



T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ



YÜKSEK LİSANS TEZİ

WINDOWS SİSTEM İŞLEMLERİ ÜZERİNDE BELLEK ANALİZİ
ile ZARARLI YAZILIM TESPİTİ

Mustafa ÇETİNKAYA

DANIŞMAN
Dr. Öğr. Üyesi Özgür Can TURNA

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

İSTANBUL-2022

Bu çalışma, 07.01.2022 tarihinde ařağıdaki jüri tarafından Bilgisayar Mühendisliğı Anabilim Dalı, Bilgisayar Mühendisliğı Programında Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Jürisi

Dr. Öğr. Üyesi Özgür Can TURNA (Danışman)
İstanbul Üniversitesi-Cerrahpařa
Mühendislik Fakültesi

Doç. Dr. Muhammed Ali AYDIN
İstanbul Üniversitesi-Cerrahpařa
Mühendislik Fakültesi

Dr. Öğr. Üyesi Ali BOYACI
İstanbul Ticaret Üniversitesi
Mühendislik Fakültesi



20.04.2016 tarihli Resmî Gazete’de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, İstanbul Üniversitesi-Cerrahpaşa’nın aboneli olduğu intihal yazılım programı kullanılarak Lisansüstü Eğitim Enstitüsü’nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

ÖNSÖZ

Tez çalışmam boyunca değerli bilgi, birikim ve tecrübesiyle destek olan danışman hocam sayın Dr. Öğr. Üyesi Özgür Can TURNA'ya sonsuz teşekkür ve saygılarımı sunuyorum.

Adli Bilişim alanında bana çalışma fırsatı sunan Adli Tıp Kurumuna ve Adli Bilişim İhtisas Dairesi Başkanı Uz. Dr. Timur Kaan GÜNDÜZ'e, araştırma sorusunu belirleme sürecinde değerli fikirlerini benimle paylaşan Emre TINAZTEPE'ye, tezin hazırlanması sürecinde Özlem MUTLU ve Yunus Emre UZUN başta olmak üzere destek olan arkadaşlarıma teşekkürlerimi sunuyorum.

Çalışmamı; yaklaşık iki yıl önce kaybettiğim babama, hayattaki en büyük destekçilerim olan annem ve ablama, eğitim hayatım boyunca maddi desteğini asla esirgemeyen abime ithaf ediyorum.

Bilime bir katkı olması ümidiyle...

Ocak 2022

Mustafa ÇETİNKAYA

İÇİNDEKİLER

Sayfa No

ÖNSÖZ	iv
İÇİNDEKİLER.....	v
ŞEKİL LİSTESİ	vii
TABLO LİSTESİ.....	viii
SİMGE VE KISALTMA LİSTESİ	ix
ÖZET	x
SUMMARY	xii
1. GİRİŞ	1
1.1. ZARARLI YAZILIM.....	2
1.2. NEDEN WINDOWS 10?.....	2
1.3. ARAŞTIRMANIN AMACI VE BÖLÜMLER	3
2. GENEL KISIMLAR.....	5
2.1. LİTERATÜR İNCELEMESİ.....	5
2.2. İŞLEM (PROCESS).....	11
2.2.1. Minimal İşlem.....	12
2.2.2. Korunan İşlem	12
2.3. İŞLEMCİ ERİŞİM MODLARI.....	13
2.3.1. Kullanıcı Modu.....	13
2.3.2. Çekirdek Modu	14
2.4. WINDOWS SİSTEM İŞLEMLERİ.....	14
2.4.1. Idle	14
2.4.2. System	15
2.4.3. Registry.....	16
2.4.4. Memory Compression	16
2.4.5. Session Manager (smss.exe).....	16
2.4.6. Client/Server Runtime Process (csrss.exe).....	18
2.4.7. Windows Initialization Process (wininit.exe).....	18
2.4.8. Service Control Manager (services.exe).....	19

2.4.9. Host Process for Windows Services (svchost.exe).....	20
2.4.10. Local Security Authority Subsystem Service (lsass.exe)	20
2.4.11. Windows Logon Application (winlogon.exe)	21
2.4.12. Windows Logon User Interface Host (logonui.exe).....	21
2.4.13. Userinit Logon Application (userinit.exe).....	22
2.4.14. Windows Explorer (explorer.exe)	22
3. MALZEME VE YÖNTEM.....	23
3.1. VMWARE WORKSTATION PRO	23
3.2. PROCESS HACKER.....	23
3.3. PROCESS MONITOR.....	24
3.4. VOLATILITY FRAMEWORK.....	24
3.5. SOREBRECT.....	27
3.5.1. Sorebrect Uygulama	29
3.5.2. Sorebrect Analizi	29
3.6. XMRIG	32
3.6.1. Xmrig Uygulama	33
3.6.2. Xmrig Analizi	33
4. BULGULAR.....	37
4.1. WINDOWS SİSTEM İŞLEMLERİ	37
4.2. SOREBRECT.....	46
4.3. XMRIG	46
4.4. VOLATILITY FRAMEWORK.....	46
5. TARTIŞMA VE SONUÇ	47
KAYNAKLAR.....	49
EKLER	52
Özgeçmiş.....	53

ŞEKİL LİSTESİ

	Sayfa No
Şekil 2.1: Program, İşlem ve İş parçacıkları arasındaki ilişki.....	11
Şekil 2.2: Kullanıcı Modu ve Çekirdek Modu Bileşenleri.....	13
Şekil 3.1: Volatility 2’de imageinfo ve pslist eklentisinin kullanımı	26
Şekil 3.2: Volatility 3’te pslist eklentisinin kullanımı	27
Şekil 3.3: Sorebreect saldırı yöntemi	28
Şekil 3.4: Sorebreect zararlı yazılımı çalıştırıldıktan sonra	29
Şekil 3.5: SOREBREECT.vmem bellek görüntüsüne ait pstree eklenti sonuçları.....	30
Şekil 3.6: SOREBREECT.vmem bellek görüntüsüne ait cmdline eklenti sonuçları	30
Şekil 3.7: SOREBREECT.vmem bellek görüntüsüne ait malfind eklenti sonuçları.....	31
Şekil 3.8: SOREBREECT.vmem bellek görüntüsüne ait procdump, vaddump sonuçları	32
Şekil 3.9: XMRig işlem ağaç yapısı.....	33
Şekil 3.10: XMRIG.vmem bellek görüntüsüne ait pstree eklenti sonuçları	34
Şekil 3.11: XMRIG.vmem bellek görüntüsüne ait pstree eklenti sonuçları	34
Şekil 3.12: XMRIG.vmem bellek görüntüsüne ait cmdline eklenti sonuçları	35
Şekil 3.13: XMRIG.vmem bellek görüntüsüne ait filescan eklenti sonuçları	35
Şekil 3.14: XMRIG.vmem bellek görüntüsüne ait malfind eklenti sonuçları	36
Şekil 3.15: XMRIG.vmem bellek görüntüsüne ait procdump eklenti sonuçları.....	36
Şekil 4.1: Windows sistem işlemlerinin üst-alt ilişkisi ve başlatılma sırası	44

TABLO LİSTESİ

	Sayfa No
Tablo 3.1: Sorebrect zararlı yazılımına ait bilgiler	28
Tablo 3.2: XMRig zararlı yazılımına ait bilgiler	32
Tablo 4.1: Windows sistem işlemlerinin sahip olduğu öznitelikler	40
Tablo 4.2: Windows sistem işlemlerinin sahip olduğu öznitelikler (devam)	41
Tablo 4.3: Windows sistem işlemlerinin sahip olduğu öznitelikler (devam)	42
Tablo 4.4: Windows sistem işlemlerinin sahip olduğu öznitelikler (devam)	43

SİMGE VE KISALTMA LİSTESİ

Simgeler Açıklama

-

Kısaltmalar Açıklama

API	: Uygulama Programlama Arayüzü (Application Programming Interface)
CPU	: Merkezî İşlem Birimi (Central Process Unit)
DLL	: Dinamik Bağlantı Kitaplığı (Dynamic Link Library)
EFW	: Bilirkişi Formatı (Expert Witness Format)
FTP	: Dosya Aktarım İletişim Kuralı (File Transfer Protocol)
GNU	: Ulusal Birlik Hükümeti (Government of National Unity)
GPU	: Grafik İşlem Birimi (Graphics Process Unit)
HKCU	: HKEY_CURRENT_USER
HKLM	: HKEY_LOCAL_MACHINE
LiME	: Linux Bellek Çıkarıcı (Linux Memory Extractor)
MD5	: Mesaj Özeti Algoritması 5 (Message Digest Algorithm 5)
PID	: İşlem Kimlik Numarası (Process ID)
PPID	: Üst İşlem Kimlik Numarası (Parent Process ID)
RAM	: Rastgele Erişimli Bellek (Random Access Memory)
RDP	: Uzak Masaüstü Protokolü (Remote Desktop Protocol)
SAM	: Güvenlik Hesabı Yöneticisi (Security Account Manager)
SAS	: Güvenli Dikkat Sırası (Secure Attention Sequence)
SHA-1	: Güvenli Karma Algoritması 1 (Secure Hash Algorithm 1)
SHA-256	: Güvenli Karma Algoritması 2 (Secure Hash Algorithm 2)
TCB	: Güvenilir Bilgi İşlem Tabanı (Trusted Computing Base)
USB	: Evrensel Seri Veriyolu (Universal Serial Bus)
VTL	: Sanal Güven Seviyesi (Virtual Trust Level)
XMR	: Monero Kripto Parası

ÖZET

WINDOWS SİSTEM İŞLEMLERİ ÜZERİNDE BELLEK ANALİZİ İLE ZARARLI YAZILIM TESPİTİ

YÜKSEK LİSANS TEZİ

Mustafa ÇETİNKAYA

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Özgür Can TURNA

Bellek analizi; adli bilişim, zararlı yazılım ve izinsiz giriş gibi incelemelerin tamamlayıcı bir araştırma yöntemidir ve sistemdeki uçucu bellek üzerinde gerçekleştirilir. Sistem hakkında diğer analizlerden elde edilemeyen eşsiz bilgilere erişilmesini sağlar ve vakaların çözümünde kritik rol oynar. Bellek analizinin zararlı yazılım analizindeki önemi ise zararlı yazılımların kendisine verilen görevi yerine getirebilmek için çalışmak zorunda olmasıdır. Çünkü çalışan her komut bellek üzerinde bulunmak zorundadır. Zararlı yazılımlar her geçen gün saldırı tekniklerini geliştirmekte ve bazen uzun bir süre sonra tespit edilebilmektedir. Bu durum zararlı yazılımların verdiği hasarı artırmaktadır. Saldırı tekniklerini tespit etmek ise bazen samanlıkta iğne aramaya dönebilir. Bu yüzden bu araştırma zararlı yazılım tespitine bu tekniklerin karşıt açısından bakmayı, yani; sistemi daha iyi tanımayı, çalışan işlemleri ve gerçekleşen etkinlikleri daha iyi anlamayı, sistem hakkında farkındalık oluşturmayı amaçlamıştır.

Literatüre daha fazla katkı sağlamak amacıyla günümüzde en çok kullanılan işletim sistemi olan Windows 10 üzerinde yapılan araştırma; Idle, System, Registry, Memory Compression, Smss, Csrss, Wininit, Services, Svchost, Lsass, Winlogon, Logonui, Userinit ve Explorer olmak üzere 14 adet sistem işleminin normallerini belirlemeye ve zararlı yazılım analizinde normalden sapmaların nasıl tespit edilebileceğini göstermeye çalışmıştır. Araştırmada Sorebrect fidye

zararlı yazılımı ve XMRig kripto para madenciliği zararlı yazılımı günümüzün en gelişmiş bellek analizi aracı olan Volatility Framework ile incelenmiştir.

Ocak 2022, 53 sayfa.

Anahtar kelimeler: Adli Bilişim, Bellek Analizi, Volatility Yazılım Çerçevesi, Windows Sistem İşlemleri, Idle, System, Registry, Memory Compression, Smss, Csrss, Wininit, Services, Svchost, Lsass, Winlogon, Logonui, Userinit, Explorer, Zararlı Yazılım Analizi, Sorebrect, XMRig



SUMMARY

MALWARE DETECTION WITH MEMORY ANALYSIS ON WINDOWS SYSTEM PROCESSES

M.Sc. THESIS

Mustafa ÇETİNKAYA

Istanbul University-Cerrahpasa

Institute of Graduate Studies

Department of Computer Engineering

Supervisor: Assist. Prof. Dr. Özgür Can TURNA

Memory analysis is a complementary investigation method of investigations such as digital forensics, malware and intrusion and is carried out on volatile memory in the system. It provides access to unique information about the system that cannot be obtained from other analysis methods and plays a critical role in resolving cases. The importance of memory analysis in malware analysis is that malware has to run in order to fulfill its assigned task. Because every running command has to be on memory. Malware develops attack techniques every day and can sometimes be detected after a long time. This increases the damage done by malicious software. Detecting attack techniques can sometimes turn into looking for a needle in a haystack. Therefore, this research aims to look at malware detection from the opposite perspective of these techniques, namely; It aimed to get to know the system better, to better understand the running processes and events, and to raise awareness about the system.

The research was carried out on Windows 10, which is the most used operating system today, in order to contribute more to the literature; Idle tried to identify the normals of 14 system processes namely System, Registry, Memory Compression, Smss, Csrss, Wininit, Services, Svchost, Lsass, Winlogon, Logonui, Userinit, and Explorer showed how deviations from

normal can be detected in malware analysis. In the research, Sorebrect ransomware and XMRig cryptocurrency mining malware were analyzed with Volatility Framework, the most advanced memory analysis tool available today.

January 2022, 53 pages.

Keywords: Digital Forensics, Memory Analysis, Volatility Framework, Windows System Processes, Idle, System, Registry, Memory Compression, Smss, Csrss, Wininit, Services, Svchost, Lsass, Winlogon, Logonui, Userinit, Explorer, Malware Analysis, Sorebrect, XMRig



1. GİRİŞ

Hayatı kolaylaştıran teknolojik gelişmelerin ve uygulamaların artması sebebiyle elektronik cihazların kullanımı gün geçtikçe artmaktadır. Bu cihazların depolama birimine sahip olması kullanıcıları hakkında çeşitli bilgiler saklamasını sağlar. İnsanlar elektronik cihazlara kimlik bilgisi, adres, telefon numarası, fotoğraf, sosyal medya hesapları, e-posta, parola, banka hesap bilgisi gibi her türden veriyi saklamaya başlamış, dolayısıyla bilgilerin miktarı ve çeşitliliği artmıştır.

Elektronik cihazlarda kullanılan dijital depolama birimleri uçucu veya uçucu olmayan (kalıcı) bellek olarak sınıflandırılabilir. Uçucu bellekler, içerisinde bulunan veriyi depolamaya devam edebilmeleri için sürekli bir elektrik akımına ihtiyaç duyar. Bu bellek türünün en yaygın örneği Rastgele Erişimli Bellek'tir (RAM). Kalıcı bellekler ise içerdiği verileri sürekli bir elektrik akımına ihtiyaç duymadan da depolamaya devam edebilir. Hard diskler ve katı hal sürücüleri kalıcı bellek türlerine örnek olarak verilebilir (Anson, 2020).

İşletim sistemi çalıştıran bilgisayar gibi elektronik cihazlar hem uçucu hem de kalıcı belleklere sahiptir. Cihaz çalıştığı sürece depolama birimleri birbirleri ile veri alışverişi halindedir. Bu yüzden kalıcı bellekte bulunan veriler üzerinde işletim sistemi veya uygulama tarafından bir işlem gerçekleştirildiğinde uçucu bellekten de bu verilere veya veriler üzerinde çalışan işlemler hakkında bilgilere erişmek mümkün hale gelir. Uçucu bellek incelendiğinde işletim sisteminin çalışma zamanındaki durumu hakkında; hangi işlemlerin çalıştığı, çalışan işlemlerin bağlantı kurduğu klasör ve dosyalar, sistem bilgileri, aktif ağ bağlantıları, dosya sistemi kayıtları, en son çalıştırılan komutlar, yüklü kütüphaneler ve çekirdek sürücüleri gibi değerli bilgilere erişilebilir. Ayrıca kalıcı bellekteki bazı veriler anlamsız veya şifrelenmiş bir şekilde depolanıyorken aynı veriler uçucu bellek üzerinde açıkça erişilebilir, anlamlandırılabilir veya şifresiz bir şekilde bulunabilir. Hatta disk şifreleme anahtarları, enjekte edilen kod parçaları, kayıt dışı sohbet mesajları, şifrelenmemiş e-postalar ve ön belleğe alınamayan internet geçmiş kayıtları gibi kritik verilere sadece uçucu bellekten erişilebilir (Ligh ve diğerleri, 2014). Uçucu belleğin sistemde çalışan işlemlere ait kritik bilgileri içermesi elbette zararlı yazılımların tespit

edilmesinde ve sistem üzerinde yaptığı değişikliklerin bulunmasında bir fırsat olarak kullanılabilir. Bu sebeple uçucu bellek analizi zararlı yazılım analizlerinde önemli bir rol oynar.

1.1.ZARARLI YAZILIM

Zararlı yazılım, kötü amaçlı eylemler gerçekleştiren bir kod parçasıdır; çalıştırılabilir, komut dosyası, kod veya başka herhangi bir yazılım biçiminde olabilir. Saldırganlar, hassas bilgileri çalmak, sistemi gözetlemek veya sistemin kontrolünü ele geçirmek için zararlı yazılım kullanır ve kullanıcı izni olmadan e-posta, web veya Evrensel Seri Veriyolu (USB) sürücüler gibi çeşitli araçlar ile sisteme girer (Krombholz ve diğerleri, 2015).

Zararlı yazılım işlevlerine ve saldırı yöntemlerine göre sınıflandırılan geniş bir terimdir. Truva atı, virüs, solucan ve rootkit gibi farklı türleri mevcuttur. Zararlı yazılım sistemde gizlenebilmesi için saldırgan tarafından özel teknikler kullanılarak geliştirilir ve her ne kadar gizlense de kendisine verilen görevi yerine getirebilmesi için çalışmak zorundadır. Günümüz bilgisayarlarının çalışma mimarisine göre çalışan her komut ve işlenen her veri bellek üzerinde bulunmak zorundadır (Von Neumann, 1993). Bellek analizinin zararlı yazılım analizinde dahil olduğu yer ise tam olarak bu noktadır çünkü çalıştırılan her bir işlem, bilgisayar belleğinde (RAM) belirli **değişikliklere neden olur** ve değişiklikler işlem tarafından genellikle uzun bir süre korunur (Ligh ve diğerleri, 2014). Zararlı yazılım bir işlem olarak çalıştığı için bellekte birtakım izler bırakır ve bu izler bellek analiziyle tespit edilebilir. Bu durum bellek analizinin zararlı yazılım analizinde neden önemli olduğunu açıklar. Ayrıca bazı zararlı yazılımlar diske veri yazmadan doğrudan bellek üzerinde çalışır. Böyle bir durumda dosya sistemi incelemesinde bir sonuca ulaşılamaz ve başarısız olunur. Bellek analizi bu tür zararlı yazılımların tespit edilmesinde de son derece faydalıdır (Monnappa, 2018).

1.2.NEDEN WINDOWS 10?

Zararlı yazılımlar genellikle maddi getiri sağlayabilecek en çok tercih edilen sistemleri hedef alır. 2020 yılı içerisinde masaüstü işletim sistemlerinin dünya genelindeki kullanım oranlarına bakıldığında Windows'un **%76,84** ile ilk sırada olduğu görülmüştür (StatCounter, 2020a). Teknoloji meraklısı kullanıcıların çoğunlukla Windows dışındaki diğer işletim sistemlerini seçtiği düşünülürse, bu kitlenin küresel ortalama ile kıyaslandığında Windows kullanıcılarına

oranla küçük bir nüfusu oluşturduğu söylenebilir. Windows kullanıcıları çoğunlukla günlük görevlerini tamamlayacak kadar ortalama bilgisayar kullanımı bilgisine sahiptir. Teknoloji meraklısı olmadıkları için kolayca sosyal mühendislik saldırılarının kurbanı olurlar. Bu durum Windows sistemlerini zararlı yazılım geliştiricilerinin birincil hedefi haline getirmiştir. 2020 yılı içerisinde Windows işletim sistemlerinin kendi sürümleri arasındaki dünya geneli kullanım oranlarına bakıldığında ise Windows 10'un **%73,03** ile ilk sırada olduğu görülmüştür (StatCounter, 2020b). Bu sebeplerden dolayı araştırma **Windows 10** işletim sistemi üzerinde gerçekleştirilmiştir. Bellek analizi yazılımı olarak **Volatility Framework** yazılımı kullanılmış ve bu yazılımın desteklediği **Windows 10.0.19041** işletim sistemi sürümü tercih edilmiştir (Ligh, 2020).

1.3.ARAŞTIRMANIN AMACI VE BÖLÜMLER

Bu araştırma ile aşağıdaki bilgilere ulaşmak amaçlanmıştır:

- Yerli literatürde Bellek Analizi konusunda yeterli bir kaynak bulunmaması ve bu alandaki eksikliğin giderilmesi için güncel bir işletim sistemi ile (Windows 10) bir başlangıç yapılması.
- Windows sistem işlemlerinin farklı kaynaklarda farklı açılardan ele alınması sebebiyle dağınık olması. Bu bilgilerin tek bir kaynaktan toplanması ve bu işlemlerin özellikleri hakkında farkındalığın artırılması.
- Günümüzün en gelişmiş bellek analiz aracı olan Volatility Framework aracını tanıtmak ve yeni eklentiler ile yeteneklerinin artırılmasını sağlamak.

Bölüm 2'de, literatürde yer alan bellek analizi ve zararlı yazılım tespiti ile ilgili çalışmalar paylaşılmış, bellek analizinde kullanılan terimler hakkında bilgilendirme yapılmış, 14 adet Windows sistem işlemine; IDLE, SYSTEM, REGISTRY, Memory Compression, SMSS, CSRSS, WININIT, SERVICES, SVCHOST, LSASS, WINLOGON, LOGONUI, USERINIT ve EXPLORER ait bilgiler ayrıntılı bir şekilde verilmiştir.

Bölüm 3'te, araştırma sürecinde bellek ve zararlı yazılım analizinde kullanılan araçlar tanıtılmış, bu araçların kullanım yönteminden bahsedilmiş, Sorebrect ve XMRig isimli iki farklı zararlı yazılımın analizi bellek görüntüsü üzerinde gerçekleştirilmiştir.

Bölüm 4’te, Windows sistem işlemlerine ait kaynaklardan ve analizlerden elde edilen bilgiler, Sorebrect-XMRig zararlı yazılımlarının analizinden elde edilen bulgular ve Volatility Framework aracının olumlu-olumsuz özellikleri paylaşılmıştır.

Bölüm 5’te, araştırma boyunca elde edilen bulgular değerlendirilmiş ve gelecek zamanda gerçekleştirilecek arařtırmaların geliştirilmesi için önerilerde bulunulmuştur.



2. GENEL KISIMLAR

Bu bölümde bellek analiziyle ilgili literatürde yer alan araştırmalara geniş bir şekilde yer verilmiş; araştırmaların odaklandığı konular, cevapladığı araştırma soruları, kullanılan araçlar ve yöntemler, araştırma sonucunda elde edilen bulgular ve ortaya konulan araçlar gerektiği ölçüde ayrıntı verilerek özetlenmiştir. Ayrıca bu araştırmanın daha iyi anlaşılabilmesi için bellek analizinin önemli bileşenlerinden olan program, işlem (process), iş parçacığı (thread), minimal işlem, korunan işlem, işlemci erişim modları (kullanıcı modu ve çekirdek modu) anlatılmıştır.

Araştırmanın asıl odak noktası olan Windows sistem işlemlerinden: IDLE, SYSTEM, REGISTRY, Memory Compression, SMSS, CSRSS, WININIT, SERVICES, SVCHOST, LSASS, WINLOGON, LOGONUI, USERINIT ve EXPLORER görevleri, çalışma modları, amaçları ve özellikleriyle referans kaynaklardan alınan bilgiler ile İnternet kaynaklarında dağınık halde bulunan veriler bir araya getirilerek ayrıntılı bir şekilde sunulmuştur.

2.1.LİTERATÜR İNCELEMESİ

(Petroni ve diğerleri, 2006), bellek analizinin ilk yıllarında çok zorlu bir süreç olan veri çıkarma işlemi pratik hale dönüştüren ve genişletilebilen Forensic Analysis ToolKit (FATKit) adında bir araç sunmuştur. Araç C programlama dilinde geliştirilmiş ve hem Windows hem de Linux işletim sistemlerine ait bellek görüntülerini analiz edebilmiştir. Ayrıca analiz sonrası sonuçları görselleştirerek analistlerin daha önemli bulgulara odaklanmasını sağlamıştır.

(Ruff, 2008), bellek görüntüsü oluşturma tekniklerine, engelleme yöntemlerine ve ücretsiz bellek analiz yazılımlarına genel bir bakış açısı sunmuştur. Bellek analizi yönteminde geçmişte aşılabilir zorluklar Windows XP işletim sistemi üzerinde örnekler ile gösterilmiştir. Araştırma sonucunda bellek analizi konusunda yoğun araştırmalar yapılmasına rağmen hala erken bir dönemde olduğu, henüz eksiksiz bir bellek görüntüsü oluşturma aracının geliştirilemediği, ücretsiz olarak sunulan araçların sınırlı olduğu ve özelleştirme gerektirdiği belirtilmiştir. Ayrıca bellek analizinin göz ardı edilemeyecek kadar önemli ve diğer analizlerin bir tamamlayıcısı olduğu ifade edilmiştir.

(White, 2013), zararlı yazılımları tanımlayabilmek ve zararlı yazılımların analizinde daha gelişmiş teknikler üretebilmenin önünü açmak için bellekte yer alan kullanıcı işlemlerine ayrılmış alanın ayrıntılarını sunmuştur. Araştırma Volatility Framework ile Windows XP ve Windows 7 işletim sistemleri üzerinde yürütülmüştür ve kullanıcı bellek alanında yer alan işlemlerdeki bilinmeyen zararlı kodların tespit edilmesi için bir yöntem önermiştir. Yöntemin sunumunda kullanıcı bellek alanının ortak bileşenleri açıklanmış, bellekte yer alan kod ve verinin birbirinden ayrıştırılabilmesi için model oluşturulmuş, bilinen ve bilinmeyen kodları tespit edebilen bir yöntem geliştirilmiştir. Yöntem test edilen tüm zararlı yazılımları başarıyla tanımlamıştır.

(Baum, 2014), bellek görüntülerinden elde edilen verilerin analiz sürecini geliştirmek için bilgi görselleştirme araçlarının uygulanmasına odaklanmıştır. Elektronik cihaz depolama kapasitelerindeki artış, veri çeşitliliğinin ve karmaşıklığının artmasına, inceleme sürelerinin uzamasına ve bazen delillerin gözden kaçmasına neden olur. Mevcut inceleme araçlarından çoğunun sonuçları metin veya liste olarak sunması da buna eklendiğinde delilleri bulmaktaki zorluk katlanarak artar. Araştırma bu zorluğun üstesinden gelmek için bilgi görselleştirme araçlarının kullanıldığı bir yöntem önermiştir. Geliştirilen yöntem, zararlı yazılım bulaşmış bellek görüntüsünde çalışan işlemlerin, işlemlere ait socket ve port numaralarının temiz bellek görüntüsünden farklı olan bilgilerini grafiklerle görselleştirerek kolayca tespit edilmesini sağlamıştır.

(Pék, 2015), zararlı yazılımların bazı Windows sistem işlemlerindeki sıra dışı etkinliklerinden tespit edilmesini sağlayan Membrane adında bir araç sunmuştur. Aracın çalışma mantığı, bellekte çalışan işlemlerin sistem çağrılarını yüksek yetki ile duraksız bir şekilde izleyerek LSASS, SERVICES, WINLOGON ve EXPLORER gibi işlemlerde sıra dışı bir davranış tespit etmektir. Araç Windows XP'de %91, Windows 7'de %75 oranında başarılı sonuçlar üretmiştir. Araştırma ayrıca mevcut donanım sanallaştırma yöntemlerinin zararlı yazılımlar karşısında zayıf kaldığını gösterip, yeni bir donanım sanallaştırma modeli önermiştir.

(Prakash ve diğerleri, 2015), zararlı yazılımlar tarafından çekirdekte çalışan işlemlere müdahale edilebildiğini ve bu durumun bellek analizinin güvenilirliğini nasıl etkilediğini Windows gibi kapalı kaynak olan işletim sistemi üzerinde yapılan deneysel bir çalışmayla göstermiştir. Araştırmada bellek analizi araçlarının ürettiği sonuçlar yeteri kadar doğrulanmadığı belirtilmiş ve Volatility Framework'ünün bazı eklentileri incelenmiştir. Araştırma sonucunda bir saldırı

tekniki geliştirilmiş ve bu teknikle bazı sistem verilerinin fark edilmeden değiştirilebileceği ve hedef sistem üzerinde olumsuz etkilere neden olabileceği gösterilmiştir.

(Bolat, 2015), belleğin tanımı ve çeşitleri hakkında genel bilgiler vermiş ve bellek analiziyle erişilecek sosyal medya uygulaması, e-posta sağlayıcıları ve disk şifreleme yöntemlerine ait parolalar gibi çeşitli bilgilere nasıl erişilebileceğini uygulamalı örnekler ile göstermiştir. Windows, Linux ve MacOS gibi farklı işletim sistemine sahip bilgisayarların ve bu bilgisayarlarda bulunan belleğe ait anlık görüntülerin nasıl ve hangi yazılımlar ile oluşturulduğu gösterilmiştir. Bellekte bulunan verilerin günlük hayatımızı etkileyebilecek kadar önemli olduğunu göstermek için incelemeler günlük hayatta kullanılan kişisel bilgisayarlarda gerçekleştirilmiştir. Araştırmada çoğunlukla açık kaynaklı yazılım olmakla birlikte bazı ücretli yazılımlar tercih edilmiştir.

(Korkmaz, 2015), hedefli saldırılarda kullanılan zararlı yazılımları davranışsal ve bellek izleri kullanılarak makine öğrenimi yöntemleriyle tespit eden ve onları sınıflandıran bir yöntem sunmuştur. Literatürde daha önce yayınlanmış olan mevcut zararlı yazılım sınıflandırma özelliklerini kullanmış ve bellek üzerinde bıraktığı bazı izleri tanımlamış, sınıflandırma yönteminde kullanmıştır. Araştırmanın dikkat çeken özelliği hedefli zararlı yazılımların makine öğrenimi yöntemleri ile sınıflandırılmasında davranışsal özniteliklere bellek üzerindeki izlerin eklendiği ilk araştırma olmasıdır. Sınıflandırma yöntemleri denetimli öğrenme algoritmaları ile test edilmiş ve hedefli saldırılarda kullanılan zararlı yazılımların tespitinde %92 doğruluk payına ulaşmıştır.

(Murray, 2016), Windows 7 bellek görüntüsü üzerinde internet tarayıcısı, anlık mesajlaşma, Dosya Aktarım İletişim Kuralı (FTP) istemcisi ve belge düzenleyici olarak 4 kategoride çalışan bazı uygulamalara ait işlemlerin ön incelemesini yapan MemTri adında bir araç sunmuştur. MemTri, uygulamalara ait bilgileri bellek görüntüsünde çalışan işlemlerinden düzenli ifadeler ile filtreleyerek Bayes Ağını ve Volatility Framework yazılımını kullanarak incelemesini gerçekleştirir. MemTri, 60 bellek görüntüsü üzerinde test edilmiş ve uygulamalar çalışma anında kullanıldığında %95,7'lik bir başarı oranına sahipken, uygulamalar sonlandırıldıktan sonra ise tarama modu sayesinde %80'lik bir başarı oranı elde etmiştir. Araştırma işlemler sonlandırıldıktan sonra bile işlemlere ait kritik verilerin elde edilmesi açısından önem arz etmektedir.

(Zapata, 2016), zararlı yazılımları geleneksel davranışlarından tespit etmenin yetersiz olduğunu ve artık sistem etkinliği hakkında daha fazla farkındalığa sahip olunması gerektiğini savunmuştur. Araştırma zararlı yazılım bulaşan sistem işlemlerinin özelliklerini analiz etmiş ve zararlı yazılımın sistem işlemine bulaşması sırasında tutulan sistem kayıtlarının önemi ve ağ bağlantılarındaki sıra dışı davranışlar gibi çeşitli etkinliklerin takibine odaklanmıştır.

(Case ve Richard, 2017), adli bilişim incelemelerinin genel olarak bilgisayarlarda kullanılan depolama birimlerine, cep telefonlarına, dijital kameralara vb. diğer elektronik cihazlara odaklandığını ve bunun artık değişmesi gerektiğine dikkat çeken bir literatür değerlendirmesi yapmıştır. Değerlendirme son zamanlarda bellek görüntü analizini incelemelere dahil eden araçlar sayesinde basit metin aramaktan çıkıp işlem ve çekirdek veri yapılarının incelenmesine dönüşse de hala yapılacak çok araştırma alanının olduğunu vurgulamıştır. Değerlendirme bellek görüntüsü inceleme yöntemlerini mevcut gücünü arttırmak ve geçmiş araştırmaların yanı sıra sistem güvenliğini savunma teknolojilerinde ön planda tutulması amacıyla, bellek görüntüsü incelemesinde mevcut araçların çözüm bulamadığı boşluklara işaret etmiş ve henüz hiç araştırma yapılmamış veya çok fazla eksik bulunan Apple iOS, Chromebooks ve Nesnelerin İnterneti gibi cihazlarda da bellek görüntülerinin incelenmesi gerektiğini vurgulamıştır.

(Henry, 2017), Stuxnet zararlı yazılımının analizinde Volatility Framework yazılımına ait Malfind, Callbacks, Devicetree, Svcsan, Yarascan, Ldrmodules, Apihooks, Psxview, Threads ve Timers eklentilerinin karşılaştırmalı bir araştırmasını sunmuş ve zararlı yazılım analiz tekniklerini incelemiştir. Volatility Framework eklentilerinin zararlı yazılım analizinde etkili sonuçlar üretmesi bu aracın değerini göstermiştir. Araştırma zararlı yazılım analiz tekniklerinden statik, dinamik, zekâ veya bellek analizinin tek başına kullanılması durumunda ayrıntılı sonuçlar vermediğini ancak bu alanların birleştirilmesi ve sonuçları görselleştirilmesi durumunda en iyi sonuç elde edildiğini ortaya koymuştur.

(Rehman ve diğerleri, 2017), bellek görüntüsü inceleme araçlarından FireEye Redline, FireEye Memoryze, Volatility Framework, Rekall Furka Framework ve Magnet Internet Evidence Finder arasında en iyi aracı belirlemek için özelliklerine ve desteklediği platformlara ait özet bir anket araştırması sunmuştur. Anket bu araçları çalışan işlemler, işlemlerin kullandığı kütüphaneler, sistem sürücüleri, kayıt defteri verileri, olay günlükleri, internet aktiviteleri, servisler, ağ bilgileri ve zararlı yazılım bulaştığına dair göstergeler üzerinden değerlendirmiştir. Araştırma sonucunda araçların bazı özelliklerde birbirlerine üstünlükleri olsa da Volatility

Framework'ün diğer araçlara göre en iyi sonucu ürettiği ancak zararlı yazılım saldırılarını önlemek veya onları daha etkin bir şekilde tespit edebilmek için yeteneklerinin geliştirilmesi gerektiği vurgulanmıştır.

(Brendmo, 2017), 2015 yılından sonra Windows 10'da yapılan çok sayıda güncelleme ile iyileştirilen sistem güvenliğini incelemiş ve tam bir bellek görüntüsünün en iyi şekilde nasıl elde edileceğine dair farklı yöntemleri tartışarak, elde edilen bellek görüntülerini analiz edebilecek Secure Kernel Forensic Toolkit (SKFT) adında bir araç geliştirmiştir. Araştırma güncellemeler sonrasında bellek görüntüsü analizinde nelerin değiştiğini tartışmış ve geliştirilen araç ile bellek görüntüsünde bulunan potansiyel veriler hakkında faydalı bilgiler sunmuştur.

(Süzen, 2018), çekirdek modunda çalışan RAM İmaj Alma Yazılımı (RİMAY) ve Dosya Kazıma ve Analizi Yazılımı (DOKAY) adında iki adet araç sunmuştur. RİMAY ile pagefile.sys dahil belleğin tüm görüntüsünü anlık olarak almayı ve alınan bellek görüntüsü üzerinde DOKAY ile metin arama, dosya imzası tarama gibi veri kazıma teknikleri kullanılarak doc, xml, pdf dosyalarını kurtarmayı başarmıştır. Bellekte bulunan dosya boyutlarının artması (belleğin yoğun kullanılması) durumunda dosya kurtarma başarı oranının düştüğü tespit edilmiştir. RİMAY'ın bellek görüntüsü oluşturma işlemi sırasında, bellek üzerinde diğer tüm yazılımlardan daha küçük alanda çalışması (156 KB), yazılımın daha küçük değişiklik yapması açısından araştırmanın dikkat çeken bulgusu olmuştur.

(Ecemiş, 2018), araştırmasında yaygın olarak görülen ve önem arz eden 10 farklı türde dosya enjeksiyon zararlı yazılımını belirlemiş ve bu zararlıların bellekte çalışan işlemlere enjekte olma yöntemlerine detaylarıyla değinmiştir. Zararlı yazılımların analizinde statik analiz için 3, dinamik analiz için 9 adet araç kullanılmıştır. Elde edilen analiz verileri kullanılarak ortak davranışsal özellikler belirlenmiş ve dosya enjeksiyon zararlı yazılımlarının tespiti için statik ve dinamik analiz algoritmaları önerilmiştir.

(Sihwail ve diğerleri, 2018), anket araştırmasında imza tabanlı ve buluşsal tabanlı zararlı yazılım tespit etme yöntemlerinin günümüzde artan zararlı yazılım sayısına ve taktiklerine karşı dezavantajlarının da artması sebebiyle bellek tabanlı zararlı yazılım tespitinin umut verici olduğu vurgulanmıştır. Araştırma; zararlı yazılım türlerine ve onları tespit etme yöntemlerine genel bir bakış açısı sağlamayı, mevcut zararlı yazılım tespit etme yöntemlerini, bu yöntemlerin

bulgularını ve sınırlarını, zararlı yazılımın gizlenme ve saldırma taktiklerini, zararlı yazılım tespitinde bellek tabanlı analiz yöntemini sunmuştur. Ayrıca statik, dinamik ve melez analiz tekniklerinden bahsedilmiş, zararlı yazılım tespiti hakkında kapsamlı bir bilgi vermeyi ve zararlı yazılım tespitinde bellek tabanlı analizin önemini tartışmıştır.

(Østerud, 2018), Windows 10 ile gelen Memory Compression isimli yeni sistem işlemini incelemiş ve MemoryDecompression adında bir araç ile bu işlemin sıkıştırdığı bellek alanlarını tekrar açmak için yöntem sunmuştur. Araştırma literatürde Memory Compression işlemi üzerinde yapılan ilk incelemedir. Araştırma bellekte etkin olmayan alanların Memory Compression işlemi tarafından metin olarak sıkıştırıldığını göstermiş ve MemoryDecompression aracı Volatility Plugin Contest yarışmasında ikinciliğe layık görülmüştür.

(Yücel, 2019), zararlı yazılımların statik ve dinamik analizinde karşılaşılan paketleme, perdeleme, ölü kod ekleme ve sanal makinenin algılanması gibi engellerden dolayı analizlerin bellek üzerinden gerçekleştirildiği bir yöntem sunmuştur. Zararlı yazılımlar için bellek işlemleri ve bellek erişim örüntüleri incelenmiş, bellek erişim görüntülerinin çıkarılması için yeni bir yaklaşım geliştirmiştir. Yaklaşımda bu görüntülerin tespiti için örüntüler, karşılaştırma amacıyla 3 boyutlu görüntülere dönüştürülmüştür.

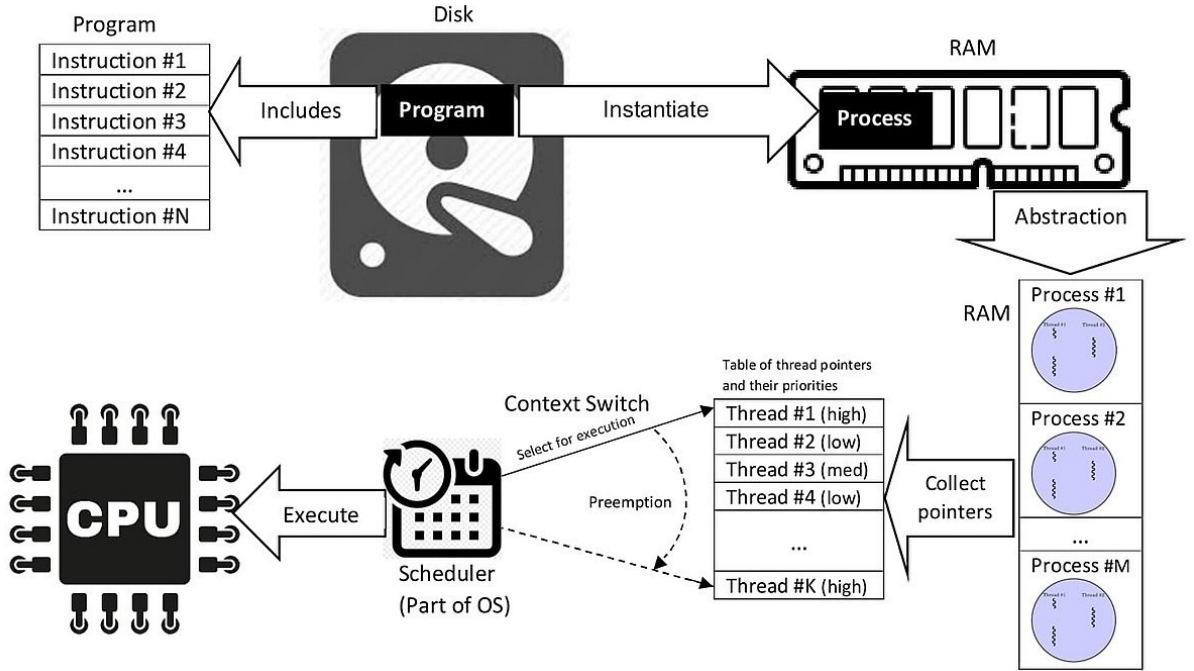
(Balogh ve Mojžiš, 2019), zararlı yazılımların davranışları üzerinden yapılan tespit işleminin küçük değişiklikler veya sıfırinci gün saldırılarıyla kolayca atlatılabildiğini, bu yüzden zararlı yazılımlardan elde edilmeyen sadece sistem özelliklerine bağlı kalarak ve sistemin işlevselliğinden ödün vermeden, kolay bir şekilde değiştirilemeyen sistem özelliklerine odaklanmanın daha iyi olacağını vurgulamıştır. Bu özelliklere sistem belleğinden elde edilen veriler de dahil edilmiştir. Araştırma zararlı yazılım bulaşan sistem işlemi ve bu işlemin özelliklerini belirlemiş, zararlı yazılım tespitinde bu özelliklerin kullanılabileceğini göstermiştir.

(Tahıllıoğlu, 2020), geçmiş yıllarda önerilen zararlı yazılım tespit yöntemlerinin artık yetersiz olduğunu vurgulamış ve çalıştırılabilir dosyaların bellekte birtakım izler bıraktığını, bu durumun zararlı yazılım davranışlarını incelemeye imkân tanıdığını düşünerek, zararlı yazılımları ve potansiyel sıfır gün saldırısını tespit etmeye yarayan yeni bir yöntem sunmuştur. Bu yöntemde zararlı yazılımların bellek üzerinde bıraktığı izler, bilgisayarlı görü tekniklerinden

faaydalanılarak farklı renk ve boyutlarda görselleştirilmiştir. Bilgisayarlı görü alanında sıklıkla kullanılan GIST ve HOG küresel tanımlayıcılarıyla resimler üzerinden özellik çıkarımı yapılmış ve vektörler elde edilmiştir. Bu vektörler XGBoost, J48, Destek Vektör Makinası, Sıralı Minimal Optimizasyon ve Rassal Orman makina öğrenmesi algoritmaları ile sınıflandırılmıştır. Yapılan çalışmanın sonuçlarına göre, en yüksek başarı oranı 4096 sabit genişlik görselleştirme yöntemi kullanıldığında %96,39 olarak elde edilmiştir.

2.2. İŞLEM (PROCESS)

Program ve işlem kavramları her ne kadar birbirlerine benzetilse de temelde farklıdır. Program, belirli bir görevi gerçekleştirmek için bilgisayar tarafından çalıştırılabilen komutlar bütünüdür. İşlemler ise programa ait komutların bir veya daha fazla iş parçacığı (thread) tarafından bellek üzerinde çalıştırılmasıdır yani programın bellek üzerinde çalıştırılan **bir örneğidir** (Silberschatz ve diğerleri, 2010). Her işlem en az bir adet iş parçacığına sahiptir. İş parçacığı olmayan bir işlem oluşturmanın doğrudan bir yolu olmadığından (böyle bir işlem zaten yararsız olacağından) bu sayı en az bir olmalıdır (Tanenbaum ve Bos, 2015). Program, işlem ve iş parçacıkları arasındaki ilişki Şekil 2.1’de gösterilmiştir.



Şekil 2.1: Program, İşlem ve İş parçacıkları arasındaki ilişki (Wikipedia, 2021a).

İş parçacığı bir işlem yapmadan önce gerekli nesnelere ile bağlantı kurması gerekir. İş parçacıklarının diğer nesnelere ile bağlantı kurmasına “**handle**” denir. İş parçacığı yeni nesnelere yaratarak, mevcut nesneyi adına göre çağırarak veya iş parçacıklarından devralarak yeni handle oluşturabilir. Bazı nesnelere sadece çekirdek modunda çalışan işlemler tarafından erişilebilir; işlemler, iş parçacıkları, erişim belirteçleri (access token), olaylar, zamanlayıcılar, oturumlar ve portlar. Bazılarıysa kullanıcı modunda çalışan işlemler tarafından da erişilebilen dosyalar, klasörler, paylaşılan bellek bölümleri ve kayıt defteri anahtarları gibi nesnelere (Fossen, 2012).

2.2.1. Minimal İşlem

Minimal işlemler kullanıcı modu adres alanına sahip değillerdir ve tüm iş parçacıkları minimaldir. Kendilerine ait çalıştırılabilir bir dosyası yoktur ancak yine de Merkezî İşlem Birimi (CPU) döngülerini tüketirler. Normal işlemler çalıştırılabilir dosyaya sahip olduklarından isimlerini de dosyadan alırlar. Minimal işlemler ise çalıştırılabilir bir dosyaya sahip olmadıklarından ortak isimleri de yoktur. Bu yüzden minimal işlemlere verilen isim, onları tanımlamak için kullanılan araçtan araca değişir. Örneğin Idle işlemini tlist aracı "System Process", Windows Görev Yöneticisi "Sistem Boşta İşlemi" olarak adlandırıyor; Process Explorer aracı bu işlemi "System Idle Process" olarak adlandırır.

2.2.2. Korunan İşlem

Windows bazı işlemlerini kritik olarak etiketler ve korur. Korunan işlemlere, yükseltilmiş ayrıcalıklara sahip olsa bile, kullanıcı modunda çalışan işlemler tarafından müdahale edilemez. Windows sistem işlemlerinin Güvenilir Bilgi İşlem Tabanı (TCB) olarak çalışması sistem güvenliği için önemlidir. Örneğin, Windows SMSS, CSRSS, WININIT ve SERVICES işlemlerini her zaman WinTcb-Lite olarak çalışacağını garanti eder. Çünkü bir işlem oluşturulurken doğru işlem koruma seviyesi belirtilmeden başlatılması mümkün değildir (Korkin, 2021). Ayrıca bu işlemler kritik olarak etiketlenmişlerdir, yani işlemin sonlandırılmayacağı anlamına gelir. Eğer işlem sonlandırılırsa işletim sistemi çalışmayı durdurur (çöker).

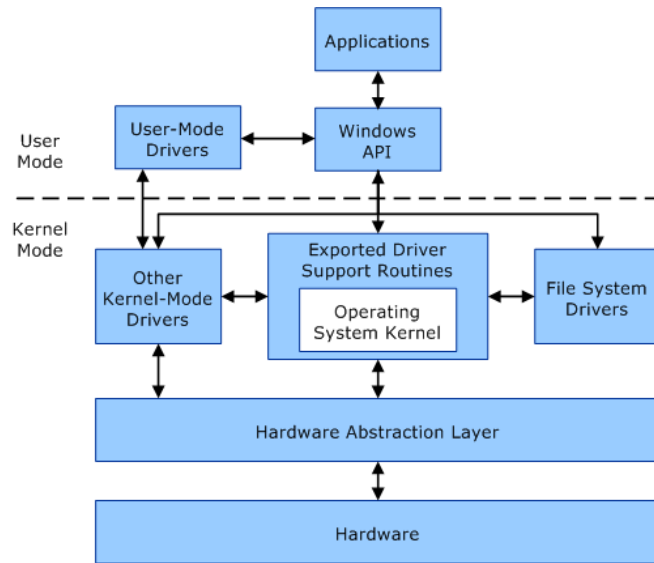
2.3. İŞLEMCI ERİŞİM MODLARI

Windows işletim sistemini çalıştıran bir bilgisayar kullanıcı uygulamalarının kritik işletim sistemi verilerine erişimini engellemek için iki farklı işlemci erişim moduna sahiptir: kullanıcı modu ve çekirdek modu. Kullanıcı uygulamalarının kodları kullanıcı modunda çalışırken işletim sistemi bileşenlerinin kodları (kritik Windows işlemleri, aygıt sürücüler vb.) ise çekirdek modunda çalışır. İşlemci çalıştıracığı kodları hangi moda ait olduğunu kontrol ederek sadece erişebileceği adres alanı içerisinde çalıştırır.

2.3.1. Kullanıcı Modu

Kullanıcı modu, uygulamaların sadece kendilerine tahsis edilen adres alanlarına erişebileceği bir yürütme modunu ifade eder. Bir uygulama başlatıldığında, Windows uygulama için bir işlem oluşturur ve işleme özel bir adres alanı tahsis eder. Kullanıcı modunda çalışan işlemlerin adres alanları kendisine özel olduğundan diğer uygulamalar tarafından **değiştirilemez** (Major, 2015). Her uygulama kendisine ayrılmış özel bir alan içerisinde çalışır ve bir uygulamanın çökmesi (işlemin yanıt vermemesi/çalışmayı durdurması) durumunda diğer uygulamalar veya işletim sistemi bu çökmeden etkilenmez.

Kullanıcı modunda çalışan uygulamalar servis çağrısı yaptıklarında çekirdek moduna geçer ve işlem tamamlandıktan sonra kullanıcı moduna tekrar döner. Bu şekilde, kritik işletim sistemi verileri korunmaya devam eder. Şekil 2.2 kullanıcı modu çekirdek modu bileşenleri arasındaki iletişimi gösterir.



Şekil 2.2: Kullanıcı Modu ve Çekirdek Modu Bileşenleri (Viviano, 2017).

2.3.2. Çekirdek Modu

Çekirdek modu, sistem belleğinin ve CPU talimatlarının tümüne erişim sağlayan bir yürütme modunu ifade eder ve çekirdek modunda çalışan tüm kodlar tek bir adres alanını paylaşır (Brendmo, 2017). Bir işletim sistemi bileşeni başlatıldığında, Windows bileşen için bir işlem oluşturur. İşlem tüm çekirdek modu uygulamaları tarafından kullanılan ortak adres alanında çalışmaya başlar ayrıca kullanıcı modu adres alanına erişim ayrıcalığına da sahiptir.

Çekirdek modunda çalışan işletim sistemi bileşenleri ve sürücüleri ortak bir adres alanını kullanmaları sebebiyle birbirlerinden izole **edilmemişlerdir**. Bu durum çekirdek modunda çalışan kodların yanlışlıkla (veya kasıtlı olarak) başka bir adrese yazmasına neden olabilir. Başka bir ifadeyle bu kodların sistem belleğinde istediği değişikliği yapabilme potansiyeline sahip olduğu anlamına gelir. Windows işletim sistemi kodunun büyük bir kısmı çekirdek modunda çalıştığından; çekirdek modunda çalışan bileşenlerin, sistem güvenliğini ihlal etmediğinden veya sistem kararsızlığına neden olmadığından dikkatlice tasarlanması ve test edilmesi çok önemlidir (Russinovich ve diğerleri, 2012a).

2.4. WINDOWS SİSTEM İŞLEMLERİ

Bir işletim sisteminin temel görevlerini yerine getirebilmesi için birtakım sistem işlemleri çalıştırması gerekir. Bu işlemlerin çoğu işletim sistemiyle birlikte başlatılır ve sistem sonlandırılana kadar çalışmaya devam eder. Başlıca görevleri sistemin kararlı, hızlı ve güvenli bir şekilde çalışmasını sağlamaktır.

Bu bölümde Windows işletim sisteminde çalışan sistem işlemleri ve bu işlemlere ait normallerin (üst işlem, alt işlem, çalıştığı oturum, ait olduğu kullanıcı, temel öncelik ve çalışma adedi vb.) neler olduğu başlıklar halinde ayrıntılı bir şekilde anlatılmıştır. Ayrıca bu normaller bir bütün halinde Tablo 4.1’de verilmiştir.

2.4.1. Idle

IDLE işleminin görevi CPU'nun boşta kalma süresini hesaplamaktır, bu hesaplamayı sistemde bulunan mantıksal her işlemci başına bir iş parçacığı çalıştırarak gerçekleştirir. Her işlemci için bir iş parçacığı bulunmasının nedeni işlemcilerin eş zamanlı olarak farklı iş parçacıklarını

çalıştırabilmesi veya bir işlemci iş parçacığı çalıştırırken diğer işlemcinin çalıştıracak bir iş parçacığı bulunmamasıdır.

IDLE iş parçacıkları, gerçekleştirilecek işi kontrol ederken sürekli bir döngü içinde çalışır (Fossen, 2012). İşlemcinin eğer çalıştıracak bir iş parçacığı kalmadıysa IDLE iş parçacığı o işlemciye gönderilir ve boşa kalma süresi hesaplanır. Döngü sırasında çalışmaya hazır başka bir iş parçacığı bulunması durumunda ise IDLE iş parçacığı sonlanır.

IDLE işleminin ve ilk iş parçacığının yeri statik olarak tahsis edilir, işlem yöneticisi ve nesne yöneticisi daha başlatılmadan sistemi önyüklemek için kullanılır. Bu yüzden IDLE işlemi ve iş parçacıkları nesne yöneticisinde bulunmazlar ve işlem kimlik numarası 0'dır (Yosifovich ve diğerleri, 2017).

IDLE işleminin ve iş parçacıklarının temel önceliği 0'dır. Aslında öncelik numarasının 0 olması bir önceliği olmadığını ifade eder çünkü yukarıda da bahsedildiği gibi bu iş parçacıkları sadece işlemcinin çalıştıracığı bir iş parçacığı kalmadığında gönderilir. Öncelikleri asla diğer iş parçacıkları ile karşılaştırılmaz ve hazır iş parçacığı kuyruğunda sıraya alınmazlar.

2.4.2. System

Çekirdek modunda çalışan iş parçacıklarının ve bunlar tarafından oluşturulan "handle"ların çoğu sistem işlemine aittir. SYSTEM işlemi, çekirdek modunda başlatılan tüm iş parçacıklarının varsayılan sahibidir (Ligh ve diğerleri, 2010). SYSTEM iş parçacıkları çekirdek modunda çalışmasına rağmen sıradan kullanıcı modu iş parçacıklarına ait donanım bağlantıları ve temel öncelik gibi tüm özniteliklerine sahiptirler. Tek fark bu iş parçacıklarının çekirdek modunda çalışmasıdır.

Windows ve çeşitli aygıt sürücülerini, sistem başlatma sırasında Giriş/Çıkış aygıtlarının hazırlanması ve beklenmesi veya bir aygıtın kontrol edilmesi gibi işlemleri gerçekleştirmek için SYSTEM iş parçacıkları oluşturur. Örneğin Bellek Yöneticisi işlemler tarafından aktif olarak kullanılmayan bellek sayfalarını sayfa dosyasına (pagefile.sys) yazmak veya aktifleşmesi halinde işlemleri sayfa dosyasından tekrar belleğe takas etmek gibi işlevler için SYSTEM iş parçacıklarını kullanır. Önbellek Yöneticisi Giriş/Çıkış'ları önceden okumak ve sonradan yazmak için SYSTEM iş parçacıklarını kullanır. Dosya sunucusu aygıt sürücüsü (Srv2.sys)

ağda paylaşılan disk bölümleri veya dosyalara yönelik Giriş/Çıkış isteklerine cevap vermek için SYSTEM iş parçacıklarını kullanır.

SYSTEM iş parçacıkları varsayılan olarak SYSTEM işlemine aittir ancak aygıt sürücülerini de herhangi bir işlemde SYSTEM iş parçacığı oluşturabilir. Örneğin Windows subsystem aygıt sürücüsü (Win32k.sys) kullanıcı modundaki verilere kolayca erişebilmek için CSRSS işleminin Canonical Display Driver (Cdd.dll) kütüphanesi ile bir SYSTEM iş parçacığı oluşturur.

Yukarıdaki sebeplerden dolayı bir analiz sırasında SYSTEM iş parçacıklarının birbirinden bağımsız olarak araştırılması, onların hangi sürücü tarafından oluşturulduğunun bilinmesi faydalı olacaktır. Böyle bir durumda ilk olarak çalışan SYSTEM iş parçacıkları belirlenir, sonra SYSTEM iş parçacığının hangi sürücüde çalışmaya başladığı bulunur. Bu en azından, iş parçacığının hangi sürücü tarafından oluşturulduğunu gösterir.

2.4.3. Registry

Registry (kayıt defteri), sistemin ön yüklenmesi ve yapılandırılması için gerekli bilgileri, işletim sisteminin yapılandırma ayarlarını depolayan hiyerarşik bir veri tabanıdır. Çekirdek bilgileri, aygıt sürücülerini ve servisler gibi temel seviye işletim sistemi bileşenlerinin yanı sıra sistemde çalışan diğer uygulamaların da ayarlarını içerir (Zhang ve diğerleri, 2011).

REGISTRY işleminin görevi, veri tabanında kayıtlı HKEY_LOCAL_MACHINE (HKLM) ve HKEY_CURRENT_USER (HKCU) gibi bölümleri bellekte depolayarak performansını artırmak ve bellek kullanımını azaltmaktır.

2.4.4. Memory Compression

Memory compression işleminin görevi işletim sisteminin performansını arttırmak ve yanıt süresini kısaltmak için kullanıcı modunda çalışan işlemlerin sıkıştırılan bellek sayfalarını, sayfa dosyasına (pagefile.sys) yazmak yerine kendisine tahsis edilmiş kullanıcı modu adres alanında saklamaktır.

2.4.5. Session Manager (smss.exe)

SMSS işlemi, Konsol veya Uzak Masaüstü Protokolü (RDP) aracılığıyla oturum açabilen çeşitli kullanıcılardan işletim sistemi servislerini izole eden oturumları oluşturmaktan sorumludur (Ligh ve diğerleri, 2014). İlk SMSS işlemi kendi örneğinden sistemde açılan her oturum için

(oturum sayacını 1 artırıp) alt işlemi olarak çoğaltır. Bu durumda tek bir kullanıcının oturum açtığı sistemlerde kendisine ait iki örneği daha alt işlem olarak başlatır. İlk örnek varsayılan olarak başlatılan oturum 0'da (servis oturumu) çalışır ve WININIT işleminin başlatılmasından yani dolaylı olarak işletim sistemi servislerinin çalıştırılmasından sorumluyken, ikinci örnek ise oturum 1'de (ilk kullanıcı oturumu) çalışır ve WINLOGON işleminin başlatılmasından sorumludur. İlk SMSS işlemi komut satırı parametresine sahip olmadan başlatılırken, örnek işlemler parametre olarak başlatılır.

İlk SMSS işleminin tek seferlik bazı başlatma adımları aşağıdaki gibidir (Yosifovich ve diğerleri, 2017):

- Kendisini ve ilk iş parçacığını kritik olarak etiketler. Kritik olarak etiketlenen bir işlem veya iş parçacığı herhangi bir sebeple sonlanırsa Windows çöker.
- Temel önceliğini 11'e yükseltir.
- HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management anahtarında saklanan sayfa dosyası (pagefile.sys) bilgilerini okur.
- HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDlls değer listesini okur ve kaydeder.
- HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment değer listesinde kayıtlı ortam değişkenlerini oluşturur.
- HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\BootExecute anahtarında bulunan disk denetimi gerçekleştiren AUTOCHK dosyasını çalıştırır.
- Kayıt defterinin geri kalanını (HKLM SOFTWARE, Güvenlik Hesabı Yöneticisi (SAM) ve SECURITY bölümlerini) başlatır.
- Oturum oluşturma isteklerine yanıt vermek için bir iş parçacığı oluşturur.
- Oturum 0'ı başlatmak için kendi örneğini oluşturur.
- Oturum 1'ı başlatmak için kendi örneğini oluşturur.

SMSS işleminin tüm örnekleri görevini tamamladıktan sonra sonlanır. Bu yüzden Windows işletim sisteminde sadece ilk SMSS işlemi çalışmaya devam eder. İşlem örneklerinin görevleri aşağıdaki gibidir:

- Her oturum için CSRSS işlemini başlatır.

- Varsayılan olarak servis oturumu için WININIT işlemini ve kullanıcı oturumları için WINLOGON işlemini başlatır.
- SMSS işlem örneği sonlanır ve başlattığı alt işlemler üst işlemsiz olarak çalışmaya devam eder.

2.4.6. Client/Server Runtime Process (csrss.exe)

CSRSS, çalıştığı oturumdaki her işlemin ve bunlara ait iş parçacığının oluşturulmasında rol oynayan önemli bir işlemdir. Bu nedenle, sistemde çalışan diğer işlemlerin (kendisi ve kendisinden önce başlayan üst işlemler hariç) çoğu için açık “handle”lara sahiptir (Monnappa, 2018). CSRSS işlemi temelde 4 adet kütüphane (basesrv.dll, winsrv.dll, sxssrv.dll, csrsrv.dll) yükler ve bu kütüphaneler aşağıdaki görevlerin gerçekleştirilmesini sağlar (Yosifovich ve diğerleri, 2017):

- İşlem ve iş parçacıklarını oluşturmak, silmek.
- Windows uygulamalarını sonlandırmak.
- Çekirdekten gelen bildirimleri (fare imleci, klavye girdileri ve masaüstü pencerelerinin işlenmesi) Windows uygulamalarında pencereler içerisinde göstermek.
- Etkileşimli oturumlarda çalışan CSRSS işlemi beşinci olarak Canonical Display Driver (Cdd.dll) kütüphanesini de yükler. Bu kütüphane sayesinde masaüstü pencerelerinin yenilenmesi için çekirdekteki DirectX desteği ile iletişim kurar.

2.4.7. Windows Initialization Process (wininit.exe)

WININIT, işletim sisteminin başlatılması sürecinde bazı işlemlerin çalıştırılmasından ve aşağıdaki ayarların yapılmasından sorumludur (Yosifovich ve diğerleri, 2017):

- Kendi işlemini ve ana iş parçacığını kritik olarak etiketler; böylece eğer işlem erken sonlandırılır ve sistem hata ayıklama modunda başlatılırsa, sistem hata ayıklayıcıya girer. Aksi takdirde sistem çökecektir.
- Hangi WINLOGON işleminin ilk başlatılacağını belirlemek için Global\FirstLogonCheck isimli bir olay oluşturur.
- Kendi işlem temel önceliğini 13'e ve ana iş parçacığının önceliğini 15'e yükseltir.
- Varsayılan ortam değişkenlerini (COMPUTERNAME, USERPROFILE, ALLUSERSPROFILE, PUBLIC ve ProgramData) ayarlar.

- %SystemRoot%\Temp klasörünü oluşturur.
- Servis oturumunda işlemlerin çalışması için bir pencere istasyonu (Winsta0) ve iki masaüstü (Winlogon ve Default) oluşturur.
- Yerel olarak depolanmasına veya etkileşimli olarak girilmesi gerekip gerekmediğine bağlı olarak LSA makine şifreleme anahtarını başlatır.
- SERVICES işlemini başlatır.
- Varsayılan olarak LSASS işlemini başlatır. Eğer kimlik bilgisi koruması (credential guard) etkinleştirilmişse Credential Guard & Key Guard (lsaiso.exe) işlemini de başlatır.
- İşletim sistemi kurulumu sonrası ilk açılışta veya büyük bir işletim sistemi yapısına güncelleme yapıldıysa kurulum programını başlatır.

2.4.8. Service Control Manager (services.exe)

SERVICES, Windows servislerini yönetir ve bu tür hizmetlerin bir listesini kendi özel bellek alanında tutar (Ligh ve diğerleri, 2014). SERVICES, servislerin bağımlılıklarına göre belirli bir sırada yüklenmesini sağlamaktan sorumludur; ayrıca sistemdeki servislerin mevcut durumuyla ilgili servislerin duraklatılması, çalıştırılması ve durdurulması gibi bilgileri de tutar (Ligh ve diğerleri, 2010). Servisler sistemde HKLM\SYSTEM\CurrentControlSet\Services altında tanımlanmıştır. Olay günlüğü ve Görev zamanlayıcısı gibi bileşenler Windows'ta çalışan servislere örnek gösterilebilir. Tüm servisler SERVICES işleminin alt öğeleri olarak çalışır. Genellikle servisleri çalıştırmaktan sorumlu olan SVCHOST işlemi tarafından başlatılırlar. Çalışan servislerin sayısı çalışan servis işlemlerinin sayısı ile her zaman aynı olmayabilir çünkü bazı servisler bazı işlemleri ortak kullanabilir.

Servisler etkileşimli oturum açma gerektirmeden sistem önyükleme zamanında otomatik olarak başlayacak şekilde ayarlanabilirler. Ayrıca hizmet yönetim aracını çalıştırarak, sc.exe aracını kullanarak veya Windows StartService fonksiyonunu çağırarak manuel olarak da başlatılabilirler. İstisnaları olsa da servisler genellikle oturum açmış olan kullanıcı ile etkileşime girmez (Goktepe, 2002). Ek olarak, çoğu servis özel hizmet hesaplarında (LOCAL SYSTEM, LOCAL SERVICE) çalışır ancak oturum açmış kullanıcı hesaplarıyla da aynı güvenlik bağlamında çalışabilir.

Servisler 3 farklı isme sahip olabilir; işlem adı (process name) servise ait çalışan işlemin, dahili adı (internal name) kayıt defteri altında tutulan ve ekran adı (display name) hizmet yönetim aracında görünen ismidir. Ayrıca servisin ne yaptığını daha ayrıntılı olarak belirten bir açıklama alanına da sahiptirler.

2.4.9. Host Process for Windows Services (svchost.exe)

SVCHOST işleminin görevi Dinamik Bağlantı Kitaplığı (DLL) dosyalarında bulunan servisleri yüklemek ve onlar için kabuk görevi görmektir. Her servis kendisine ait işlemleriyle gruplanarak SVCHOST altında çalıştırılır böylece bir serviste oluşan sorun diğer servisin çalışmasını etkilemez (Simpson, 2017). Bu sayede servisler arasında bir tür yalıtım sağlanmış olur.

SVCHOST işlemi her zaman -k bayrağı ile başlatılır ve -k bayrağından sonra gelen ifade servise ait HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost kayıt defteri anahtarında tutulan Internal Name değeridir. Zararlı yazılım çalıştırmayan bir sistemde tüm SVCHOST işlemleri **SERVICES** tarafından başlatılır (Monnappa, 2018).

2.4.10. Local Security Authority Subsystem Service (lsass.exe)

LSASS işlemi, kullanıcı kimliğinin doğrulanması, parolaların yönetilmesi ve erişim belirteçlerinin oluşturulması gibi güvenlikle ilgili temel görevleri yerine getirmekten sorumludur. Bu nedenle, LSASS işleminin belleğinde önemli miktarda hassas veri mevcuttur ve bu veriler kimlik bilgilerini çalmaya çalışan herhangi bir saldırgan için LSASS işlemini birincil hedef haline getirir (Ah-Fat ve diğerleri, 2020).

LOGONUI işlemi tarafından kimlik doğrulama talep edildiğinde LSASS işlemi, parolanın aktif dizinde (active directory) veya SAM'de (kullanıcı ve grup bilgilerini içeren kayıt defteri bölümü) eşleşip eşleşmediğini kontrol etmek için kimlik doğrulama paketini içeren uygun kütüphaneyi çağırır. Eğer credential guard etkinse ve bu bir etki alanında oturum açma ise, LSASS kimlik doğrulama isteğinin geçerliliğini sorgulamak için Credential Guard & Key Guard (lsaiso.exe) ile iletişim kurar.

LSASS kullanıcıların access token hash değerlerini kendi belleğinde tutmak yerine LSAISO işlemi kullanılır. Çünkü LSAISO, Sanal Güven Seviyesi (VTL) 1'de çalışan bir Trustlet (izole kullanıcı modu) işlemi olduğundan, hiçbir işlem (çekirdek modunda çalışan işlemler bile) adres

alanına erişemez. LSASS ise ihtiyaç duyulan şifre hash değerinin sadece şifrelenmiş bir “blob”unu saklar (Yosifovich ve diğerleri, 2017). Kimlik doğrulaması başarılı olursa LSASS tarafından kullanıcının güvenlik profilini içeren bir access token oluşturulur.

2.4.11. Windows Logon Application (winlogon.exe)

WINLOGON, etkileşimli kullanıcı oturumlarını açma ve kapatma isteklerine yanıt verir, gerektiğinde ekran koruyucuyu başlatır ve kullanıcı profillerinin yüklenmesine yardımcı olur (Ligh ve diğerleri, 2014). Sadece etkileşimli oturumlarda çalışır. Ancak yalnızca kullanıcı oturumu açma ve kapatma sırasında değil, kullanıcının Güvenli Dikkat Sırası (SAS) tuş kombinasyonunu her kullanımında da etkinleştirir.

SAS her işletim sisteminde farklı tuşlardan oluşmakla birlikte Windows'ta Ctrl + Alt + Delete tuşlarına öncekiler kaldırılmadan sırasıyla basılmasına verilen isimdir. SAS'ın amacı, kullanıcıları oturum açma sürecini taklit eden parola yakalama programlarından korumaktır, çünkü bu tuş kombinasyonu kullanıcı modunda çalışan uygulamalar tarafından engellenemez.

Windows'un normal açılışında bizi LOGONUI işleminin oturum açma kullanıcı arayüzü karşılar. Arayüz kullanıcının oturum açma bilgilerini girmesini bekler. LOGONUI işlemi girilen kullanıcı bilgilerini doğrulanması için LSASS işlemine gönderir. LSASS işlemi kullanıcı bilgilerini bir hesapla eşleştirdiğinde kimlik doğrulamasını başarıyla gerçekleştirmiş olur ve bir access token oluşturulur. Bu access token daha sonra WINLOGON tarafından giriş yapan kullanıcı oturumundaki ilk işlemleri başlatmak için kullanılır. İlk işlemler, HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon kayıt defteri anahtarı altındaki Userinit değerinde depolanır. Değer varsayılan olarak sadece "C:\Windows\system32\userinit.exe," 'dir ancak birden fazla işlem de olabilir.

2.4.12. Windows Logon User Interface Host (logonui.exe)

LOGONUI işlemi kullanıcının giriş yapması için gerekli bilgileri girmesini sağlayan bir arayüz uygulamasıdır. Windows açılışında LOGONUI işlemi WINLOGON tarafından otomatik olarak başlatılır ve bizi oturum açma kullanıcı arayüzü ile karşılar. Ancak oturum açık olduğu durumlarda WINLOGON işlemi kullanıcının SAS'ı tuşlamasını bekler. Bu durumda LOGONUI arayüzü; hesabı kilitleme, kullanıcı değiştirme, oturumu kapatma ve görev yöneticisini başlatma vb. seçenekleri sunar.

LOGONUI kullanıcı tanımlama ve kimlik doğrulama işlemlerini kimlik bilgisi sağlayıcı kütüphaneleri (parola, pin kodu, yüz tarama, parmak izi okuma, fiziksel güvenlik anahtarı, akıllı kart vb.) aracılığıyla gerçekleştirilir. Akıllı kart gibi sağlayıcılar kullanılması durumunda LOGONUI işlemi alt öğelere sahip olabilir.

2.4.13. Userinit Logon Application (userinit.exe)

USERINIT işleminin görevi oturum açma komut dosyasını çalıştırmak, ağ bağlantılarını kurmak ve Windows kullanıcı arayüzü gibi ortamları başlatmaktır. Sonra kayıt defterinde HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon konumunda bulunan shell değerini çalıştırır. Shell'in varsayılan değeri “explorer.exe” 'dir. Shell değerini başlattıktan sonra USERINIT işlemi sonlanır.

2.4.14. Windows Explorer (explorer.exe)

EXPLORER, Windows Gezgini olarak bilinen dosya yöneticisi uygulamasıdır ve Microsoft Windows işletim sisteminin Windows 95'ten sonraki sürümlerinde bulunur. Dosya sistemlerine erişim için bir kullanıcı arayüzü sağlar. Aynı zamanda başlangıç menüsü, görev çubuğu ve masaüstü gibi monitörde birçok kullanıcı arabirimi öğesini sunan işletim sistemi bileşenidir. EXPLORER işlemi kullanıcının doğrudan etkileşimde olduğu bir arayüze sahiptir, bu yüzden kullanıcı tarafından arayüzden başlatılan çoğu işlemin üst öğesidir.

3. MALZEME VE YÖNTEM

Bu bölümde bellek ve zararlı yazılım analizinde kullanılan VMware Workstation Pro, Process Hacker, Process Monitor araçları hakkında kısaca bilgi verilmiş ve araştırmanın temel aracı olan Volatility Framework; geliştirilme amacı, kullanım şekli ve iki büyük sürüm arasındaki farkıyla anlatılmıştır. Ayrıca analizler sırasında Volatility Framework tarafından üretilen çıktılar ekran görüntüleriyle birlikte sunulmuştur. Volatility Framework'ün her iki sürümü de sanal makinede çalışan Kali işletim sistemine eksiksiz bir şekilde yüklenmiş ve bellek analizleri bu makinede ana bilgisayardan bağımsız bir şekilde gerçekleştirilmiştir.

Bellek analizi için son zamanlarda Windows kullanıcılarına çokça zarar vermiş iki farklı zararlı yazılım seçilmiştir. Birincisi **Sorebrex** adında dosyasız bir fidye yazılımıdır ve dosyaları şifreleyerek kullanıcıdan para talep eder. İkincisi **XMRig** adında kripto para madenciliği yapan bir zararlı yazılımdır ve bilgisayarın donanımını yüksek seviyelerde kullanmak için zorlar. Her iki zararlı yazılımın da gelecek araştırmalarda kullanılabilmesi için Mesaj Özeti Algoritması 5 (MD5), Güvenli Karma Algoritması 1 (SHA-1) ve Güvenli Karma Algoritması 2 (SHA-256) değerleri ve Virustotal kaynakları verilmiştir.

3.1.VMWARE WORKSTATION PRO

VMware Workstation Pro, Linux veya Windows bilgisayarlarda sanal makine oluşturmak ve çalıştırmak için kullanılan bir sanallaştırma yazılımıdır. Kullanıcılara tek bir fiziksel bilgisayarda sanal makineler oluşturmasına ve bunları ana bilgisayarla birlikte aynı anda kullanmasına imkân tanır (Wikipedia, 2021b). Her sanal makine kendi işletim sistemini çalıştırabilir. İstenildiğinde sanal makinenin ana bilgisayardan bağımsız çalıştırılabilmesi, duraklatılabilmesi ve anlık görüntüsünün oluşturulabilmesi zararlı yazılımın güvenli bir ortamda test edilmesini sağlar.

3.2.PROCESS HACKER

Zararlı yazılım çalıştırırken gerçekleşen işlem etkinliklerini analiz etmek için kullanımı kolay ücretsiz bir araç olan Process Hacker kullanılmıştır (Liu, 2021). Process Hacker ile sistemde gerçekleşen tüm işlem aktiviteleri görülebilir. Yeni başlatılan işlemler yeşil renkte ve

sonlandırılmak üzere olan işlemler ise kırmızı renkte görünür. Ayrıca işlemlerin özellikleri, istatistikleri, İnternet kullanımını, disk erişimi ve CPU-RAM tüketim grafikleri kontrol edilebilir. Zararlı yazılım analizi için oldukça yetenekli bir araçtır.

3.3.PROCESS MONITOR

İşletim sistemi çalışırken tüm işlemlere ait etkinlikleri gerçek zamanlı bir şekilde izlemek ve onları kaydetmek için Sysinternals aracı olan Process Monitor kullanılmıştır (Russinovich, 2021a). Process Monitor ile gerçek zamanlı dosya sistemi etkinliği, kayıt defteri kayıtları, işlem ve iş parçacığı etkinlikleri izlenebilir. Etkinlikler kayıpsız bir şekilde filtrelenerek işlem özelinde etkinlikler görüntülenebilir. Ayrıca işlemlere ait üst işlem, alt işlem, kimlik numarası, temel öncelik numarası, kullanıcı adı, oturum kimliği ve dosya yolu gibi bilgiler de görülebilir. Zararlı yazılım analizi için oldukça yetenekli bir araçtır.

3.4.VOLATILITY FRAMEWORK

Volatility Framework, adli bilişim topluluğu üyeleri tarafından Windows, Mac, Linux ve Linux çekirdeğinden türetilen işletim sistemlerinin uçucu bellek görüntülerini incelemek için dünyanın en yaygın kullanılan ve 2007 yılından beri tamamen açık kaynak olarak Genel Kamu Lisansı (GNU General Public License) altında geliştirilen ücretsiz bir yazılım çerçevesidir.

Volatility Framework; adli bilişim incelemelerine, olay müdahalesine ve zararlı yazılım analizine odaklanmasına rağmen uçucu bellek görüntülerindeki veriyi anlamlandırmayı ve bu araştırma alanında daha fazla çalışma gerçekleştirilmesi için bir platform oluşturmayı amaçlar. Verileri anlamlandırma işlemi daha önceden oluşturulmuş uçucu bellek görüntüsü üzerinden yapılacağı için görüntünün alındığı ve inceleneceği sistemden tamamen bağımsız olarak gerçekleştirilir ve sonuçlar sistemin çalışma zamanında görünürlük sağlar (Foundation, 2021a). Volatility Framework daha iyi anlaşılabilmesi için bazı özellikleri aşağıda maddeler halinde verilmiştir.

- Volatility Framework bellek görüntüsü oluşturmaz, oluşturulan görüntüleri incelemeye yarar ve sonraki maddede belirtilen farklı biçimlerde oluşturulmuş görüntüleri birbirleri arasında dönüştürebilir.

- Desteklediği dosya türleri şunlardır: VirtualBox, VMware ve QEMU sanal makineleri ile oluşturulan bellek görüntüleri, Windows 7 ve öncesine ait uyku dosyaları (hibernation file), Hata (Crash) dosyaları, Raw (dd), Bilirkişi Formatı (EWF), Mach-O, Firewire, HPAK (FDPro), Linux Bellek Çıkarıcı (LiME) dökümleri.
- 200'den fazla eklentiye sahiptir. Eklentiler modüler bir şekilde geliştirilmesi sebebiyle yeni yayınlanan işletim sistemi sürümlerini kolayca desteklemesi sağlanır.
- Grafik bir ara yüze sahip değildir, terminalden çalışır.
- Terminalden çalıştırılabilir olması, eklentilerin ayrı ayrı çalıştırılabilmesi ve birçok tersine mühendislik aracında kullanılan Python ile geliştirilmiş olması sebebiyle diğer yazılımlar ile kolayca birleştirilebilir ve otomatikleştirilebilir.

Volatility Framework'ün şu an aralarında büyük fark olan iki farklı sürümü mevcuttur. İlki Python 2.6 kütüphaneleri ile çalışan Volatility 2 sürümüdür. Bu sürümde eklentileri kullanabilmek için bellek görüntüsünün ait olduğu işletim sistemi sürümünün Volatility profili bilinmeli veya imageinfo eklentisi ile bulunmalıdır. Profili bilinen bellek görüntüsünü inceleme işlemi diğer eklentiler ile Şekil 3.1'de gösterildiği gibi yapılabilir. Şekilde ilk önce imageinfo eklentisi kullanılarak image1.vmem isimli görüntü dosyasının profili Win10x64_19041 olarak tanımlanmış ve bu profil bilgisi ile bellek görüntüsü alındığı sırada çalışan işlemleri listelemeye yarayan pslist eklentisi çalıştırılmıştır.

```

kali@kali: ~/Downloads
File Actions Edit View Help

(kali@kali)~/Downloads
$ python2 volatility2/vol.py -f images/image1.vmem imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s) : Win10x64_19041
                           AS Layer1 : SkipDuplicatesAMD64PagedMemory (Kernel AS)
                           AS Layer2 : FileAddressSpace (/home/kali/Downloads/images/image1.vmem)
                           PAE type : No PAE
                           DTB : 0x1ad002L
                           KDBG : 0xf8074a215b20L
      Number of Processors : 2
      Image Type (Service Pack) : 0
      KPCR for CPU 0 : 0xfffff80748558000L
      KPCR for CPU 1 : 0xfffff80019a7e0000L
      KUSER_SHARED_DATA : 0xfffff78000000000L
      Image date and time : 2021-01-16 12:48:06 UTC+0000
      Image local date and time : 2021-01-16 15:48:06 +0300

(kali@kali)~/Downloads
$ python2 volatility2/vol.py pslist -f images/image1.vmem --profile=Win10x64_19041
Volatility Foundation Volatility Framework 2.6.1
Offset(V)      Name      PID      PPID     Thds     Hnds     Sess     Wow64     Start      Exit
-----
0xffffe58f0489f040 System    4         0       116      0         0         0  2021-01-16 12:41:33 UTC+0000
0xffffe58f04888080 Registry  92         4         4         0         0         0  2021-01-16 12:41:25 UTC+0000
0xffffe58f09585040 smss.exe 336         4         3         0         0         0  2021-01-16 12:41:33 UTC+0000
0xffffe58f07711140 csrss.exe 424        416         9         0         0         0  2021-01-16 12:41:50 UTC+0000
0xffffe58f0cca5080 wininit.exe 496        416         1         0         0         0  2021-01-16 12:41:50 UTC+0000
0xffffe58f0be58080 services.exe 628        496         8         0         0         0  2021-01-16 12:41:50 UTC+0000
0xffffe58f07708080 lsass.exe 644        496         8         0         0         0  2021-01-16 12:41:50 UTC+0000
0xffffe58f0d1c2240 svchost.exe 748        628        27         0         0         0  2021-01-16 12:41:50 UTC+0000
0xffffe58f0d1c4080 fontdrvhost.exe 756        496         5         0         0         0  2021-01-16 12:41:50 UTC+0000
0xffffe58f0db152c0 svchost.exe 836        628        14         0         0         0  2021-01-16 12:41:50 UTC+0000

```

Şekil 3.1: Volatility 2’de imageinfo eklentisi ile profil tanımlama ve pslist eklentisinin kullanımı.

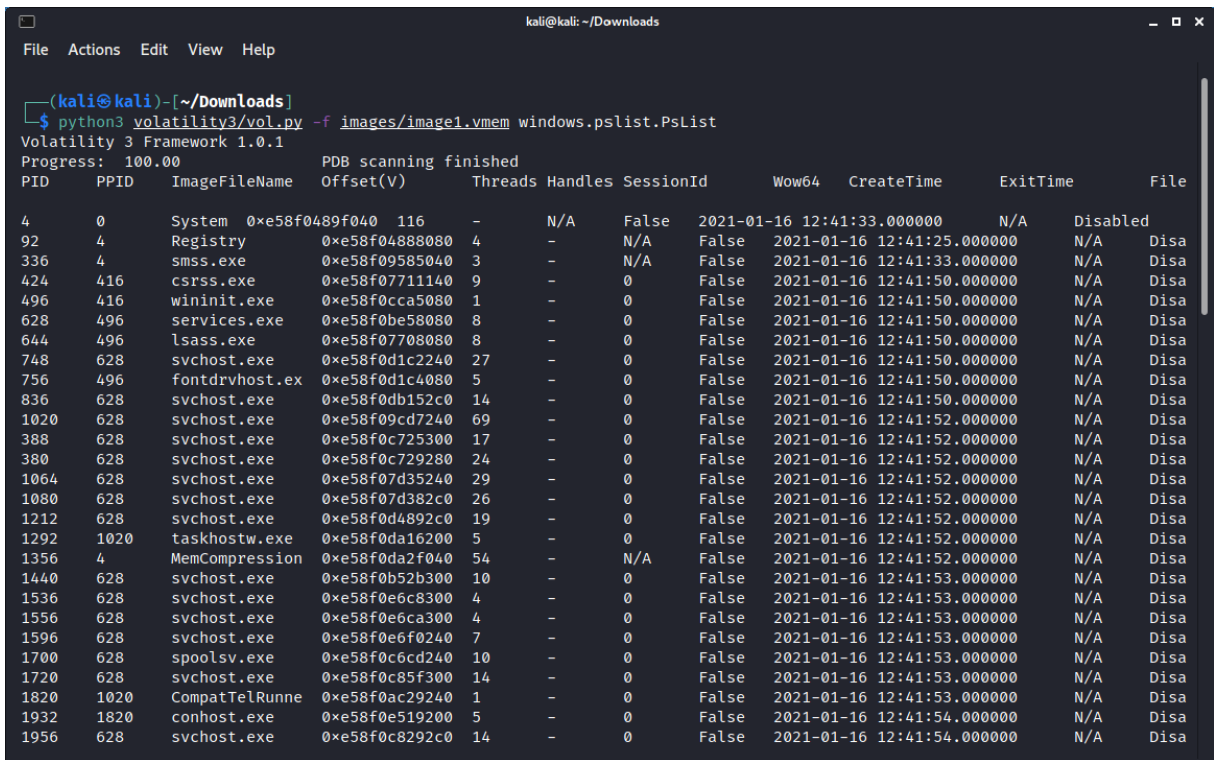
Diğer Volatility sürümü ise Python 3 ile çalışan Volatility 3 sürümüdür. Volatility Kuruluşu, Volatility yazılım çerçevesinin tamamını yeniden yazarak Volatility 3 sürümüyle yayınladı. Proje, önceki 10 yılda belirgin hale gelen orijinal kod tabanıyla ilişkili birçok teknik ve performans sorununu ele almayı amaçlıyordu (Foundation, 2021b). Bu bağlamda 2019 yılında 1.0.0 beta sürümü ve 1 Şubat 2021’de Volatility 3’ün 1.0.1 release sürümü yayımlandı.

Volatility 3’ün bazı özellikleri ve Volatility 2 ile arasındaki bazı farklar aşağıda maddeler halinde belirtilmiştir (Case, 2020).

- Bellek görüntülerinin ait olduğu profilleri otomatik tanımlama gerçekleştirildi. Tüm eklentiler profil belirtmeden doğrudan kullanılır.
- Volatility Framework’ünün tamamı (arka plan kodları, eklentiler vb.) tamamen yeniden tasarlandı ve Python 3 ile yazıldı.
- Eklentilere sürüm eklendi.
- Eklentiler gerektiğinde diğer eklentileri doğrudan çağırabilir.
- Performans artışları sağlandı.
- Diğer kütüphaneler ile çalıştırma ve kullanıcı ara yüzü basitleştirildi.

- Geliştiriciler için kapsamlı Uygulama Programlama Arayüzü (API) belgeleri yayınlandı.
- Aynı anda birden fazla görüntü dosyasını otomatik analiz etme özelliği eklendi.

Volatility 3 ile bellek görüntüsü alındığı sırada çalışan işlemleri listelemeye yarayan pslist eklentisinin kullanımı Şekil 3.2’de gösterilmiştir, eklenti Volatility 2’nin aksine profil belirtmeden çalıştırılmıştır.



```

kali@kali: ~/Downloads
File Actions Edit View Help
(kali@kali) - [~/Downloads]
$ python3 volatility3/vol.py -f images/image1.vmem windows.pslist.PsList
Volatility 3 Framework 1.0.1
Progress: 100.00
PDB scanning finished

```

PID	PPID	ImageFileName	PDB scanning	Offset(V)	Threads	Handles	SessionId	Wow64	CreateTime	ExitTime	File
4	0	System	0xe58f0489f040	116	-	N/A	False	2021-01-16	12:41:33.000000	N/A	Disabled
92	4	Registry	0xe58f04888080	4	-	N/A	False	2021-01-16	12:41:25.000000	N/A	Disa
336	4	smss.exe	0xe58f09585040	3	-	N/A	False	2021-01-16	12:41:33.000000	N/A	Disa
424	416	csrss.exe	0xe58f07711140	9	-	0	False	2021-01-16	12:41:50.000000	N/A	Disa
496	416	wininit.exe	0xe58f0cca5080	1	-	0	False	2021-01-16	12:41:50.000000	N/A	Disa
628	496	services.exe	0xe58f0be58080	8	-	0	False	2021-01-16	12:41:50.000000	N/A	Disa
644	496	lsass.exe	0xe58f07708080	8	-	0	False	2021-01-16	12:41:50.000000	N/A	Disa
748	628	svchost.exe	0xe58f0d1c2240	27	-	0	False	2021-01-16	12:41:50.000000	N/A	Disa
756	496	fontdrvhost.exe	0xe58f0d1c4080	5	-	0	False	2021-01-16	12:41:50.000000	N/A	Disa
836	628	svchost.exe	0xe58f0db152c0	14	-	0	False	2021-01-16	12:41:50.000000	N/A	Disa
1020	628	svchost.exe	0xe58f09cd7240	69	-	0	False	2021-01-16	12:41:52.000000	N/A	Disa
388	628	svchost.exe	0xe58f0c725300	17	-	0	False	2021-01-16	12:41:52.000000	N/A	Disa
380	628	svchost.exe	0xe58f0c729280	24	-	0	False	2021-01-16	12:41:52.000000	N/A	Disa
1064	628	svchost.exe	0xe58f07d35240	29	-	0	False	2021-01-16	12:41:52.000000	N/A	Disa
1080	628	svchost.exe	0xe58f07d382c0	26	-	0	False	2021-01-16	12:41:52.000000	N/A	Disa
1212	628	svchost.exe	0xe58f0d4892c0	19	-	0	False	2021-01-16	12:41:52.000000	N/A	Disa
1292	1020	taskhostw.exe	0xe58f0da16200	5	-	0	False	2021-01-16	12:41:52.000000	N/A	Disa
1356	4	MemCompression	0xe58f0da2f040	54	-	N/A	False	2021-01-16	12:41:52.000000	N/A	Disa
1440	628	svchost.exe	0xe58f0b52b300	10	-	0	False	2021-01-16	12:41:53.000000	N/A	Disa
1536	628	svchost.exe	0xe58f0e6c8300	4	-	0	False	2021-01-16	12:41:53.000000	N/A	Disa
1556	628	svchost.exe	0xe58f0e6ca300	4	-	0	False	2021-01-16	12:41:53.000000	N/A	Disa
1596	628	svchost.exe	0xe58f0e6f0240	7	-	0	False	2021-01-16	12:41:53.000000	N/A	Disa
1700	628	spoolsv.exe	0xe58f0c6cd240	10	-	0	False	2021-01-16	12:41:53.000000	N/A	Disa
1720	628	svchost.exe	0xe58f0c85f300	14	-	0	False	2021-01-16	12:41:53.000000	N/A	Disa
1820	1020	CompatTelRunne	0xe58f0ac29240	1	-	0	False	2021-01-16	12:41:53.000000	N/A	Disa
1932	1820	conhost.exe	0xe58f0e519200	5	-	0	False	2021-01-16	12:41:54.000000	N/A	Disa
1956	628	svchost.exe	0xe58f0c8292c0	14	-	0	False	2021-01-16	12:41:54.000000	N/A	Disa

Şekil 3.2: Volatility 3’te pslist eklentisinin kullanımı.

3.5.SOREBRECT

Sorebrect 2017’de tanımlanan Windows işletim sistemini hedefleyen dosyasız bir fidye yazılımıdır. Fidye yazılımı, bilgisayar sistemlerinin önemli işlevlerini devre dışı bırakarak veya cihazdaki kullanıcı dosyalarını şifreleyerek kullanıcılardan zorla para almak için tasarlanmış bir zararlı yazılım sınıfıdır. Sorebrect’e ait bilgiler Tablo 3.1’de verilmiştir.

Tablo 3.1: Sorebrect zararlı yazılımına ait bilgiler (Virusotal, 2017).

Özellik	Değer
BOYUT	999.00 KB (1022978 bytes)
MD5	83e824c998f321a9179efc5c2cd0a118
SHA-1	16b84004778505afbcc1032d1325c9bed8679b79
SHA-256	4142ff4667f5b9986888bdc2a727db6a767f78fe1d5d4ae3346365a1d70eb76

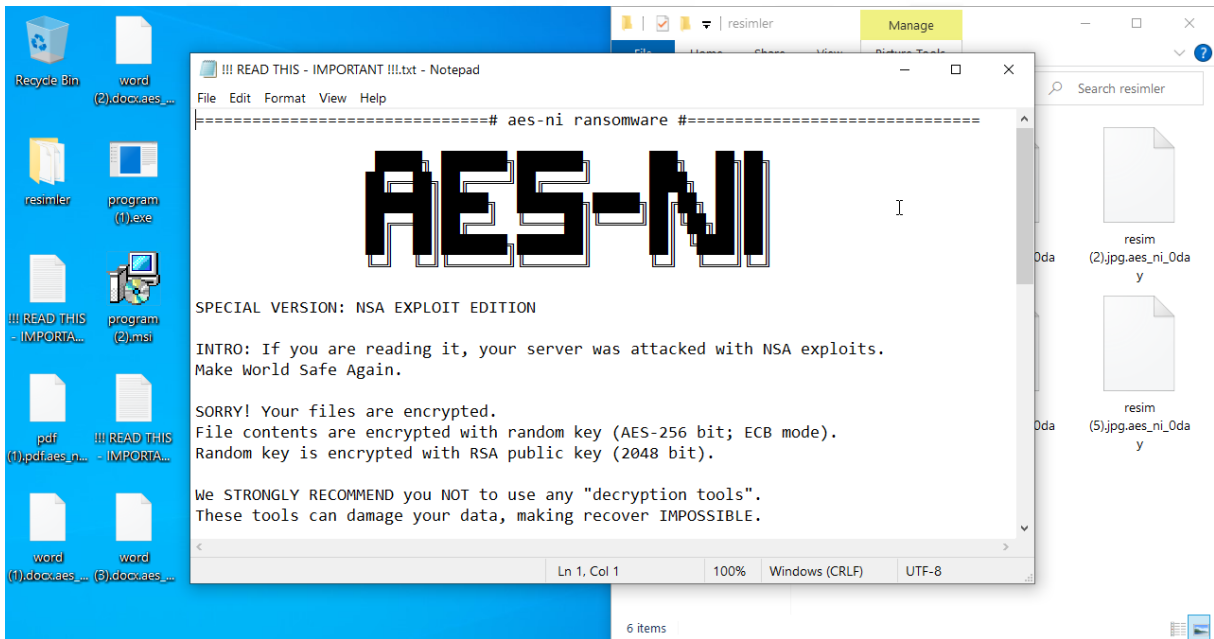
Sorebrect tamamen gizlilik içinde çalıştığından tespit edilmesi zordur. Zararlı kodlarını meşru bir sistem işlemine (SVCHOST) enjekte ettikten sonra kendini silmesi onu dosyasız bir tehdit haline getirir. Ayrıca kayıt defterinde herhangi bir bilgi depolamaz; olay günlüklerini, zaman damgalarını (appcompat, shimcache ve prefetch), Windows sistem geri yükleme noktalarını ve Volume Shadow kopyalarını temizler. SVCHOST işlemine enjekte edilen zararlı kod parçası ise dosyaların şifrlenmesini sağlar. Sunucu ile olan bağlantısını anonimleştirmek için Tor ağ protokolünü kullanır (TorProject, 2019). Sorebrect'in saldırı yöntemi Şekil 3.3'te gösterilmiş ve genel olarak aşağıda anlatılmıştır:

- Hedef sistemde yetkili kullanıcıya ait kimlik bilgilerini çalmak için açık bulunan RDP bağlantı noktalarına kaba kuvvet (brute-force) yöntemiyle saldırır.
- Erişim sağlandıktan sonra kendisini PsExec (Rusinovich, 2021b) aracılığıyla çalıştırır.
- Sorebrect zararlı kodu SVCHOST işlemine enjekte ederek, zararlı kodun yüksek ayrıcalıklarla çalışmasını sağlar. Sonra kendisini sonlandırır ve siler.
- Bundan sonraki tüm işlemleri SVCHOST işlemine enjekte edilen zararlı kod parçası yürütür. Adında "Windows", "Desktop" kelimesini içeren klasör ve sys, exe, msi, lnk, dll uzantılı dosyalar haricindeki ağda paylaşılanlar dahil her dosyayı şifreler (Keijzer, 2020).

**Şekil 3.3:** Sorebrect saldırı yöntemi (Tancio ve Yaneza, 2017).

3.5.1. Sorebrect Uygulama

Sorebrect zararlı yazılımı çalıştırılmadan önce sanal makineye örnek resim, belge ve program dosyaları kopyalanmış ve Windows Defender gerçek zamanlı koruması kapatılmıştır. SOREBRECT.exe dosyası yönetici olarak çalıştırılmıştır. Sorebrect kendi dosyasını hemen silmiş ve birkaç dakika bekledikten sonra Şekil 3.4'te gösterildiği gibi belirli uzantılardaki dosyaların adına “.aes_ni_0day” uzantısını ekleyerek şifrelemiş, şifrelenen dosyaların yanına **!!! READ THIS – IMPORTANT !!!.txt** isimli içeriği şekilde gösterilen metin dosyasını oluşturmuştur. Bu aşamada sanal makinenin anlık görüntüsü alınmış ve bellek dosyasının adı incelenmek üzere **SOREBRECT.vmem** olarak değiştirilmiştir.



Şekil 3.4: Sorebrect zararlı yazılımı çalıştırıldıktan sonra.

3.5.2. Sorebrect Analizi

SOREBRECT.vmem bellek görüntüsünde çalışan işlemler Volatility pstree eklentisi ile ağaç yapısında Şekil 3.5'teki gibi awk komutu ile filtrelenerek listelenmiştir. Filtreye WININIT ve SERVICES işlemleri SVCHOST'un üst işlemleri olduğu için eklenmiştir.

Tablo 4.1'de verilen işlemlerin üst-alt ilişkisine göre tüm SVCHOST işlemlerinin üst işlemi SERVICES'tir. Şekil 3.5'e dikkat edildiğinde SVCHOST işlemleri Üst İşlem Kimlik Numarası (PPID) 592 olarak SERVICES işleminin altında çalışırken, işaretlenmiş SVCHOST işlemi ise 640 PPID numarası ile listelenmeyen bir işlemin altında çalışmaktadır. Bu durum **6076**

İşlem Kimlik Numarasına (PID) sahip SVCHOST işleminin şüpheli olduğunu gösterir. Ayrıca işlemin kullanıcı (1 numaralı) oturumunda çalışması, Wow64 parametresinin True olmasından anlaşılacağı üzere 64-bit olması ve diğer SVCHOST işlemlerinden 3 dakika daha geç çalışmaya başlaması gibi farklılıklar vardır.

```
(kali@kali)-[~/Downloads]
└─$ python3 volatility3/vol.py -f images/SOREBRECT.vmem windows.pstree | awk 'NR<5 || /winit|services|svchost/'
Volatility 3 Framework 1.0.1 PDB scanning finished
```

PID	PPID	ImageFileName	Offset(V)	Threads	Handles	SessionId	Wow64	CreateTime	ExitTime
484	404	wininit.exe	0x898ffbd9e300	1	-	0	False	2021-04-22 14:28:22.000000	N/A
* 592	484	services.exe	0x898ffbd9e300	6	-	0	False	2021-04-22 14:28:22.000000	N/A
** 1892	592	svchost.exe	0x898ffbd9e300	14	-	0	False	2021-04-22 14:28:24.000000	N/A
** 2920	592	svchost.exe	0x898ffbd9e300	3	-	1	False	2021-04-22 14:30:43.000000	N/A
** 2548	592	svchost.exe	0x898ffbd9e300	9	-	1	False	2021-04-22 14:28:27.000000	N/A
** 1272	592	svchost.exe	0x898ffbd9e300	16	-	0	False	2021-04-22 14:28:23.000000	N/A
** 2172	592	svchost.exe	0x898ffbd9e300	13	-	0	False	2021-04-22 14:28:25.000000	N/A
6076	640	svchost.exe	0x898ffbd9e300	8	-	1	True	2021-04-22 14:33:08.000000	N/A

Şekil 3.5: SOREBRECT.vmem bellek görüntüsüne ait pstree eklenti sonuçları.

SOREBRECT.vmem bellek görüntüsünde çalışan işlemlerin komut satırı parametreleri Volatility cmdline eklentisi ile Şekil 3.6'daki gibi awk komutu ile filtrelenerek listelenmiştir.

Bölüm 2.3.9'da SVCHOST işleminin ayrıntılarında değinildiği üzere tüm işlemler -k bayrağı ile başlatılır. Şekil 3.6'da SVCHOST işlemleri -k bayrağı ile başlatıldığı görülüyorken 6076 PID numaralı işlem ise -k bayrağı olmadan başlatılmıştır. İşlem yolu her ne kadar "C:\Windows\SysWOW64\svchost.exe" olarak meşru olsa da -k bayrağı ile başlatılmamış olması 6076 PID numaralı SVCHOST işlemi üzerindeki şüpheleri attırmıştır.

```
(kali@kali)-[~/Downloads]
└─$ python3 volatility3/vol.py -f images/SOREBRECT.vmem windows.cmdline | awk 'NR<5 || /winit|services|svchost/'
Volatility 3 Framework 1.0.1 PDB scanning finished
```

PID	Process	Args
484	wininit.exe	wininit.exe
592	services.exe	C:\Windows\system32\services.exe
8	svchost.exe	C:\Windows\system32\svchost.exe -k ClipboardSvcGroup -p
4132	svchost.exe	C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted
5380	svchost.exe	C:\Windows\system32\svchost.exe -k LocalServiceAndNoImpersonation -p
5304	svchost.exe	C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted -p
2920	svchost.exe	C:\Windows\system32\svchost.exe -k UdkSvcGroup
6076	svchost.exe	"C:\Windows\SysWOW64\svchost.exe"

Şekil 3.6: SOREBRECT.vmem bellek görüntüsüne ait cmdline eklenti sonuçları.

SOREBRECT.vmem bellek görüntüsünde çalışan işlemlerin muhtemel kod enjeksiyonu içeren bellek alanlarını gösteren Volatility malfind eklentisi ile Şekil 3.7'deki gibi 6076 PID numarası ile filtrelenerek listelenmiştir.

Malfind eklentisi işlem belleğinde **0xb90000** ve **0x3e00000** adreslerinde başlayan iki adet kod enjekte edilmiş bellek alanı tespit etmiştir. Şekil 3.7’de gösterildiği üzere bellek koruması **PAGE_EXECUTE_READWRITE** olarak ayarlanmıştır. **PAGE_EXECUTE_READWRITE** bellek koruması ayarı zararlı yazılımın kod enjekte etmek için yeni bölümler tahsis edebileceği anlamına gelir. Bu durum 6076 PID numaralı SVCHOST işlemi üzerinde manipülasyon gerçekleştirildiğini gösterir.

```
(kali@kali)-[~/Downloads]
└─$ python3 volatility3/vol.py -f images/SOREBRECT.vmem windows.malfind --pid 6076
Volatility 3 Framework 1.0.1
Progress: 100.00 PDB scanning finished
PID Process Start VPN End VPN Tag Protection CommitCharge PrivateMemory File output Hexdump Disasm
6076 svchost.exe 0xb90000 0xc8dfff VadS PAGE_EXECUTE_READWRITE 254 1 Disabled
4d 5a 90 00 03 00 00 00 MZ.....
04 00 00 00 ff ff 00 00 .....
b8 00 00 00 00 00 00 00 .....
40 00 00 00 00 00 00 00 @.....
00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 .....
00 00 00 00 e8 00 00 00 .....
0xb90000: dec ebp
0xb90001: pop edx
0xb90002: nop
0xb90003: add byte ptr [ebx], al
0xb90005: add byte ptr [eax], al
0xb90007: add byte ptr [eax + eax], al
0xb9000a: add byte ptr [eax], al
6076 svchost.exe 0x3e00000 0x3fc2fff VadS PAGE_EXECUTE_READWRITE 451 1 Disabled
4d 5a 90 00 03 00 00 00 MZ.....
04 00 00 00 ff ff 00 00 .....
b8 00 00 00 00 00 00 00 .....
40 00 00 00 00 00 00 00 @.....
00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 .....
00 00 00 00 f0 00 00 00 .....
0x3e00000: dec ebp
0x3e00001: pop edx
0x3e00002: nop
0x3e00003: add byte ptr [ebx], al
0x3e00005: add byte ptr [eax], al
0x3e00007: add byte ptr [eax + eax], al
0x3e0000a: add byte ptr [eax], al
```

Şekil 3.7: SOREBRECT.vmem bellek görüntüsüne ait malfind eklenti sonuçları.

Procdump ve vaddump eklentileri ile 6076 PID numaralı SVCHOST işleminin çalıştırılabilir örneği ve **0xb90000** ve **0x3e00000** adreslerinde başlayan bellek alanları Şekil 3.8’de gösterildiği gibi SOREBRECT.vmem bellek görüntüsünden dışarı aktarılmıştır.

```

(kali@kali)-[~/Downloads]
└─$ python2 volatility2/vol.py -f images/SOREBRECT.vmem --profile=Win10x64_19041 procdump --pid 6076 --dump-dir ./SONUC
Volatility Foundation Volatility Framework 2.6.1
Process(V)      ImageBase      Name      Result
-----
0xfffff8980014b2300 0x0000000000b80000 svchost.exe OK: executable.6076.exe

(kali@kali)-[~/Downloads]
└─$ python2 volatility2/vol.py -f images/SOREBRECT.vmem --profile=Win10x64_19041 vaddump --pid 6076 --base 0xb90000 --dump-dir ./SONUC
Volatility Foundation Volatility Framework 2.6.1
Pid      Process      Start      End      Result
-----
6076    svchost.exe  0x0000000000b90000 0x0000000000c8dfff ./SONUC/svchost.exe.80b2300.0x0000000000b90000-0x0000000000c8dfff.dmp

(kali@kali)-[~/Downloads]
└─$ python2 volatility2/vol.py -f images/SOREBRECT.vmem --profile=Win10x64_19041 vaddump --pid 6076 --base 0x3e0000 --dump-dir ./SONUC
Volatility Foundation Volatility Framework 2.6.1
Pid      Process      Start      End      Result
-----
6076    svchost.exe  0x0000000003e00000 0x0000000003fc2fff ./SONUC/svchost.exe.80b2300.0x0000000003e00000-0x0000000003fc2fff.dmp

```

Şekil 3.8: SOREBRECT.vmem bellek görüntüsüne ait procdump ve vaddump eklenti sonuçları.

İki bellek alanında bulunan zararlı kod parçalarını doğrulamak için virustotal.com sitesine yüklenerek çevrimiçi tarama yapılmış ve **35** adet zararlı yazılım aracı tarafından tespit edilmiştir. Oluşturulan dosyalardan daha fazla bilgi elde etmek için statik ve dinamik analiz gibi tersine mühendislik teknikleri uygulanabilir.

3.6. XMRIG

XMRig Monero Miner 2017'de tanımlanan Windows işletim sistemini hedefleyen kripto para mandenciliği yapan bir zararlı yazılımdır. Bilgisayarın sistem kaynaklarını ele geçirir ve onları kripto para oluşturmak için maksimum seviyede kullanır. XMRig adını aslında Monero kripto para (XMR) madenciliği yapmak amacıyla geliştirilen XMRig yazılımından almıştır. XMRig açık kaynak kodlu, çapraz platformlarda Merkezî İşlem Birimi (CPU) ve Grafik İşlem Birimi (GPU) üzerinde çalışabilen bir madenci yazılımdır. XMRig'e ait bilgiler Tablo 3.2'de verilmiştir.

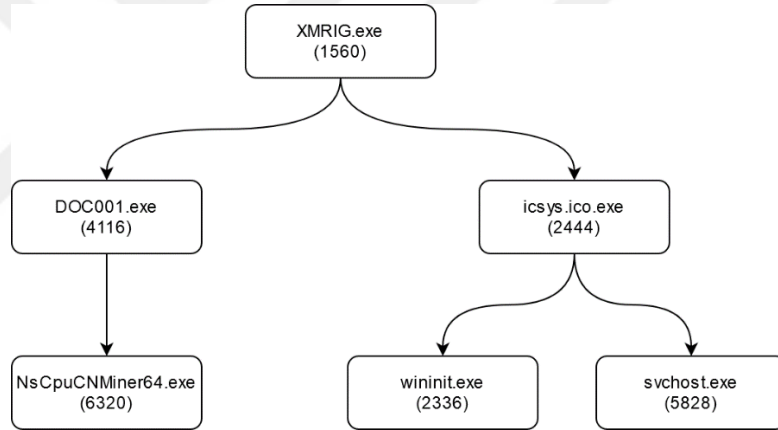
Tablo 3.2: XMRig zararlı yazılımına ait bilgiler (Virustotal, 2015).

Özellik	Değer
BOYUT	4.13 MB (4334887 bytes)
MD5	81ab069475bf534067c004daf2264c79
SHA-1	aec901033cbbbbfa00d4963c03abb309a51c2c4b
SHA-256	91c6e3cbe1c061a3dfd15ef47894436a6fc4341078a566e38131a6d9b655edac

XMRig kullanıcının göreceği hiçbir arabirime sahip olmadığı için tespit edilmesi kolay değildir, tüm işlemleri arka planda yürütür. Bir sisteme bulaştığının göstergeleri ise normalden fazla CPU tüketimi, sistemin genel olarak yavaşlaması ve donmaların yaşanmasıdır.

3.6.1. Xmrig Uygulama

XMRig zararlı yazılımı çalıştırılmadan önce Windows Defender gerçek zamanlı koruması kapatılmıştır. XMRig.exe dosyası yönetici olarak çalıştırılmıştır. XMRig kendisini DOC001.exe ismiyle "C:\Users\ali\AppData\Roaming\Temp\DOC001.exe" yoluna kopyalamış ve bu dosyayı da çalıştırarak Şekil 3.9'da görülen diğer işlemleri (WININIT ve SVCHOST) oluşturmaya devam etmiştir. Tüm işlemler çalıştıktan sonra sanal makinenin anlık görüntüsü alınmış ve bellek dosyasının adı incelenmek üzere **XMRIG.vmem** olarak değiştirilmiştir.



Şekil 3.9: XMRig işlem ağaç yapısı.

3.6.2. Xmrig Analizi

XMRIG.vmem bellek görüntüsünde çalışan işlemler Volatility pstree eklentisi ile ağaç yapısında Şekil 3.10'daki gibi awk komutu ile filtrelenerek listelenmiştir. Filtreye SERVICES ve LSASS işlemleri WININIT'in alt işlemleri olduğu için eklenmiştir.

Tablo 4.1'de verilen işlemlerin üst-alt ilişkisine göre tüm WININIT işleminin alt işlemleri SERVICES ve LSASS'tir. Ayrıca WININIT işlemi sadece 1 adet olmalıdır. Bu bilgilere göre Şekil 3.10'a dikkat edildiğinde 2 adet WININIT işlemi olduğu ve bunlardan işaretlenmiş 2336 PID numaralı işlemin meşru bir işlem olmadığı anlaşılmıştır. Ayrıca işlemin kullanıcı (1 numaralı) oturumunda çalışması ve meşru WININIT işleminin alt işlemlerinden sonra

çalışmaya başlaması gibi uyumsuzluklar vardır. Bu durum 2336 PID numaralı WININIT işlemini şüpheli kılar.

```
(kali@kali) - [~/Downloads]
└─$ python3 volatility3/vol.py -f images/XMRIG.vmem windows.pstree | awk 'NR<5 || /System|smss|wininit|services|lsass/'
Volatility 3 Framework 1.0.1 PDB scanning finished
```

PID	PPID	ImageFileName	Offset(V)	Threads	Handles	SessionId	Wow64	CreateTime	ExitTime
4	0	System	0x898ffe364080 124	-	N/A	False	2021-04-22 14:28:20.000000	N/A	
* 328	4	smss.exe	0x898ffe364080	2	-	N/A	False	2021-04-22 14:28:20.000000	N/A
484	404	wininit.exe	0x898ffe364080	1	-	0	False	2021-04-22 14:28:22.000000	N/A
* 592	484	services.exe	0x898ffe364080	9	-	0	False	2021-04-22 14:28:22.000000	N/A
** 504	736	SystemSettings	0x898ffe364080	2	-	1	False	2021-04-22 14:30:43.000000	N/A
* 624	484	lsass.exe	0x898ffe364080	9	-	0	False	2021-04-22 14:28:22.000000	N/A
2336	2444	wininit.exe	0x898ffe364080	1	-	1	True	2021-04-27 14:23:02.000000	N/A

Şekil 3.10: XMRIG.vmem bellek görüntüsüne ait pstree eklenti sonuçları.

XMRIG.vmem bellek görüntüsünde çalışan işlemler Volatility pstree eklentisi ile bu sefer 2336 PID numaralı (şüpheli) WININIT işleminin çalışma tarihine göre Şekil 3.11'deki gibi filtrelenerek listelenmiştir. Şekil 3.11'e dikkat edildiğinde işaretli işlemlerin Windows sistem işlemi olmadığı isimlerinden anlaşılmıştır. Ayrıca birbirlerine çok yakın zamanlarda çalışmaya başlamış olmaları ve **5828** PID numaralı SVCHOST işlemi ile 2336 WININIT işleminin PPID numaralarının aynı olması bu işlemlerin birbirleri ile bağlantılı olabileceği şüphesini artırır.

```
(kali@kali) - [~/Downloads]
└─$ python3 volatility3/vol.py -f images/XMRIG.vmem windows.pstree | awk 'NR<5 || /2021-04-27/'
Volatility 3 Framework 1.0.1 PDB scanning finished
```

PID	PPID	ImageFileName	Offset(V)	Threads	Handles	SessionId	Wow64	CreateTime	ExitTime
1560	3548	XMRIG.exe	0x898ffe364080	0	-	1	True	2021-04-27 14:23:01.000000	2021-04-27 14:23:04.000000
* 4116	1560	DOC001.exe	0x898ffe364080	3	-	1	True	2021-04-27 14:23:04.000000	N/A
** 6320	4116	NsCpuCNMiner64	0x898ffe364080	0	-	1	False	2021-04-27 14:24:09.000000	2021-04-27 14:24:09.000000
** 6788	4116	cmd.exe	0x898ffe364080	1	-	1	True	2021-04-27 14:24:10.000000	N/A
** 3168	6788	conhost.exe	0x898ffe364080	3	-	1	False	2021-04-27 14:24:10.000000	N/A
** 6888	6788	net.exe	0x898ffe364080	5	-	1	True	2021-04-27 14:25:41.000000	N/A
2336	2444	wininit.exe	0x898ffe364080	1	-	1	True	2021-04-27 14:23:02.000000	N/A
5828	2444	svchost.exe	0x898ffe364080	1	-	1	True	2021-04-27 14:23:02.000000	N/A

Şekil 3.11: XMRIG.vmem bellek görüntüsüne ait pstree eklenti sonuçları.

XMRIG.vmem bellek görüntüsünde çalışan şüpheli işlemlerin komut satırı parametreleri Volatility cmdline eklentisi ile Şekil 3.12'deki gibi PID numaraları ile filtrelenerek listelenmiştir.

Şekil 3.12'ye dikkat edildiğinde 2336 PID numaralı WININIT ve 5828 PID numaralı SVCHOST işlemlerinin Tablo 4.1'de verilen gerçek konumlarında bulunmadığı, SVCHOST işleminin ise -k bayrağı ile başlatılmadığı anlaşılmıştır. Şekil 3.12'de işaretlenmiş 2336, 5828 ve 4116 PID numaralı işlemlerin bulunduğu klasörler bellekten hala okunabiliyorken 1560 ve 6320 PID numaralı işlemler sonlandığı için okunamamıştır. Bunun yanında 6788, 3168 ve 6888 PID numaralı şüpheli diğer işlemlerin hangi parametreler ile başlatıldığı Şekil 3.12'de

listelenmiştir. Bu tür işlemler Windows sistem klasörlerinde saklanan ve meşru işlemler olmasına rağmen görüldüğü üzere zararlı yazılımlar tarafından kötü amaçlarla kullanılabilir.

```
(kali@kali)-[~/Downloads]
└─$ python3 volatility3/vol.py -f images/XMRIG.vmem windows.cmdline --pid 1560 4116 6320 6788 3168 6888 2336 5828
Volatility 3 Framework 1.0.1
Progress: 100.00          PDB scanning finished
PID      Process Args
-----
1560     XMRIG.exe Required memory at 0x2bf020 is not valid (process exited?)
2336     wininit.exe c:\windows\wininit.exe
5828     svchost.exe c:\users\ali\appdata\local\svchost.exe
4116     DOC001.exe "C:\Users\ali\AppData\Roaming\Temp\DOC001.exe"
6320     NsCpuCNMiner64 Required memory at 0x3ac020 is not valid (process exited?)
6788     cmd.exe "C:\Windows\system32\cmd.exe" /v:on /c (for /f "usebackq tokens=1,*" %i in ('net view^|find /i "\\* ^|^| arp -a^|find /i " 1") do set str !random=%i!& for /f "usebackq tokens=1* delims=" %j in ('set str_') do set s=%k& set s=!s:\=!& set l=!s:-PC=!& set l=:!-IF=!& set f=DOC001.exe& if not "!s!"="%COMPUTERNAME%" (for /f "usebackq tokens=1,*" %j in ('net view \\s!^|find /i " " do echo f|xcopy /y /d "C:\Users\ali\AppData\Roaming\Temp\DOC001.exe" "\\s!\%j\DOC001.exe") & net use * /delete /y & (for %u in (1!! administrator user admin ääëïëñöäöäö) do @for %p in (0 " %u 1 123) do ping -n 3 localhost & (for %c in (\\s!\C$ \\s!\Users) do (if not "%p%"="01" net use %c "%p" /user:"%u") && ((for %d in ("%c\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup\!f!" "%c\Documents and Settings\%u\Start Menu\Programs\Startup\!f!" "%c\%u\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\!f!") do echo f|xcopy /y /d "C:\Users\ali\AppData\Roaming\Temp\DOC001.exe" %d) & net use %c /delete /y & ping -n 20 localhost)))
3168     conhost.exe \?C:\Windows\system32\conhost.exe 0x4
6888     net.exe net use \\192.168.65.2\C$ "123" /user:"1"
```

Şekil 3.12: XMRIG.vmem bellek görüntüsüne ait cmdline eklenti sonuçları.

XMRIG.vmem bellek görüntüsünde çalışan işlemler tarafından kullanılan dosya nesneleri Volatility filescan eklentisi ile Şekil 3.13'teki gibi listelenmiştir. Volatility cmdline eklentisi ile okunamayan işlemlerin bulunduğu klasörler filescan eklentisi ile şekildeki gibi okunabilmiştir.

```
(kali@kali)-[~/Downloads]
└─$ python3 volatility3/vol.py -f images/XMRIG.vmem windows.filescan | awk 'NR<5 || /XMRIG|DOC001|NsCpuCNMiner64|wininit|icsys/'
Volatility 3 Framework 1.0.1
PDB scanning finished

Offset Name Size
-----
0x898002dd9e20 \Windows\wininit.exe 216
0x898002fa0b50 \Users\ali\Desktop\XMRIG.exe 216
0x898002fa1190 \Users\ali\AppData\Roaming\Temp\DOC001.exe 216
0x898002fa7ef0 \Users\ali\AppData\Roaming\icsys.ico.exe 216
0x89800300d9b0 \Users\ali\AppData\Roaming\Temp\NsCpuCNMiner64.exe 216
0x89800346eb00 \DOC001.exe 216
0x898004890530 \Users\ali\Desktop\XMRIG.exe 216
0x898004be1c90 \Windows\wininit.exe 216
```

Şekil 3.13: XMRIG.vmem bellek görüntüsüne ait filescan eklenti sonuçları.

XMRIG.vmem bellek görüntüsünde çalışan işlemlerin muhtemel kod enjeksiyonu içeren bellek alanlarını gösteren Volatility malfind eklentisi ile Şekil 3.14'teki gibi PID numaraları ile filtrelenerek listelenmiştir.

Malfind eklentisi işlem belleğinde **0x950000** ve **0x2840000** adreslerinde başlayan iki adet kod enjekte edilmiş bellek alanı tespit etmiştir. Şekil 3.14'te gösterildiği üzere bellek koruması PAGE_EXECUTE_READWRITE olarak ayarlanmıştır. Bu durum 2336 ve 5828 PID numaralı işlemler üzerinde manipülasyon gerçekleştirildiğini gösterir.


```
(kali@kali)-[~/Downloads]
└─$ python3 volatility3/vol.py -f images/XMRIG.vmem windows.malfind --pid 1560 4116 6320 6788 3168 6888 2336 5828
Volatility 3 Framework 1.0.1
Progress: 100.00      PDB scanning finished
PID      Process Start VPN      End VPN Tag      Protection      CommitCharge      PrivateMemory      File output      Hexdump Disasm
-----
2336     wininit.exe      0x950000      0x950fff      VadS      PAGE_EXECUTE_READWRITE      1      1      Disabled
00 00 00 00 59 e9 d6 36 ...Y..6
b7 ff e8 f5 ff ff ff 00 .....
00 00 00 00 00 00 00 e8 .....
e8 ff ff ff 0a 00 95 00 .....
0x950000:      add      byte ptr [eax], al
0x950002:      add      byte ptr [eax], al
0x950004:      pop      ecx
0x950005:      jmp      0x4c36e0
0x95000a:      call    0x950004
5828     svchost.exe      0x284000      0x2840ff      VadS      PAGE_EXECUTE_READWRITE      1      1      Disabled
00 00 00 00 59 e9 d6 36 ...Y..6
c8 fd e8 f5 ff ff ff 00 .....
00 00 00 00 00 00 00 e8 .....
e8 ff ff ff 0a 00 84 02 .....
0x2840000:     add      byte ptr [eax], al
0x2840002:     add      byte ptr [eax], al
0x2840004:     pop      ecx
0x2840005:     jmp      0x4c36e0
0x284000a:     call    0x2840004
```

Şekil 3.14: XMRIG.vmem bellek görüntüsüne ait malfind eklenti sonuçları.

Volatility Procdump eklentisi ile 5828 ve 2336 PID numaralı işlemlerin çalıştırılabilir örneği Şekil 3.15'te gösterildiği gibi XMRIG.vmem bellek görüntüsünden dışarı aktarılmıştır.

```
(kali@kali)-[~/Downloads]
└─$ python2 volatility2/vol.py -f images/XMRIG.vmem --profile=Win10x64_19041 procdump --pid 5828 --dump-dir ./SONUC
Volatility Foundation Volatility Framework 2.6.1
Process(V)      ImageBase      Name      Result
-----
0xffff898003fcf080 0x00000000000400000 svchost.exe      OK: executable.5828.exe

(kali@kali)-[~/Downloads]
└─$ python2 volatility2/vol.py -f images/XMRIG.vmem --profile=Win10x64_19041 procdump --pid 2336 --dump-dir ./SONUC
Volatility Foundation Volatility Framework 2.6.1
Process(V)      ImageBase      Name      Result
-----
0xffff89800441a080 0x00000000000400000 wininit.exe      OK: executable.2336.exe
```

Şekil 3.15: XMRIG.vmem bellek görüntüsüne ait procdump eklenti sonuçları.

İki işlemde bulunan zararlı kod parçalarını doğrulamak için virustotal.com sitesine yüklenerek çevrimiçi tarama yapılmış ve 99 adet zararlı yazılım aracı tarafından tespit edilmiştir. Oluşturulan dosyalardan daha fazla bilgi elde etmek için statik ve dinamik analiz gibi tersine mühendislik teknikleri uygulanabilir.

4. BULGULAR

Bu bölümde Windows sistem işlemlerine, Sorebrect-XMRig zararlı yazılımlarına ve analiz sürecinde kullanılan Volatility Framework aracına ait bulgular sunulmuştur.

4.1.WINDOWS SİSTEM İŞLEMLERİ

Windows işletim sisteminde çalışan IDLE, SYSTEM, REGISTRY, Memory Compression, SMSS, CSRSS, WININIT, SERVICES, SVCHOST, LSASS, WINLOGON, LOGONUI, USERINIT ve EXPLORER sistem işlemlerine ait bulgular işletim sistemi henüz yalın iken; güncelleme yapılmadan, program kurulmadan ve zararlı yazılım çalıştırılmadan önce Process Hacker, Process Monitor ve Volatility Framework araçları ile elde edilmiş, Tablo 4.1’de toplu olarak verilmiştir. SMSS, LOGONUI ve USERINIT gibi sistem işlemleri başlatıldıktan saniyeler sonra sonlandığı için üst-alt ilişkilerinin çıkarılmasında Process Monitor aracına ait **Process Tree** ve **Enable Boot Logging** özelliklerinin büyük katkısı olmuştur.

IDLE minimal bir işlemdir. Çekirdek modunda çalışır. İşlem, kimlik numarası olarak 0 değerini alır ve alt bir işlem başlatmaz. İşletim sistemi sonlandırılana kadar çalışmaya devam eder ve çalışan örnek sayısı 1'den fazla olamaz.

SYSTEM minimal bir işlemdir. Çekirdek modunda çalışır. Çekirdek dosyasında (ntoskrnl.exe) yer alan işlem yöneticisi fonksiyonu tarafından oluşturulur. İşlem kimlik numarası her zaman sabittir (4) ve üç adet alt işlem çalıştırır. İşletim sistemi sonlandırılana kadar çalışmaya devam eder ve çalışan örnek sayısı 1'den fazla olamaz.

REGISTRY minimal bir işlemdir. Çekirdek modunda çalışır. SYSTEM işlemi tarafından başlatılır. İşlem, kimlik numarası olarak çeşitli değerler alır ve alt bir işlem başlatmaz. İşletim sistemi sonlandırılana kadar çalışmaya devam eder ve çalışan örnek sayısı 1'den fazla olamaz.

Memory Compression minimal bir işlemdir. Çekirdek modunda çalışır. SYSTEM işlemi tarafından başlatılır. İşlem, kimlik numarası olarak çeşitli değerler alır ve alt bir işlem başlatmaz. İşletim sistemi sonlandırılana kadar çalışmaya devam eder ve çalışan örnek sayısı 1'den fazla olamaz. Çekirdek modunda çalışan diğer minimal işlemlerden farklı olarak kullanıcı modu adres alanını kullanır.

SMSS korunan ve kritik olarak etiketlenmiştir. Kullanıcı modunda çalıştırılan ilk işlemdir. SYSTEM işlemi tarafından başlatılır. İşlem, kimlik numarası olarak çeşitli değerler alır ve işlemin çalıştırılabilir dosyası %SystemRoot%\System32\smss.exe konumundadır. İşletim sistemi sonlandırılana kadar çalışmaya devam eder ve çalışan örnek sayısı 1'den fazla olamaz.

CSRSS korunan ve kritik olarak etiketlenmiştir. Kullanıcı modunda çalışır. SMSS işleminin her örneği tarafından (her oturum için) 1 adet başlatılır. İşlem, kimlik numarası olarak çeşitli değerler alır ve alt bir işlem başlatmaz. İşlemin çalıştırılabilir dosyası %SystemRoot%\System32\csrss.exe konumundadır. Oturum sonlandırılana kadar çalışmaya devam eder ve çalışan örnek sayısı açık oturum sayısı kadardır.

WININIT korunan ve kritik olarak etiketlenmiştir. Kullanıcı modunda çalışır. SMSS işleminin servis oturumu örneği tarafından başlatılır. İşlem, kimlik numarası olarak çeşitli değerler alır ve işlemin çalıştırılabilir dosyası %SystemRoot%\System32\wininit.exe konumundadır. İşletim sistemi sonlandırılana kadar çalışmaya devam eder ve örnek sayısı 1'den fazla olamaz.

SERVICES korunan ve kritik olarak etiketlenmiştir. Kullanıcı modunda çalışır. WININIT işlemi tarafından başlatılır. İşlem, kimlik numarası olarak çeşitli değerler alır ve işlemin çalıştırılabilir dosyası %SystemRoot%\System32\services.exe konumundadır. İşletim sistemi sonlandırılana kadar çalışmaya devam eder ve çalışan örnek sayısı 1'den fazla olamaz.

SVCHOST kullanıcı modunda çalışır. SERVICES işlemi tarafından başlatılır. İşlem, kimlik numarası olarak çeşitli değerler alır ve işlemin çalıştırılabilir dosyası %SystemRoot%\System32\svchost.exe konumundadır. Çeşitli zamanlarda başlayıp sonlanabilirler ve çalışan örnek sayısı oldukça fazladır.

LSASS kullanıcı modunda çalışır. WININIT işlemi tarafından başlatılır. İşlem, kimlik numarası olarak çeşitli değerler alır ve alt bir işlem başlatmaz. İşlemin çalıştırılabilir dosyası %SystemRoot%\System32\lsass.exe konumundadır. İşletim sistemi sonlandırılana kadar çalışmaya devam eder ve çalışan örnek sayısı 1'den fazla olamaz.

WINLOGON kullanıcı modunda çalışır. SMSS işleminin her kullanıcı örneği tarafından (her etkileşimli kullanıcı oturumu için) 1 adet başlatılır. İşlem, kimlik numarası olarak çeşitli değerler alır ve işlemin çalıştırılabilir dosyası %SystemRoot%\System32\winlogon.exe

konumundadır. Oturum sonlandırılana kadar çalışmaya devam eder ve çalışan örnek sayısı açık kullanıcı oturum sayısı kadardır.

LOGONUI kullanıcı modunda çalışır. WINLOGON işlemi tarafından başlatılır. İşlem, kimlik numarası olarak çeşitli değerler alır ve işlemin çalıştırılabilir dosyası %SystemRoot%\System32\logonui.exe konumundadır. Kilit ekranına sahip bağlı veya oturum açmış her etkileşimli kullanıcı için 1 adet başlatılır, kimlik bilgileri girildiğinde veya oturum açma arayüzü kapatıldığında işlem sonlanır.

USERINIT kullanıcı modunda çalışır. WINLOGON işlemi tarafından başlatılır. İşlem, kimlik numarası olarak çeşitli değerler alır ve işlemin çalıştırılabilir dosyası %SystemRoot%\System32\userinit.exe konumundadır. Her etkileşimli kullanıcı oturumu için 1 adet başlatılır, görevini tamamladıktan sonra sonlanır.

EXPLORER kullanıcı modunda çalışır. USERINIT işlemi tarafından başlatılır. İşlem, kimlik numarası olarak çeşitli değerler alır ve işlemin çalıştırılabilir dosyası %SystemRoot%\explorer.exe konumundadır. Her etkileşimli kullanıcı oturumu için 1 adet başlatılır. Oturum sonlandırılana kadar çalışmaya devam eder ve çalışan örnek sayısı açık kullanıcı oturum sayısı kadardır.

Tablo 4.1: Windows sistem işlemlerinin sahip olduğu öznitelikler.

İşlem	Öznitelikler			
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
IDLE	-	0	LOCAL SYSTEM	0
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	-	0	-	1
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
SYSTEM	-	4	LOCAL SYSTEM	0
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	Registry Session Manager (smss.exe) Memory Compression	8	-	1
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
REGISTRY	SYSTEM	-	LOCAL SYSTEM	0
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	-	8	-	1
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
Memory Compression	SYSTEM	-	LOCAL SYSTEM	0
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	-	8	-	1
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
SMSS	SYSTEM	-	LOCAL SYSTEM	0
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	Auto Check Utility (autochk.exe) Session Manager (smss.exe)	11	%SystemRoot%\System32\smss.exe	1
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
SMSS Örneği	İlk SMSS işlemi	-	LOCAL SYSTEM	Her oturum
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	Client/Server Runtime Process (csrss.exe) Windows Initialization Process (wininit.exe) Windows Logon Application (winlogon.exe)	11	%SystemRoot%\System32\smss.exe	Her oturum için 1 adet başlatılır ve görevini tamamlayınca sonlanır
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum

Tablo 4.2: Windows sistem işlemlerinin sahip olduğu öznitelikler (devam).

İşlem	Öznitelikler			
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
CSRSS	Her oturum için başlatılan SMSS işlem örneği	-	LOCAL SYSTEM	Her oturum
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	-	13	%SystemRoot%\System32\csrss.exe	Her oturum için 1 adet
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
WININIT	Servis oturumu için başlatılan SMSS işlem örneği	-	LOCAL SYSTEM	0
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	Service Control Manager (services.exe) Local Security Authentication Subsystem Service (lsass.exe) Usermode Font Driver Host (fontdrvhost.exe)	13	%SystemRoot%\System32\wininit.exe	1
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
SERVICES	WININIT	-	LOCAL SYSTEM	0
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	Host Process for Windows Services (svchost.exe)	9	%SystemRoot%\System32\services.exe	1
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
SVCHOST	SERVICES	-	LOCAL SYSTEM LOCAL SERVICES NETWORK SERVICES LOCAL USERS	0
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	Çok sayıda olabilir	8	%SystemRoot%\System32\svchost.exe	Çok sayıda olabilir
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
LSASS	WININIT	-	LOCAL SYSTEM	0
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	-	9	%SystemRoot%\System32\lsass.exe	1
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum

Tablo 4.3: Windows sistem işlemlerinin sahip olduğu öznitelikler (devam).

İşlem	Öznitelikler			
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
WINLOGON	Oturum X için başlatılan SMSS işlem örneği	-	LOCAL SYSTEM	X
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	Usermode Font Driver Host (fontdrvhost.exe)			
	Windows Logon User Interface Host (logonui.exe)			
	Desktop Window Manager (dwm.exe)	13	%SystemRoot%\System32\winlogon.exe	Oturum X için 1 adet başlatılır
Userinit Logon Application (userinit.exe)				
Windows Logon Reminder (wlrmdr.exe)				
LOGONUI	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
	Oturum X için başlatılan WINLOGON	-	LOCAL SYSTEM	X
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	-	13	%SystemRoot%\System32\logonui.exe	Oturum X için 1 adet başlatılır ve görevini tamamlayınca sonlanır
USERINIT	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
	Oturum X için başlatılan WINLOGON	-	Oturum açan kullanıcı	X
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	Windows Explorer (explorer.exe)	13	%SystemRoot%\System32\userinit.exe	Oturum X için 1 adet başlatılır ve görevini tamamlayınca sonlanır

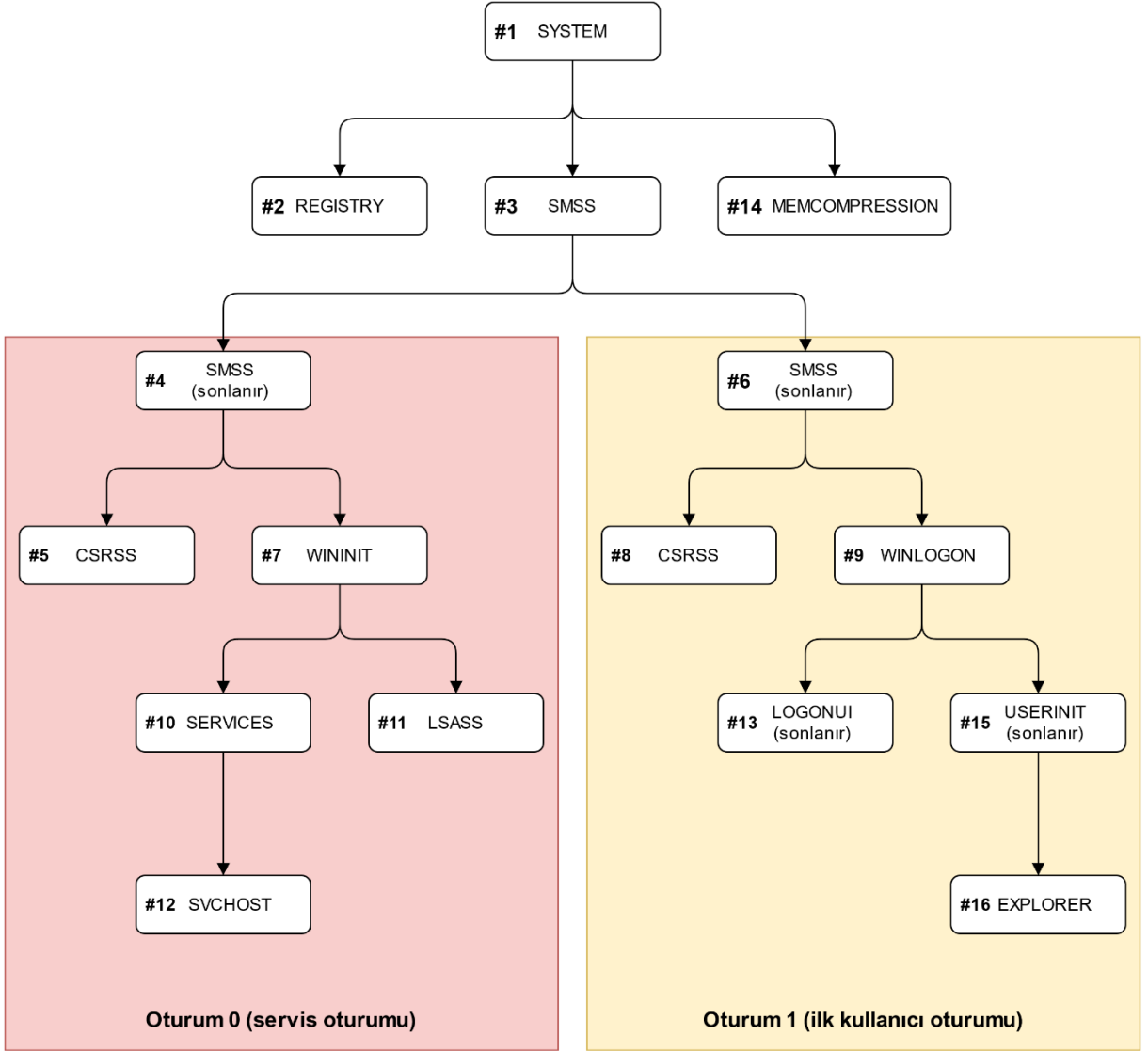
X: 1'den başlayarak artan şekilde devam eden etkileşimli kullanıcı oturum sayacı.

Tablo 4.4: Windows sistem işlemlerinin sahip olduğu öznitelikler (devam).

İşlem	Öznitelikler			
	Üst İşlem	Kimlik Numarası	Kullanıcı	Oturum
EXPLORER	Oturum X için başlatılan USERINIT	-	Oturum açan kullanıcı	X
	Alt İşlem	Temel Öncelik	Dosya Yolu	Adet
	Çok sayıda olabilir	8	%SystemRoot%\explorer.exe	Oturum X için 1 adet başlatılır

X: 1'den başlayarak artan şekilde devam eden etkileşimli kullanıcı oturum sayısı.

Çalışan bir Windows sisteminde en az iki oturum vardır: servis oturumu (oturum 0) ve ilk kullanıcı oturumu (oturum 1). Servis oturumu Windows sistem işlemlerinin ve işletim sistemi servislerinin başlatılmasını sağlayan kullanıcı etkileşimi gerektirmeyen bir oturumdur. İlk kullanıcı oturumu ise etkileşimli bir oturumdur. Her ikisi de ilk SMSS işleminin çoğaltılan örnekleri tarafından sistemin açılış zamanına çok yakın bir sürede oluşturulur (Rusinovich ve diğerleri, 2012b). Windows sistem işlemlerinin varsayılan üst-alt ilişkisi ve başlatılma sırası (#1 biçiminde) Şekil 4.1'de gösterilmiştir. Şekilde verilen işlem başlatılma sırası sistem açılışının performansına göre değişiklik gösterebilir. Ayrıca sistem açılış sürecinde çalışan tüm işlemlerin bu işlemlerden ibaret olmadığı unutulmamalıdır.



Şekil 4.1: Windows sistem işlemlerinin üst-alt ilişkisi ve başlatılma sırası.

Şekil 4.1’de gösterilen sistem açılışı sürecinde aşağıdaki işlemler sırasıyla gerçekleşir:

- SYSTEM, çekirdek dosyası tarafından başlatılır.
- SYSTEM, REGISTRY minimal işlemini başlatır. REGISTRY, kayıt defterinin bazı bölümlerini sistem belleğine yükler.
- SYSTEM, ilk SMSS işlemini başlatır. İlk SMSS, en az iki oturumu başlatmakla görevlidir.
- SMSS işlemi servis oturumu için kendi örneğini oluşturur.
- SMSS işleminin servis oturumu örneği, CSRSS işlemini servis oturumu için başlatır. CSRSS, servis oturumunda çalıştırılacak işlem ve iş parçacıklarının yönetiminden sorumludur.

- SMSS işlemi kullanıcı oturumu için kendi örneğini oluşturur. Ek bir kullanıcı oturum açması durumunda SMSS işlemi kendisine ait yeni bir kullanıcı örneği daha oluşturur ve bu örnek (alt işlemleri de dahil) ilk kullanıcı oturumunda olduğu gibi çalışır.
- SMSS işleminin servis oturumu örneği, WININIT işlemi başlatır ve sonlanır. SMSS örneğinin sonlanması sebebiyle canlı analiz araçlarında, çalışan işlemler üst-alt ilişkisine göre listelendiğinde WININIT'in üst işlemi yokmuş gibi görünecektir. Bu durum üst işlemi sonlandırılan diğer işlemler için de (CSRSS, WINLOGON, EXPLORER vs.) geçerlidir. WININIT, genel olarak bazı ortam değişkenlerinin başlatılması, pencere istasyonu ve masaüstü oluşturulması gibi sistem ayarlarının yapılmasını sağlar.
- SMSS işleminin kullanıcı oturumu örneği, CSRSS işlemi kullanıcı oturumu için başlatır. CSRSS, kullanıcı oturumunda çalıştırılacak işlem ve iş parçacıklarının yönetiminden sorumludur.
- SMSS işleminin kullanıcı oturumu örneği, WINLOGON işlemi başlatır ve sonlanır. WINLOGON, kullanıcı oturum açma işlemlerini ve kullanıcı güvenliğini yönetir, gerektiği durumlarda (SAS tuşlanması gibi) LOGONUI işlemi başlatır.
- WININIT, SERVICES işlemi başlatır. SERVICES, aygıt sürücülerini ve Windows servislerini başlatmaktan sorumludur.
- WININIT, LSASS işlemi başlatır. LSASS, kullanıcı kimliğinin doğrulanmasını sağlar.
- SERVICES, SVCHOST işlemi başlatır. SVCHOST, kayıt defteri anahtarında bulunan servislerin başlatılmasını sağlar.
- WINLOGON, LOGONUI işlemi başlatır. LOGONUI, etkileşimli kullanıcı girişi için bir arayüz sunar ve kullanıcı girişi başarıyla gerçekleştirildikten sonra sonlanır.
- SYSTEM, MEMCOMPRESSION minimal işlemi başlatır. MEMCOMPRESSION, az kullanılan bellek sayfalarını sıkıştırarak yine bellek üzerinde depolar ve sistemin performansını artırır.
- WINLOGON, USERINIT işlemi başlatır. USERINIT, kayıt defterinde kayıtlı bulunan kabuk değerini başlatmaktan sorumludur.
- USERINIT, EXPLORER işlemi başlatır ve sonlanır.

4.2.SOREBRECT

Sorebrect dosyasız fidye yazılımı sanal makinede çalıştırıldıktan sonra oluşturulan SOREBRECT.vmem bellek görüntüsü incelendiğinde SVCHOST işleminin SERVICES işlemi tarafından başlatılmadığı ve kullanıcı oturumunda çalıştığı Volatility pstree eklentisi ile tespit edilmiştir. Bu işlemin -k bayrağı olmadan başlatıldığı Volatility cmdline eklentisi ile listelenmiş ve işleme ait bellek alanlarındaki muhtemel kod enjeksiyonları Volatility malfind eklentisi ile bulunmuştur. Zararlı kod içeren bellek alanları Volatility procdump ve vaddump eklentileri ile dışarı aktarılmıştır. Bellek alanları virustotal.com taramasından geçirildiğinde 35 adet araç tarafından zararlı olarak işaretlenmiştir.

4.3.XMRIG

XMRig kripto para madenci yazılımı sanal makinede çalıştırıldıktan sonra oluşturulan XMRig.vmem bellek görüntüsü incelendiğinde birden fazla WININIT işleminin çalıştığı listelenmiş ve meşru olmayan WININIT işlemi Volatility pstree eklentisi ile tespit edilmiştir. Bu işlemin başlangıç zamanına yakın işlemler filtrelendiğinde XMRIG.exe, DOC001.exe, NsCpuCNMiner.exe ve svchost.exe işlemleri görülmüştür. WININIT ve SVCHOST işlemlerinin başlangıç parametrelerinin meşru işlemlerden farklı olduğu Volatility cmdline eklentisi ile listelenmiş ve diğer işlemlerin bulunduğu klasörler filescan eklentisi ile tespit edilmiştir. İşlemlere ait bellek alanlarındaki muhtemel kod enjeksiyonları Volatility malfind eklentisi ile bulunmuş ve zararlı kod içeren bellek alanları Volatility procdump eklentisi ile dışarı aktarılmıştır. Bellek alanları virustotal.com taramasından geçirildiğinde 99 adet araç tarafından zararlı olarak işaretlenmiştir.

4.4.VOLATILITY FRAMEWORK

Yapılan analizlerde Volatility yazılım çerçevesinde mümkün olduğunca Volatility 3 eklentileri kullanılmıştır. Volatility 3'te plist, pstree, cmdline, filescan ve malfind eklentilerinin daha doğru sonuçlar ürettiği ancak procdump ve vaddump eklentilerinin dosyaları dışarı aktaramadığı görülmüştür. Bu yüzden zararlı yazılımların çalıştırılabilir örnekleri ve bellek alanları Volatility 2'nin procdump ve vaddump eklentileri ile dışarı aktarılmıştır.

5. TARTIŞMA VE SONUÇ

RAM’de çalışan her işlemin çalıştırılabilir bir dosyaya sahip olmadığı; IDLE, SYSTEM, REGISTRY ve Memory Compression gibi minimal işlemlerde görülmüştür. Ayrıca SOREBRECT zararlı yazılımı çalıştıktan sonra kendisine ait exe dosyasını silerek mevcut işleme enjekte olmuştur. Bu durum çalışma anında RAM’de bulunan tüm işlemlerin çalıştırılabilir bir dosyaya sahip olma şartı aranmadan **DLL dosyasından başlatılmış olsalar bile** incelenmesi gerektiğini gösterir.

Windows sistem işlemlerine ait Şekil 4.1’de verilen üst-alt ilişkisinin ve başlatılma sırasının bilinmesi analiz edilen her iki zararlı yazılımda da şüpheli işlemlerin tespit edilmesini **en kolaylaştıran** yöntem olduğu görülmüştür. Sorebrect zararlı yazılımı SVCHOST işlemini manipüle ederek sistemdeki zararlı faaliyetlerini sürdürmüştür, SVCHOST işlemi sistem normallerinde sadece SERVICES işlemi tarafından başlatılması gerekir ancak Sorebrect bu işlemi kendisi başlatmıştır. Benzer şekilde XMRig zararlı yazılımı WININIT ve SVCHOST işlemlerini manipüle etmiştir; WININIT işlemi sistem normallerinde sadece 1 adet olmalı ve LSASS, SERVICES alt işlemlerini başlatmış olmalıdır ancak XMRig meşru olan WININIT işlemi (kritik olarak etiketlenmesi sebebiyle) sonlandıramamış ve sistemde gizlenebilmek için aynı isimde yeni bir işlem başlatarak zararlı faaliyetlerine bunun üzerinden devam etmiştir.

Bu araştırmada en iyi bilinmesi gereken ve zararlı yazılımlar tarafından araştırmanın gerçekleştirildiği tarihe kadar en çok manipüle edilen veya gelecekte manipüle edilme olasılığı en yüksek olan 14 adet Windows sistem işlemi (IDLE, SYSTEM, REGISTRY, Memory Compression, SMSS, CSRSS, WININIT, SERVICES, SVCHOST, LSASS, WINLOGON, LOGONUI, USERINIT ve EXPLORER) incelenmiş ve mümkün olan en derin şekilde ayrıntılarına inilmiştir. Ancak Windows’un tüm sistem işlemleri bu işlemlerden **ibaret değildir** ve bu işlemler sabit kalmayacaktır. Çünkü Windows sistem işlemleri tıpkı günlük işlerimizde kullandığımız diğer yazılımlar gibi zamanla güncellenir. Bunun başlıca sebepleri işletim sistemi güvenliğinin artırılması, işlemlerin görev dağılımında değişiklikler yapılması veya yeni ihtiyaçlara çözüm üretilmesidir. Bu sebeple gelecek çalışmalarda (Tablo 4.1’de bazılarının isimleri zaten verilmiştir); Interrupts, Secure System, Auto Check Utility (autochk.exe), Desktop Window Manager (dwm.exe), Windows Logon Reminder (wlrmrdr.exe), Console Window Host (Conhost.exe), Usermode Font Driver Host (fontdrvhost.exe), Host Process for

Windows Tasks (taskhostw.exe), COM Surrogate (dllHost.exe), Runtime Broker (runtimebroker.exe) gibi ve zamanla eklenecek olan diğerk Windows işlemlerinin de ayrıntılı araştırması yapılmalıdır.

Araştırma sonucunda elde edilen bulgularda Windows sistem işlemlerinin özellikleri Volatility yazılım çerçevesinin her iki sürümünde bulunan eklentilerle de elde edilememiştir. Bu durum dikkate alındığında gelecek çalışmalarda bu araştırma sonucunda elde edilen tüm bulguların Volatility yazılım çerçevesine uygun **bir eklenti geliştirilmesi** sistem normalinin dışına çıkan tüm durumların tespit edilmesini hem kolaylaştıracak hem de bellek görüntüsünün incelenme süresini önemli bir ölçüde kısaltacaktır. Bu eklenti geliştirilirken işletim sistemi güncellemelerinde Windows sistem işlemlerinde yapılan belirgin değişiklikler dikkate alınmalıdır. Bu durumda geliştirilmesi planlanan eklenti bu araştırmanın aksine birden fazla işletim sistemi sürümünü desteklemelidir.

Araştırma süresince Volatility yazılım çerçevesinin her iki sürümüne de (Volatility 2 ve Volatility 3) ihtiyaç duyulmuştur. Volatility 3 henüz geliştirme aşamasında olması sebebiyle tüm eklentileri henüz çalışmamaktadır. Bu yüzden bellek görüntüsü incelemelerinde her iki sürümün de kurulumları eksiksiz yapılmalı ve incelemeyi daha güvenilir kılmak için her iki sürümden aynı eklentilerin **sonuçları karşılaştırılmalıdır**.

KAYNAKLAR

- AH-FAT, P., HUTH, M., MEAD, R., BURRELL, T. & NEIL, J. Effective Detection of Credential Thefts from Windows Memory: Learning Access Behaviours to Local Security Authority Subsystem Service. 23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020), 2020. 181-194.
- ANSON, S. 2020. *Applied Incident Response*, Wiley.
- BALOGH, Š. & MOJŽIŠ, J. New direction for malware detection using system features. 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2019. IEEE, 176-183.
- BAUM, J. B. 2014. *Windows memory forensic data visualization*.
- BOLAT, Y. 2015. *Memory Forensics*. İstanbul Bilgi Üniversitesi.
- BRENDMO, H. K. 2017. *Live forensics on the Windows 10 secure kernel*. NTNU.
- CASE, A. 2020. *Public Beta: Insider's Preview* [Çevrimiçi]. The Volatility Foundation. Bağlantı: <https://www.volexity.com/wp-content/uploads/2020/11/Volexity-Cyber-Session-April-2020-Volatility3-Public-Beta.pdf> [Erişim 2021].
- CASE, A. & RICHARD, G. G. 2017. Memory forensics: The path forward. *Digital Investigation*, 20, 23 - 33.
- ECEMIŞ, A. 2018. *Dosya enjeksiyon zararlılarının tespiti ve analizi için bir algoritma önerisi*. Süleyman Demirel Üniversitesi.
- FOSSEN, J. 2012. *Windows Exploratory Surgery With Process Hacker* [Çevrimiçi]. Bağlantı: https://web.archive.org/web/20151113022252/http://blogs.sans.org/windows-security/files/Process_Hacker_SANS_Jason_Fossen.pdf [Erişim 2021].
- FOUNDATION, T. V. 2021a. *An advanced memory forensics framework* [Çevrimiçi]. The Volatility Foundation. Bağlantı: <https://github.com/volatilityfoundation/volatility> [Erişim 2021].
- FOUNDATION, T. V. 2021b. *Volatility 3.0 development* [Çevrimiçi]. The Volatility Foundation. Bağlantı: <https://github.com/volatilityfoundation/volatility3> [Erişim 2021].
- GOKTEPE, M. 2002. *Windows XP Operating System Security Analysis*. NAVAL POSTGRADUATE SCHOOL MONTEREY CA.
- HENRY, C. 2017. *Survey and Use of Ten Volatility Framework Plugins for Malware Analysis*. Utica College.
- KEIJZER, N. 2020. *The new generation of ransomware: an in depth study of Ransomware-as-a-Service*. University of Twente.
- KORKIN, I. Protected Process Light is not Protected: MemoryRanger Fills The Gap Again. 2021 IEEE Security and Privacy Workshops (SPW), 2021. IEEE, 298-308.
- KORKMAZ, Y. 2015. *Hedefli saldırılarda kullanılan zararlı yazılımların makine öğrenimi kullanılarak tespiti ve sınıflandırılması*. Bilkent University.
- KROMBHOLZ, K., HOBEL, H., HUBER, M. & WEIPPL, E. 2015. Advanced social engineering attacks. *Journal of Information Security and applications*, 22, 113-122.
- LIGH, M. 2020. *2.6 Win Profiles* [Çevrimiçi]. Bağlantı: <https://github.com/volatilityfoundation/volatility/wiki/2.6-Win-Profiles> [Erişim 18.03.2021].
- LIGH, M., ADAIR, S., HARTSTEIN, B. & RICHARD, M. 2010. *Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code*, Wiley Publishing.
- LIGH, M. H., CASE, A., LEVY, J. & WALTERS, A. 2014. *The art of memory forensics: detecting malware and threats in windows, linux, and Mac memory*, John Wiley & Sons.

- LIU, W. J. 2021. *Process Hacker* [Çevrimiçi]. Bağlantı: <https://processhacker.sourceforge.io/> [Erişim 2021].
- MAJOR, M. 2015. *A Taxonomic Evaluation of Rootkit Deployment, Behavior and Detection*. University of Idaho.
- MONNAPPA, K. 2018. *Learning Malware Analysis: Explore the concepts, tools, and techniques to analyze and investigate Windows malware*, Packt Publishing Ltd.
- MURRAY, R. 2016. *MemTri: A memory forensics triage tool using bayesian network and volatility*. University of Westminster.
- ØSTERUD, A. 2018. *Windows 10 Memory Compression in Digital Forensics-Uncovering Digital Evidence in Compressed Swap*. NTNU.
- PÉK, G. 2015. *New methods for detecting malware infections and new attacks against hardware virtualization*.
- PETRONI, N. L., WALTERS, A., FRASER, T. & ARBAUGH, W. A. 2006. FATKit: A framework for the extraction and analysis of digital forensic data from volatile system memory. *Digital Investigation*, 3, 197 - 210.
- PRAKASH, A., VENKATARAMANI, E., YIN, H. & LIN, Z. 2015. On the Trustworthiness of Memory Analysis An Empirical Study from the Perspective of Binary Execution. *IEEE Transactions on Dependable and Secure Computing*, 12, 557-570.
- REHMAN, Z. U., AHMAD, A. & SALEEM, S. 2017. A Brief Survey of Memory Analysis Tools. *NUST Journal of Engineering Sciences*, 10, 57-64.
- RUFF, N. 2008. Windows memory forensics. *Journal in Computer Virology*, 4, 83-100.
- RUSSINOVICH, M. 2021a. *Process Monitor* [Çevrimiçi]. Bağlantı: <https://docs.microsoft.com/en-us/sysinternals/downloads/procmon> [Erişim 2021].
- RUSSINOVICH, M. 2021b. *PsExec v2.33* [Çevrimiçi]. Bağlantı: <https://docs.microsoft.com/en-us/sysinternals/downloads/psexec> [Erişim 2021].
- RUSSINOVICH, M. E., SOLOMON, D. A. & IONESCU, A. 2012a. *Windows Internals, Part 1*, Microsoft Press.
- RUSSINOVICH, M. E., SOLOMON, D. A. & IONESCU, A. 2012b. *Windows Internals, Part 2*, Microsoft Press.
- SIHWAIL, R., OMAR, K. & ARIFFIN, K. Z. 2018. A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis. *International Journal on Advanced Science, Engineering and Information Technology*, 8, 1662.
- SILBERSCHATZ, A., GALVIN, P. B. & GAGNE, G. 2010. *Operating system concepts with Java*, John Wiley & Sons Software.
- SIMPSON, D. 2017. *Changes to Service Host grouping in Windows 10* [Çevrimiçi]. Bağlantı: <https://docs.microsoft.com/en-us/windows/application-management/svchost-service-refactoring> [Erişim 2021].
- STATCOUNTER. 2020a. *Desktop Operating System Market Share Worldwide* [Çevrimiçi]. Bağlantı: <https://gs.statcounter.com/os-market-share/desktop/worldwide#monthly-202001-202012-bar> [Erişim 09.03.2021].
- STATCOUNTER. 2020b. *Desktop Windows Version Market Share Worldwide* [Çevrimiçi]. Bağlantı: <https://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide#monthly-202001-202012-bar> [Erişim 09.03.2021].
- SÜZEN, A. A. 2018. *Adli bilişim için ram imajı alınarak elektronik delil elde etme*. Süleyman Demirel Üniversitesi.
- TAHİLLİOĞLU, E. 2020. *Bilgisayarlı görü ve bellek analizinden yararlanılarak zararlı yazılım tespiti için gelişmiş bir yaklaşım*. Hacettepe Üniversitesi.

- TANCIO, B. & YANEZA, J. 2017. *Analyzing the Fileless SOREBRECT Ransomware* [Çevrimiçi]. Bağlantı: https://www.trendmicro.com/en_us/research/17/f/analyzing-fileless-code-injecting-sorebrect-ransomware.html [Erişim 2021].
- TANENBAUM, A. S. & BOS, H. 2015. *Modern operating systems*, Pearson.
- TORPROJECT. 2019. *Tor Network* [Çevrimiçi]. Bağlantı: <https://2019.www.torproject.org/about/overview.html.en> [Erişim 2021].
- VIRUSTOTAL. 2015. *Xmrig* [Çevrimiçi]. Virustotal. Bağlantı: <https://www.virustotal.com/gui/file/91c6e3cbe1c061a3dfd15ef47894436a6fc4341078a566e38131a6d9b655edac/details> [Erişim 2021].
- VIRUSTOTAL. 2017. *Sorebrect* [Çevrimiçi]. Virustotal. Bağlantı: <https://www.virustotal.com/gui/file/4142ff4667f5b9986888bdc2a727db6a767f78fe1d5d4ae3346365a1d70eb76/details> [Erişim 2021].
- VIVIANO, A. 2017. *User mode and kernel mode* [Çevrimiçi]. Bağlantı: <https://docs.microsoft.com/tr-tr/windows-hardware/drivers/gettingstarted/user-mode-and-kernel-mode> [Erişim 2021].
- VON NEUMANN, J. 1993. First Draft of a Report on the EDVAC. *IEEE Annals of the History of Computing*, 15, 27-75.
- WHITE, A. J. 2013. *Identifying the unknown in user space memory*. Queensland University of Technology.
- WIKIPEDIA. 2021a. *Process (computing)* [Çevrimiçi]. Bağlantı: [https://en.wikipedia.org/wiki/Process_\(computing\)](https://en.wikipedia.org/wiki/Process_(computing)) [Erişim 2021].
- WIKIPEDIA. 2021b. *VMware Workstation Pro* [Çevrimiçi]. Bağlantı: https://en.wikipedia.org/wiki/VMware_Workstation [Erişim 2021].
- YOSIFOVICH, P., RUSSINOVICH, M. E., IONESCU, A. & SOLOMON, D. A. 2017. *Windows Internals, Part 1: System architecture, processes, threads, memory management, and more*, Microsoft Press.
- YÜCEL, Ç. 2019. *Zararlı yazılımlar için bellek erişimlerinin görüntülenmesi ve değerlendirilmesi*. Yaşar Üniversitesi.
- ZAPATA, M. A. 2016. *Indicators of compromise in the analysis of system processes*. Utica College.
- ZHANG, S., WANG, L. & ZHANG, L. Extracting windows registry information from physical memory. 2011 3rd International Conference on Computer Research and Development, 2011. IEEE, 85-89.

EKLER

-

