini hedef alan zararlı yazılımların davranış tabanlı analizini gerçekleştirmeyi hedeflemektedir.

Virmon *(Virus Monitor)* zararlı yazılım analiz sistemi, amacının ne olduğu öğrenilmek istenen uygulamaları, ağ erişim izolasyonu sağlanmış kontrollü ortamlarda otomatik olarak çalıştırarak izlemektedir. Analizi yapılan uygulamanın işletim sistemi üzerinde gerçekleştirmiş olduğu proses, registry, dosya sistemi ve ağ aktiviteleri takip edilerek rapor üretilmektedir. Virmon, Windows işletim sistemi tarafından sunulan çekirdek geri bildirim mekanizması aracılığıyla ağ aktiviteleri dışındaki olayları gözlemlemektedir. Kullanılan yöntemin işletim sistemi tarafından izin verilen legal bir teknik olması, ileride çıkabilecek yeni Windows versiyonlarına da kolay bir şekilde adapte edilebilmesine olanak sağlamaktadır. Ağ aktiviteleri analiz sistemi genelinde HTTP, DNS, IDS, NetFlow gibi farklı tipteki ağ teknolojileri kullanılarak izlenmektedir. Toplanan bu veriler analizi yapılan yazılımın davranışını ortaya koymada yardımcı olmaktadır.

Tez çalışması 5 ana bölümden oluşmaktadır. İlk bölümde siber güvenliğin önemine ve tezin amacına yönelik genel bir değerlendirmede bulunulmuştur.

İkinci bölüm zararlı yazılım türleri ve fonksiyonlarını, zararlı yazılımları hedef sistemlere bulaştırmak için kullanılan ve zararlı yazılımların analizi için şimdiye kadar ortaya konulup hali hazırda kullanılan teknikleri içermektedir.

Üçüncü bölümde literatürde geçen ve daha önceden yapılan benzer çalışmalar hakkında bilgi verilmektedir.

Dördüncü bölüm sistem bileşenlerinin tasarımını içermekte, analiz için kullanılan tekniklerin ve gerçeklenen uygulamaların detaylı anlatımını yapmaktadır.

Beşinci bölümde örnek bir zararlı yazılımın analizi yapılmıştır. Analiz sonucu elde edilen bilgiler gösterilmiş ve açıklanmıştır.

Son bölümde sistemde bulunan kısıtlar belirtilmiş ve gelecek çalışmalarda nelerin yapılabileceği hakkında bilgi verilerek çalışma sonlandırılmıştır.

2. ZARARLI YAZILIMLAR

Kötü niyetli kişiler tarafından bilgisayar sistemlerine yetkisiz erişim elde etmek, sistemleri çalışmaz hale getirmek ya da bu sistemlerdeki hassas bilgileri çalmak amacıyla oluşturulmuş, çalıştırılabilir her türlü dosyaya *zararlı yazılım* denilmektedir [4]. Zararlı yazılım ifadesi İngilizce *malware (malicious software)* teriminin karşılığı olarak kullanılmaktadır. Bu bölümde, ilk olarak zararlı yazılım türleri ve işlevleri hakkında bilgi verilecektir. Akabinde, zararlı yazılımların analizi için kullanılan tekniklerden bahsedilecektir.

2.1 Zararlı Yazılım Çeşitleri

Günümüz güvenlik dünyasında zararlı yazılımlar, göstermiş oldukları benzer davranışlara göre sınıflandırılmıştır. Bununla birlikte, zararlı yazılım sınıfları birbirlerini tamamen dışlamamaktadırlar. Sınıflar arasında yüksek oranda benzerlikler görülebilmekte, spesifik zararlı yazılım örneklerinde birden fazla sınıfın özellikleri bulunabilmektedir. Bu bölümde, zararlı yazılım türleri hakkında genel bir bilgi verilecektir. Zararlı yazılım türleri hakkında yapılan daha detaylı çalışmalar da bulunmaktadır [5].

2.1.1 Adware

Bu tip yazılımlar, bir bilgisayar kullanıcısına istemediği halde reklam sunmaktadır. Bazı güvenlik araştırmacıları, reklam yazılımlarını zararlı yazılım olarak sınıflandırmamaktadır.

2.1.2 Backdoor

Arka kapı olarak ifade edilen bu yazılımlar, saldırgan kişilerin kullanıcı yetkilendirme işlemini atlatarak hedef sistemlere erişebilmelerine olanak sağlamaktadır.

2.1.3 Bot

Bu yazılımlar, saldırganın ele geçirdiği sistemleri uzaktan yönetebilmesine olanak sağlar. Çok sayıda botu içeren ağlar *botnet* olarak ifade edilmektedir. Botnetler genellikle spam mail göndermek ve DDOS adı verilen dağıtık servis dışı bırakma saldırıları için kullanılmaktadırlar.

2.1.4 Dropper

Zararlı yazılım geliştiricileri bu tip yazılımları, ele geçirmek istedikleri sistemlerde yeni zararlı yazılımları internet üzerinden indirmek ve kurmak için kullanırlar. Dropper, yeni kurulacak olan zararlı yazılımı indirmekle birlikte kendi içinde de barındırabilir.

2.1.5 Rootkit

Bu tip zararlı yazılımlar kendilerini, bulaştıkları sistemin kullanıcısından saklayabilirler. İşletim sistemi üzerinde belirli bilgileri değiştirebilmeleri proses, dosya, ağ bağlantısı gibi aktivitelerini gizlemelerine olanak sağlar. Rootkitler kullanıcı ve çekirdek modunda çalışabilirler.

2.1.6 Spyware

Hedef bilgisayarlar üzerindeki hassas bilgileri saldırganın belirlediği bir yere transfer etmek amacıyla kullanılan zararlı yazılımlardır.

2.1.7 Truva atı

İyi niyetli bir yazılım gibi gözüküp, arka planda zararlı aktivitelerde bulunan yazılımlar *trojan horse* (truva atı) olarak kategorilendirilmektedir.

2.1.8 Virüs

Kendisini başka programlara ekleyen yazılımlardır. Virüsler tek başlarına çalışamazlar. Aktivitelerini yerine getirebilmek için host uygulamalara ihtiyaç duyarlar.

2.1.9 Worm

Bağımsız çalışabilen ve kendisini başka makinelere bulaştırabilen zararlı yazılımlar *worm* (solucan) olarak isimlendirilmektedir. En meşhur worm örneklerinden biri

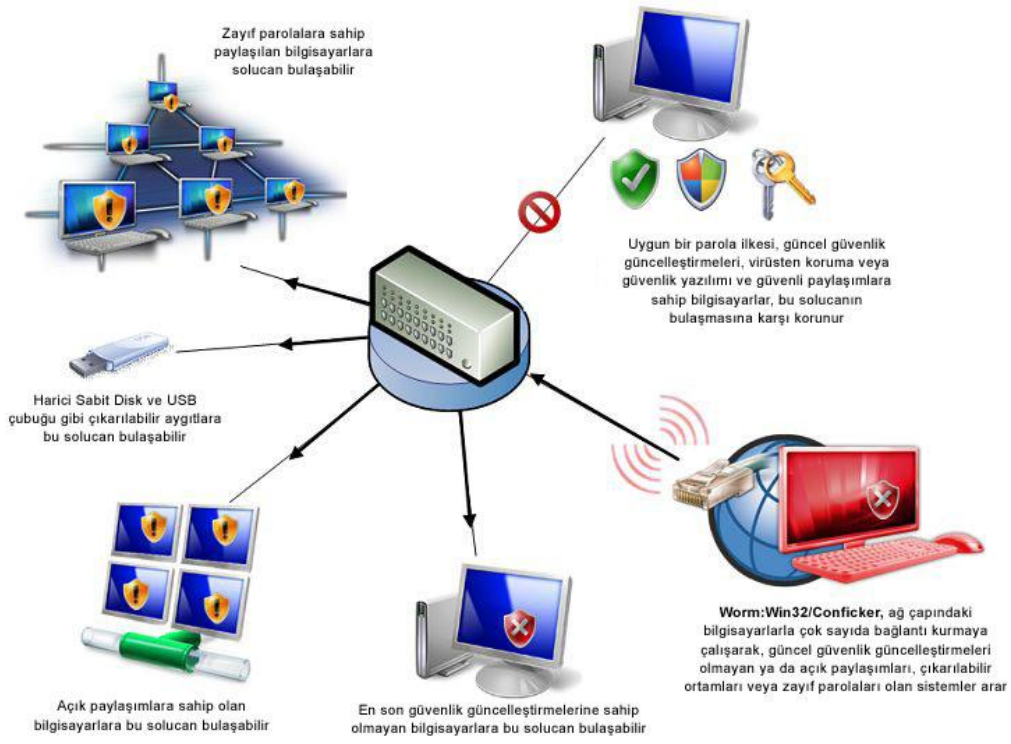
olup 2008 yılında tespit edilen *Conficker*, ağ üzerinden otomatik olarak 200 ülkede milyonlarca Windows makineye bulaşmıştır [6].

2.2 Bulaşma Teknikleri

Bu bölüm zararlı yazılımların hedef makinelere bulaşabilmesi için kullandıkları teknikleri açıklamaktadır. Örnek olaylarla zararlı yazılımların nasıl yayıldığı hakkında bilgi verilecektir.

2.2.1 Ağ üzerinden zaafiyetli servislerin exploit edilmesi

Bu teknikte, internet üzerinde kullanıcılara hizmet veren sunucular hedef alınmaktadır. Bu tip sunucular genellikle dosya paylaşım, web ve DNS hizmeti veren sunuculardan oluşmaktadır. Bu tip sunucular üzerlerinde belirli ağ servislerini barındırmaktadırlar. Bu servislerin üzerinde tespit edilmiş ya da bilinmeyen zaafiyetlerin bulunması durumunda, bu zaafiyetler exploit edilerek sistem ele geçirilebilmektedir. Örnek olarak Conficker zararlı yazılımı verilebilir. Bu yazılım, herhangi bir kullanıcı etkileşimi olmadan ağ üzerindeki dosya paylaşım servisleri üzerindeki zaafiyetleri kullanarak diğer bilgisayarlara yayılabilmektedir [7]. Şekil 2.1 conficker zararlı yazılımının çalışma tekniğini göstermektedir.

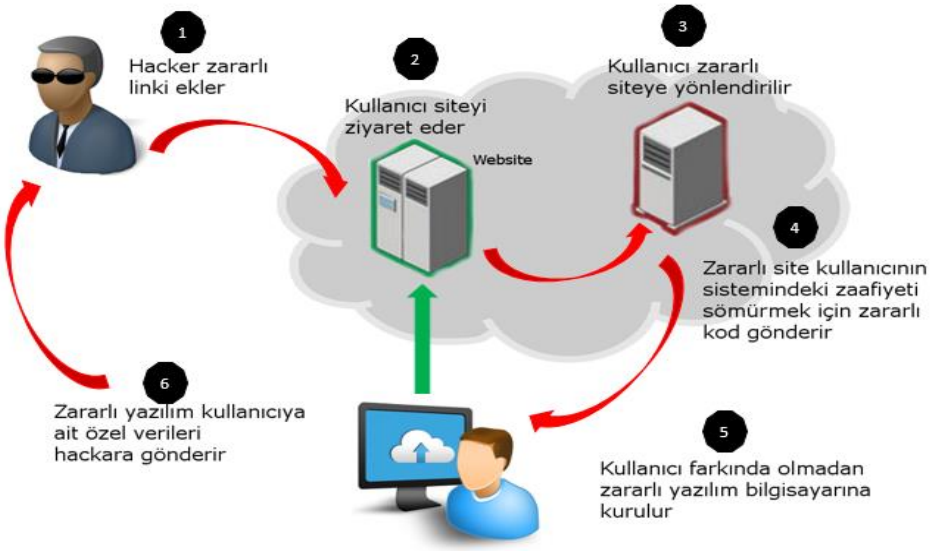


Şekil 2.1 : Conficker zararlı yazılımının çalışma tekniği.

Bir diğer örnek uzaktan erişime açık olan sunucuların yetkilendirmesinin sözlük saldırıları *(dictionary attacks)* ile kırılmasıdır. Bu tip saldırılar ile güçlü parolalara sahip olmayan servisler otomatik olarak kırılabilmekte ve sunucuya zararlı yazılım bulaştırılabilmektedir.

2.2.2 Drive-by download

Bu bulaşma tekniğinde saldırganlar genel olarak web browser uygulamalarını hedef alırlar. Web browserlar üzerinde bulunan zaafiyetler exploit edilerek hedef makineler üzerinde zararlı kod çalıştırılabilmektedir. Şekil 2.2'de drive-by download tekniği ile hedef sistemlere nasıl zararlı yazılım bulaştırıldığı gösterilmiştir. Bu teknikte ilk olarak saldırgan kişiler, çok fazla ziyaret edilen bir web sitesini ele geçirirler. Ele geçirilen sitenin kaynak koduna zararlı içerik barındıran bir url eklenilir. Son kullanıcılar bu siteyi ziyaret edip zararlı linke tıkladıklarında kötü amaçla kurulmuş bir web sitesine yönlendirilirler. Zararlı içerik barındıran site son kullanıcının haberi olmadan, zaafiyet barındıran web browser uygulaması aracılığıyla zararlı kodu çalıştırır. Zararlı yazılım hedef sisteme bu şekilde kurulmuş olur [8].

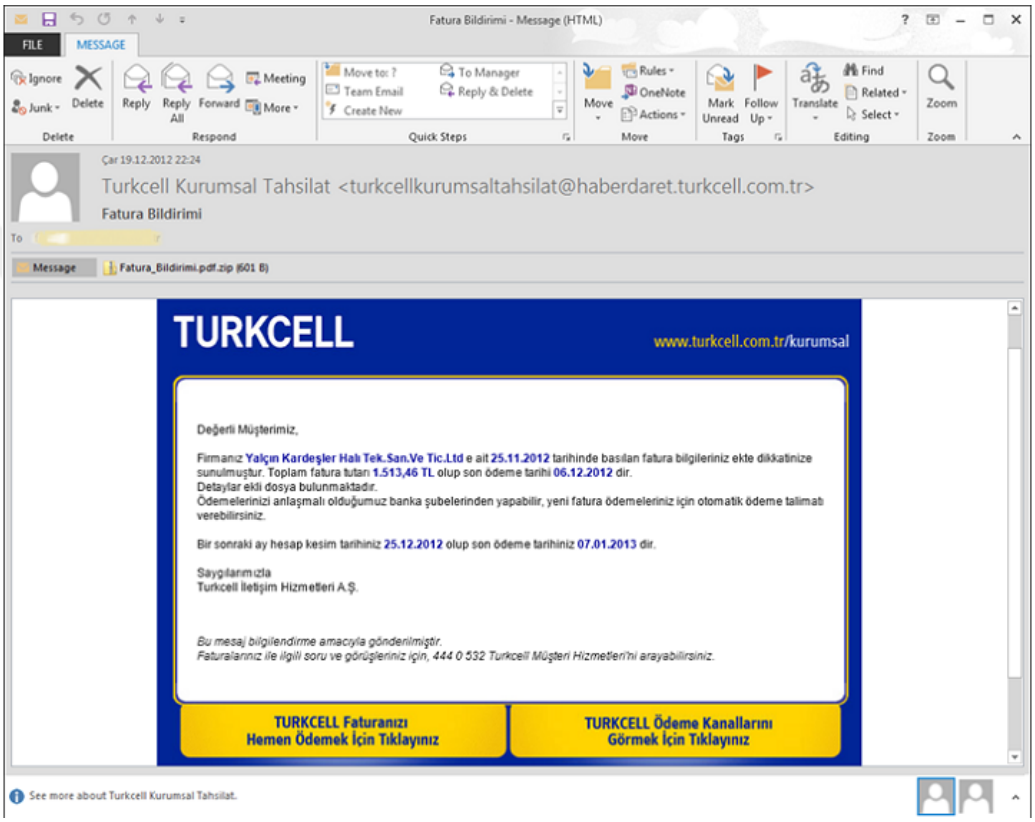


Şekil 2.2 : Drive-by download ile zararlı yazılım bulaştırma.

2.2.3 Sosyal mühendislik

Son kullanıcıların, çeşitli yollarla kandırılırak zararlı yazılımların kullanıcı makineleri üzerinde çalıştırılması tekniğidir. Sosyal mühendislik saldırılarında en çok kullanılan yöntem mail yoluyla yapılan oltalama saldırılarıdır. Hedef seçilen kişiye gönderilen mail içeriğindeki linke tıklanması ya da ekte gönderilen dosyaların

çalıştırılması ile kurbanın makinesi ele geçirilebilmektedir. Örnek bir olay geçtiğimiz yıl ülkemizde yaşanmıştır. Hedef seçilen banka ve GSM müşterilerine özel hazırlanmış mailler ile fatura eki gibi gösterilen *Fatmal* isimli zararlı yazılım dağıtılmıştır. Bu zararlı yazılım Türkiye'de çok sayıda kişinin makinesine bulaşmıştır. Ek olarak gönderilen dosyanın uzantısı pdf gibi gösterilerek kullanıcılar tarafından farkedilmesi zorlaştırılmaya çalışılmıştır [9]. Şekil 2.3'te Fatmal isimli zararlı yazılımın dağıtımı için kullanılan örnek bir oltalama maili gösterilmiştir.



Şekil 2.3 : Fatmal zararlı yazılımının dağıtımında kullanılan oltalama maili.

2.3 Analiz Teknikleri

Günümüz anti-virüs uygulamaları büyük oranda imza tabanlı çalışmaktadırlar. Bir zararlı yazılımın bu uygulamalarca tespit edilebilmesi için imzasının, anti-virüsün imza veritabanında bulunması gerekmektedir. İmza oluşturma işlemi de genellikle el ile yapılmaktadır. Şüpheli bir dosyanın asıl amacını anlayabilmek ve imza oluşturabilmek için bugüne kadar çeşitli teknikler önerilmiştir. Bu teknikler genel olarak iki farklı başlık altında incelenmektedir: *statik* ve *dinamik analiz*. İki yöntem arasındaki temel fark analiz edilmek istenen dosyanın çalıştırılıp çalıştırılmamasıdır. Analiz tekniklerinin ilerlemesi ile birlikte, zararlı yazılım geliştiricileri de kendi

uygulamalarının analiz edilmesini zorlaştırmak ya da engellemek için paketleme, karmaşıklaştırma gibi çeşitli yöntemler ortaya koymuşlardır. Bu bölümde, bahsedilen iki farklı teknik, analiz işlemi açısından avantajları ve dezavantajları ile birlikte açıklanacaktır.

2.3.1 Statik analiz

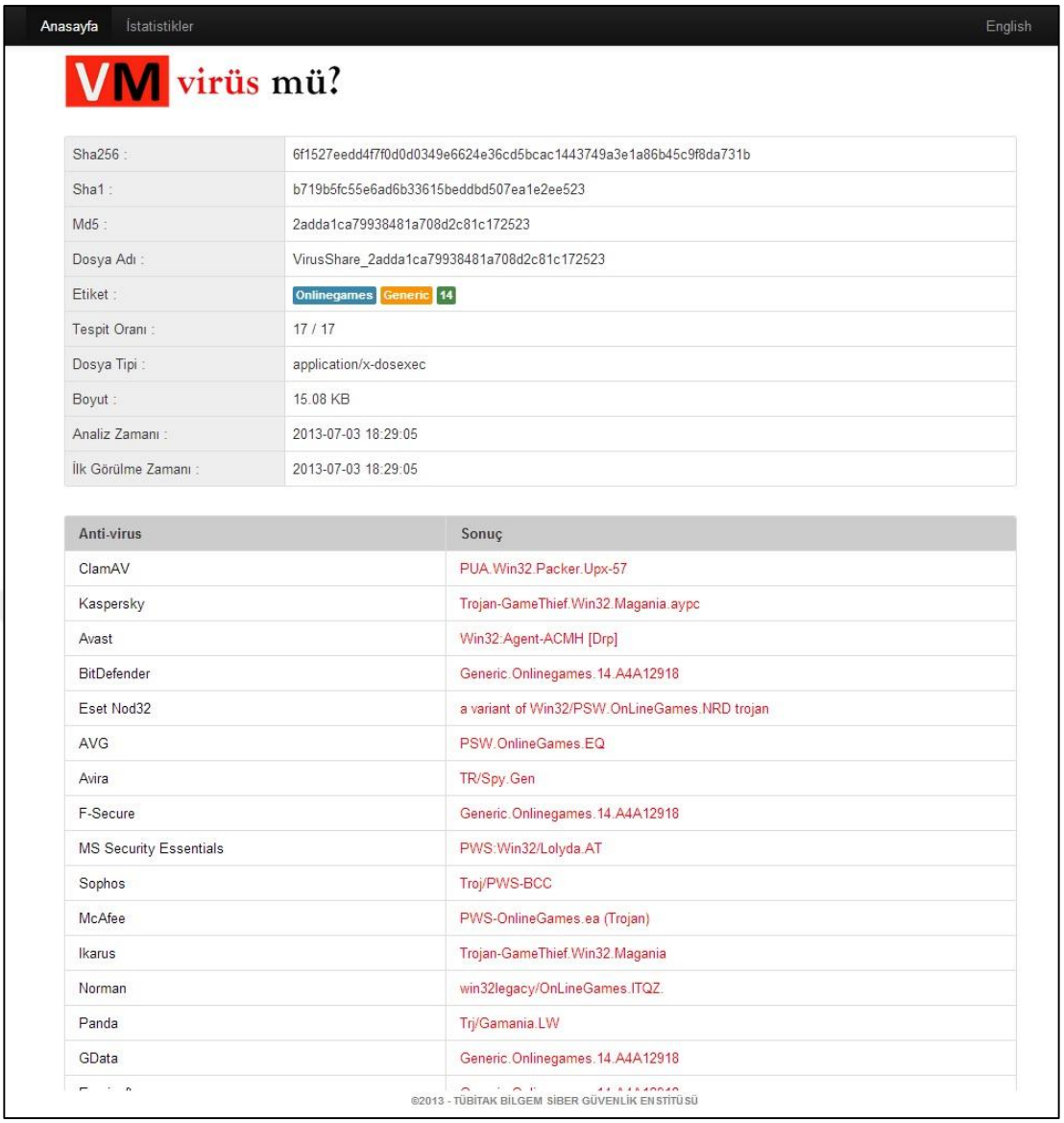
Statik analiz tekniği, hedef programın kodunun veya yapısının incelenmesi olarak tanımlanabilir. Bu inceleme işlemi esnasında program çalıştırılmaz. Programa ait kaynak kod mevcut ise çok değerli veriler elde edilebilir. Bununla birlikte çoğu zararlı yazılımın kaynak kodu mevcut değildir ve analiz işleminin assembly kodu üzerinde yapılması gerekir. Statik analiz işlemi hızlı gerçeklenebilmekle birlikte, zararlı yazılım geliştiricileri tarafından kullanılan karşı teknikler bu tip analiz işlemlerini zorlaştırmakta ve analiz için harcanan süreyi de artırmaktadır [10]. Takip eden bölümler en çok kullanılan statik analiz tekniklerinden bahsedilmektedir.

2.3.1.1 Anti-virüs taraması

Şüpheli bir dosyanın statik analizine başlarken atılacak en mantıklı ilk adım, dosyanın çoklu anti-virüs sistemlerinden geçirilmesidir. Daha öncede bahsedildiği üzere anti-virüs sistemleri imza tabanlı çalışmaktadırlar. Eğer zararlı yazılıma ait bir imza veritabanlarında mevcut değilse, ilgili dosya anti-virüsler tarafından tespit edilememektedir. Bilinen bir zararlı yazılımın kodunun bir miktar değiştirilmesi de anti-virüs sistemlerini kolayca atlatabilmektedir. Bu yüzden tek anti-virüs sistemi kullanmak yerine çok sayıda anti-virüs içeren daha büyük bir sistemin kullanılması daha doğru olabilir. Şekil 2.4'te çoklu bir anti-virüs sisteminden geçirilen bilinen bir zararlı yazılımın analiz sonuçları gösterilmektedir.

2.3.1.2 Kriptografik özet çıkarma

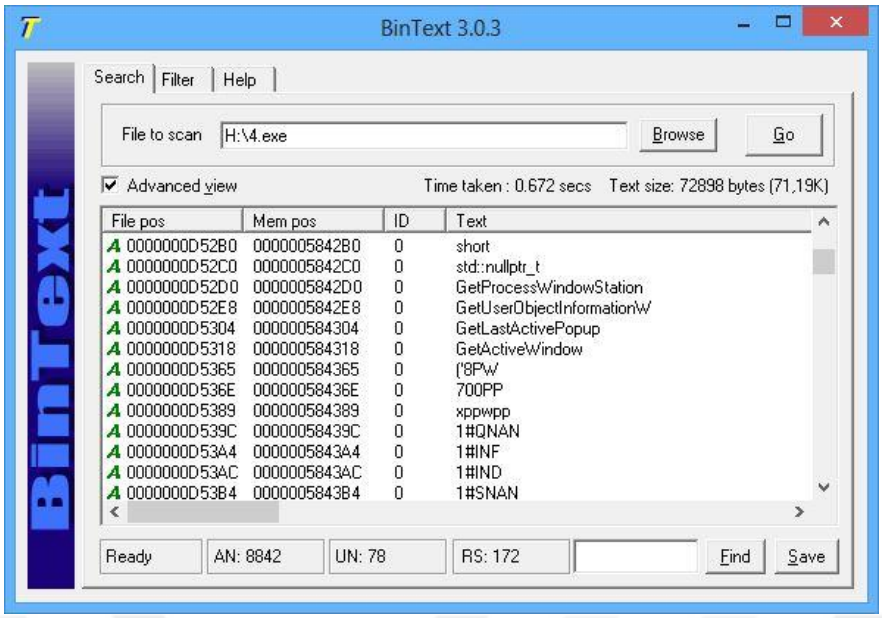
Kriptografik özet (*hash*) çıkarma, zararlı yazılımları tekil olarak tanımlamak için en sık kullanılan yöntemlerden biridir. MD5, SHA1, SHA256 gibi çok sayıda kriptografik özet algoritması bulunmaktadır. Şüphe edilen bir dosyanın hash değeri diğer analistler ile paylaşılarak onların bu yazılımı da tanıması sağlanabilir, hesaplanan değer internet üzerinde aratılarak da daha önceden tespit edilip edilmediği öğrenilebilir.



Şekil 2.4 : Bilinen bir zararlı yazılımın çoklu anti-virüs tarama sonuçları.

2.3.1.3 Metin bulma

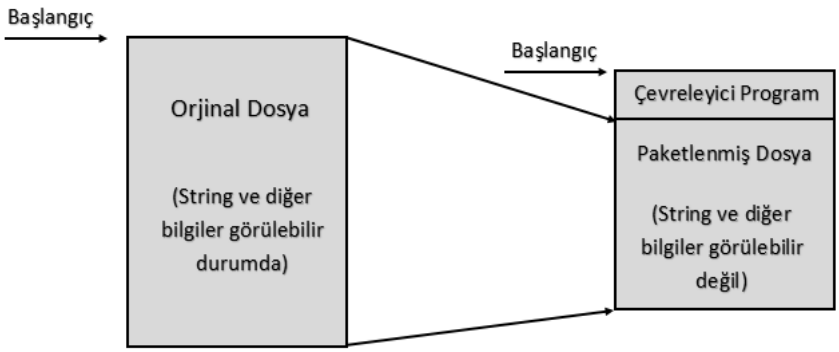
Uygulamalar içerisindeki metinleri bulma statik analiz için büyük önem arz etmektedir. Bulunan bu metinler, zararlı yazılımların bağlanmak istediği komuta kontrol sunucuların *(CC)* adreslerini, uygulamanın import ettiği dll dosyalarını, kullandığı Windows API fonksiyonlarını içerebilir. Bu bilgiler de hedef dosyanın ne yapmak istediğinin anlaşılmasına yardımcı olmaktadır. Şekil 2.5'te örnek bir zararlı yazılımın içerisindeki metinleri çıkaran BinText aracı gösterilmiştir.



Şekil 2.5 : BinText ile zararlı yazılımların içerisindeki metinlerin çıkarılması.

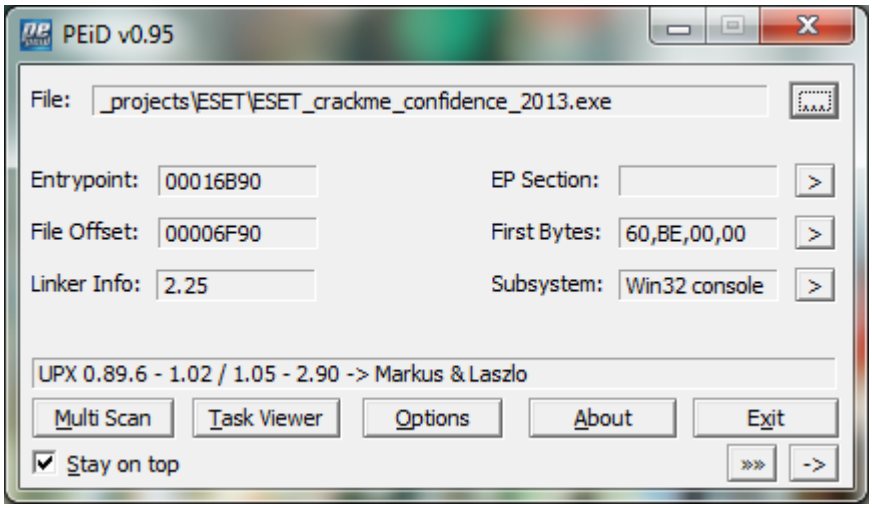
2.3.1.4 Paketleyici tespiti

Zararlı yazılım geliştiricileri, uygulamalarının statik analizinin yapılmasını zorlaştırmak için uygulamanın saklamak istedikleri kısımlarını sıkıştırırlar. Normal uygulamalar bol miktarda metin içerirlerken, paketlenmiş zararlı yazılımlar tersine az miktarda içerirler. Şekil 2.6'da aynı ugulamanın normal ve paketlenmiş halleri görülmektedir. Paketlenmiş dosya çalıştırıldığında uygulamanın içinde bulunan küçük bir *wrapper* program paketlenmiş dosyayı açarak çalıştırmaktadır.



Şekil 2.6 : Normal ve paketlenmiş uygulamanın görünümü.

Paketlenmiş uygulamaların tespiti için geliştirilmiş uygulamalar bulunmaktadır. Örnek bir uygulama olan PeiD Şekil 2.7'te göstermiştir. Bu uygulama içerdiği pluginler sayesinde farklı algoritmalarla paketlenmiş uygulamaları tespit edebilmektedir. Eğer zararlı yazılım bilinmeyen bir yöntemle paketlenmiş ise bunun tespiti kolay olmamaktadır.



Şekil 2.7 : PEiD uygulaması ile paketleyici tespiti.

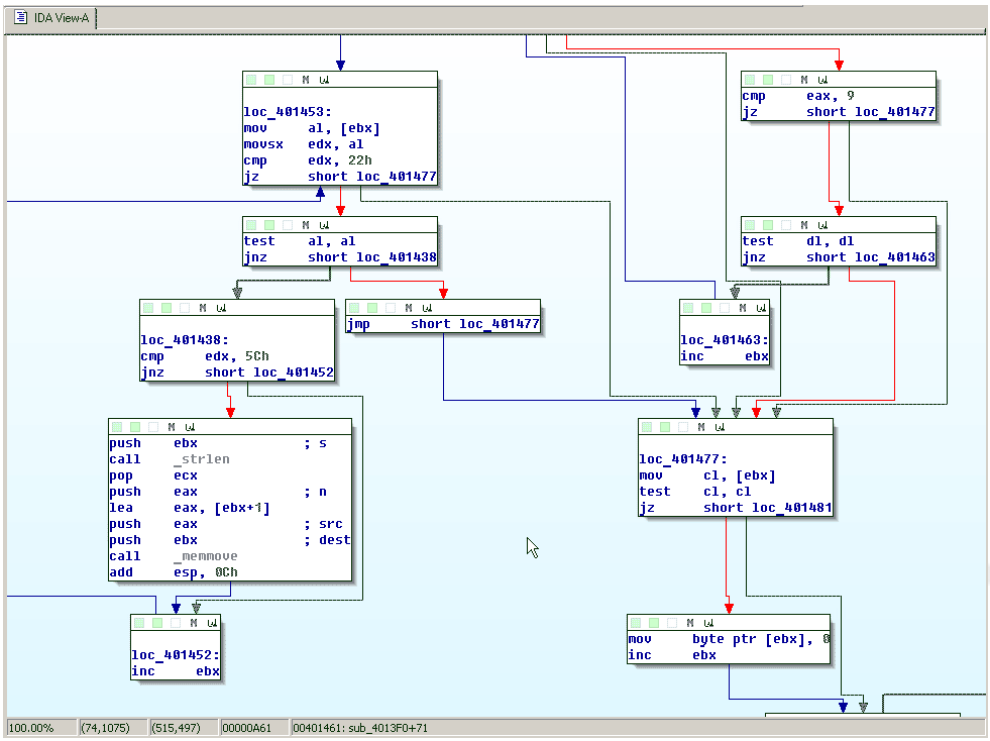
2.3.1.5 Disassembly

Statik analiz tekniklerinin en önemlisi denilebilir. Hedef uygulamanın makine kodlarının assembly kodlarına ayrıştırılması ile program akışı incelenebilir ve uygulamanın asıl yapmak istedikleri hakkında bilgi sahibi olunabilir. Bu teknik için kullanılan programlardan en popüler olanı IDA Pro'dur. Şekil 2.8, örnek bir uygulamanın IDA Pro ile disassembly edilmiş halini göstermektedir. Zararlı yazılımlar tarafından kullanılan anti-dissassembly teknikleri ile kodlar karmaşıklaştırılabilmekte, olduğundan farklı bir program akışı ile karşı karşıya kalınabilmektedir.

2.3.2 Dinamik analiz

Analiz edilmek istenen şüpheli dosyanın kontrollü bir ortamda çalıştırılıp, sistem üzerindeki aktivitelerinin izlenerek amacının belirlenmesi işlemine *dinamik analiz* denilmektedir [11]. Dinamik analiz, zararlı yazılım analizinde genellikle statik analizden sonra ikinci aşamayı teşkil eder. Karmaşıklaştırma, paketleme gibi zararlı yazılımların kullandığı analiz zorlaştırma yöntemlerinden dolayı, statik analiz teknikleri kullanılarak dinamik analiz metodları ile elde edilebilecek sonuçlara ulaşabilmek pek mümkün olmamaktadır. Bu yöntemler kullanılarak zararlı yazılımın sistem üzerinde gerçekten ne yaptığı sorusuna cevap bulunabilmektedir. Dinamik analiz işlemlerinin otomatikleştirilebilmesi analiz işlemini kolaylaştırarak çok geniş ölçekli analizlerin yapılabilmesine olanak sağlamaktadır. Avantajları ile birlikte dinamik analiz yönteminin de belli eksik yönleri bulunmaktadır. Dinamik analiz ile, hedef uygulamaya ait kodun çalışma yollarından *(execution paths)* yalnızca biri

incelenebilmektedir. Özellikle komut satırından parametre alan uygulamaların analizi tam olarak yapılamayabilmektedir. Farklı çalışma yollarının da içerilebilmesi için yapılmış çeşitli araştırmalar da bulunmaktadır [12]. Ayrıca analiz ortamının enfekte olması, zararlı yazılımın farklı ağlara ve makinelere bulaşabilme potansiyelinin bulunması da analiz ortamının izolasyonunun sağlanmasını zorunlu kılmaktadır. Dinamik analiz tekniklerinin en bilinenleri fonksiyon çağrılarının ve parametrelerinin izlenmesi ve hata ayıklama *(debugging)* yöntemleridir.



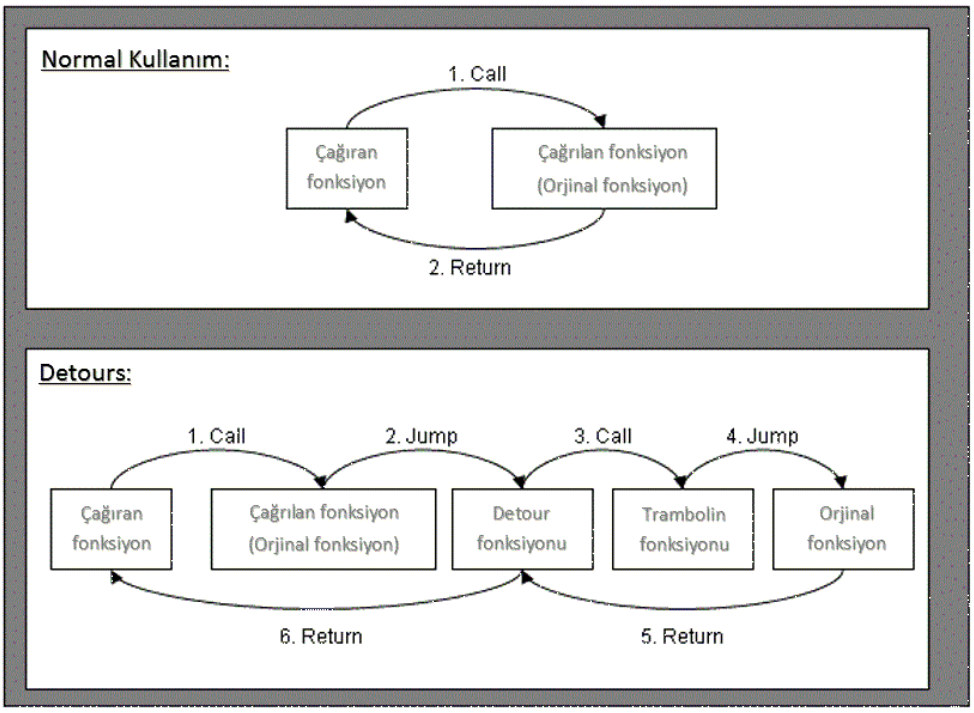
Şekil 2.8 : Örnek bir uygulamanın IDA Pro ile disassembly edilmiş hali.

2.3.2.1 Fonksiyon çağrıları izleme

Belirli bir işi yapan kod parçaları fonksiyon olarak tanımlanmaktadır. Fonksiyonlar, programlamada kod yeniden kullanılabilirliğini *(reusability)* sağlamak adına tasarlanmıştır. İşletim sistemleri de üzerinde çalışacak uygulamaların geliştirilmesini kolaylaştırmak adına API adı verilen uygulama programlama arayüzleri sunmaktadırlar. Her program gibi zararlı yazılımlar da işletim sistemi tarafından sunulan bu fonksiyonları sıklıkla kullanmaktadır. Zararlı yazılımlar tarafından çağrılan bu fonksiyonların tespiti, zararlı yazılımların asıl amaçlarını ortaya koymaya yardımcı olmaktadır. Bu teknik *hooking* olarak tanımlanmaktadır. Hooking tekniği ile bir uygulama tarafından bir fonksiyona yapılan çağrılar istenen bir fonksiyona yönlendirilebilir [13]. Çağrının yönlendirildiği bu yeni fonksiyonun, hook edilmek

istenen fonksiyonla aynı geri dönüş tipine, aynı sayıda ve tipte parametrelere sahip olması gerekmektedir. İlgili fonksiyona ait parametrelerin incelenmesi ile çağrının hangi amaçla yapıldığı anlaşılabilmektedir.

Kullanıcı ve çekirdek *(kernel)* uzayında olmak üzere iki farklı seviyede hooking yapılabilmektedir. Windows işletim sistemleri kullanıcı uzayında, Windows tarafından sunulan dökümantasyonu yapılan Windows API ve dökümantasyonu yapılmayan Windows Native API fonksiyonları hook edilebilmektedir. Bu işlem için geliştiriciler kendileri uygulama geliştirebildikleri gibi *Detours* gibi Microsoft tarafından sunulan hazır kütüphaneler de kullanabilmektedirler. Şekil 2.9, Detours'un hook işlemini nasıl gerçekleştirdiğini anlatmaktadır [14].



Şekil 2.9 : Detours hooking diyagramı.

Çekirdek uzayında hooking işlemi, işletim sistemi çekirdeğine kod enjekte edilerek yapılmaktadır [15]. Bu yöntemde hedef alınan fonksiyonlar sistem çağrılarıdır. Sistem çağrılarına yapılan isteklerin başka fonksiyonlara yönlendirilebilmesi zararlı yazılım analizcilerinin kullandığı bir teknik olmakla birlikte, zararlı yazılım geliştiricilerinin de sıklıkla kullandığı bir yöntem olmuştur. Bu yöntemle tespiti çok zor olan çekirdek seviyesinde çalışan rootkitler geliştirilebilmiştir. Windows çekirdeğinin değiştirilebilme özelliğinin, zararlı yazılım geliştiricileri tarafından kötüye kullanılması Microsoft'u yeni arayışlara itmiş ve XP Service Pack 3 ile

birlikte *patchguard* adı verilen bir çekirdek koruma mekanizması geliştirilmiştir [16]. Bu mekanizma, windows çekirdeğine kod enjekte edilmek istendiğinde devreye girmekte ve sistemin mavi ekran *(BSOD)* vererek çökmesini sağlamaktadır. Bu mekanizmanın devreye sokulması hem zararlı yazılım geliştiricilerinin hem de analistlerin işini zorlaştırmıştır.

2.3.2.2 Debugging

Debugger, başka bir programın çalışma akışını incelemek için kullanılan yazılım veya donanım parçasıdır [17]. Bu araçlar, analiz edilmek istenen uygulama çalıştırıldığında dinamik bir bakış açısı kazanılmasını sağlar. Örneğin, programın çalışma akışıyla birlikte kullanılan bellek adreslerinin değişimleri izlenebilmektedir.

Debugging tekniği zararı yazılım analizi sırasında, analiste çok önemli verileri sunabilmektedir. Anlık olarak register değerleri, fonksiyonlara girdi olan parametreler debuggerlar aracılığıyla görülebilmektedir. Bu araçlar programın çalışması sırasında değiştirilmesine de olanak sağlamaktadır. Uygulamaların kaynak koduna ihityaç duyulmadan debug edilebilmesi zararlı yazılım analizi için büyük önem taşımaktadır. Hem kullanıcı, hem de çekirdek uzayında debugging yapılabilmektedir. Bu sayede çekirdek uzayında çalışan rootkitlerin de analizi mümkün olmaktadır.

.

3. BENZER ÇALIŞMALAR

3.1 Anubis

Bu system TTAnalyze isimli bir tez çalışmasının sonucunda ortaya çıkmıştır [18]. Sistem halihazırda online olarak dinamik analiz hizmeti vermektedir [19]. Bu sistem Qemu emülatörü üzerinde çalışan bir Windows XP işletim sisteminden oluşmaktadır. Anubis, uygulamalar tarafından Windows API ve Windows Native API fonksiyonlarına yapılan çağrıları parametreleri ile birlikte izlemektedir.

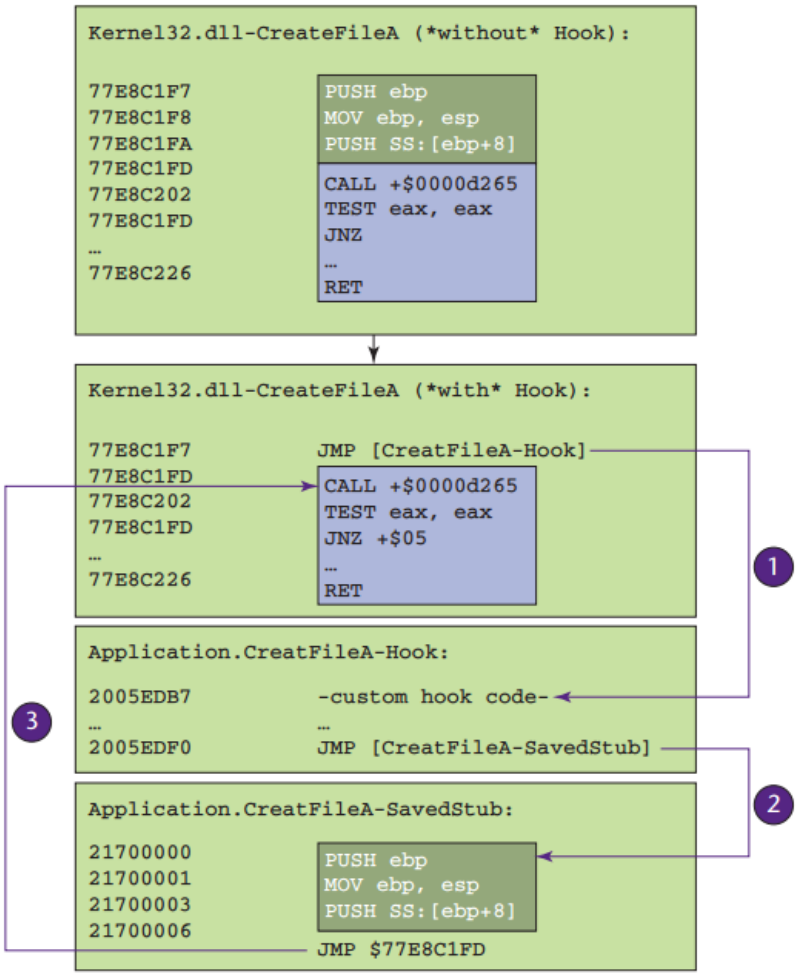
Anubis, analiz etmek istediği prosesleri CR3 merkezi işlemci birimi kütüğü *(register)* değerini kontrol ederek takip etmektedir. Bu kütük, halihazırda çalışan prosesin bellek üzerindeki fiziksel adresini tutmaktadır. İlgili proses tarafından seçilmiş bazı fonksiyonlara yapılan çağrılar, emüle edilmiş merkezi işlem birimine ait komut işaretçisinin *(instruction pointer)* anlık değeri izlenerek belirlenmektedir. İzlenmek istenen fonksiyonların bellekteki fiziksel adresleri çalışma zamanında dinamik olarak belirlenmektedir.

Bazı fonksiyonlara yapılan çağrıların izlenmesi ile birlikte Anubis, bu çağrılara gönderilen parametreleri de incelemektedir. Belirli callback rutinlerine gönderilen bu parametreler analiz edilerek fonksiyona yapılan çağrıların detayları çıkarılmaktadır. Örneğin Windows registry yapısında gerçekleştirilen değişimler ya da yeni oluşturulan dosyalar, bu callback rutinlerinde yapılan analizlerle belirlenmektedir.

3.2 CWSandbox

Williams, doktora çalışması kapsamında kullanıcı uzayında API hooking tekniğini kullanarak bir dinamik zararlı yazılım analiz sistemi geliştirmiştir [20]. Bu sistemin asıl hedefi dosya sistemi ve registry üzerindeki değişimlerin izlenmesi ve yapılan ağ aktivitelerinin takip edilmesidir. Analiz işlemleri, sanallaştırma sistemleri üzerinde 32-bit işletim sistemleri kullanılarak gerçekleştirilmektedir.

Şekil 3.1, CWSandbox tarafından kullanılan hooking tekniğini göstermektedir. Bu teknik, hedef proses belleğe yüklenir yüklenmez uygulanmaktadır. Hook edilmek istenen her bir fonksiyona özel bir hook fonksiyonu yazılmaktadır. Sistemin çalışma şekli şu şekildedir. Analiz edilmek istenen uygulama *suspend* modda çalıştırılarak uygulamanın kullanacağı kütüphanelerin belleğe yüklenmesi sağlanır. Uygulamaya ait bellekteki kodun ilk 5-baytı daha önceden yazılmış olan hook fonksiyonuna şartsız dallanma yapan kod parçası ile değiştirilir. Hook fonksiyonunda istenilen analiz işlemleri yapılır ve üzerine yazılan 5 baytlık kod çalıştırılarak orjinal fonksiyondaki kodun ilgili bölümüne şartsız dallanma yapılır.



Şekil 3.1 : CWSandbox tarafından kullanılan hooking tekniği.

CWSandbox genel olarak iki bileşenden oluşmaktadır: analiz işlemini kontrol eden ana proses ve hook fonksiyonlarını içeren dll. Sistem kendi içinde barındırdığı hook fonksiyonlarını dll enjeksiyon yöntemini kullanarak hedef prosesin uzayına entegre etmektedir. Bununla birlikte, zararlı yazılımların kendisini tespit etmesini engellemek için kullanıcı uzayında rootkit teknikleri CWSandbox tarafından

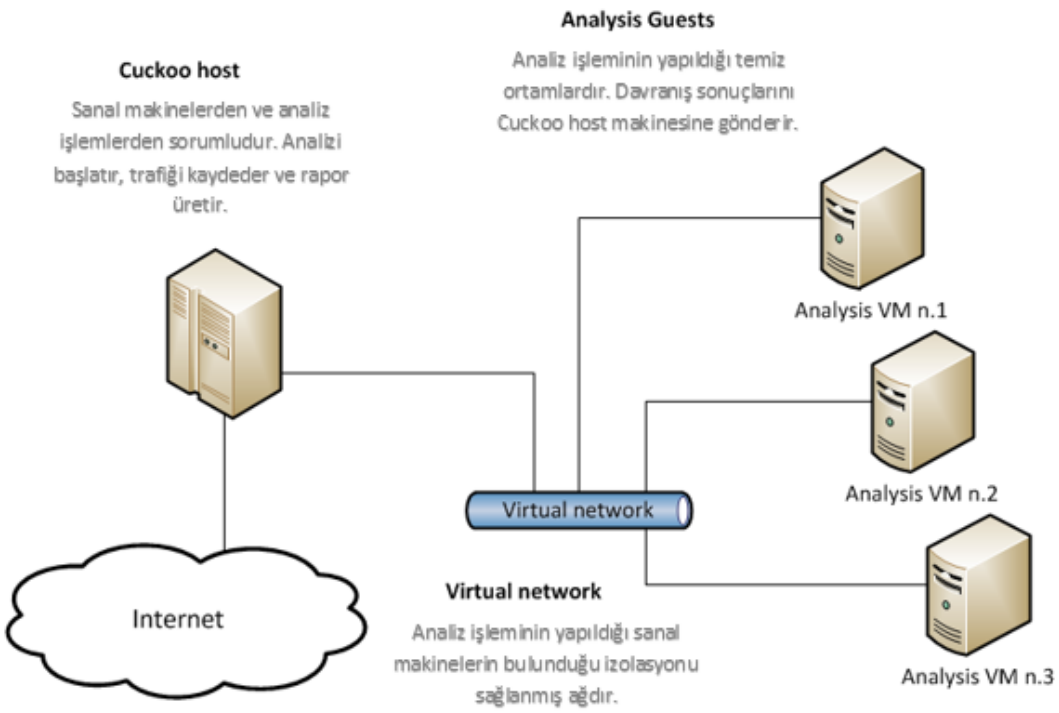
kullanılmaktadır. Sistem analiz işlemi sonrasında uygulamanın yaptığı aktiviteleri gösteren bir rapor üretmektedir.

3.3 Cuckoo Sandbox

Açık kaynak kodlu, otomatikleştirilmiş bir zararlı yazılım analiz sistemi olan Cuckoo [21], izole edilmiş Windows ortamlarında şüpheli dosyaları çalıştırıp ne yaptığını anlamak amacıyla geliştirilmiştir. Sistem kullanıcı uzayında API hooking tekniğini kullanarak istediği verileri toplamaktadır. Cuckoo ile aşağıda belirtilen çıktılar elde edilebilmektedir:

- Prosesler tarafından seçilmiş Win32 API fonksiyonlarına yapılan çağrıların zaman sıralı listesi,
- Dosyanın çalışması esnasında silinen, oluşturulan ya da internetten indirilen dosyaların listesi,
- Zararlı yazılıma ait prosesin bellekteki tüm çıktısı,
- Zararlı yazılımın çalışması esnasında sistemde gerçekleşen ağ trafiğinin PCAP formatında kaydı,
- Anlık ekran görüntüleri

Sistem Windows çalıştırılabilir dosyaların yanı sıra, Office ve Pdf dökümanlarını, dll dosyalarını da analiz edebilmektedir. Analiz işlemleri enfekte olmamış Windows sanal makinaları üzerinde gerçekleştirmektedir. Analiz işlemi host makine üzerinde bulunan uygulama tarafından yönetilmektedir. Şekil 3.2, Cuckoo'nun genel mimarisini göstermektedir.



Şekil 3.2 : Cuckoo mimarisi.

4. GERÇEKLEME

Virmon, Windows çalıştırılabilir dosyalarının davranış tabanlı analizlerini yapabilen, dağıtık ve ölçeklenebilir, otomatikleştirilmiş bir dinamik zararlı yazılım analiz sistemidir. Bu amaçla, hedef uygulamalar sanal sistemler üzerinde çalıştırılarak yapmış oldukları aktiviteler izlenmektedir.

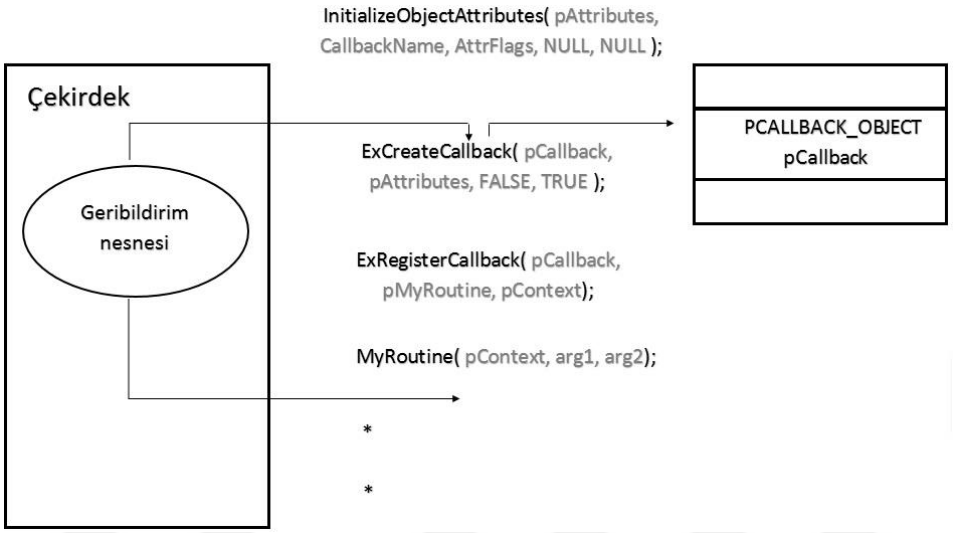
Virmon çok farklı bileşenlerden oluşan, geniş bir alana yayılmış bir analiz sistemidir. Mevcut analiz sistemlerinin aksine, Windows 8 64-bit işletim sistemi de dahil olmak üzere tüm Windows işletim sistemlerini hedef alan çalıştırılabilir dosyaları analiz edebilme kabiliyetine sahiptir. Analiz için ihtiyaç duyduğu bilgileri, işletim sisteminin sağlamış olduğu yöntemlerle toplaması yeni çıkacak Windows sürümlerine kolay bir şekilde adapte edilebilmesine olanak sağlamaktadır.

Bu bölümde Virmon'un tasarım ve gerçekleme detayları ayrıntılı bir şekilde ele alınacaktır. İlk olarak gerçeklemede kullanılan en önemli teknik olan Windows çekirdek geri bildirim *(callback)* mekanizması hakkında bilgi verilecektir. İkinci bölümde sistem mimarisi detaylarıyla açıklanacak, işletim sistemi bazında hangi verilerin nasıl toplandığı, genel sistem çerçevesinde ağ aktivitelerinin hangi teknikler kullanılarak elde edildiği ve sistemin analiz işlemini nasıl otomatize ettiğinden bahsedilecektir.

4.1 Windows Çekirdek Geri Bildirim Mekanizması

Windows çekirdeğinde, sürücülerin *(driver)* ihtiyaç duyduğu bilgilerin temini ve sürücüler arasında senkronizasyonun sağlanması için geliştirilmiş bir mekanizmadır [22]. Belirli şartlar gerçekleştiğinde her bir sürücü bu bilgilerden faydalanabilmektedir. Sürücüler, bekledikleri bir olay gerçekleştiğinde bu olaydan haberdar olabilmek için kendilerini geri bildirim mekanizmasına kayıt ettirirler. Sistem tarafından oluşturulan geri bildirim nesnelerinin bulunmasıyla birlikte, her bir sürücü geri bildirim nesneleri oluşturarak bu nesneleri başka sürücülerin kullanımına da açabilir. Her geri bildirim nesnesinin bir ismi ve belirli özellikleri *(attribute)* bulunmaktadır. Bu yüzden herbir geri bildirim nesnesi farklı bir isimde

oluşturulmaktadır. Şekil 4.1, sürücü tanımlı bir geri bildirim nesnesinin kullanımını göstermektedir.



Şekil 4.1 : Sürücü tanımlı geri bildirim nesnesinin kullanımı.

Sistem ya da bir başka sürücü tarafından oluşturulmuş bir geri bildirim nesnesinden bildirim alabilmek için, sürücüler ilgili geri bildirim nesnesini kullanırlar ve belirlemiş oldukları bir geri bildirim fonksiyonunu kayıt ettirirler. Geri bildirim nesnesi tarafından belirlenen şartlar gerçekleştiğinde, geri bildirim nesnesini oluşturan sürücü sırayla kendisini kaydettiren tüm geri bildirim fonksiyonlarını çağırır. Bu mekanizma ile sürücüler ihtiyaç duydukları bilgilere erişebilmektedirler.

Dinamik analiz teknikleri bölümünde, Windows çekirdeğinin bütünlüğünün sağlanabilmesi, çekirdeğe kod enjeksiyonunun engellenebilmesi amacıyla geliştirilmiş patchguard [16] mekanizmasından bahsedilmişti. Bu mekanizma, zararlı yazılım analizi için gerekli bilgilerin sistem çağrılarının hook edilerek toplanmasını imkansız hale getirmektedir. Bu yüzden, geliştirmiş olduğumuz zararlı yazılım analiz sistemi, analiz edilen uygulamaların işletim sisteminde gerçekleştirmiş olduğu aktiviteleri Windows çekirdek geri bildirim mekanizması aracılığıyla toplamaktadır. Bu mekanizmanın kullanım detayları bir sonraki bölümde ayrıntılı bir şekilde ele alınacaktır.

4.2 Sistem Mimarisi

Virmon dinamik zararlı yazılım analiz sistemi bileşenleri genel olarak 4 ana grupta kategorilendirilebilir:

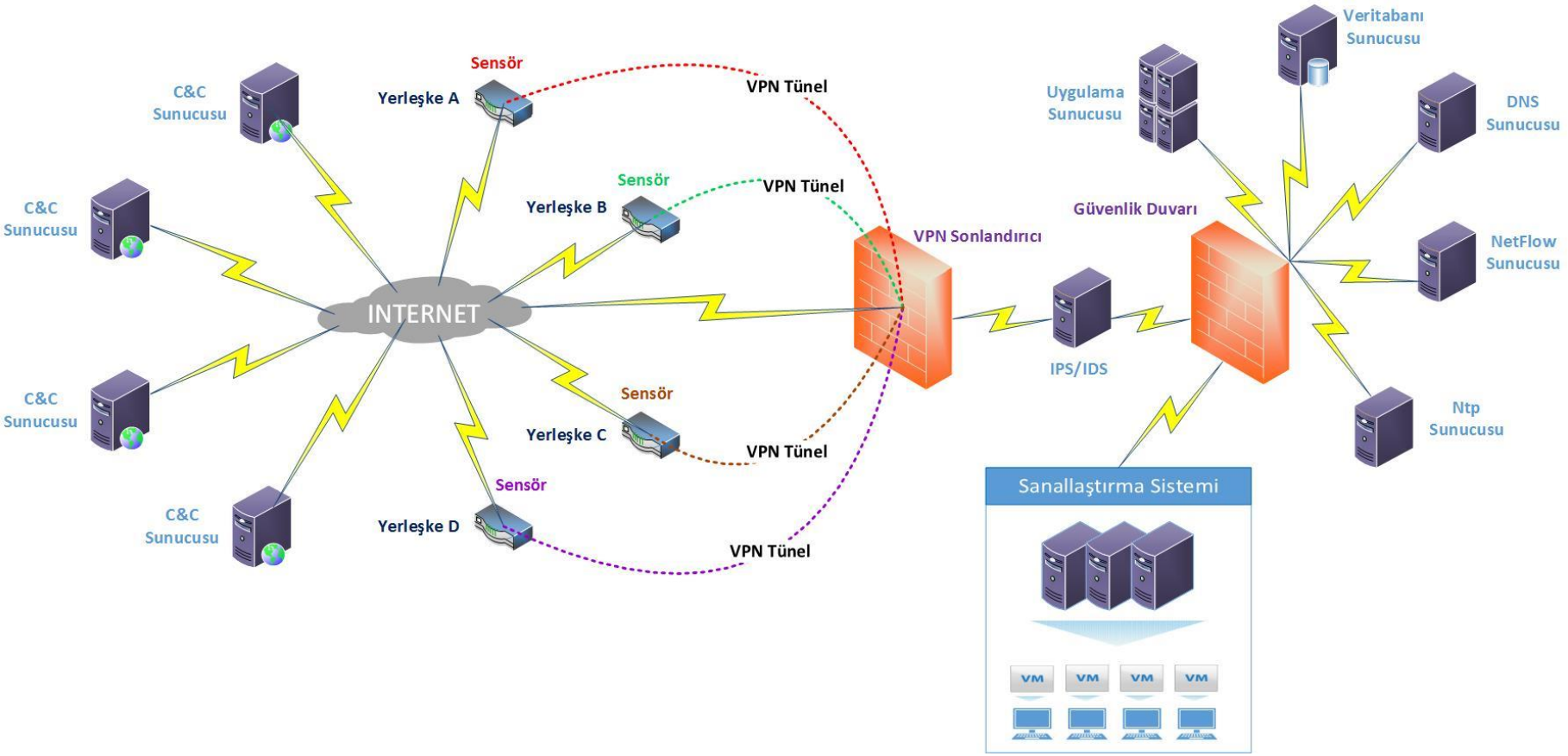
- *Sanallaştırma altyapısı:* Analiz işlemini gerçekleştirecek olan Windows işletim sistemi yüklü sanal makineleri içerir.
- *Analiz makinesi bileşenleri:* Analiz edilecek uygulamanın işletim sistemi üzerinde yapmış olduğu proses, registry ve dosya sistemi aktivitelerini izlemekle yükümlü bileşenlerdir.
- *Ağ bileşenleri:* Analiz edilecek uygulamanın gerçekleştirmiş olduğu ağ aktivitelerini takip etmekle yükümlüdür. Bu bileşenler farklı lokasyonlara yerleştirilmiş ve analiz makinesi trafiğini uzaktaki C&C sunucularına ileten dağıtık sensör ağı, DNS isteklerine cevap veren bir DNS sunucusu, saldırı alarmları üreten, ağ trafiğinden HTTP isteklerini ve dosyaları çıkarabilen, gerektiğinde ağ trafiğini kesebilen bir IPS/IDS, ve tüm sistemlerin zaman senkronizasyonunu sağlamak için kullanılacak olan bir NTP sunucusudur.
- *Otomatik analiz bileşenleri:* Analiz sisteminin otomatik olarak çalışmasını ve yönetimini sağlayacak olan bileşenlerdir. Sanallaştırma sistemi üzerindeki makinelere analiz edilecek dosyaları gönderen ve tüm analiz sürecini yöneten bir uygulama sunucusu, analiz makineleri üzerinde çalışan ve uygulama sunucusu ile iletişime geçerek analiz işlemini kontrol eden analiz uygulamaları, sistemin yönetimi için kullanılacak olan bir web uygulaması ve verilerin depolanması için kullanılacak olan veritabanı sunucusundan oluşmaktadır.

Şekil 4.2, bileşenleri tanıtılan Virmon dinamik zararlı yazılım analiz sisteminin genel topolojisini göstermektedir.

4.2.1 Sanallaştırma altyapısı

Zararlı yazılım analiz makinelerini barındıracak olan sanallaştırma ortamı için açık kaynak sanallaştırma çözümü olan Oracle VirtualBox seçilmiştir. VirtualBox;

- açık kaynak olması,
- 32 ve 64 bit platformda çalışabilmesi,
- 32 ve 64 bit işletim sistemlerini çalıştırabilmesi,



Şekil 4.2 : Virmon sistem topolojisi.

- endüstri standardında oluşturulmuş OVF formatı sayesinde, oluşturulan sanal makinelerin sanallaştırma uygulamasına alınması ve sanallaştırma uygulamasından dışarı aktarılmasına imkân (import, export) vermesi,
- Intel VT-x veya AMD-V gibi donanım sanallaştırması olmadan da sanal makineleri çalıştırabilmesi,
- uzaktan kontrol edilebilmesi,
- web ara yüzü ile yönetilebilmesi,
- komut satırı desteği olması,
- sanal makinelerin dışarıdan komut alabilmesi,
- VLAN desteği sağlaması,

gibi özelliklere sahiptir. Özellikle analiz makinelerinde belirli işleri otomatik olarak yapılabilmesine olanak sağlayan VirtualBox, sanal makine dışından komut çalıştırma özelliği sayesinde analiz makinelerinin otomatik olarak oluşturulmasına ve başta ağ ayarları olmak üzere yapılandırılabilmesine olanak sağlamaktadır [23].

Sanallaştırma altyapısı, çok sayıda fiziksel sunucuyu içerebilecek şekilde tasarlanmıştır. Analiz makinelerinin performanslı çalışabilmesi için her bir fiziksel sunucunun sahip olduğu çekirdek sayısı kadar analiz makinesi içermesi sağlanmıştır. Sistemdeki fiziksel sunucu sayısı *s*, her bir sunucudaki işlemci sayısı *p*, her bir işlemcinin sahip olduğu çekirdek sayısı *n* olmak üzere sistemin toplam analiz makinesi kapasitesi *c*'dir **(4.1)**.

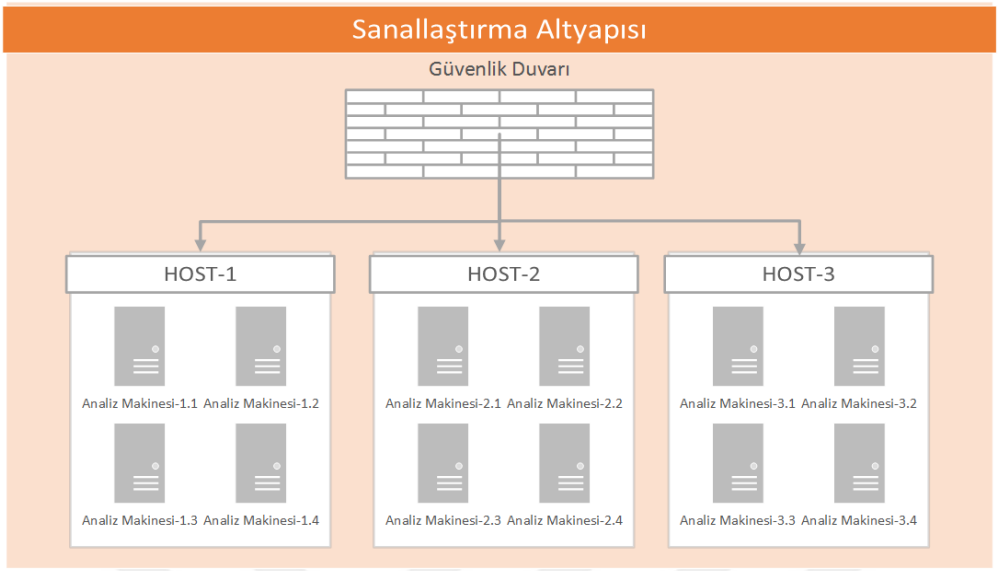
$$c = s.p.n \tag{4.1}$$

Örnek olarak bir blade sunucu sisteminin altyapıda kullanıldığını varsayalım. Bu sistemin özellikleri;

- 16 adet fiziksel sunucu içermesi,
- her sunucunun 2 adet 6'şar çekirdekli Xeon işlemciye sahip olması

dır. Böyle bir sistemin altyapıda kullanılması durumunda, toplam analiz makinesi kapasitesi 192 olmaktadır. Bu durum da 192 adet işletim sisteminin paralel olarak zararlı yazılım analizi yapabileceği anlamına gelmektedir. Sisteme fiziksel

sunucuların eklenmesi ile bu kapasite daha da artırılabilir. Şekil 4.3 örnek bir sanallaştırma topolojisini göstermektedir.



Şekil 4.3 : Sanallaştırma altyapısı.

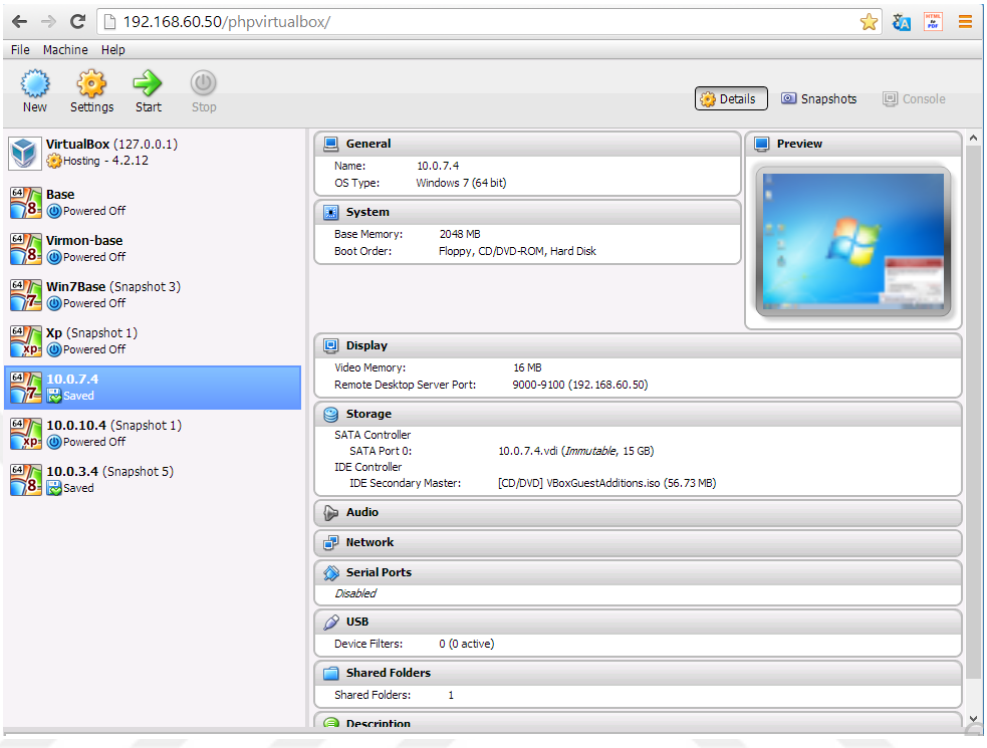
Sanallaştırma altyapısının yönetimi *phpvirtualbox* adı verilen açık kaynak kodlu bir web uygulaması aracılığı ile yapılmaktadır [24]. Bu uygulama çok sayıda fiziksel sunucu içeren Virtualbox sistemlerini kolaylıkla yönetebilmektedir. RDP protokolü ile sanal makinelere direkt bağlanılabilmesine olanak sağlamaktadır. Şekil 4.4 phpvirtualbox uygulamasına ait arayüzü göstermektedir.

4.2.2 Analiz makinesi bileşenleri

Zararlı yazılım analizi için işletim sistemi bazında genel olarak 3 farklı tipte veri toplanmaktadır. Bu veriler analiz edilen uygulamanın gerçekleştirmiş olduğu *proses*, *registry* ve *dosya sistemi aktiviteleri*dir. Toplanan veriler yalnızca analiz edilen uygulamaya ait olmayıp, ilgili uygulamanın çalıştırılması ile elde edilen prosesin oluşturduğu tüm alt proseslerin ve etkileşime geçtiği proseslerin de aktivitelerini içermektedir. Bu veriler çekirdek uzayında çalışan 3 farklı bileşen tarafından izlenmektedir : *proses*, *registry* ve *dosya sistemi izleyicileri*. Bu bileşenler tek bir dosya sistemi filtreleme sürücüsü *(file system filter driver)* altında geliştirilmiştir.

Dosya sistemi filtreleme sürücüleri, bir dosya sisteminin davranışını değiştirmek için kullanılmaktadır [25]. Bu tip sürücüler Windows yönetici (Windows executive) servisinin bir parçası olarak çekirdek uzayında çalışmaktadırlar. Bu sürücüler, dosya sistemlerinin yapmış oldukları giriş/çıkış (I/O) işlemlerini filtreleme özelliğine

sahiptir. Burada filtreleme özelliğinden kastedilen giriş/çıkış işleminin kayıt altına alınması, izlenmesi, değiştirilmesi ya da tamamen engellenmesidir. Dosya sistemi filtreleme sürücülerine örnek olarak anti-virüs araçları, şifreleme uygulamaları ve hiyerarşik depolama yönetim sistemleri verilebilir.

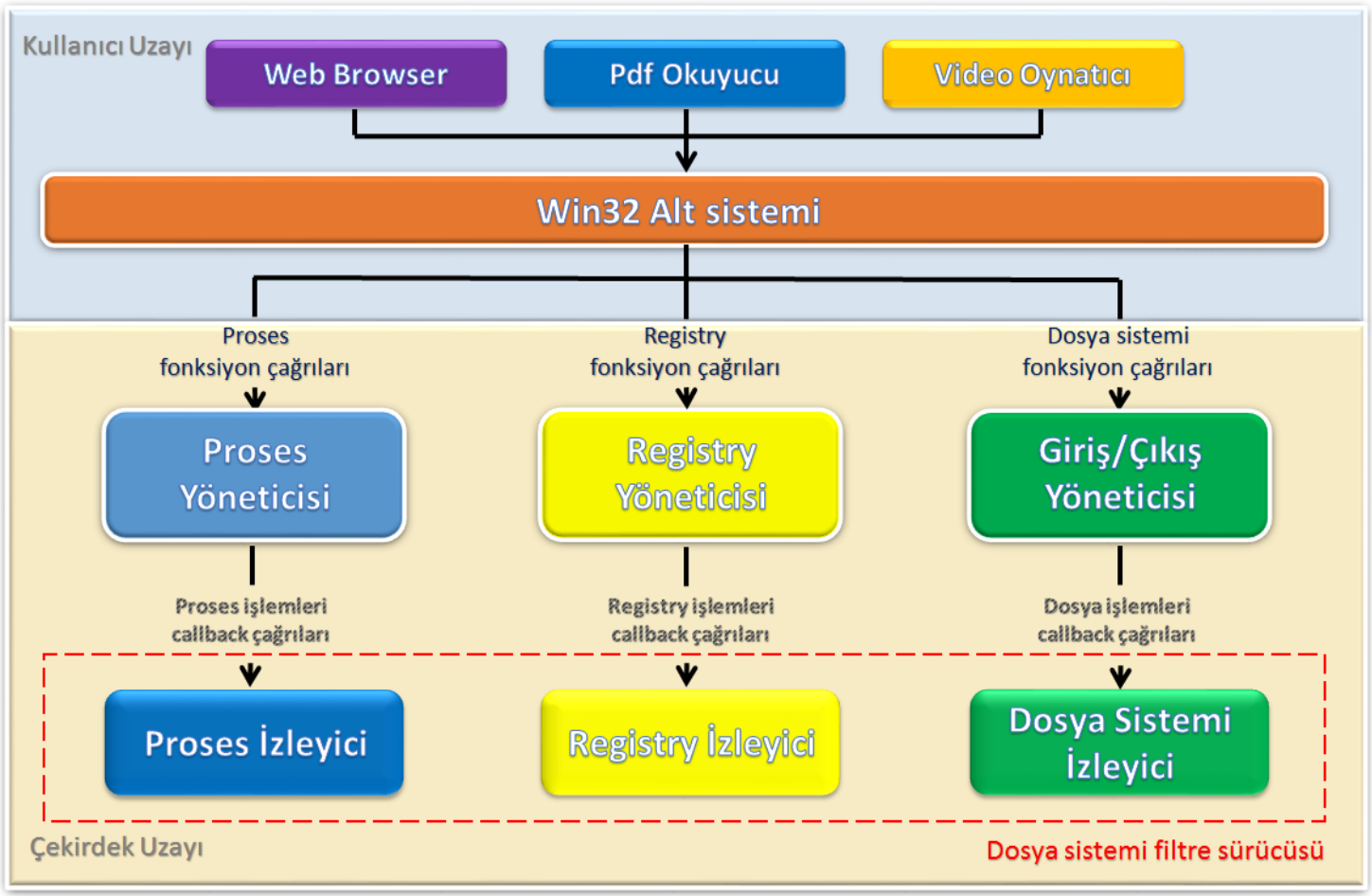


Şekil 4.4 : Phpvirtualbox yönetim arayüzü.

Analiz makinesi bileşenleri ihtiyaç duydukları proses, registry ve giriş/çıkış olaylarından, daha önce tanıtılan Windows çekirdek geri bildirim mekanizması aracılığıyla haberdar olmaktadırlar. Her bir bileşen ilgili olduğu geri bildirim nesnesine belirlemiş olduğu fonksiyonu, sürücünün sisteme yüklenme zamanında kayıt ettirir. İlgili olay gerçekleştiğinde geri bildirim nesnesi, belirlenen fonksiyonu çağırır. Şekil 4.5 açıklanan analiz makinesi bileşenlerini göstermektedir.

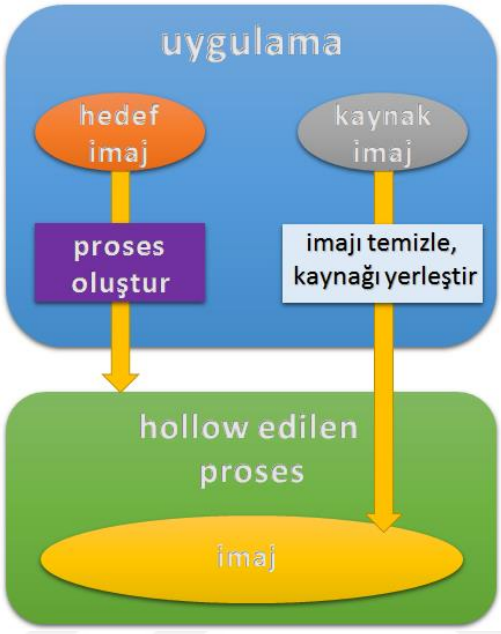
4.2.2.1 Proses izleyici

Zararlı yazılım analizinde ihitiyaç duyulan en önemli bilgilerden biri, hedef uygulamanın yapmış olduğu proses ve thread aktiviteleridir. Zararlı yazılımlar kendilerini proses bazında gizlemek için farklı teknikler kullanmaktadırlar. Örnek olarak *process hollowing* tekniği verilebilir [26]. Şekil 4.6'da process hollowing tekniğini anlatan bir diyagram gösterilmiştir. Bu teknikte, zararlı yazılım çalışmaya



Şekil 4.5 : Analiz makinesi bileşenleri.

başladığında kendi kopyası olan bir çocuk prosesi suspend modda oluşturur. Anne proses oluşturmuş olduğu çocuk prosesin bellekteki zararsız kodunu, zararlı kodla değiştirerek çalışmasını devam ettirir. Çocuk proses çalıştıktan sonra, anne proses kendisini sonlandırır. Görüldüğü gibi bazı uygulamalar zararsızmış gibi gözüktükleri halde farklı prosesler altında faaliyetlerini devam ettirebilmektedirler. Bu tip durumlar da uygulamaların proses aktivitelerinin takibini zorunlu kılmaktadır.



Şekil 4.6 : Proses hollowing tekniği.

Zararlı yazılımların kendilerini gizleme amacıyla kullandıkları bir diğer teknik *kod enjeksiyonu* yöntemidir. Bu yöntemde zararlı yazılım kendisine hedef seçtiği, sistem üzerinde hali hazırda çalışan faydalı bir prosesin bağlamında zararlı kod çalıştırmaktadır [27]. Bu yöntemin kullanılma amaçlarından biri de anti-virüs ve güvenlik duvarlarını atlatmaktır. Şaşırtıcıdır ki bu işleme işletim sistemi tarafından izin verilmektedir. Tekniğin temeli uzak *(remote)* thread oluşturmaya dayanmaktadır. Eğer analiz edilen uygulama, çalışma esnasında başka proseslerin bağlamında uzak thread açıyorsa bu şüpheli bir durumdur ve dikkat edilmesi gerekmektedir. Bu anlatılan tekniklerin dışında yine proses ve threadler ile ilgili başka teknikler de bulunmaktadır. Bu durum, başarılı bir zararlı yazılım analizi yapılabilmesı için proses ve threadlerin izlenmesini zorunlu kılmaktadır.

Virmon dinamik zararlı yazılım analiz sistemi proses izleyici bileşeni aracılığıyla işletim sistemi üzerinde oluşturulan ya da sonlandırılan proses ve threadlerin takibini yapmaktadır. Bu bileşen, analiz edilen uygulamanın çalışmasıyla oluşan prosesi, bu

prosesin oluşturduğu prosesleri ve uzak thread açılan tüm proseslerin aktivitelerini takip etmektedir. Proses oluşturulması/sonlandırılması, uzak thread açılması takip edilen olaylar arasında bulunmaktadır. Takip edilen proseslerden herhangi biri bu eylemlerden birini gerçekleştirdiğinde Proses izleyici bileşeni, Windows çekirdek geri bildirim mekanizması aracılığıyla haberdar edilmektedir.

Proses İzleyici bileşeni iki ana bölümden oluşmaktadır: ilgili geri bildirim nesnelerine kayıt işlemini gerçekleştiren bölüm ve olay sırasında geri bildirim nesneleri tarafından çağrılan bölüm. İlgili geri bildirim nesnelerine kayıt işleminin yapılabilmesi için işletim sistemi tarafından sunulan kayıt fonksiyonlarının sürücünün sisteme yüklenme zamanında çağrılması gerekmektedir.

Şekil 4.7'de proses kayıt fonksiyonunun prototipi gösterilmiştir. Bu fonksiyon iki parametre almaktadır. İlki geri bildirim nesnesinin yeni bir proses oluşturulduğunda ya da sonlandığında çağıracağı CreateProcessNotifyEx prototipindeki fonksiyonu gösteren bir işaretçi, ikincisi ise belirtilen fonksiyonun geri bildirim nesnesinin kayıt listesine eklenileceğini ya da çıkarılacağını belirten ikili değişkendir. Fonksiyon işlemin başarılı olması durumunda STATUS_SUCCESS, kayıt işleminin hali hazırda yapılmış olması ya da geri bildirim nesnesinin kayıt listesinin kapasitesini doldurmuş olması durumunda STATUS_INVALID_PARAMETER, geri bildirim fonksiyonunu gösteren işaretçinin erişim yetkisinin bulunmaması durumunda STATUS_ACCESS_DENIED değerini geri döndürmektedir [28]. Yeni bir proses oluşması durumunda geri bildirim fonksiyonu oluşturma işlemini gerçekleştiren threadin bağlamında, proses sonlanması durumunda ise sonlanan prosesin son threadinin bağlamında çalışmaktadır. Bu yüzden herhangi bir prosesin çalışması engellenebilirken, proses sonlanması engellenememektedir. Bununla birlikte Proses izleyici yeni prosesi oluşturan prosesin kimliğini tanımlayabilirken, sonlanma işlemini başlatan prosesin kimliğini tanımlayamamaktadır.

```
NTSTATUS PsSetCreateProcessNotifyRoutineEx(
        _In_  PCREATE_PROCESS_NOTIFY_ROUTINE_EX NotifyRoutine,
        _In_  BOOLEAN Remove
      );
```

Şekil 4.7 : Proses kayıt fonksiyonu prototipi.

Şekil 4.8'de ise thread kayıt fonksiyonunun prototipi gösterilmiştir. Bu fonksiyon, geri bildirim nesnesinin yeni bir thread oluşturulduğunda ya da sonlandığında

çağıracağı CREATE_THREAD_NOTIFY_ROUTINE prototipindeki fonksiyonu gösteren bir işaretçiyi parametre olarak almaktadır. Fonksiyon işlemin başarılı olması durumunda STATUS_SUCCESS, işlemin başarısız olması durumunda STATUS_INSUFFICIENT_RESOURCES değerini döndürmektedir [29]. Proses olaylarına benzer şekilde yeni bir thread oluşması durumunda geri bildirim fonksiyonu oluşturma işlemini gerçekleştiren threadin bağlamında, thread sonlanması durumunda ise sonlanan threadin bağlamında çalışmaktadır. Bu yüzden herhangi bir threadin oluşturulması engellenebilirken, thread sonlanması engellenememektedir. Proses izleyici yalnızca uzak thread oluşması olaylarını takip etmektedir.

```
NTSTATUS PsSetCreateThreadNotifyRoutine(
        _In_  PCREATE_THREAD_NOTIFY_ROUTINE NotifyRoutine
    );
```

Şekil 4.8 : Thread kayıt fonksiyonu prototipi.

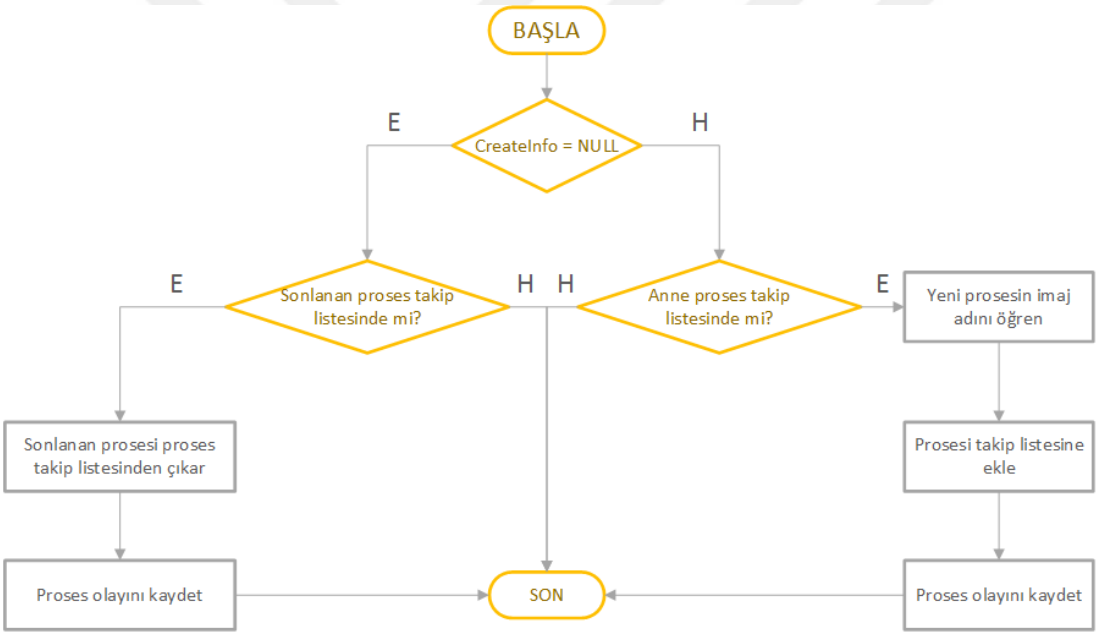
Proses izleyici bileşeninin ikinci bölümü callback nesnelerine kayıt sırasında belirtilen uygun formattaki proses ve thread geri bildirim fonksiyonlarının tanımlarından oluşmaktadır. Şekil 4.9'da proses geri bildirim fonksiyonuna ait prototip görülmektedir. Proses geri bildirim fonksiyonu 3 parametre almaktadır. İlk parametre Microsoft tarafından dökümantasyonu yapılmamış EPROCESS yapısındaki *(struct)* değişkeni gösteren işaretçidir. İkinci parametre, oluşturulacak olan prosese ayrılmış proses numarası ya da sonlanacak olan prosesin numarasıdır. Son parametrenin NULL olmaması durumunda yeni bir proses oluşturulduğu, NULL olması durumunda ise işlemin proses sonlanma olayı olduğu anlaşılmaktadır. Bu parametre yeni prosesi tanımlayan PS_CREATE_NOTIFY_INFO yapısındaki bir değişkeni gösteren işaretçidir. Proses olayına ait bilgiler bu değişken üzerinden sağlanmaktadır.

```
VOID CreateProcessNotifyEx(
  _Inout_   PEPROCESS Process,
  _In_      HANDLE ProcessId,
  _In_opt_  PPS_CREATE_NOTIFY_INFO CreateInfo
);
```

Şekil 4.9 : Proses geri bildirim fonksiyonu prototipi.

Proses geri bildirim fonksiyonu çalışmaya başladığında ilk olarak oluşturma bilgisinin (CreateInfo) mevcut olup olmadığını kontrol eder. Oluşturma bilgisi

mevcut ise (yeni proses oluşturuluyorsa), oluşturan prosesin proses takip listesinde olup olmadığına bakılır. Proses takip edilmiyorsa işlem sonlandırılır. Proses takip listesinde ise yeni prosesin imaj adı *KeStackAttachProcess* fonksiyonu yardımıyla, prosesin adres uzayına geçilerek *ZwQueryInformationProcess* fonksiyonu yardımıyla öğrenilir. Proses imaj adı öğrenildikten sonra *KeUnstackDetachProcess* yardımıyla geri dönülür. Yeni oluşan prosesin numarası ve imaj adı proses takip listesine eklenir. Proses oluşturma olayı kayıt altına alınarak işlem sonlandırılır. Oluşturma bilgisi mevcut değilse (proses sonlanıyorsa), sonlanan prosesin proses takip listesinde olup olmadığına bakılır. Proses listede değilse işlem sonlandırılır. Proses listede ise proses takip listesinden çıkarılır, proses sonlanma olayı kayıt altına alınarak işlem sonlandırılır. Şekil 4.10 açıklanan proses geri bildirim fonksiyonuna ait akış diyagramını göstermektedir.



Şekil 4.10 : Proses geri bildirim fonksiyonu akış diyagramı.

Thread geri bildirim fonksiyonuna ait prototip Şekil 4.11'de gösterilmiştir. Bu fonksiyon da proses geri bildirim fonksiyonu gibi 3 parametre almaktadır. İlk ve ikinci parametre sırayla, oluşturulan ya da sonlanan threadin ait olduğu prosesin ve threadin numarasını, son parametre ise işlemin oluşturulma ya da sonlanma eylemi olduğunu göstermektedir.

Thread geri bildirim fonksiyonu çalışmaya başladığında ilk olarak threadin ait olduğu prosesin takip listesinde olup olmadığını kontrol etmektedir. Proses takip listesinde değilse işlem sonlandırılır. Proses takip listesinde ise işlemin oluşturma mı

sonlanma mı olduğuna bakılır. Olay sonlanma ile ilgili ise fonksiyondan çıkılır. Eğer yeni bir thread oluşturuluyorsa, işlemci üzerinde hali hazırda çalışan prosesin numarası *PsGetCurrentProcessId* fonksiyonu yardımıyla öğrenilir. Eğer oluşturulan thread hali hazırda çalışan proses ise işlem sonlandırılır. Eğer uzak thread oluşturuluyorsa , uzak prosesin imaj adı öğrenilir, proses takip listesine eklenir ve uzak thread oluşturma eylemi kayıt altına alınarak işlem sonlandırılır. Şekil 4.12 proses geri bildirim fonksiyonuna ait akış diyagramını göstermektedir.

```
VOID
(*PCREATE_THREAD_NOTIFY_ROUTINE) (
    IN HANDLE  ProcessId,
    IN HANDLE  ThreadId,
    IN BOOLEAN  Create
    );
```

Şekil 4.11 : Thread geri bildirim fonksiyonu prototipi.

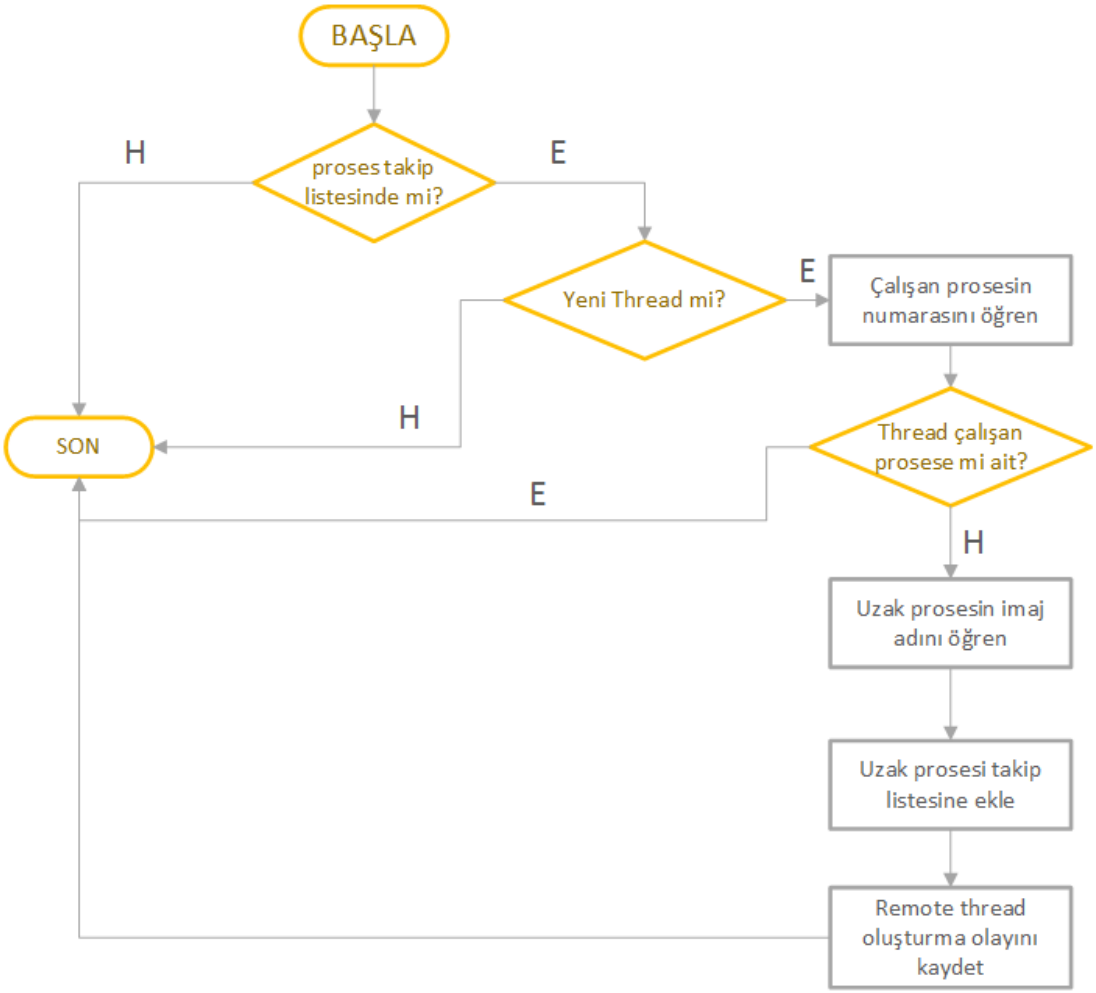
4.2.2.2 Registry izleyici

Zararlı yazılım analizinde ihitiyaç duyulan bir diğer önemli bilgi, analiz edilen uygulamanın yapmış olduğu Windows registry aktiviteleridir. Registry, Windows işletim sistemlerinde kullanıcıları, uygulamaları ve donanım aygıtlarını konfigüre etmek için gerekli olan bilgilerin tutulduğu, merkezi ve hiyerarşik bir veritabanıdır [30]. Windows, çalışması sırasında sürekli registry yapısını kullanmaktadır. Örneğin, sistem üzerinde kurulu donanımların listesi alınırken ya da o an kullanılan portlar öğrenilirken registry veritabanına başvurulmaktadır. Windows ile birlikte gelen *regedit* uygulaması ile registry sorgulanabilir ve değiştirilebilir.

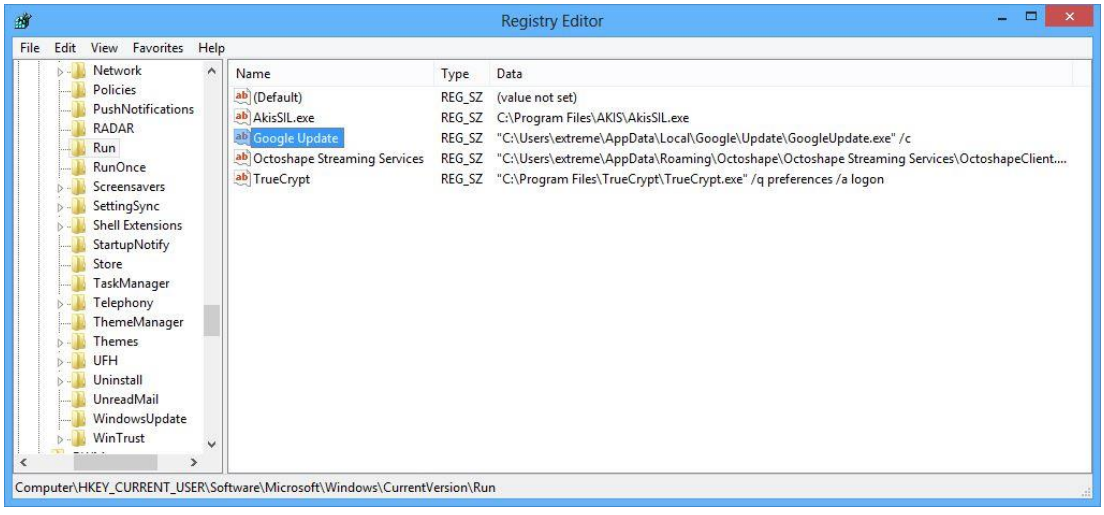
Zararlı yazılımlar, Windows registry yapısını farklı amaçlarla kullanabilmektedirler. Bu amaçlardan en bilineni sistem üzerinde kalıcılığın sağlanmasıdır. Zararlı uygulamalar, bulaşmış oldukları sistemler yeniden başlatıldığında otomatik olarak aktive olabilmek için registry üzerindeki belirli lokasyonlarda değişiklik yapmaktadırlar. Şekil 4.13'te gösterilen örnek lokasyonda sistem yeniden başlama sırasında otomatik olarak çalışmak isteyen uygulamalar görülmektedir.

Otomatik başlama dışında zararlı yazılımlar, kendilerine ait konfigürasyon bilgilerini de bu yapı üzerinde tutabilmektedirler. Registry yapısının bir diğer kullanım amacı da güvenlik uygulamalarının ya da görev yöneticisinin kullanıcılar tarafından çağrılmasını engellemektir. Tüm bu değişimler zararlı yazılımların tespit edilmeyi

zorlaştırma amacıyla kullandıkları tekniklerdir. Sağlıklı bir zararlı yazılım analizi yapılabilmesi için registry takibi bu sebeplerden ötürü önem arz etmektedir.



Şekil 4.12 : Thread geri bildirim fonksiyonu akış diyagramı.



Şekil 4.13 : Örnek otomatik çalışma lokasyonu.

Virmon dinamik zararlı yazılım analiz sistemi registry izleyici bileşeni aracılığıyla, Windows registry yapısı üzerinde yapılan değişimleri izlemektedir. Bu bileşen, analiz edilen uygulamanın çalışmasıyla oluşan prosesi, bu prosesin oluşturduğu prosesleri ve uzak thread açılan tüm proseslerin registry aktivitelerini takip etmektedir. Registry yapısında yeni düğümün oluşturulması ve silinmesi, düğümlere ait değerlerin değiştirilmesi ve silinmesi olayları takib edilen eylemler arasında bulunmaktadır. Takip edilen proseslerden herhangi biri bu eylemlerden birini gerçekleştirdiğinde registry izleyici bileşeni, Windows çekirdek geri bildirim mekanizması aracılığıyla haberdar edilmektedir.

Registry izleyici bileşeni, proses izleyici bileşeni gibi iki ana bölümden oluşmaktadır: ilgili geri bildirim nesnesine kayıt işlemini gerçekleştiren ve olay sırasında geri bildirim nesnesi tarafından çağrılan bölüm. Registry geri bildirim nesnelerine kayıt işleminin yapılabilmesi için işletim sistemi tarafından sunulan kayıt fonksiyonunun sürücünün sisteme yüklenme zamanında çağrılması gerekmektedir.

Registry geri bildirim nesnesine registry fonksiyonunun kayıt edilmesi için kullanılan fonksiyonun prototipi Şekil 4.14'te gösterilmiştir. Fonksiyon 6 parametre almaktadır. İlki registry olayı gerçekleştiğinde registry nesnesi tarafından çağrılacak olan EX_CALLBACK_FUNCTION yapısındaki fonksiyonu gösteren bir işaretçi, ikincisi kayıt işlemini gerçekleştiren sürücünün yüklenme sırasını belirten UNICODE_STRING tipindeki değişkeni gösteren işaretçi, üçüncüsü kayıt işlemini gerçekleştiren sürücüyü gösteren DRIVER_OBJECT yapısındaki bir işaretçi, dördüncüsü sürücü tarafından tanımlanan ve geri bildirim bağlamını gösteren bir işaretçi, beşincisi geri bildirim fonksiyonunu tanımlamak için kullanılan LARGE_INTEGER tipindeki değişkeni gösteren bir işaretçidir. Registry kayıt işlemi kaldırılırken bu tanımlayıcı kullanılmaktadır. Son parametre ise Microsoft tarafından sonraki işletim sistemlerinde kullanılmak üzere rezerve edilmiştir. Kayıt işlemi başarılı bir şekilde gerçekleştiği zaman fonksiyon STATUS_SUCCESS, bellek ataması sırasında hata oluştuğunda STATUS_INSUFFICIENT_RESOURCES, çağırma işlemini gerçekleştiren sürücü ya da aynı yüksekliğe sahip (altitude) bir başka sürücü halihazırda kayıt işlemini gerçekleştirmişse STATUS_FLT_INSTANCE_ALTITUDE_COLLISION değerini geri döndürmektedir [31].

```
NTSTATUS CmRegisterCallbackEx(
  _In_        PEX_CALLBACK_FUNCTION Function,
  _In_        PCUNICODE_STRING Altitude,
  _In_        PVOID Driver,
  _In_opt_    PVOID Context,
  _Out_       PLARGE_INTEGER Cookie,
  _Reserved_  PVOID Reserved
);
```

Şekil 4.14 : Registry kayıt fonksiyonu prototipi.

Registry izleyici bileşeninin ikinci bölümü geri bildirim nesnelerine kayıt sırasında belirtilen uygun formattaki registry geri bildirim fonksiyonunun tanımını içermektedir. Şekil 4.15'te prototipi gösterilen geri bildirim fonksiyonu registry işlemlerini izlemek, değiştirmek ya da engellemek amacıyla kullanılmaktadır. Geri bildirim fonsiyonunun ilk parametresi sürücünün geri bildirim nesnesine kayıt sırasında parametre olarak verilen bağlam işaretçisidir. İkinci parametre REG_NOTIFY_CLASS tipinde gerçekleşen registry olayının çeşidini belirtmektedir. Son parametre ise registry olayının tipine bağlı olarak değişen ve registry olayına ait bilgileri içeren bir yapıyı gösteren işaretçidir. Registry izleyici olayları bu iki parametreyi birlikte değerlendirerek kayıt altına almaktadır.

```
EX_CALLBACK_FUNCTION RegistryCallback;

NTSTATUS RegistryCallback(
  _In_      PVOID CallbackContext,
  _In_opt_  PVOID Argument1,
  _In_opt_  PVOID Argument2
);
```

Şekil 4.15 : Registry geri bildirim fonksiyonu prototipi.

Registry geri bildirim fonksiyonu çalışmaya başladığında ilk olarak registry işlemini gerçekleştiren prosesin proses takip listesinde olup olmadığını kontrol eder. Proses takip edilmiyorsa işlem sonlandırılır. Proses takip listesinde ise registry bağlamı oluşturulur ve ilgili registry işleminin tipi belirlenir. Geri bildirim bir registry anahtarı silme işleminin öncesinde geliyorsa (RegNtPreDeleteKey) silinen anahtarın yolu belirlenir ve olay başarısız olarak kayıt altına alınır. Eğer geri bildirim bir registry anahtarı silme işleminin sonrasında geliyorsa (RegNtPostDeleteKey), ilgili registry olayı başarılı olarak güncellenir. Herbir registry işlemi tipi için kullanılan pre ve post geri bildirimleri sayesinde işlemin gerçekleşip gerçekleşmediği anlaşılmaktadır. Örnek olarak silme isteği gönderilen bir registry anahtarı mevcut

değilse, pre bildiriminde istek hatalı olarak kayıt altına alınmaktadır. Eğer anahtar mevcut ise post geri bildiriminde kayıt başarılı olarak güncellenmektedir.

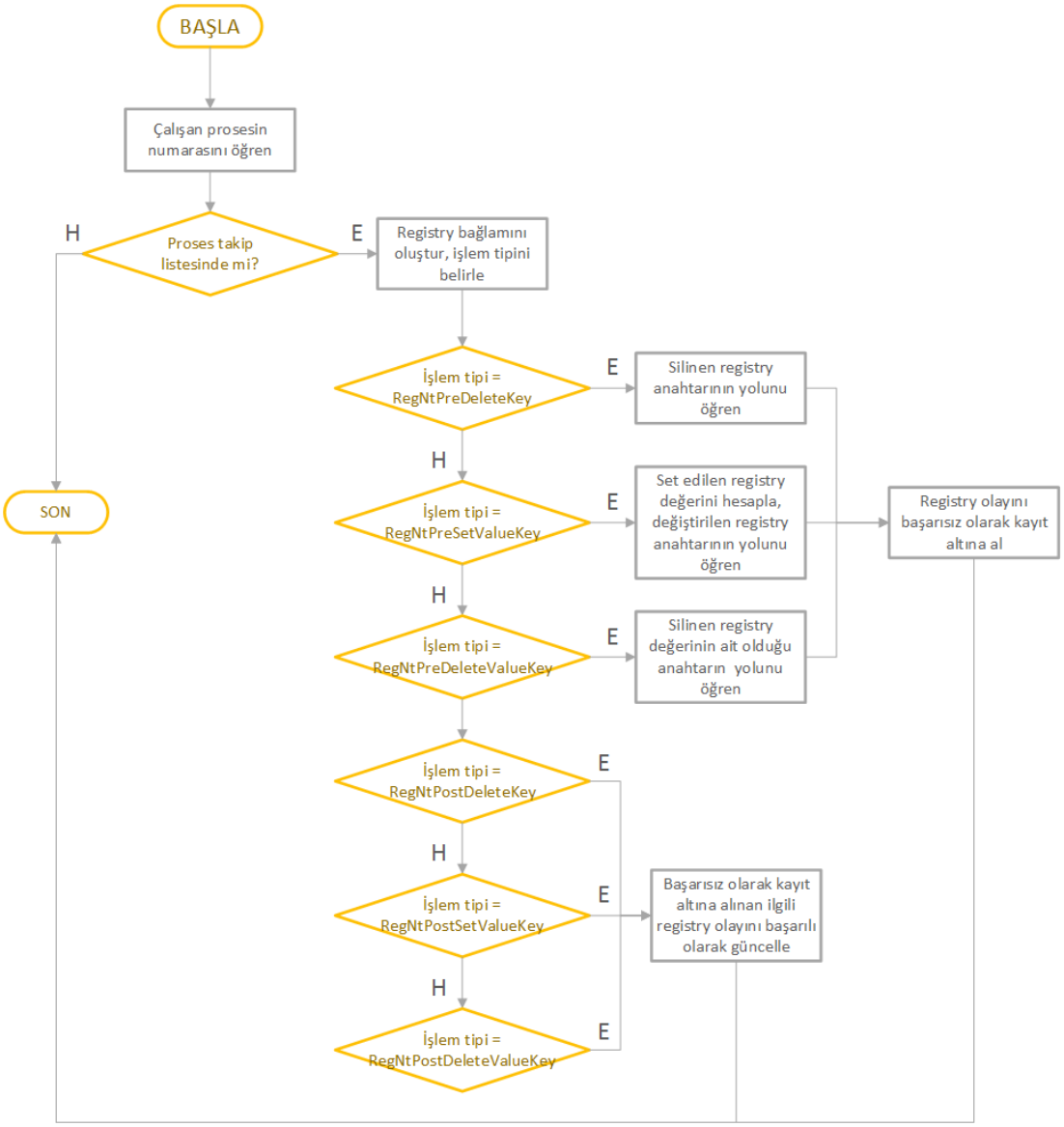
Geri bildirim tipi bir registry anahtarına değer atama işlemi öncesi (RegNtPreSetValueKey) ise atanan değer hesaplanır, ilgili registry anahtarının yolu öğrenilir ve olay başarısız olarak kayıt altına alınır. Eğer geri bildirim bir registry anahtarı değerini atama işleminin sonrasında geliyorsa (RegNtPostSetValueKey), ilgili registry olayı başarılı olarak güncellenir.

Geri bildirim tipinin bir registry anahtarına ait değerin silinmesi işlemi öncesi (RegNtPreDeleteValueKey) ise registry anahtarının yolu öğrenilir ve olay başarısız olarak kayıt altına alınır. Eğer geri bildirim bir registry anahtarına ait değerin silinmesi işleminin sonrasında geliyorsa (RegNtPostDeleteValueKey), ilgili registry olayı başarılı olarak güncellenir. Bu işlemlerin akabinde geri bildiri fonksiyonu sonlandırılır. Şekil 4.16 registry geri bildirim fonksiyonuna ait akış diyagramını göstermektedir.

4.2.2.3 Dosya sistemi izleyici

Davranış tabanlı zararlı yazılım analizinde proses ve registry aktivitesi takibi dışında izlenmesi gereken bir diğer aktivite tipi dosya sistemi olaylarıdır. Zararlı aktivite gösteren uygulamaların çoğunluğu çalıştırıldığı zaman gizlenmek adına kendilerini işletim sistemi üzerinde başka bir noktaya taşıyıp çalıştırılan dosyayı silmeyi tercih etmektedir. Bunun dışında bazı zararlı yazılımlar periyodik olarak kendilerini güncelleyip komuta kontrol sunucuları aracılığıyla yeni zararlı yazılımlar indirip sistemlere yükleyebilmektedirler.

Virmon zararlı yazılım analiz sistemi dosya sistemi izleyici bileşeni ile işletim sistemi üzerinde gerçekleşen dosya yazma ve silme olaylarını takip etmektedir. Dosya okuma işlemleri takip edilebilmekle birlikte çok fazla okuma işlemi gerçekleştiği için bu bileşen yalnızca sistem üzerinde değişimlere sebebiyet veren aktiviteleri takip etmektedir. Dosya sistemi izleyici bileşeni, analiz edilen uygulamanın çalışmasıyla oluşan prosesin, bu prosesin oluşturduğu proseslerin ve uzak thread açılan tüm proseslerin dosya sistemi aktivitelerini izlemektedir. Takip edilen proseslerden herhangi biri izlenen eylemlerden birini gerçekleştirdiğinde Dosya sistemi izleyici bileşeni, Windows çekirdek geri bildirim mekanizması aracılığıyla haberdar edilmektedir.



Şekil 4.16 : Registry geri bildirim fonksiyonu akış diyagramı.

Analiz makinesi bileşenlerinin bir dosya sistemi filtreleme sürücüsü olarak gerçeklendiği önceki bölümlerde belirtilmişti. Dosya sistemi izleyici bileşeni de diğer analiz makinesi bileşenleri gibi giriş/çıkış yöneticisine filtreleme işlemi gerçekleştireceğini sürücünün sisteme yükleme zamanında belirtmektedir. Bu bileşen de diğerleri gibi geri bildirim fonksiyonlarının giriş/çıkış yöneticisine kayıt ettirilmesi ve fonksiyon tanımlarından oluşan iki bölümden oluşmaktadır.

Uygulamaların yapmış olduğu her bir dosya sistemi işlemi, giriş/çıkış yöneticisi tarafından bir fonksiyon kodu ile tanımlanmaktadır. IRP adı verilen bu kodlar geri bildirim fonksiyonlarının tanımlanmasında kullanılmaktadır [32]. Virmon dosya yazma ve silme işlemlerini takip ettiği için IRP_MJ_CREATE, IRP_MJ_WRITE ve

IRP_MJ_SET_INFORMATION kodlarını geri bildirim fonksiyonlarının kayıt edilmesi için kullanılır. IRP_MJ_CREATE silinmek üzere açılan dosyaların takibi, IRP_MJ_WRITE dosya yazma işlemlerinin takibi, IRP_MJ_SET_INFORMATION ise daha önceden açılmış dosyalardan silinmek ve adının değiştirilmesi istenenlerin takibi için kullanılmaktadır. IRP_MJ_SET_INFORMATION var olan bir dosyaya link oluşturulması olaylarını izlemek için de kullanılmaktadır. İşlem öncesi ve sonrası olmak üzere her bir IRP isteği için iki adet farklı geri bildirim fonksiyonu tanımlanabilmektedir. İstenilirse tüm IRP tipleri için tek fonksiyon yazılabilmektedir. Dosya sistemi izleyici bileşeni tüm IRP kodları için aynı işlem öncesi/sonrası fonksiyonunu kullanmaktadır.

Şekil 4.17 giriş/çıkış işlemi öncesinde geri bildirim mekanizması tarafından çağrılan fonksiyonun prototipini göstermektedir. Fonksiyonun ilk parametresi giriş/çıkış işlemini simgeleyen verileri tutan, FLT_CALLBACK_DATA tipindeki bir değişkeni gösteren işaretçidir. İkinci parametre FLT_RELATED_OBJECTS tipinde bir değişkeni gösteren işaretçi olup hali hazırdaki giriş/çıkış işlemiyle ilişkili olan nesneler hakkında bilgi içermektedir. İşlemin ilgili olduğu dosyalara bu parametre aracılığıyla ulaşılabilmektedir. Üçüncü parametre ise seçmeli olup, işlem sonrası geri bildirim fonksiyonuna parametre olarak verilmektedir.Bu parametre işlem öncesi geri bildirim fonksiyonunda istenilen verileri içeren bir bağlamı göstereren işaretçidir.

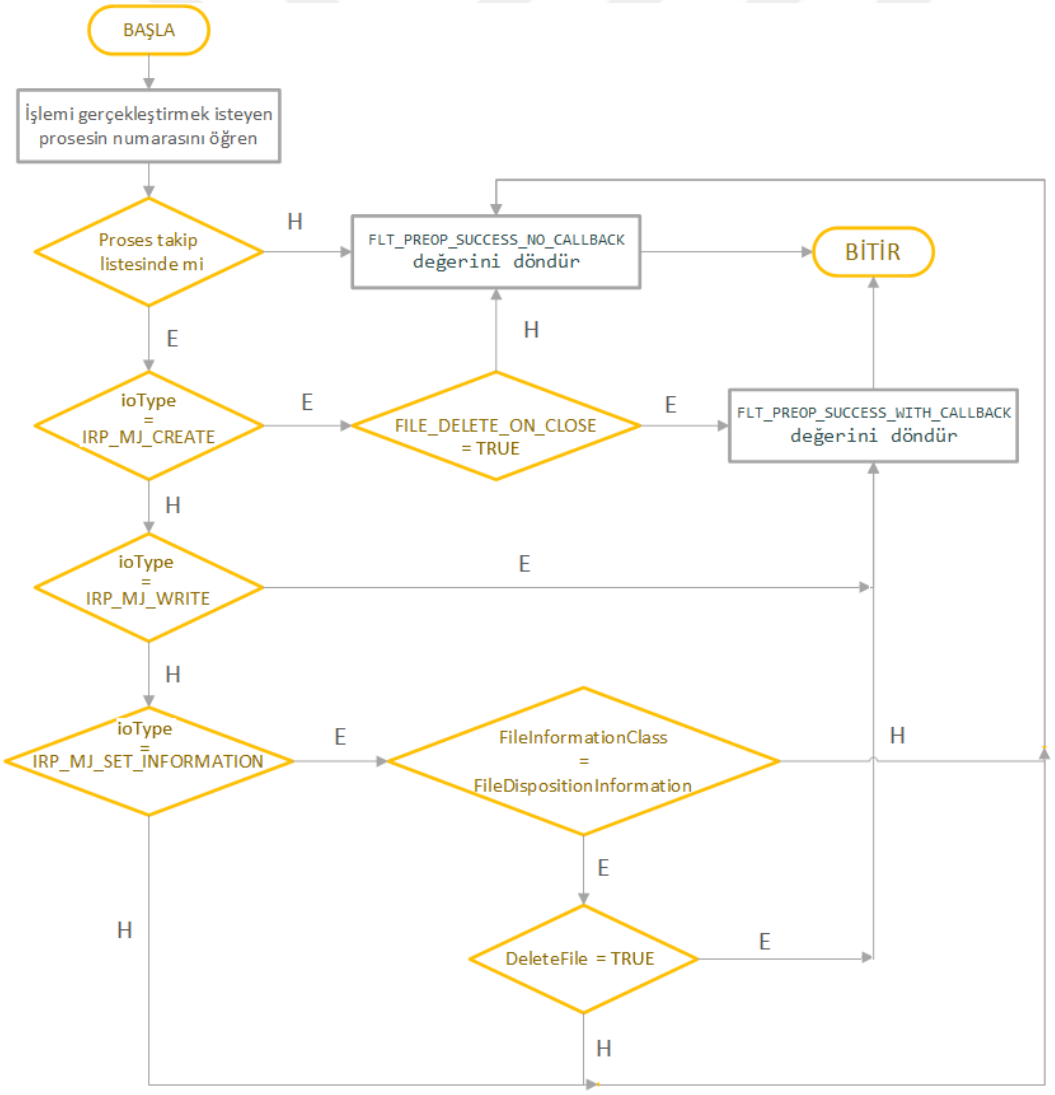
Şekil 4.18 giriş/çıkış işlemi öncesinde çalışan geri bildirim fonksiyonuna ait akış diyagramını göstermektedir. İşlem öncesi geri bildirim fonksiyonunun asıl görevi Dosya sistemi izleyici bileşeninin ilgi alanında olan eylemleri belirlemektir. Fonksiyon, geri bildirim nesnesi tarafından çağrıldığında ilk olarak işlemi

```
typedef FLT_PREOP_CALLBACK_STATUS ( *PFLT_PRE_OPERATION_CALLBACK)(
  _Inout_  PFLT_CALLBACK_DATA Data,
  _In_     PCFLT_RELATED_OBJECTS FltObjects,
  _Out_    PVOID *CompletionContext
);
```

Şekil 4.17 : G/Ç işlemi öncesi çalışan geri bildirim fonksiyonu prototipi.

gerçekleştiren prosesin kimliğini belirler. Eğer proses takip listesinde değilse fonksiyon, işlem sonrası fonksiyonunun çağrılmasına gerek olmadığını belirten FLT_PREOP_SUCCESS_NO_CALLBACK değerini geri döndürür. Proses takip listesinde ise IRP kodu kontrol edilir. IRP kodu IRP_MJ_CREATE ise ilgili dosyanın silinme amaçlı açılıp açılmadığı kontrol edilir. Dosya silinme amaçlı

açılmamışsa FLT_PREOP_SUCCESS_NO_CALLBACK değeri geri döndürülür. Dosya silinme amaçlı açılmış ise işlem sonrası fonksiyonunun çağrılmasının gerektiğini belirten FLT_PREOP_SUCCESS_WITH_CALLBACK değeri döndürülür. IRP kodu IRP_MJ_WRITE ise başka bir kontrole gerek duyulmadan FLT_PREOP_SUCCESS_WITH_CALLBACK değeri döndürülür. IRP kodunun IRP_MJ_SET_INFORMATION olması durumunda ise bir diğer dosya silme yöntemi olan FILE_DISPOSITION_INFORMATION yapısının kullanılıp kullanılmadığına bakılır. Eğer kullanılıyor ve DeleteFile değeri TRUE olarak set edilmişse fonksiyon FLT_PREOP_SUCCESS_WITH_CALLBACK, aksi takdirde FLT_PREOP_SUCCESS_NO_CALLBACK değeri ile döner. Diğer tüm IRP isteklerinde de fonksiyon FLT_PREOP_SUCCESS_NO_CALLBACK değerini döndürerek sonlanır [33].



Şekil 4.18 : G/Ç işlemi öncesi çalışan geri bildirim fonksiyonunun akış diyagramı.

Şekil 4.19'da gösterilen giriş/çıkış işlemi sonrası fonksiyonu prototipinin işlem öncesininkinden tek farkı dördüncü parametredir. Bu parametre işlem sonrası geri bildiriminin nasıl işleyeceğini belirtmektedir.

```
typedef FLT_POSTOP_CALLBACK_STATUS ( *PFLT_POST_OPERATION_CALLBACK)(
  _Inout_    PFLT_CALLBACK_DATA Data,
  _In_       PCFLT_RELATED_OBJECTS FltObjects,
  _In_opt_   PVOID CompletionContext,
  _In_       FLT_POST_OPERATION_FLAGS Flags
);
```

Şekil 4.19 : G/Ç işlemi sonrası çalışan geri bildirim fonksiyonu prototipi.

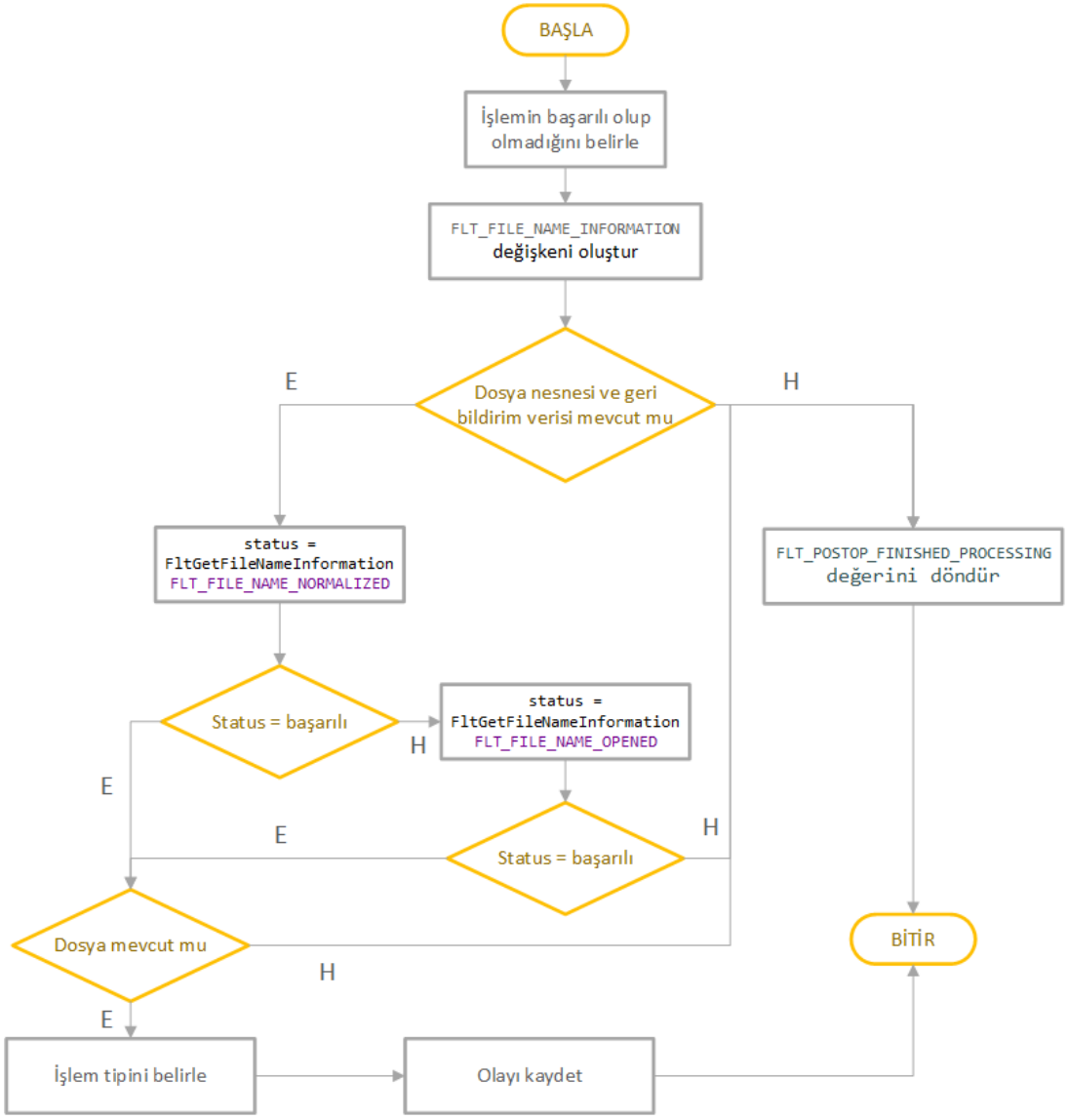
Şekil 4.20 giriş/çıkış işlemi sonrası çağrılan geri bildirim fonksiyonuna ait akış diyagramını göstermektedir. Bu fonksiyon yalnızca takip edilen proseslere ait istenilen tipteki olayların gerçekleşmesi durumunda çağrılmaktadır. Bu yüzden fonksiyon çalışmaya başladığında herhangi bir proses kontrolü gerçekleştirmemektedir. Fonksiyon çalışmaya başladığında parametre olarak aldığı geri bildirim verisinden dosya işleminin başarılı olup olmadığını belirler. Akabinde ilgili dosyanın bilgilerini tutacak olan FLT_FILE_NAME_INFORMATION tipinde bir değişken oluşturulur. Parametre olarak alınan veriler içerisinde dosya nesnesinin mevcut olmaması durumunda işlem FLT_POSTOP_FINISHED_PROCESSING değeri döndürülerek sonlandırılır. Eğer istenen bilgiler mevcut ise FltGetFileNameInformation fonksiyonu yardımıyla dosyanın adına ulaşılmaya çalışır. Bu fonksiyona parametre olarak verilen FLT_FILE_NAME_INFORMATION tipindeki değişkene bellek atanmamışsa, bu fonksiyon değişkene yeterli belleği atayarak hata ile döner. Fonksiyonun ikinci kez çağrılması ile dosya adına ulaşılması beklenmektedir. İşlemin başarılı bir şekilde sonlanması durumunda IRP tipine göre dosya işlemi tipi belirlenerek olay kayıt altına alınır. Hata durumunda işlem sonlandırılır.

4.2.3 Ağ bileşenleri

Zararlı yazılımlar genellikle bulaşmış oldukları sistemler üzerinden hassas verileri çalma amacını taşımaktadırlar. Saldırganın bu verilere erişim sağlayabilmesi için internet üzerinden ele geçirilen bilgisayar ile iletişime geçmesi gerekmektedir. Bunun dışında, bot olarak ifade edilen köle bilgisayarlar da saldırganlar tarafından ele geçirilmiş komuta kontrol sunucularından komutlar almakta ve bu komutları

yerine getirmektedirler. Bu tip işlemler gerçekleştiren uygulamaların tespiti için analiz makinelerinin ağ trafiğinin izlenmesi zorunluluk arzetmektedir.

Virmon zararlı yazılım analiz sistemi ağ bileşenleri aracılığıyla analiz makinelerinin ağ trafiğini izleme yeteneğine sahiptir. Önerilen sistemde VLAN, VPN, IPS/IDS, güvenlik duvarı gibi farklı ağ çözümleri kullanılmıştır. Çekirdek seviyesinde ağ sürücüsü *(NDIS driver)* geliştirmenin zorluğu sebebiyle sistem genelinde ağ aktivitelerinin toplanması fikri benimsenmiştir. Bu yöntemin avantajları ve dezavantajlarından bahsedilecektir.



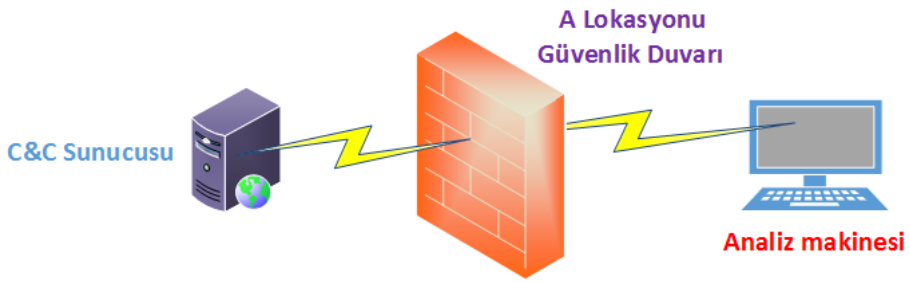
Şekil 4.20 : G/Ç sonrası çalışan geri bildirim fonksiyonunun akış diyagramı.

4.2.3.1 Dağıtık sensör ağı

Zararlı yazılımların ağ adresine bakarak analiz ortamlarını tespit etmeye çalışmaları analiz sisteminin ağ trafiğinin farklı lokasyonlara dağıtılması fikrini doğurmuştur. Bu amaçla bir önceki çalışmamızda gerçeklemiş olduğumuz dağıtık zararlı yazılım toplama sistemindeki sensör ağı analiz sistemine entegre edilmiştir [34].

Sensör olarak belirtilen ağ cihazları birer mini bilgisayardan oluşmaktadır. Bu cihazlar internet üzerinde herhangi bir lokasyon üzerine kurulabilmektedir. Sensör, sahip olduğu public ağ adresinin zararlı yazılım analiz sistemindeki bir ya da birden fazla makinenin private ağ adresi ile eşleştirilmesi için kullanılmaktadır. Temel işlevi eşleşmiş olduğu analiz makinelerinin trafiğini internete iletmektir. Şekil 4.2'de görüldüğü gibi sensör ağındaki her bir cihaz güvenli kanallar *(VPN)* aracılığı ile analiz merkezine bağlanmaktadır.

VPN kullanımı ile her bir analiz makinesi doğrudan internete bağlı gibi gösterilebilmektedir. Analiz makinelerinden çıkan paketlerin IP katmanındaki bilgileri (kaynak-hedef ağ adresi, port bilgileri vb.) sensörler üzerinde güncellenerek paketi gönderen makinenin sensörmüş gibi algılanması sağlanabilmektedir. Bu yöntemle zararlı yazılımın bizim analiz sistemimizde değil sıradan bir son kullanıcının sisteminde çalıştırıldığı imajı verilmeye çalışılmaktadır. Şekil 4.21'de her bir analiz makinesinin mantıksal görünümü belirtilmiştir. Analiz makinesi bu yöntemle sensörün bulunduğu ağın güvenlik duvarının arkasındaymış gibi görülmektedir.



Şekil 4.21 : Analiz makinelerinin sanal topolojisi.

Analiz sistemindeki her bir makineye ayrı bir C class ağ ayrılmıştır. Analiz makinesi üzerinde çalıştırılan şüpheli bir uygulama internete doğru ağ trafiği gerçekleştirdiğinde, VPN sonlandırıcı ilgili sensörü otomatik olarak belirleyip trafiği bu sensöre yönlendirmektedir. Ek olarak sistemin izolasyonunu sağlamak, analiz

makinelerinin birbirini görmesini engellemek için farklı VLAN'lar iç ağdaki güvenlik duvarında ve sanallaştırma sunucularında tanımlanmıştır.

4.2.3.2 Alan adı sunucusu

DNS, internete ya da özel ağlara *(private network)* bağlı bilgisayar, servis ya da herhangi bir kaynağa ait alan adlarını ya da makine isimlerini ağ adreslerine çeviren dinamik bir veritabanı servisidir. Internet ağ adreslerine dayalı olarak çalıştığı için çok önemli bir servistir. DNS servisi sayesinde son kullanıcılar ulaşmak istedikleri sunucunun ağ adresini bilmek zorunda kalmamaktadırlar.

Uygulamalar tarafından yapılan DNS isteklerinin takibi iyi bir zararlı yazılım analizi yapabilmek adına önemlidir. Genellikle botnetler tarafından kullanılan fast-flux adlı bir DNS tekniği ile oltalama *(phishing)* saldırıları ve zararlı yazılım dağıtımı yapan sunucular gizlenebilmektedir. Fast-flux tekniğinin arkasındaki temel fikir, bir alan adının çok sayıda ağ adresleri ile eşleştirilmesinden ibarettir [35]. Çok yüksek frekanslarda ilgili alan adına ait DNS kayıtları değiştirilerek karşı düşen ağ adresi gizlenebilmektedir. Analiz edilen uygulamaların yapmış olduğu DNS isteklerinin mevcut alan adı karalisteleri ile karşılaştırılabilmesi ya da güvenilir olmayan alan adlarına yapılan isteklerin izlenebilmesi zararlı yazılım analizinin başarısını yükseltmektedir.

Virmon dinamik zararlı yazılım analiz sistemi içerdiği DNS sunucusu aracılığıyla analiz makinelerinin yapmış olduğu alan adı isteklerini takip edebilmektedir. DNS sunucusu için FreeBSD işletim sistemi üzerine kurulu BIND yazılımı tercih edilmiştir. Analiz sisteminde yer alan her bir makineye DNS sunucusunun ağ adresi otomatik olarak atanmıştır. Bununla birlikte zararlı yazılımların alan adlarını dışarıya açık DNS sunucularına (8.8.8.8 adresli Google DNS sunucusu gibi) sormalarını engellemek için analiz makinelerinden gelen DNS istekleri sistem içinde yer alan merkezi güvenlik duvarı aracılığıyla DNS sunucusuna yönlendirilmektedir. Bu yöntemle zararlı yazılımlar tarafından yapılan her bir DNS isteği cevaplanabilmekte ve kayıt altına alınabilmektedir. DNS sunucusu sorgulanan alan adını, sorgulama zamanını ve isteği yapan makinenin ağ adresini kaydetmektedir. İlgili kayıt dosyası belirli periyotlarla okunup ayrıştırılarak istekler veritabanına kaydedilmektedir. Toplanan kayıtlar arasında analiz edilen uygulamalara ait olmayan, zararlı içerik

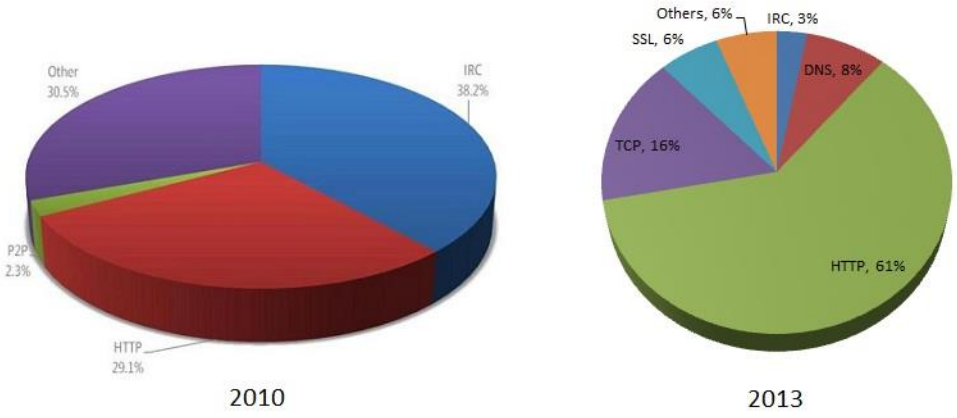
içermeyen alan adlarına yapılan istekler de bulunmaktadır. Kayıtların analizi sırasında bu alan adlarına yapılan istekler kapsam dışında bırakılmaktadır.

4.2.3.3 IPDS

Saldırı tespit ve engelleme sistemleri zararlı aktiviteleri tespit etmek amacıyla ağ trafiğini ve sistem aktivitelerini izleyebilen ağ güvenliği araçlarıdır. Bu sistemler, ağ trafiğini izlerken içerdiği kural setleri ile trafiğin kötü niyet içerip içermediğini tespit etmeye çalışır. Konfigürasyona göre kuralların eşleşmesi durumunda zararlı trafiği engelleme yeteneğine de sahiptir.

Zararlı yazılımların dinamik analizi için çalıştırılmasının gerekmesi, olası muhtemel ağ saldırılarının meydana gelmesi durumunu ortaya çıkarmaktadır. Analiz edilen uygulamanın yayılma ve internet üzerindeki başka makinelere saldırma ihtimaline karşı saldırı engelleme sistemlerinin kullanılması zorunlu olmuştur.

Günümüz bilgisayar sistemleri içinde barındırdığı güvenlik duvarları aracılığıyla son kullanıcıları ağ üzerinden yapılan saldırılara karşı bir nebze koruyabilmektedir. Kullanıcı bilgisayarları ile yalnızca belirli protokoller (HTTP, FTP vb.) aracılığı ile iletişime geçilebilmektedir. Bu yüzden günümüz zararlı yazılımları komuta kontrol sunucuları ile bu tip protokolleri kullanarak haberleşmektedir. Özellikle botnetlerin komuta kontrol sunucuları ile yaptıkları haberleşmede IRC protokolünden HTTP protokolüne doğru geçiş yaptıkları gözlemlenmiştir. Şekil 4.22'de görüldüğü üzere 2010 yılında botnet haberleşmesi için en kullanılan protokol IRC iken 2013 yılında en çok kullanılan protokol belirgin bir şekilde HTTP olmuştur [36-37] Bu durum analiz makinelerinin HTTP trafiğinin izlenmesini gerekli kılmaktadır.



Şekil 4.22 : 2010 ve 2013 yıllarında botnetler tarafından kullanılan protokoller.

Zararlı yazılım türleri bölümünde dropper adı verilen bir yazılım çeşidinden bahsedilmişti. Bu tip zararlı yazılımların temel kullanım amacı asıl işi yapacak zararlı yazılımı internet üzerinden indirip hedef sistemde çalıştırmaktır. Bu sebeple ağ trafiğinden dosya ayıklanabilmesi yeni zararlı yazılımların toplanabilmesine ve analiz edilebilmesine yardımcı olmaktadır.

Virmon zararlı yazılım analiz sistemi saldırı tespit ve engelleme sistemi olarak açık kaynak kodlu bir proje olan Suricata'yı kullanmaktadır [38]. Saldırı tespit ve engelleme yeteneğinin yanı sıra Suricata, gerçek zamanlı trafik üzerinden HTTP isteklerini ve dosyaları çıkarabilmektedir. HTTP isteklerini tutan kayıt dosyası belirli aralıklarla okunarak ilgili kayıtlar veritabanında saklanmaktadır. Eğer trafik üzerinden yeni bir çalıştırılabilir dosya çıkarılmışsa bu dosya da analiz edilmek üzere kuyruğa eklenmektedir.

4.2.3.4 NetFlow sunucusu

NetFlow, Cisco tarafından geliştirilen ve IP trafik bilgisini toplayan bir ağ protokolüdür. NetFlow ayrıca açık kaynak işletim sistemlerince de üretilebilmektedir. Dolayısıyla Linux, FreeBSD ve OpenBSD platformlarıncada da desteklenmektedir [39].

Virmon, NetFlow sunucusunu analiz makinelerinin gerçekleştirdiği ağ trafiğinin özetini çıkarmak için kullanmaktadır. Bu özet, trafiğin hangi ağ adresleri ve portları arasında gerçekleştiğini, hangi protokolün kullanıldığını, veri akışının ne kadar sürdüğünü, aktarılan verinin boyutu gibi önemli bilgiler içermektedir. Bu bilginin çıkarılabilmesi için sistem içerisinde yer alan güvenlik duvarı aracılığıyla analiz makinelerine ait ağ trafiğinin kopyası NetFlow sunucusuna gönderilmektedir. NetFlow sunucusu periyodik olarak bu bilgileri işleyerek veritabanında saklamaktadır.

4.2.4 Otomatik analiz sistemi

Bu bölümde zararlı yazılımların analizinin otomatikleştirilmesi ve izlenebilmesini sağlayan bileşenler tanıtılacaktır.

4.2.4.1 Uygulama sunucusu

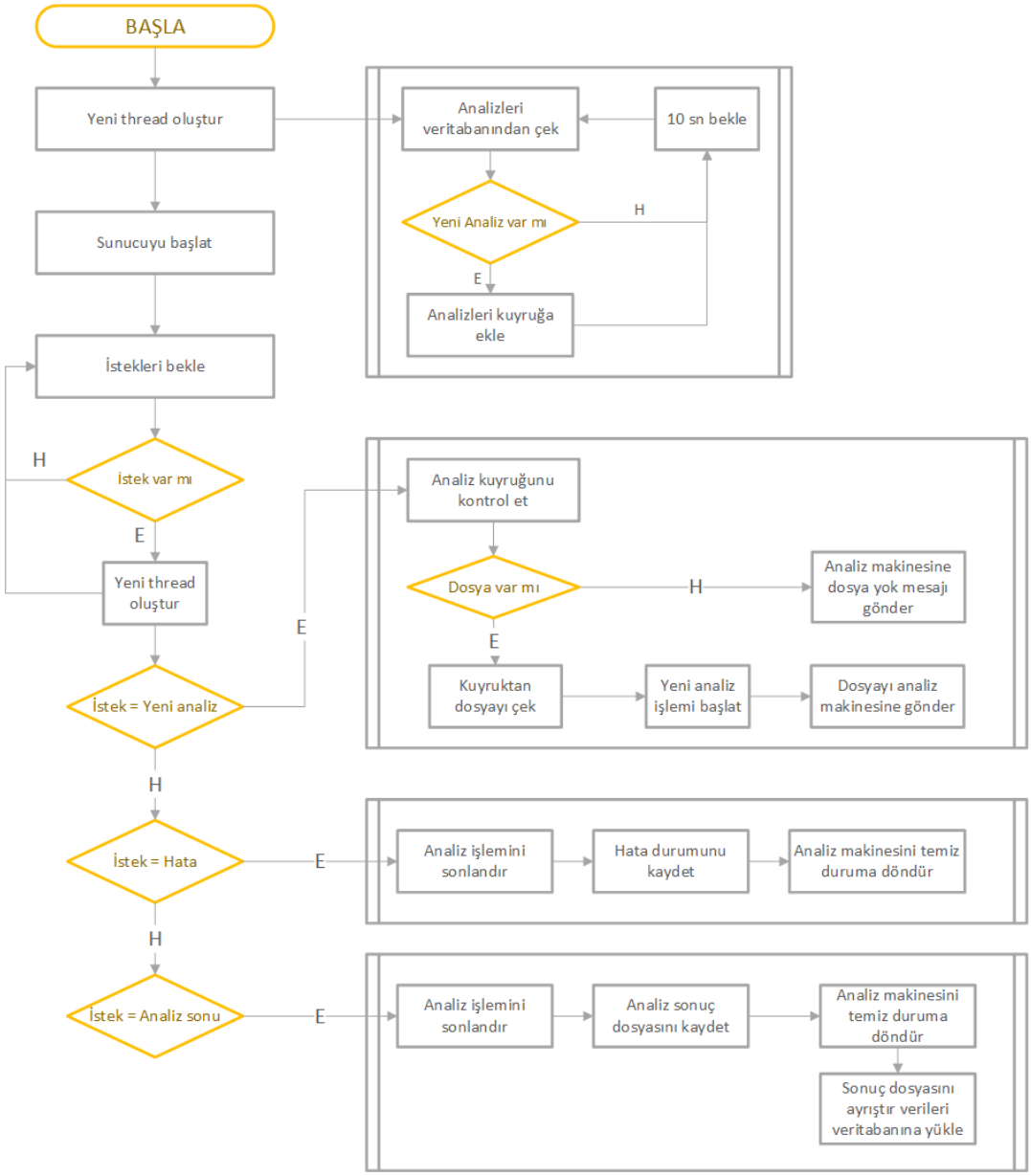
Zararlı yazılım analiz süreçleri uygulama sunucusu tarafından yönetilmektedir. Sunucu multithread çalışarak aynı anda gelen birden fazla analiz isteklerine cevap verebilmektedir. Analiz makinelerinden gelen talepler sunucu tarafından değerlendirilerek dosyalar isteği gönderen makinelere iletir. Uygulama sunucusu, periyodik olarak veritabanını sorgulayarak yeni gelen dosyaları öncelikli kuyruğuna *(priority queue)* ekler. Virmon zararlı yazılım analiz sisteminde bulunan tüm bileşenlerden ilgili aktiviteleri toplayıp veritabanında toplar. Analiz işlemleri sonrasında makinelerin temiz duruma *(clean state)* döndürülmesinden sorumludur.

Uygulama sunucusu, farklı ağlar üzerinde bulunan analiz makineleri ile haberleşmeyi Facebook tarafından geliştirilmiş olan *Thrift* uygulamasını kullanarak sağlamaktadır. Thrift uygulaması verimli ve ölçeklenebilir alt sistemlerin geliştirilebilmesini kolaylaştırmayı amaçlamaktadır. Farklı programlama dillerinde geliştirilmiş uygulamalar arasındaki haberleşmenin kolaylaştırılması da Thrift'in amaçları arasında yer almaktadır [40]. C# programlama dili ile geliştirilen uygulama sunucusu ile C++ dili ile geliştirilmiş olan analiz makinesi uygulamaları arasındaki veri iletişimi Thrift'in sağlamış olduğu altyapı sayesinde mümkün olmaktadır.

Şekil 4.23'te uygulama sunucusuna ait akış diyagramı gösterilmiştir. Sunucu çalışmaya başladığında yeni bir thread oluşturarak öncelikli dosya kuyruğunu güncel tutmaya çalışmaktadır. Bu thread belirli aralıklarla veritabanını sorgulayarak analizi yapılmak istenen yeni dosyaların mevcut olup olmadığını kontrol etmektedir. İşlem sonunda belirli bir süre uyku moduna geçerek beklemektedir. Kuyruk güncellendikten sonra sunucu multithreadli olarak ve belirli bir port üzerinde çalışacak şekilde başlatılır.

Sunucu çalışmaya başladıktan sonra analiz makinelerinden gelecek olan istekleri beklemeye başlar. Yeni istek geldiğinde gelen isteğin tipi kontrol edilir. Analiz makineleri ve uygulama sunucusu arasında 3 farklı tipte mesaj alış-verişi gerçekleşmektedir. Bunlar yeni analiz işlemi isteği, hata bildirim ve analiz işlemi tamamlama isteğidir. Sunucuya gelen istek yeni analiz isteği ise, uygulama sunucusu öncelikle analiz kuyruğunu kontrol eder. Kuyrukta dosya yoksa sunucu analiz makinesine analiz edilecek dosya olmadığı mesajını gönderir. Dosya var ise ilgili dosya kuyruktan çekilir, dosyaya ait bilgiler (analiz başlangıç zamanı, analiz

makinesi vb.) veritabanında güncellenir ve dosya analiz makinesine gönderilerek yeni analiz işlemi başlatılır. Analiz makinesinden gelen istek hata durumunu belirtiyor ise veritabanında ilgili dosyaya ait analiz başlangıç zamanı ve analiz makinesi alanları silinerek dosyanın hiç analiz edilmemiş gibi yorumlanması sağlanır. Hata durumuna ait bilgiler kaydedilir ve ilgili analiz makinesi temiz duruma döndürülür. Gelen istek analiz işleminin sonlandığını belirtiyor ise veritabanında ilgili dosyaya ait analiz bitiş zamanı bilgisi güncellenir, analiz makinesinden gelen sonuç dosyası ayrıştırılarak elde edilen aktiviteler veritabanına kaydedilir. Bu işlemin akabinde analizi gerçekleştiren makine temiz duruma döndürülürek işlem sonlandırılmış olunur.

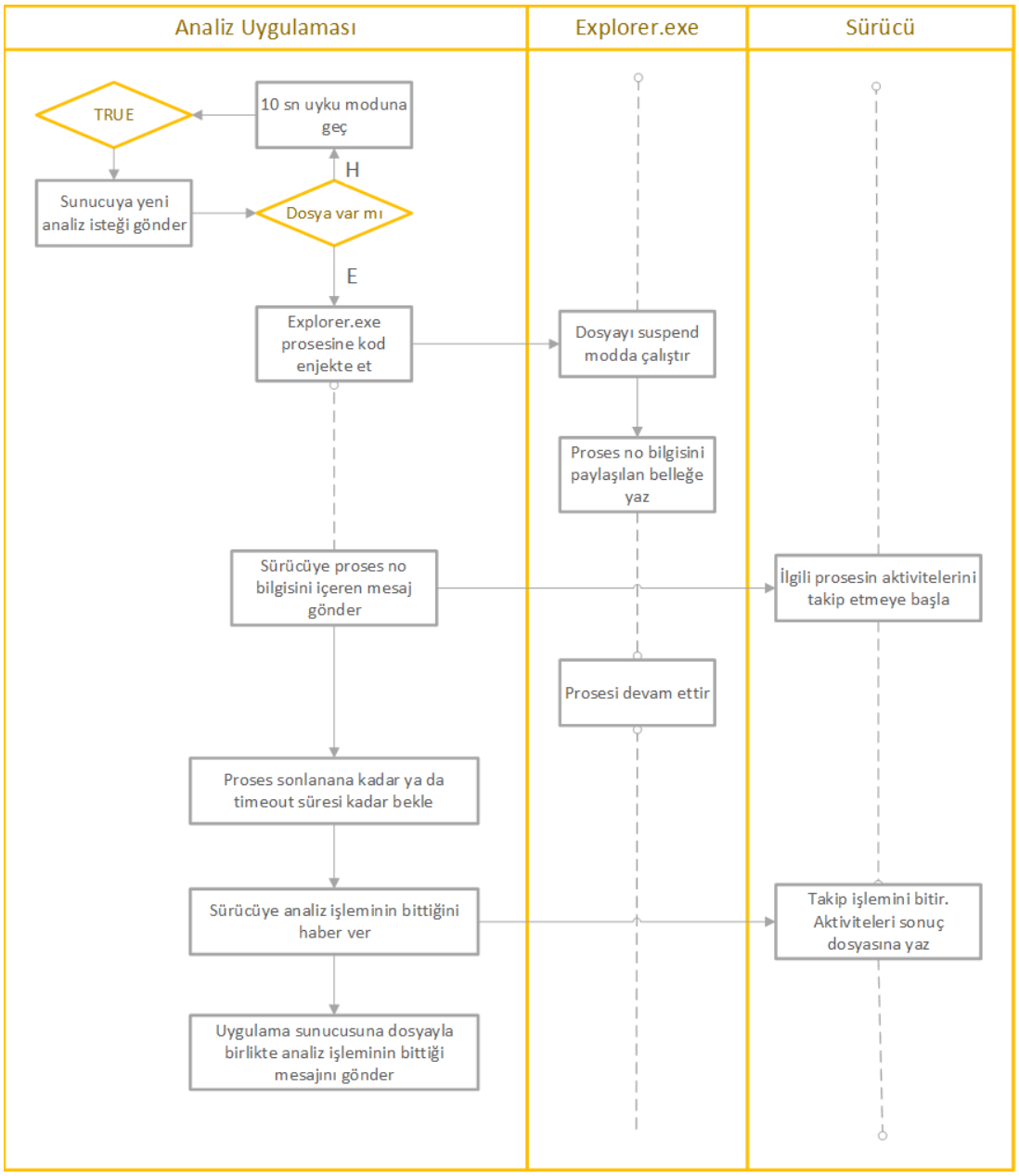


Şekil 4.23 : Uygulama sunucusu akış diyagramı.

4.2.4.2 Analiz Uygulaması

Otomatik analiz sisteminin analiz makineleri üzerindeki bileşenini oluşturmaktadır. Analiz uygulamaları, uygulama sunucusu ile analiz makineleri arasında bir köprü görevi görmektedir. Uygulama sunucusundan alınan dosyalar analiz uygulamaları aracılığıyla çalıştırılmaktadır. Analiz işlemi sonucunda elde edilen aktivite dosyası analiz uygulaması aracılığıyla uygulama sunucusuna gönderilmektedir. Uygulama sunucusu bölümünde de belirtildiği üzere analiz uygulamaları C++ dilinde geliştirilmiştir. Aynı zamanda Thrift uygulamasını kullanarak uygulama sunucusu ile haberleşmeyi sağlamaktadır.

Şekil 4.24 analiz uygulamasına ait akış diyagramını göstermektedir. Analiz uygulaması çalışmaya başladığında ilk olarak uygulama sunucusu ile bağlantıya geçerek analiz edilmek üzere yeni dosya talebinde bulunur. Eğer uygulama sunucusundan dosya yok mesajı dönerse, uygulama 10 saniyelik uyku moduna geçip akabinde isteği yeniler. Eğer dosya gönderilirse, analiz uygulaması sistem üzerinde o an çalışan explorer.exe prosesine kod enjekte ederek uygulamayı suspend modda çalıştırması istenir. Bu işlemdeki amaç analiz edilen uygulamalara parent proses olarak explorer.exe'nin gösterilmek istenmesidir. Bazı zararlı yazılımlar parent prosesi kontrol ederek analiz ortamında ya da normal bir kullanıcı tarafından mı çalıştırıldığını anlamaya çalıştıkları için bu yöntem tercih edilmiştir. Explorer.exe prosesi suspend modda çalıştırdığı dosyadan elde ettiği proses no bilgisini paylaşılan belleğe yazar. Analiz uygulaması paylaşılan bellekten okuduğu proses no bilgisi ile analiz makinesi bileşeni olan sürücüye mesaj göndererek ilgili prosese ait aktivitelerin toplanmaya başlamasını ister. Akabinde sürücü takip işlemini başlatır ve explorer.exe prosesi suspend modda bekleyen prosesi devam ettirir. Analiz uygulaması analiz edilen dosyaya ait proses sonlanana kadar ya da belirli bir zamanaşımı süresi kadar bekler. Bu sürenin sonunda analiz uygulaması sürücüye mesaj göndererek takip işlemini bitirmesini söyler. Sürücü takip işlemini bitirir ve analiz edilen uygulamaya ait aktiviteleri bir dosyaya yazar. Analiz uygulaması oluşturulan aktivite dosyasını uygulama sunucusuna göndererek analiz işleminin tamamlandığını bildirmiş olur.



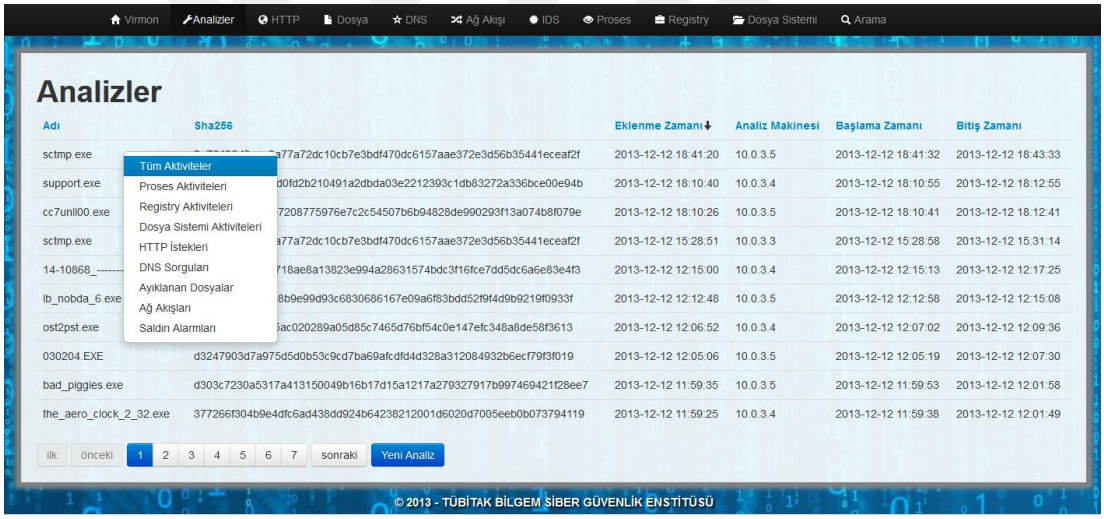
Şekil 4.24 : Analiz uygulaması akış diyagramı.

4.2.4.3 Web uygulaması

Zararlı yazılım analiz sisteminin son kullanıcıya bakan kısmını oluşturmaktadır. PHP programlama dili ile Zend Framework 2 kütüphanesi kullanılarak geliştirilmiştir. Zend Framework, açık kaynak kodlu, nesneye dayalı programlama tekniğini kullanan, PHP 5 için web uygulama altyapısı sunan bir kütüphanedir. ZF, ilgilerin ayrımı *(separation of concerns)* yaklaşımı ve ileri MVC *(Model-View-Controller)* gerçeklemesi ile uygulama geliştirmeyi basitleştirmekte, kod esnekliği, ölçeklenebilirliği ve yeniden kullanılabilirliği kazandırmaktadır [41].

Web uygulaması, dosyalara ait analiz sonuçlarının izlenmesine olanak sağlamaktadır. Uygulamalar tarafından gerçekleştirilen HTTP istekleri, yapılan DNS sorguları, ağ trafiği üzerinden geçen çalıştırılabilir dosyalar, gerçekleşen saldırı alarmları ve NetFlow bilgileri bu arayüz sayesinde takip edilebilmektedir. Ayrıca analiz makinelerinden elde edilen proses, registry ve dosya sistemi aktiviteleri de arayüz vasıtasıyla görülebilmektedir. Gerçekleşen DNS ve HTTP aktivitelerinde ilgili alan adlarından güvenilir olanlar süzülebilmektedir. Ek olarak bağlantı kurulan uzak sunucuların da bulundukları ülkeler GEOIP veritabanı sayesinde gösterilebilmektedir.

Web uygulamasının bir diğer özelliği de el ile ya da otomatik olarak analiz sistemine dosya submit edilebilmesine olanak sağlamaktadır. Analiz edilecek dosya sayısının çok fazla olması durumunda sunulan submit arayüzü sayesinde işlemler kolaylaşmaktadır. Web uygulamasına ait örnek bir arayüz Şekil 4.25'te gösterilmiştir.



Şekil 4.25 : Virmon web uygulaması.

5. DEĞERLENDİRME

Bu bölümde Virmon zararlı yazılım analiz sisteminin incelemiş olduğu uygulamaların yaptığı aktiviteleri başarılı bir şekilde topladığını göstermek için 2013 Aralık ayında tespit edilmiş [42], hali hazırda aktif olan bir zararlı yazılımın analiz sonuçları verilecektir. Analiz edilen uygulamanın Virmon'un sahip olduğu tüm yetenekleri gösterebilecek aktiviteler gerçekleştirdiği görülmüştür. Yazılım Windows 8 64-bit işletim sistemine sahip bir sanal makine üzerinde otomatik olarak incelenmiştir. Analiz için süre aşımı 2 dakika olarak belirlenmiştir. Analiz işlemi sonunda oluşan tüm proseslerin süre aşımından önce sonlandıkları görülmüştür. İşlem sonunda her bir aktivite tipine göre elde edilen olay sayısı Çizelge 5.1'de gösterilmiştir.

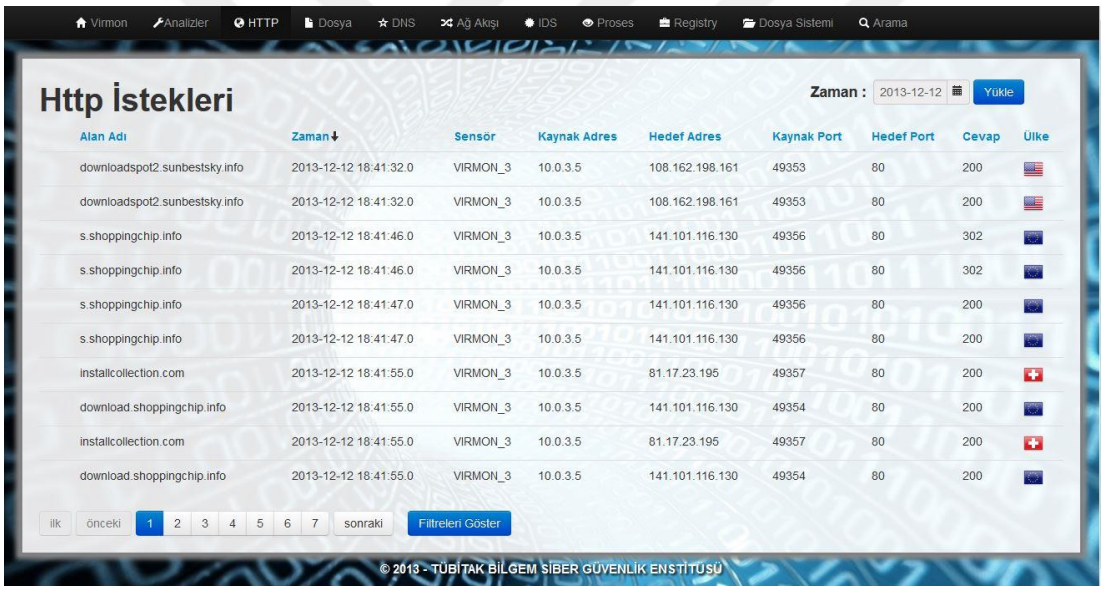
Çizelge 5.1 : Aktivite tipine göre elde edilen olay sayıları.

Aktivite Tipi	Olay Sayısı
Proses	30
Registry	522
Dosya sistemi	371
HTTP isteği	10
DNS sorgusu	5
Dosya indirme	3
Ağ akışı	9
Saldırı tespit sistemi	1

Elde edilen alarm sayılarına göz atıldığında zararlı yazılımın çok sayıda proses oluşturduğu anlaşılmaktadır. Ayrıca zararlı yazılımın internet üzerinden bazı sunucularla bağlantı kurarak yeni zararlı yazılımlar indirdiği de görülmektedir. Takip eden bölümlerde her bir aktivite tipinde hangi olayların gerçekleştiği mümkün olduğunca anlatılmaya çalışılmıştır. Registry ve dosya sistemi aktivitelerinin sayısının çok fazla olması nedeniyle önemli görülen bazı olayların üzerinde durulmuştur. Diğer aktivite tiplerinde gerçekleşen olayların daha iyi anlatılabilmesi için ilk olarak yapılan ağ aktiviteleri hakkında bilgi verilmiştir.

5.1 Yapılan HTTP İstekleri

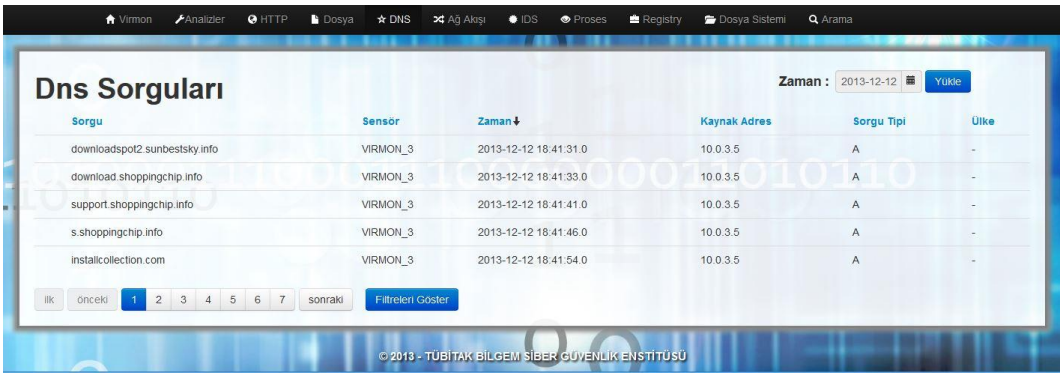
Analiz makinesi trafiğinin incelenmesi sonucunda çıkarılan HTTP istekleri Şekil 5.1'de gösterilmiştir. Görüldüğü üzere Amerika, İsviçre ve Avrupa Birliği ülkelerinde bulunan 5 farklı web sunucusuna yakın zamanlarda HTTP isteği gönderilmiştir. Ağ bileşenleri bölümünde anlatıldığı gibi, hangi prosesin hangi HTTP isteğini yaptığı tam olarak bilinememektedir. Analiz sürecinde şüpheli tek bir dosya çalıştığı için yapılan isteklerin büyük ölçüde analiz edilen uygulama tarafından yapıldığı kabul edilmektedir. Sistemin sahip olduğu güvenilir liste özelliği sayesinde "*.microsoft.com" gibi bilinen alan adlarına yapılan istekler ayıklanabilmekte, bu sayede mümkün olduğunca doğru sonuca varılabilmektedir. İnceleme sonucunda yazılımın zararlı olduğu kanaatine varılırsa, HTTP isteklerinin yapılmış olduğu ağ adresi karalistelere alınarak farklı kişi veya kuruluşlarla paylaşılabilinecektir.



Şekil 5.1 : Yapılan HTTP istekleri.

5.2 Gönderilen DNS Sorguları

DNS sunucusundan elde edilen bilgilere göre, analiz zamanı aralığında işlemi gerçekleştiren makineden yapılan DNS sorguları Şekil 5.2'de gösterilmiştir. Analiz süresince güvenilirliği belli olmayan 5 adet alan adı sorgulanmıştır. Bu alan adlarının HTTP isteği gönderilen web sitelerine ait olduğu görülmektedir. Farklı bir alan adının sorgulanmaması ağırlıklı olarak HTTP trafiğinin gerçekleştiğini göstermektedir.



Şekil 5.2 : Sorgulanan alan adları.

5.3 İndirilen Dosyalar

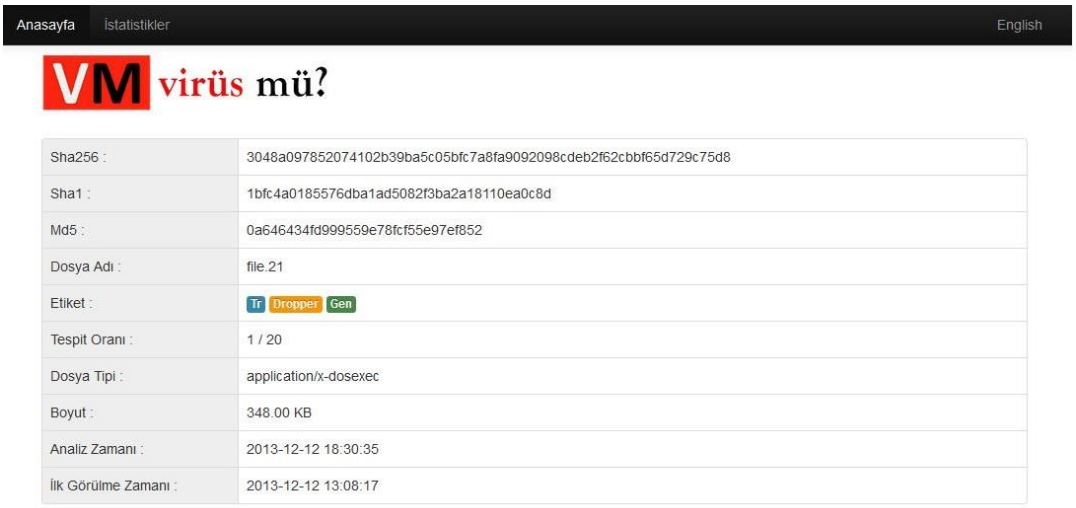
Analizi gerçekleştirilen uygulama çalışması esnasında HTTP üzerinden 3 farklı çalıştırılabilir dosya indirmiştir (Şekil 5.3). Bu dosyaların indirildikleri sunucuların, alan adı sorgulanan ve HTTP isteği gönderilen 5 sunucudan 3'ünü teşkil ettiği anlaşılmaktadır.

Şekil 5.3 : Ağ trafiğinden çıkarılan çalıştırılabilir dosyalar.

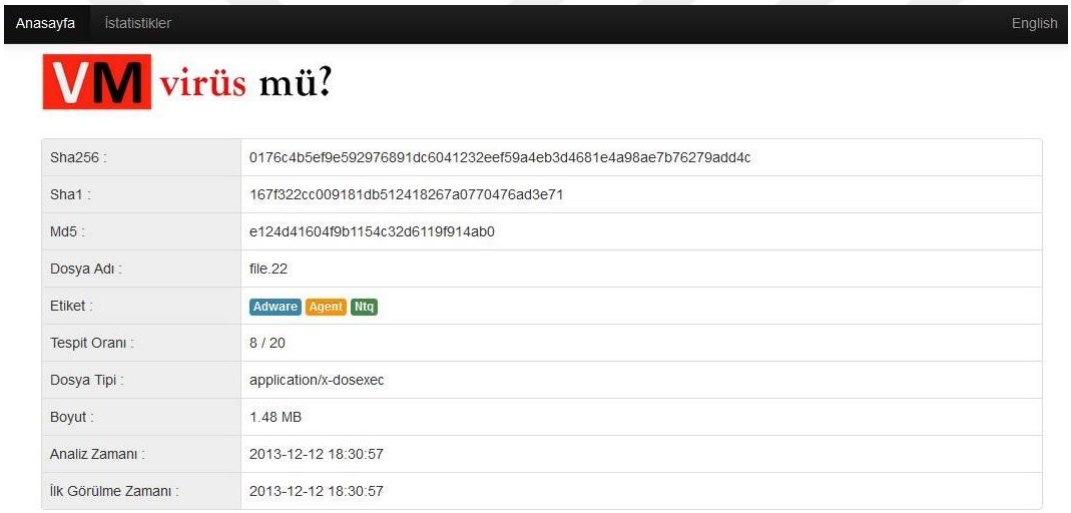
Şekil 5.4, Şekil 5.5 ve Şekil 5.6'da indirilen dosyalara ait çoklu anti-virüs tarama sonuçları gösterilmiştir. Analiz makinesi tarafından indirilen dosyalar incelendiğinde, bu dosyaların da aslında birer zararlı yazılım olduğu anlaşılmaktadır. Bir diğer dikkat çeken durum ise dosyaların indirildiği sayfalara arka arkaya istek gönderildiğinde aynı dosyanın farklı adlar altında oluşturulmasıdır.

5.4 Gerçekleşen Ağ Trafiği

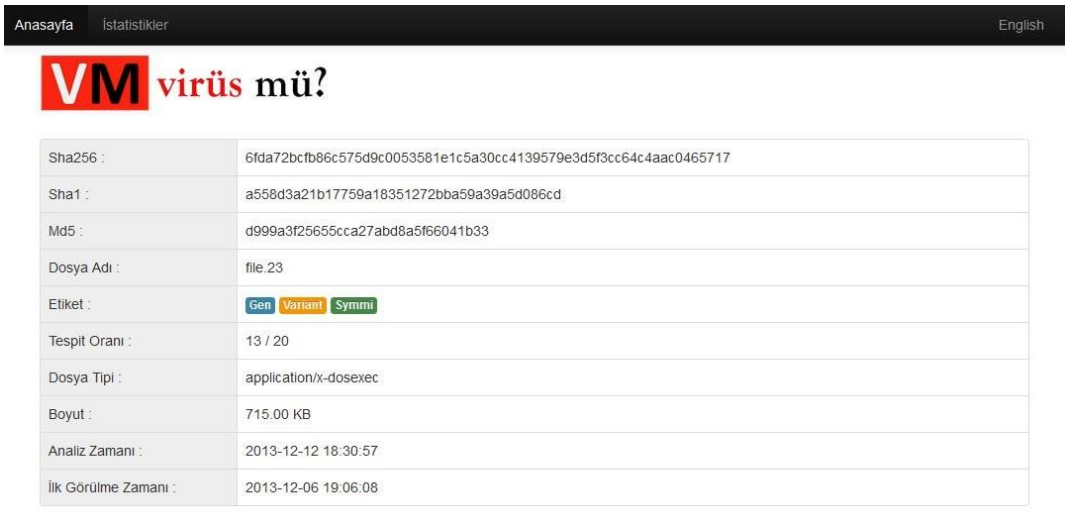
Şekil 5.7'de görülen analiz makinesinin ağ trafiğinden elde edilen NetFlow kayıtları incelendiğinde, HTTP isteklerinin gönderildiği sunucuların ağ adreslerinden başka adreslerle haberleşme yapılmadığı anlaşılmaktadır. DNS ve HTTP'de olduğu gibi bu bölümde de güvenilir kaynaklarla yapılan trafik ihmal edilmiştir.



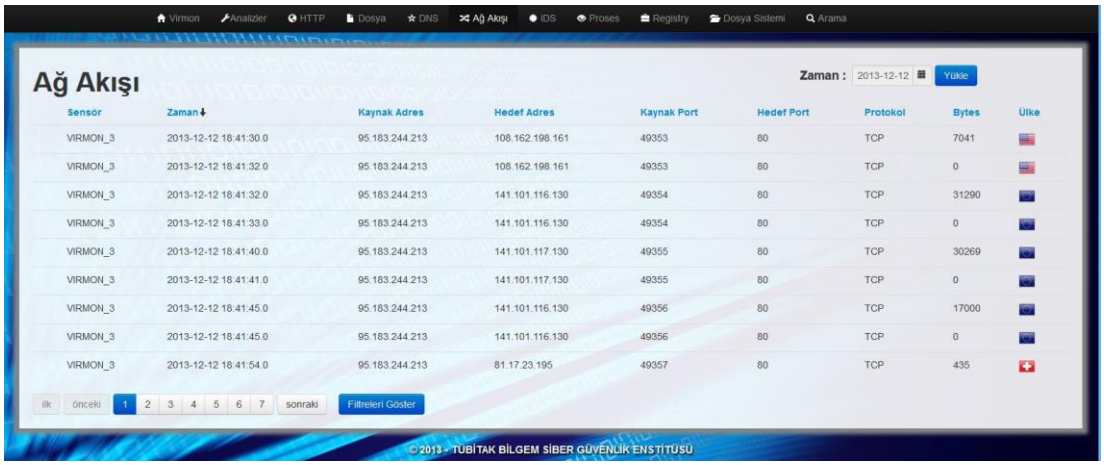
Şekil 5.4 : Tmps.i01 dosyasının anti-virüs tarama sonuçları.



Şekil 5.5 : T8iJOoLsam.exe dosyasının anti-virüs tarama sonuçları.



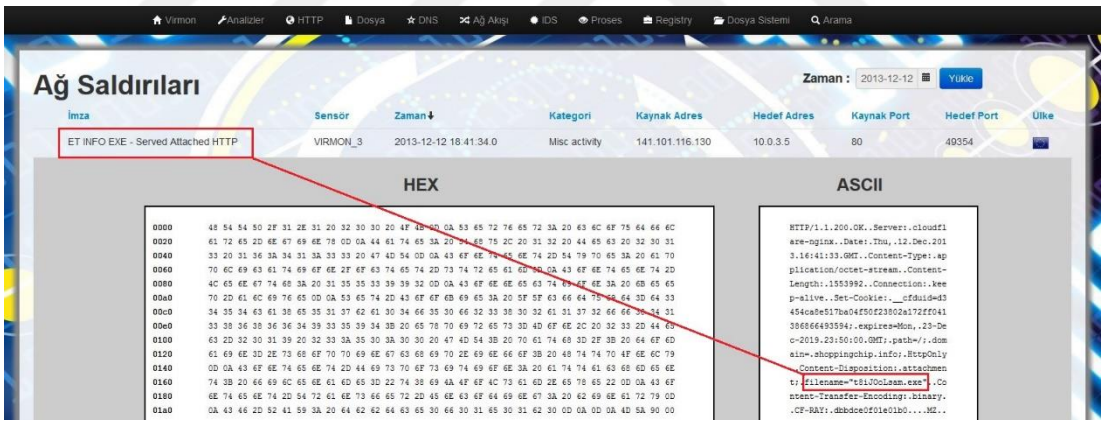
Şekil 5.6 : Supporter_.exe dosyasının anti-virüs tarama sonuçları.



Şekil 5.7 : NetFlow kayıtları.

5.5 Oluşan Saldırı Tespit Sistemi Alarmları

Saldırı tespit ve engelleme sistemi olarak kullanan Suricata'nın fazla bir alarm üretmediği gözlenmiştir. Oluşan tek alarm yalnızca t8iJOoLsam.exe dosyasının indirildiğini belirtmektedir (Şekil 5.8). Başka alarmın gözükmemesi zararlı yazılımın analiz süresince herhangi bir saldırı da bulunmadığını göstermektedir.



Şekil 5.8 : Saldırı tespit sistemi alarmları.

5.6 Proses Aktiviteleri

Zararlı yazılımın çalıştırılması ile elde edilen proses, çok sayıda alt prosesin oluşturulmasını tetiklemektedir. Toplanan olay sayısının yarısının (15), proses oluşturma işlemi olduğu görülmektedir. Genel olarak proses oluşturma işlemlerinden önce çalıştırılacak dosyanın ana proses tarafından oluşturulduğu gözlenmiştir. Bu durum analiz edilen zararlı yazılımın bir dropper olma ihtimalini kuvvetlendirmektedir.

Şekil 5.9, oluşturulan prosesler arasındaki ilişkiyi göstermektedir. Görüldüğü gibi karmaşık bir yapıda bağlantı kurulmuştur. Proseslere verilen isimlerin anlamsız olması, rastgele isimlerin verildiğini göstermektedir. Ana prosesin drop ettiği dosyadan 5 adet aynı prosesi oluşturması dikkat çekmektedir. Yine oluşturulan proseslerin çoğu 4 numaralı "Te.exe" prosesi tarafından çalıştırılmıştır. Bu proses tarafından yapılan registry ve dosya işlemlerinin daha dikkatli bir şekilde incelenmesi önem arzetmektedir.

Bir diğer dikkat çeken husus ana proses tarafından oluşturulan "cmd.exe" prosesidir. Bu prosesin de "conhost.exe" prosesini oluşturması ana proses tarafından bir scriptin çalıştırıldığına işaret olabilmektedir. Yine benzer şekilde "regsvr32.exe" prosesinin oluşması bir dll dosyasının register edilmeye çalışıldığını göstermektedir. Tüm proses oluşturulma olayları Şekil A.1'de gösterilmiştir.

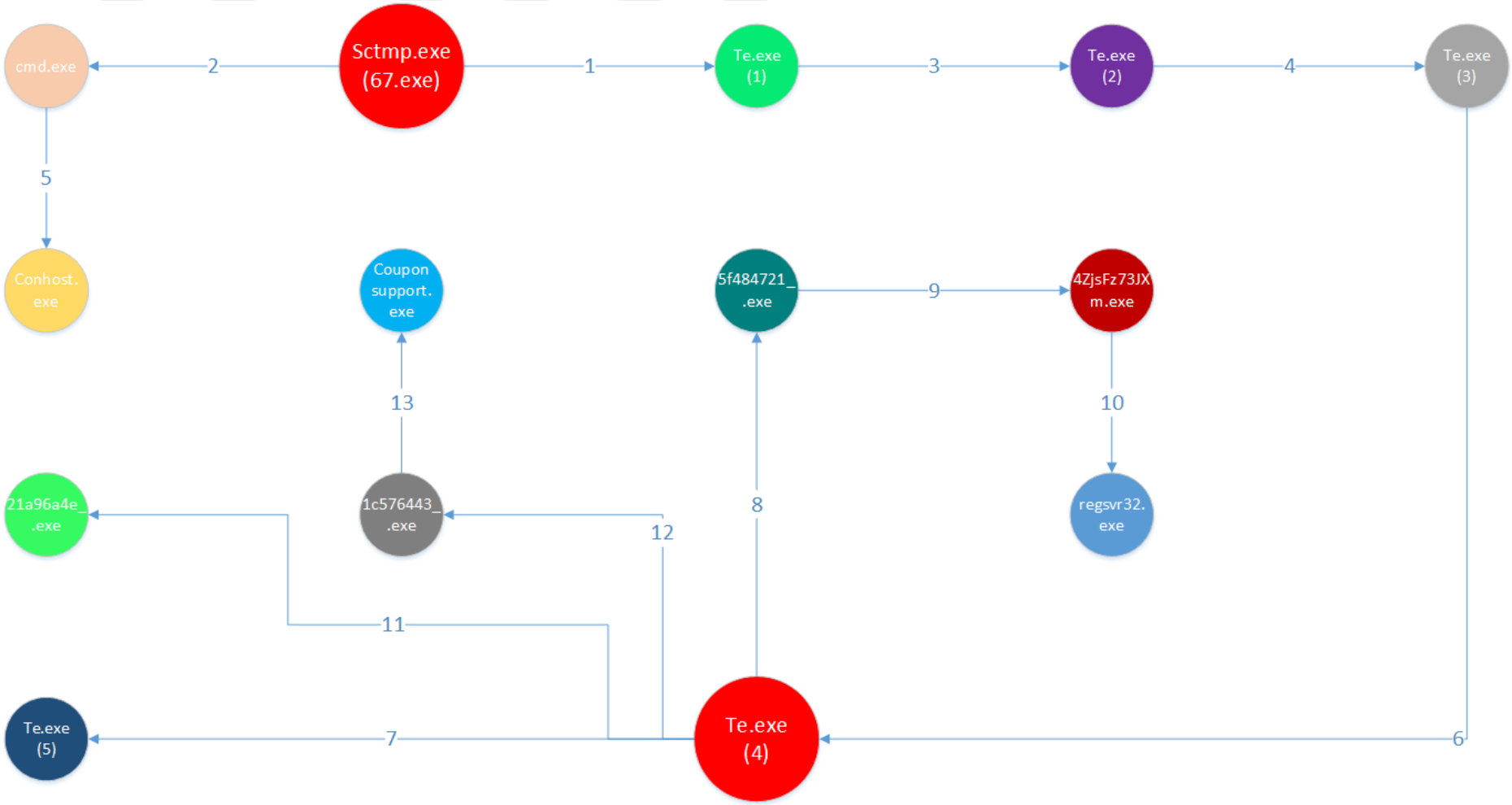
Proses sonlanma olaylarında da dikkat çeken noktalar bulunmaktadır. Ana proses drop ettiği "Te.exe" dosyasını çalıştırdıktan sonra sonlanmaktadır. Buradan ana prosesin görevinin işlemleri gerçekleştirecek olan asıl prosesi oluşturması olduğu anlaşılmaktadır. Tüm proses oluşturulma olayları Şekil A.2'de gösterilmiştir.

5.7 Registry Aktiviteleri

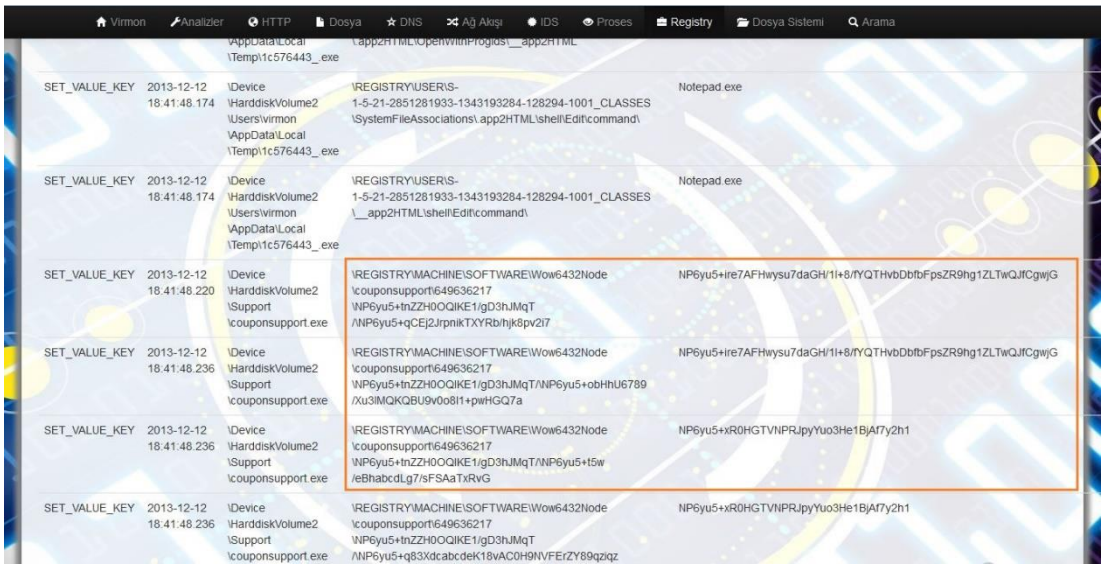
Analiz edilen dosyanın ve alt proseslerin çok fazla sayıda registry işlemi yaptıkları gözlenmiştir. Registry anahtarı okuma işlemlerine göz atıldığında hangi zamanlarda internet bağlantısı kurulduğu anlaşılabilmektedir. Registry işlemlerinde dikkat çeken bir durum anahtar yollarının ve değerlerinin şifrelenmiş olarak oluşturulduğudur. Örnek registry olayları Şekil 5.10'da gösterilmiştir. Beklendiği halde zararlı uygulama, kendisini bilgisayar yeniden başlarken otomatik çalıştırılması için herhangi bir registry işlemi yapmamıştır.

5.8 Dosya Sistemi Aktiviteleri

Analiz edilen zararlı yazılım ve alt prosesler registry gibi dosya sistemi üzerinde çok sayıda çalıştırılabilir dosya, dinamik kütüphaneler ve scriptler oluşturmuş ve silmişlerdir. Dosya işlemlerinde en dikkat çeken husus şüphesiz ki incelenen dosyanın çalıştıktan bir süre sonra silinmesidir. Bu durum Şekil 5.11'de gösterilmiştir.



Şekil 5.9 : Prosesler arası ilişki.

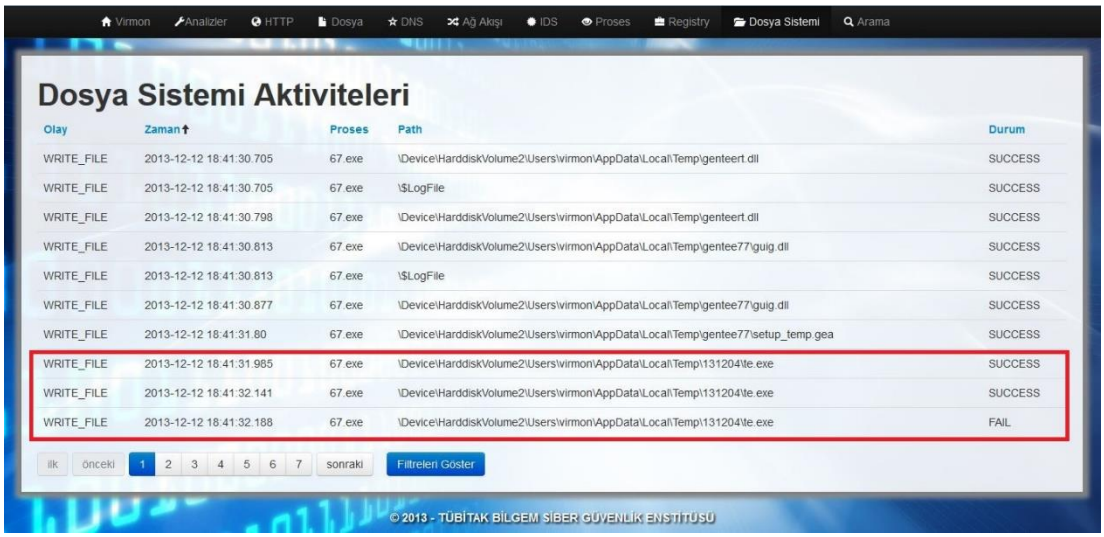


Şekil 5.10 : Şifrelenmiş registry değerleri.

Şekil 5.11 : Analiz edilen yazılımın silinmesi işlemi.

Bir diğer önemli husus çalıştırılan proseslerin ait oldukları dosyaların oluşturulma işlemleridir. En çok aktivite gösteren proseslerden biri olan “Te.exe” dosyası çalıştırılmadan önce ana proses tarafından oluşturulmuştur. Yazma işlemi başlama ve sonlanma zamanı kontrol edildiğinde bu dosyanın aslında internetten indirilen bir diğer zararlı yazılım (tmps.i01) olduğu görülmektedir. Bahsedilen durum Şekil 5.12’de gösterilmiştir.

Görüldüğü üzere analiz edilen yazılım çok fazla sayıda şüpheli aktivite gerçekleştirmiştir. Virmon zararlı yazılım analiz sistemi farklı bileşenleri ile bu aktiviteleri başarılı bir şekilde kayıt altına alabilmiştir.



Şekil 5.12 : İndirilen dosyanın ana proses tarafından dosyaya yazılması.

6. GELECEK ÇALIŞMALAR VE SONUÇ

Siber güvenliği tehdit eden en büyük faktörlerden biri olan zararlı yazılımların mümkün olduğunca erken tespit edilebilmesi hem son kullanıcılar hem de kurumlar açısından büyük önem arz etmektedir. Hali hazırda kullanılan anti-virüs gibi güvenlik çözümlerinin hedef gözeterek kişiye ya da bir kuruma özel geliştirilmiş zararlı yazılımları tespitte zayıf kalması da büyük bir sorun teşkil etmektedir. Bu sebeple kamu ve özel sektördeki büyük kuruluşların kendilerine özel tehdit tespit sistemlerine sahip olmaları gerekmektedir.

Gerçek zamanlı zararlı yazılım tespit sisteminin bir parçası olarak tasarlanıp geliştirilen Virmon dinamik zararlı yazılım analiz sistemi etkili bir şekilde çalışmakla birlikte zayıf kaldığı önemli noktalar da bulunmaktadır. Virmon, şu an yalnızca komut satırından parametre almadan çalışan çalıştırılabilir (executable) dosyaların analizini gerçekleştirebilmektedir. Parametre alarak çalışan ve grafik arayüzüne sahip zararlı yazılımların analizi için yeni çalışmalar yapılmalı ve sisteme entegre edilmelidir. Bu tip uygulamaların bir kullanıcı aktivitesi gerektirmesi, analiz işleminin otomatik olarak gerçekleştirilmesinin önündeki en büyük engellerden biri olarak görülmektedir. Sistemin daha aktif bir şekilde yürüyebilmesi için çalıştırılabilir dosyalar dışındaki office, pdf gibi son kullanıcılar tarafından sıklıkla kullanılan dosya tiplerinin de analizini gerçekleştirebilecek bir yapıya sahip olması gerekmektedir.

Zararlı yazılım analiz sistemlerinin popüler hale gelmesiyle birlikte, zararlı yazılım geliştiricileri de uygulamalarının otomatik sistemler tarafından çalıştırılabilmesinin önüne geçmek için çeşitli yollara başvurmaktadırlar. Örnek olarak sanal makinelerin tespiti verilebilir. Bazı zararlı yazılımlar yalnızca fiziksel sistemler üzerinde çalışmayı tercih etmekte, sanal sistemler üzerinde çalıştırıldıklarında davranış değiştirmektedirler. Sanal makine tespiti dışında bazı zararlı yazılımlar, çalıştırıldıkları ortamlar üzerinde fare veya klavye tıklaması gibi herhangi bir kullanıcı aktivitesi olup olmadığını kontrol etmektedirler. Bu sebeple, ortam tespiti

yapmaya çalışan zararlı yazılımların da toplanan aktivitelerden tespit edilebilmesi, Virmon'a kazandırılabilecek yeni yetenekler arasında yer almaktadır.

Sistem gerçeklemesi anlatılırken NDIS filtreleme sürücülerinin geliştirilmesinin zorluğundan bahsedilmişti. Virmon analiz sisteminin taşınabilir hale getirilmesi için yapılabilecek bir diğer çalışma ağ aktivitelerini toplayabilecek bir NDIS filtreleme sürücüsü geliştirilmesidir.

Virmon, bir izleme sistemi olarak tasarlanmış ve geliştirilmiştir. Virmon'un bir veri toplama sisteminin de ötesine taşınarak bir tespit sistemi haline getirilmesi fikri önümüzdeki çalışmaların temelini oluşturmaktadır. Analiz işlemleri ile elde edilen büyük verinin, geliştirilecek dağıtık makine öğrenmesi algoritmaları tarafından kullanılmasını sağlayan gerçek zamanlı bir zararlı yazılım tespit sisteminin geliştirilmesi en büyük hedefimizdir.

KAYNAKLAR

[1] **URL-1** <http://www.vupen.com/english/company.php>, alındığı tarih: 02.12.2013.

[2] **Farwell, P. J., ve Rohozinski, R., (**2011). Stuxnet and the Future of Cyber War, *Survival: Global Politics and Strategy*, vol. 53, no. 1, pp. 23-40.

[3] **Maslennikov, D., ve Namestnikov, Y., (**2012). Kaspersky Security Bulletin 2012: The overall statistics for 2012, Alındığı tarih: 03.12.2013, adres: http://www.securelist.com/en/analysis/204792255/Kaspersky_Security_Bulletin_2012_The_overall_statistics_for_2012.

[4] **Tirli, H., Pektas, A., Falcone, Y., Erdogan, N.**, (2013). Virmon: A Virtualization-Based Automated Dynamic Malware Analysis System. *Proceedings of the 6th International Information Security and Cryptology Conference*, pp. 59-64 Ankara, Turkey, September 20-21.

[5] **Skoudis, E. ve Zeltser, L.,** (2003). Malware: Fighting Malicious Code. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[6] **Porras, P., Saidi, H., Yegneswaran, V.,** (2009). Conficker C Analysis. Technical report, Alındığı tarih: 12.09.2013, adres: http://mtc.sri.com/Conficker/addendumC/index.html.

[7] **URL-2** <http://www.microsoft.com/tr-tr/security/pc-security/conficker.aspx>, alındığı tarih: 12.09.2013.

[8] **Cova, M., Kruegel, C., Vigna, G.,** (2010). Detection and analysis of drive-by-download attacks and malicious JavaScript code. *Proceedings of the 19th international conference on World wide web*. pp. 281-290 USA.

[9] **Pamuk, O., Şişeci, N., E.,** (2012). Fatura Zararlı Yazılımı (FatMal). Alındığı tarih: 05.12.2013, adres: http://www.bilgiguvenligi.gov.tr/zararli-yazilimlar/fatura-zararli-yazilimi-fatmal.html.

[10] **Sikorski, M., Honig, A.,** (2012) Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. ISBN-10: 1-59327-290-1, ISBN-13: 978-1-59327-290-6, San Francisco, CA 94103.

[11] **Egele, M., Scholte, T., Kirda, E., Kruegel, C.,** (2012). A Survey on Automated Dynamic Malware-Analysis Techniques and Tools. ACM Computing Surveys, Vol. 44, No. 2, Article 6.

[12] **Moser, A., Kruegel, C., Kirda, E.,** (2007). Exploring Multiple Execution Paths for Malware Analysis. In: IEEE Symposium on Security and Privacy.

[13] **Ivanov, I.,** (2012). API Hooking Revealed. The Code Project, Alındığı tarih: 09.10.2013, adres: http://www.codeproject.com/system/hooksys.asp.

[14] **Hunt, G., C., ve Brubacher, D.,** (1999). Detours: Binary Interception of Win32 Functions, *Proceedings of the 3rd Usenix Windows NT Symposium*, pp.135–143, USA.

[15] **Wang, Z., Jiang, X., Cui, W., Wang, X.,** (2008). Countering persistent kernel rootkits through systematic hook discovery, *Proceedings of the Recent Advances in Intrusion Detection (RAID)*.

[16] **URL-3** <http://msdn.microsoft.com/enus/library/windows/hardware/gg48735 3.aspx>, alındığı tarih: 12.09.2013.

[17] **Chen, X., Andersen, J., Mao, Z., M., Bailey, M., Nazario, J.,** (2008). Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. *International Conference on Dependable Systems and Networks*, Anchorage, AK.

[18] **Bayer, U., Kruegel, C., Kirda, E.,** (2006). TTAnalyze: A Tool for Analyzing Malware. *Proceedings of the 15th European Institute for Computer Antivirus Research (EICAR) Annual Conference.*

[19] **URL-4** <http://anubis.iseclab.org>, alındığı tarih: 12.06.2013.

[20] **Willems, C., Holz, T., Freiling, F.,** (2007). CWSandbox: Towards automated dynamic binary analysis, IEEE Security and Privacy, 5(2).

[21] **URL-5** <http://cuckoosandbox.org>, alındığı tarih: 15.06.2013.

[22] **URL-6** <http://msdn.microsoft.com/en-us/library/windows/hardware/ff540718 (v=vs.85).aspx>, alındığı tarih: 16.06.2013.

[23] **URL-7 <**https://www.virtualbox.org/wiki/Documentation>, alındığı tarih: 05.12.2013.

[24] **URL-8 <**http://sourceforge.net/projects/phpvirtualbox>, alındığı tarih: 07.12.2013.

[25] **URL-9** <http://msdn.microsoft.com/en-us/library/windows/hardware/ff557282 (v=vs.85).aspx> alındığı tarih: 07.12.2013.

[26] **Ligh, M., H., Adair, S., Hartstein, B., Richard, M.,** (2011). Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code. Indianapolis, USA: Wiley Publishing, Inc, pp. 596 - 599.

[27] **Butler, J., ve Kendal, K.,** (2008). Blackout: What Really Happened. Black Hat Conference, USA.

[28] **URL-9** <http://msdn.microsoft.com/en-us/library/windows/hardware/ff559953 (v=vs.85).aspx>, alındığı tarih: 11.03.2013.

[29] **URL-10** <http://msdn.microsoft.com/en-us/library/windows/hardware/ff559954 (v=vs.85).aspx>, alındığı tarih: 15.04.2013.

[30] **URL-11** <http://windows.microsoft.com/en-id/windows-vista/what-is-the-registry>, alındığı tarih: 21.06.2013.

[31] **URL-12** <http://msdn.microsoft.com/en-us/library/windows/hardware/ff541921 (v=vs.85).aspx>, > alındığı tarih: 22.06.2013.

[32] **URL-13** <http://msdn.microsoft.com/en-us/library/windows/hardware/ff550706(v=vs.85).aspx>, alındığı tarih: 25.06.2013.

[33] **URL-14** <http://msdn.microsoft.com/en-us/library/windows/hardware/ff551109(v=vs.85).aspx> alındığı tarih: 25.06.2013.

[34] **Şiseci, N., E., Emre, B., Tirli, H.,** (2013). Case Study: Malicious Activity in the Turkish Network, Alındığı tarih: 21.06.2013. adres: http://www.syssecproject.eu/media/page-media/3/syssec-d5.3-TurkishNetworkCaseStudy.pdf.

[35] **Nazario, J., ve Holz, T.,** (2008).As the Net Churns: Fast-Flux Botnet Observations, *Proceedings of 3rd International Conference on Malicious and Unwanted Software*, pp. 24-31.

[36] **URL-15** <http://www.microsoft.com/security/sir/story/default.aspx#!botnetsection_irc> alındığı tarih: 04.12.2013.

[37] **URL-16** <http://blogs.mcafee.com/mcafee-labs/periodic-links-to-controlserver-offer-new-way-to-detect-botnets> alındığı tarih: 04.12.2013.

[38] **URL-17** <http://suricata-ids.org> alındığı tarih: 06.12.2013.

[39] **Claise, B.,** (2004). Cisco Systems NetFlow Services Export Version 9. RFC3954, Internet Engineering Task Force.

[40] **Slee, M., Agarwal, A., Kwiatkowski, M.,** Thrift: Scalable Cross-Language Services Implementation, Facebook.

[41] **URL-19** <http://framework.zend.com/manual/1.12/en/learning.quickstart.intro.html> alındığı tarih: 08.12.2013.

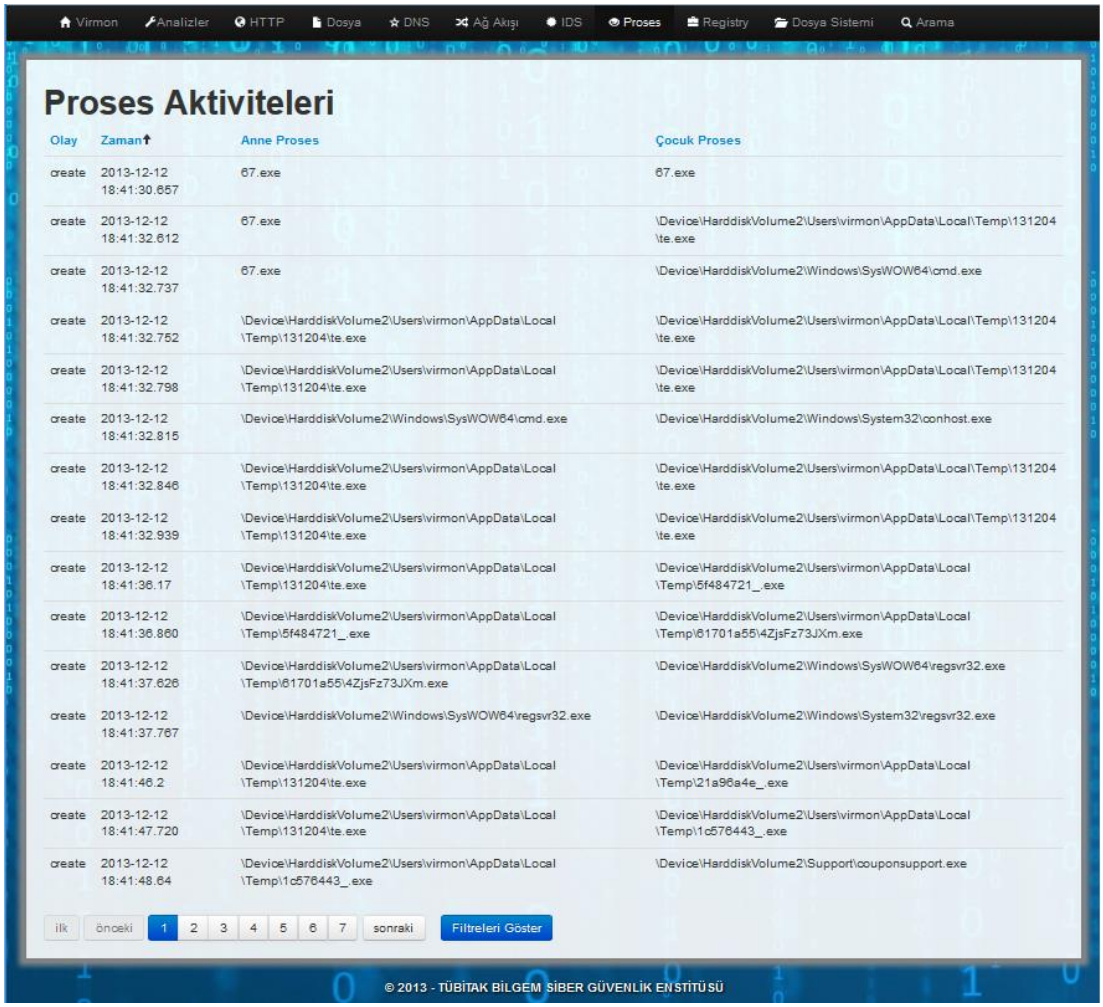
[42] **URL-20** <https://www.virustotal.com/en/file/2a76406d6eea9a77a72dc10cb7e3bdf470dc6157aae372e3d56b35441eceaf2f/analysis/> alındığı tarih: 12.12.2013.

EKLER

EK A.1 : Oluşturulan Prosesler

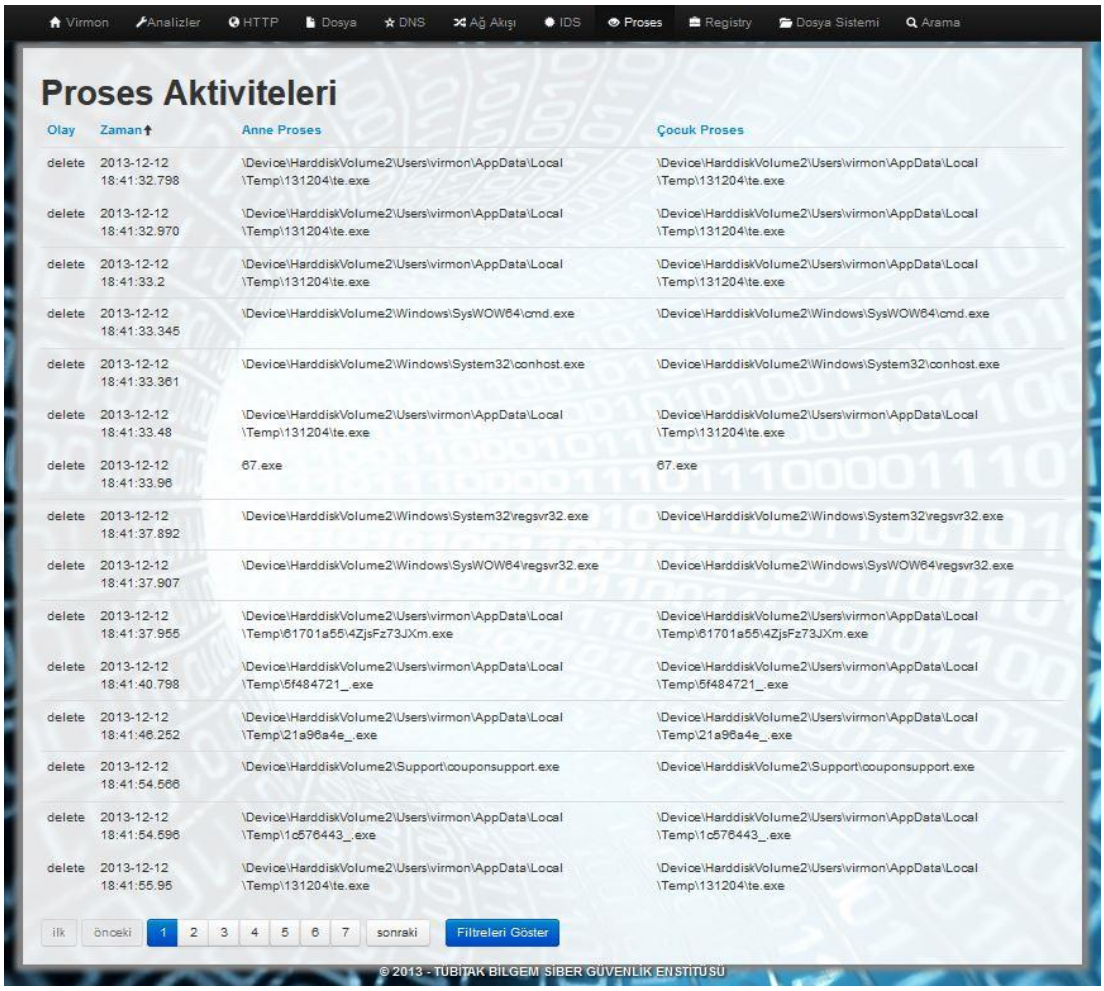
EK A.2 : Sonlanan Prosesler

EK A.1



Şekil A.1 : Oluşturulan prosesler.

EK A.2



Şekil A.2 : Sonlanan prosesler.

ÖZGEÇMİŞ

Ad Soyad: Hüseyin Tirli

E-Posta: tirli.huseyin@gmail.com

Lisans: İstanbul Teknik Üniversitesi, Bilgisayar Mühendisliği

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- **Tirli, H., Pektas, A., Falcone, Y., Erdogan, N. (2013)** Virmon: A Virtualization-Based Automated Dynamic Malware Analysis System. 6th International Information Security and Cryptology Conference. Ankara, Turkey.