5

CYSA+

DUYGU KAÇAR

- Uç Nokta İzleme(Endpoint Monitoring)

İçindekiler

1. Endpoint Analysis 3
1.1. Uç Nokta Analizi Nedir? 3
1.2. Uç Nokta Güvenlik Araçları: 3
Gelişen Teknolojiler: 4
2. Sandboxing 5
3. Reverse Engineering 6
Kötü Amaçlı Yazılım Analizinde Tersine Mühendislik 7
Statik Analiz ve Disassembly 7
Makine Kodu ve Assembly Kodu 7
Dosya İmzaları ve Sihirli Sayılar 8
Kod Çözme (Decompilation) ve Yüksek Seviye Diller 8
Dizeler (Strings) ve İmza Tabanlı Tespit 8
Paketleyiciler (Packers) ve Obfuscation 8
4. Malware Exploitation 8
5. Behavior Analysis 10
6. Malware Analysis demo 13
7. EDR Configuration 13
8. Block Lists and Allow Lists 23

1. Endpoint Analysis

1.1. Uç Nokta Analizi Nedir?

- Uç nokta analizi, bir organizasyonun ağındaki cihazların (uç noktaların) güvenlik durumunu izlemek, kayıt altına almak ve analiz etmek için kullanılan bir süreçtir.

- Uç nokta: Ağımıza bağlanabilen herhangi bir cihazdır (örneğin, masaüstü bilgisayarlar, dizüstü bilgisayarlar, akıllı telefonlar, tabletler). Bu cihazlar, genellikle saldırganların hedefi haline gelir ve bu nedenle uç nokta güvenliği kritik öneme sahiptir.

Siber Güvenlik Analisti Olarak Uç Nokta Analizinde Yapılması Gerekenler:

- Davranışsal Anomalileri Tanımlamak: Araçlar kullanarak normalden sapma gösteren davranışları tespit etmek. Bu, uç noktalarda meydana gelebilecek olası güvenlik ihlallerini erken tespit etmek açısından önemlidir.

- Kötü Amaçlı Yazılımların Tespiti: Kötü amaçlı yazılımların ayrıcalık elde etme, yükseltme ve kalıcılık sağlama tekniklerini tanımlamak. Örneğin, bir zararlı yazılım sistemde yönetici ayrıcalıkları elde etmeye çalışabilir.

1.2. Uç Nokta Güvenlik Araçları:

- Antivirüs:
    - **İşlevi:** Virüsleri, solucanları, Truva atlarını, kök kitlerini, reklam yazılımlarını, casus yazılımları ve diğer kötü amaçlı yazılım türlerini tespit eder ve ortadan kaldırır.
    - **Çalışma Prensibi:** Genellikle imza tabanlı tespit kullanılır. İmza tabanlı tespit, kötü amaçlı yazılımların bilinen davranış kalıplarını (imzalarını) tanıyarak çalışır.
    - **Gelişmiş Özellikler:** Modern antivirüs yazılımları, makine öğrenimi ve davranışsal analiz kullanarak bilinmeyen tehditleri de tespit edebilir.

- HIDS/HIPS (Host-Based Intrusion Detection/Prevention Systems):
    - **İşlevi:** Belirli bir uç noktadaki sistem durumunu izleyerek beklenmeyen davranışları ve sistemdeki köklü değişiklikleri tespit eder.
    - **Çalışma Prensibi**: İmza tabanlı tespit, dosya sistemi bütünlüğü izleme, günlük dosyalarının izlenmesi gibi yöntemler kullanılır.
    - **Örnek Kullanım:** İşletim sistemi dosyalarının değiştirilip değiştirilmediğini veya sürücülerde izinsiz değişiklik yapılıp yapılmadığını kontrol eder.
    - **Tespit Edebilecekleri:** Rootkit'ler, işletim sistemi üzerinde yapılan yetkisiz değişiklikler, beklenmedik süreçler.

- EPP (Endpoint Protection Platform):

    - İşlevi: Birden fazla güvenlik görevini yerine getiren entegre bir yazılım ajanı ve izleme sistemidir.
    - **Çalışma Prensibi:** Antivirüs, HIDS/HIPS, güvenlik duvarı, veri kaybı önleme (DLP), dosya şifreleme gibi çeşitli güvenlik özelliklerini tek bir platformda birleştirir.
    - **Gelişmiş Özellikler:** Birçok EPP, merkezi yönetim imkanı sağlar, böylece güvenlik politikaları tüm uç noktalara tek bir yönetim konsolu üzerinden uygulanabilir.

- EDR (Endpoint Detection and Response):
    - **İşlevi:** Davranışsal ve anomali analizi odaklıdır. Uç noktaların gözlemlenebilir olaylarını (indicators of compromise - IoC) ve loglarını toplayarak tehditlerin erken tespitini sağlar.
    - **Çalışma Prensibi:** EDR sistemleri, uç noktada gerçekleşen olayları kaydeder ve bu kayıtları analiz ederek potansiyel tehditleri tespit eder. Davranış analizi ve makine öğrenimi teknikleri kullanılır.
    - **Özellikler:** Tehditlerin geçmişe dönük izlenebilirliği ve bir ihlalin gerçekleştiği durumda hızlı bir yanıt ve olay müdahalesi sağlar. Bu sistemler, uzaktan erişim ile zararlı yazılımların analiz edilmesine ve sistemlerin temizlenmesine olanak tanır.

- UEBA (User and Entity Behavior Analytics):
    - **İşlevi:** Kullanıcı hesapları ve cihazlar tarafından gerçekleştirilen şüpheli etkinlikleri tanımlar.
    - **Çalışma Prensibi:** Normal kullanıcı davranışlarının bir taban çizgisi oluşturulur ve bu taban çizgisinin dışına çıkan davranışlar şüpheli olarak işaretlenir. Bu, makine öğrenimi ve yapay zeka kullanarak gerçekleştirilir.
    - **Kapsam:** Yalnızca uç nokta veri toplama değil, aynı zamanda kullanıcı ve varlıkların genel davranış analizi de yapılır. Bu sistemler, çok büyük miktarda veriyi işleyebilmek için ileri hesaplama tekniklerine bağımlıdır.
    - **Önemli Özellikler:** Özellikle iç tehditlerin tespitinde kullanışlıdır. İç tehditlerin davranışları genellikle dış tehditlere göre daha belirsiz olabilir, bu nedenle UEBA'nin güçlü analiz yetenekleri gereklidir.

Piyasadaki Önemli Oyuncular:

- Antivirüs ve EPP: Microsoft, CrowdStrike, Symantec gibi önde gelen firmalar, uç nokta koruma platformları (EPP) sunmaktadır. Bu platformlar, Gartner'ın Magic Quadrant raporlarına göre değerlendirilmekte ve sıralanmaktadır.

- UEBA: Microsoft Advanced Threat Analytics ve Splunk User Behavior Analytics, UEBA çözümleri sunan önemli firmalardır. Bu araçlar, gelişmiş analiz ve gösterge tablosu yetenekleriyle tehditlerin tespit edilmesini ve yönetilmesini sağlar.

Gelişen Teknolojiler:

- Güvenlik teknolojileri sürekli olarak gelişmekte ve daha entegre çözümler ortaya çıkmaktadır:

- Gelişmiş Tehdit Koruması (ATP): Yeni nesil tehditlere karşı daha sofistike koruma sağlayan sistemler.
- Gelişmiş Uç Nokta Koruması (AEP): EPP ve EDR gibi teknolojilerin birleşimiyle daha kapsamlı koruma sağlar.
- NextGen AV (NGAV): Geleneksel antivirüs çözümlerinin ötesine geçerek, makine öğrenimi ve davranış analizi ile geliştirilmiş kötü amaçlı yazılım tespiti sunar.

Bu teknolojiler, uç nokta güvenliğinin sağlanmasında kritik rol oynar ve siber güvenlik analistlerinin tehditlere karşı proaktif bir yaklaşım geliştirmelerine yardımcı olur.

2. Sandboxing

Günümüzde, imza tabanlı güvenlik araçları, kötü amaçlı yazılımların otomatik olarak engellenmesi konusunda giderek daha az etkili hale geliyor. Bunun sebebi, kötü amaçlı yazılım yaratıcılarının tekniklerini geliştirerek kötü niyetlerini daha iyi gizleyebilmeleridir. Bu durum, kötü amaçlı yazılım analistlerinin de tekniklerini geliştirmeye ve manuel analiz yeteneklerini artırmaya zorlamaktadır.

Sandboxing, kötü amaçlı yazılımların kontrollü bir ortamda yürütülmesini sağlayan ve bu süreçte sistemin güvenliğini garanti altına alan bir tekniktir. Bu ortam, ana bilgisayar sisteminden izole edilmiştir ve sandbox ile ana bilgisayar arasında herhangi bir iletişim bağlantısı tamamen yasaktır. Çoğunlukla, bir sandbox ortamı sanal makine (VM) veya VirtualBox gibi sanallaştırma araçlarıyla oluşturulur.

Bu teknikte, kötü amaçlı yazılımı izole bir sanal ortamda çalıştırarak, kötü amaçlı yazılımın sistemde yapacağı değişiklikler gözlemlenir. Bu değişiklikler arasında kayıt defteri değişiklikleri, yeni dosyaların oluşturulması gibi işlemler bulunur. Sandboxing, bir dosyanın gerçekten kötü amaçlı olup olmadığını belirlememize ve sistem üzerinde yaratacağı etkileri anlamamıza olanak tanır. Ayrıca, kötü amaçlı yazılımın bağımlılıklarını ve bu bağımlılıkların ana bilgisayarda nasıl yer aldığını da tanımlayabiliriz.

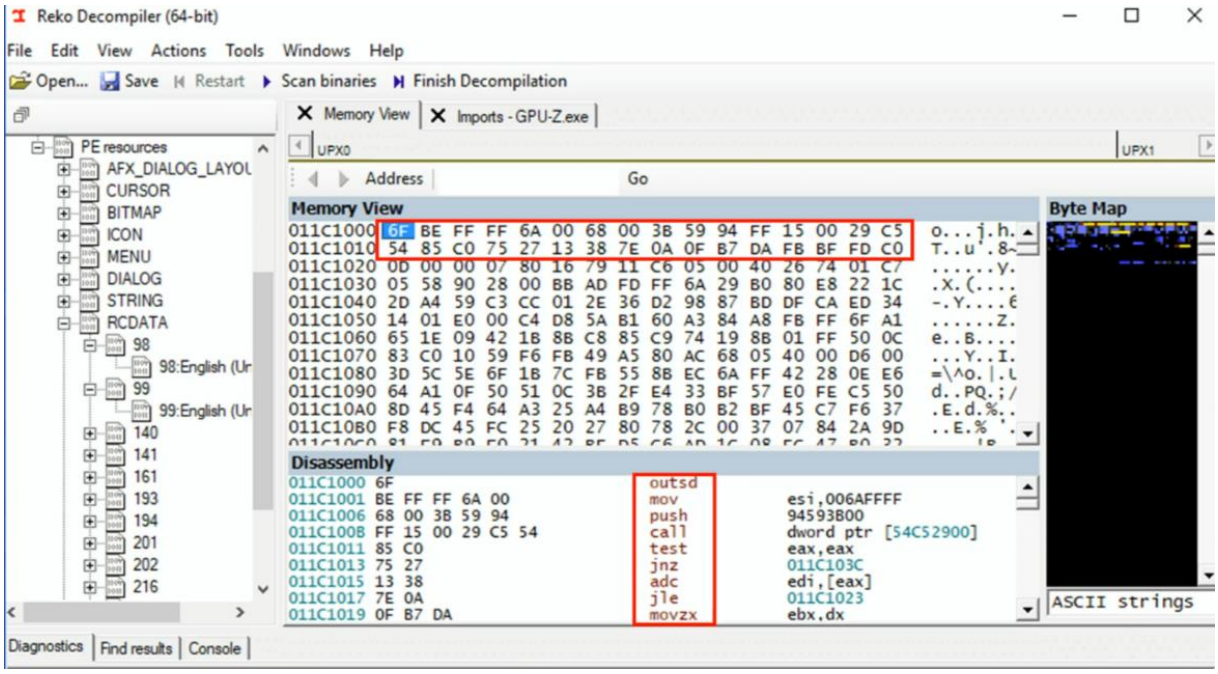
Kötü amaçlı yazılım analizinde sandboxing kullanarak, kötü amaçlı yazılımı çeşitli ortam ve işletim sistemlerinde, temel ana bilgisayara zarar vermeden test edebiliriz. Bu süreçte, sistem değişikliklerini, ağ bağlantılarını ve sistem çağrılarını izleyebiliriz. Ayrıca, sanal belleği boşaltarak kötü amaçlı yazılımın şifrelenmemiş veya paketlenmemiş sürümünü analiz edebiliriz.

FLARE VM, Cuckoo ve Joe Sandbox gibi araçlar, sandboxing sürecinde yaygın olarak kullanılır. Örneğin, FLARE VM, bir Windows 10 üzerinde çalışan ve kötü amaçlı yazılımın yarattığı değişiklikleri gözlemlememize olanak tanıyan ücretsiz bir programdır. Cuckoo ise birden fazla işletim sisteminde otomatik olarak kötü amaçlı yazılım çalıştıran ve analiz eden bir sandbox programıdır. Joe Sandbox ise kötü amaçlı yazılımları analiz etme sürecini otomatikleştirir ve bir güvenlik araştırmacısının kötü amaçlı yazılımın davranışını anlamasına yardımcı olur.

Bu araçlar, kötü amaçlı yazılım analistlerine, sistem değişikliklerini izleme, ağ bağlantılarını gözlemleme ve zararlı yazılımları analiz etme konusunda büyük kolaylıklar sağlar.

3. Reverse Engineering

Tersine mühendislik (Reverse Engineering), bir cihazın, sistemin veya yazılımın iç yapısını anlamak amacıyla onu analiz etme sürecidir. Yazılım dünyasında tersine mühendislik, genellikle mevcut bir yazılımı inceleyerek nasıl çalıştığını, hangi bileşenlere sahip olduğunu ve bu bileşenlerin birbirleriyle nasıl etkileştiğini anlamayı içerir. Bu süreç, özellikle kötü amaçlı yazılımların analizi sırasında son derece önemlidir.



Kötü Amaçlı Yazılım Analizinde Tersine Mühendislik

Kötü amaçlı yazılımlar (malware), bilgisayar sistemlerine zarar vermek veya izinsiz erişim sağlamak amacıyla tasarlanmış yazılımlardır. Bu yazılımların nasıl çalıştığını anlamak için tersine mühendislik teknikleri kullanılır. Kötü amaçlı yazılımların dinamik analiz (sandboxing) ve statik analiz yoluyla incelenmesi, bu yazılımların etkilerini ve amaçlarını belirlemek için önemli yöntemlerdir.

Statik Analiz ve Disassembly

Tersine mühendislik sırasında kullanılan tekniklerden biri statik analizdir. Statik analiz, kötü amaçlı yazılımın kodunu çalıştırmadan, doğrudan ikili dosya üzerinde yapılan analizdir. Bu süreçte, bir "disassembler" kullanılarak makine dili (binary code) assembly diline dönüştürülür. Assembly dili, düşük seviyeli bir dildir ve CPU tarafından doğrudan yürütülebilen komutlardan oluşur.

Makine Kodu ve Assembly Kodu

Makine kodu, işlemcinin doğrudan çalıştırabileceği ikili komutlardır ve genellikle onaltılık (hexadecimal) formatta ifade edilir. Bu komutlar, bellek adreslerine erişim, veri transferi ve matematiksel işlemler gibi temel işlemleri içerir. Assembly kodu ise makine kodunun insan tarafından okunabilir hale getirilmiş halidir. Örneğin, "MOV EAX, 1" komutu, EAX register'ına 1 değerinin atanmasını belirtir. Tersine mühendisler, makine kodunu assembly koduna dönüştürerek kötü amaçlı yazılımın işlevlerini analiz eder.

Dosya İmzaları ve Sihirli Sayılar

Bir dosyanın türünü belirlemek için dosya imzaları veya "sihirli sayılar" kullanılır. Sihirli sayılar, dosyanın başında yer alan belirli bir dizi onaltılık kodlardır ve dosya türünü tanımlar. Örneğin, Windows taşınabilir yürütülebilir dosyaları (PE dosyaları), genellikle "4D 5A" onaltılık kodlarıyla başlar. Bu, dosyanın bir EXE veya DLL dosyası olduğunu gösterir.

Kod Çözme (Decompilation) ve Yüksek Seviye Diller

Disassembly işlemi, makine kodunu assembly diline dönüştürmekle sınırlı kalmaz; bazen daha yüksek seviyeli dillere geri döndürme işlemi de yapılabilir. Kod çözme (decompilation), bir ikili dosyanın orijinal yüksek seviye kaynak koduna (C, Java vb.) geri döndürülmesi sürecidir. Örneğin, Java tabanlı kötü amaçlı yazılımlar, Java kod çözücüler (decompilers) kullanılarak oldukça okunabilir hale getirilebilir.

Dizeler (Strings) ve İmza Tabanlı Tespit

Kötü amaçlı yazılım analizinde kullanılan bir başka teknik ise dizelerin (strings) çıkarılmasıdır. Dizeler, kötü amaçlı yazılımın içindeki metin parçalarıdır ve genellikle kullanıcı adları, dosya yolları, URL'ler veya işlev çağrıları gibi önemli bilgiler içerir. Örneğin, bir kötü amaçlı yazılım, "InternetOpenUrl" işlevi ve bir URL içeriyorsa, bu, kötü amaçlı yazılımın bir web kaynağından veri indirmeye çalıştığını gösterebilir. Dizelerin tespiti, imza tabanlı kötü amaçlı yazılım tespiti için kritik öneme sahiptir.

Paketleyiciler (Packers) ve Obfuscation

Kötü amaçlı yazılım yazarları, analiz sürecini zorlaştırmak için genellikle paketleyiciler (packers) kullanır. Paketleyiciler, bir yürütülebilir dosyayı sıkıştırarak veya şifreleyerek, analiz edilmesini zorlaştırır. Bu teknik, kötü amaçlı yazılımın kodunu gizler ve tersine mühendislerin işini zorlaştırır. Paketlenmiş kötü amaçlı yazılımların analiz edilebilmesi için önce bu paketleme işleminin tersine çevrilmesi, yani dosyanın paketinin açılması (unpacking) gereklidir.

4. Malware Exploitation (Kötü Amaçlı Yazılım İstismarı)

Bu başlık altında , kötü amaçlı yazılımların nasıl istismar edildiği ve modern saldırı tekniklerinin nasıl çalıştığı üzerinde durulacaktır.

1. İstismar Teknikleri ve Kötü Amaçlı Yazılımın Bulaşma Yöntemleri

- İstismar Tekniği: Kötü amaçlı yazılım kodunun, hedeflenen bir ana bilgisayara bulaşmasını sağlayan belirli yöntemlerdir. Eski yöntemlerde, kötü amaçlı yazılımlar, hedef diskteki yürütülebilir dosyaları veya makro dosyalarını yeniden yazarak çalıştırıldığında virüsün yüklenmesini sağlardı. Modern teknikler ise dosyasız istismarları içerir.

- Dosyasız Teknikler: Modern kötü amaçlı yazılımlar, antivirüs ve ana bilgisayar tabanlı saldırı tespit sistemlerinden kaçınmak için dosyasız teknikler kullanır. Bu teknikler, kötü amaçlı yazılımın doğrudan bir komut dosyası olarak veya sistem belleğinde bir işlem oluşturan küçük bir kabuk kodu olarak çalıştırılması anlamına gelir. Bu tür yazılımlar, genellikle geçici bir dizinde çalıştırıldıktan sonra kendilerini silerek izlerini kaybettirirler.

2. APT'ler ve Damlalık (Dropper) Kullanımı

- İleri Seviye Sürekli Tehditler (APT): Modern kötü amaçlı yazılımlar, APT'ler tarafından kullanılır. APT'ler, hedef sistemlere dosyasız bir şekilde sızmak için damlalık (dropper) veya indirici (downloader) kullanır. Damlalıklar, küçük ve hafızada kolayca çalışabilen dosyalardır. Bu damlalıklar, sistemde hafif kabuk kodu çalıştırarak, ikinci aşama indiriciyi indirir ve böylece kötü amaçlı yazılımın bilgisayara bulaşmasını sağlar.

- Erişim Sağlama: APT'ler, genellikle kullanıcıyı bir dosyaya tıklamaya veya kodu çalıştırmaya ikna eder. Bu, sisteme ilk sızma aşamasıdır ve ardından ikinci aşama indirici, uzaktan erişim Truva atı (RAT) gibi zararlı yazılımları indirerek saldırganın sistemdeki kontrolünü sağlar.

3. Kötü Amaçlı Yazılımın İstismar Adımları

1. **Dropper ve Downloader (Damlalık ve İndirici):** Kötü amaçlı yazılımın ilk aşaması, damlalık ve indirici kullanarak sisteme bulaşmayı içerir. Damlalık, küçük bir kod parçası ile zararlı yazılımı sisteme yerleştirir. Ardından indirici, uzaktan erişim Truva atı (RAT) gibi ek zararlı yazılımları indirir ve bu, saldırganın sistemi kontrol etmesini sağlar.

2. **Erişimi Sürdürme (Maintain Access):** Kötü amaçlı yazılım sisteme bulaştıktan sonra saldırgan, erişimi sürdürmek için gerekli araçları yerleştirir. Bu aşamada, saldırganın sisteme tekrar erişim sağlamasına yönelik adımlar atılır.

3. **Erişimi Güçlendirme (Strengthen Access):** Saldırgan, yanal hareket yaparak diğer sistemlere de bulaşmaya çalışır. Bu aşamada, genellikle sunucular veya etki alanı denetleyicileri gibi yüksek değerli hedefler seçilir. Yanal hareket, saldırganın ek ayrıcalıklar elde etmesini ve sistemdeki varlığını güçlendirmesini sağlar.

4. **Hedeflere Yönelik Eylemler (Actions on Objectives):** Saldırgan, hedef sistemlerde belirli eylemleri gerçekleştirmeye başlar. Bu eylemler, dosyaların çalınması, şifrelenmesi veya başka zararlı işlemler olabilir.

5. **Gizlenme (Concealment):** Saldırgan, sistemdeki izlerini silerek ve günlük dosyalarını temizleyerek varlığını gizlemeye çalışır. Anti-adli teknikler kullanarak, saldırgan sistemdeki varlığını uzun süre koruyabilir.

4. Teknik İstismar Yöntemleri

- Kod Enjeksiyonu: Kötü amaçlı kod, meşru bir sürecin kimlik numarası ile çalıştırılarak sisteme bulaştırılır. Bu, saldırganın kötü amaçlı yazılımı gizlemesine olanak tanır.
- Masquerading: Damlalık, meşru bir yürütülebilir dosyayı kötü amaçlı bir dosya ile değiştirir.
- DLL Enjeksiyonu: Damlalık, bir işlemi kötü amaçlı bir DLL dosyasını yüklemeye zorlar.
- DLL Sideloading: Damlalık, meşru bir programın zafiyetini kullanarak kötü amaçlı bir DLL yükler.
- Process Hollowing: Damlalık, bir işlemi askıya alır ve bellek konumlarını kötü amaçlı yazılımla yeniden yazar.

5. Anti-Adli Teknikler ve Tespit Zorluğu

- **Anti-Adli Teknikler:** Kötü amaçlı yazılımlar, yüklerini şifrelemek, sıkıştırmak veya gizlemek gibi yöntemlerle tespit edilmelerini zorlaştırır. Saldırganlar, sistemdeki varlıklarını daha uzun süre koruyabilmek için bu teknikleri kullanırlar.
- **Topraktan Geçinmek (Living off the Land):** Saldırganlar, sistemde zaten mevcut olan araçları (örneğin PowerShell veya Bash) kullanarak izinsiz girişlerini gerçekleştirirler. Bu yöntem, saldırganların tespit edilmesini zorlaştırır çünkü kullanılan araçlar, sistemde zaten meşru olarak bulunmaktadır.

5. Davranış Analizi

Önceki başlıkta kabuk kodu hakkında konuştuk ve bu kodu gizlemenin gerçekten çok kolay olduğunu belirtmiştik. Bir saldırgan olarak kabuk kodunu imza tabanlı antivirüs algılamasından kaçınmak için kullanabilirsiniz. Bu nedenle, tehdit avcılığı ve güvenlik izleme, enfeksiyonları tanımlamak için davranışa dayalı teknikler kullanmalıdır.

Bunu gerçekleştirmek için birçok araç bulunmaktadır ve en yaygın olarak kullanılanlardan biri, Sysinternals adlı bir araç paketidir. Sysinternals, Windows ile ilgili sorunları gidermeye yardımcı olmak amacıyla sistem yöneticileri için tasarlanmıştır, ancak bu araçların çoğu güvenlik sorunlarını araştırmak için de oldukça uygundur. Bu araçlar, Microsoft'un web sitesinde ücretsiz olarak sunulmaktadır ve araç takımlarının bir parçasıdır.

Sysinternals araçları, normalin ne olduğunu belirlemek için bir temel oluşturmamıza olanak tanır. Bu araçlardan biri olan Process Explorer, bir Windows sisteminde hangi süreçlerin çalıştığını anlamamıza yardımcı olur. Process Explorer, meşru etkinlikleri filtreleyebilir ve bu sayede anormal davranışları hızlı bir şekilde tanımlayabiliriz. İşlem Gezgini ile bir sistemin temelini oluşturup, ardından bir sandbox'ta kötü amaçlı yazılım çalıştırarak, bu temel ile mevcut durumu karşılaştırabiliriz. Bu da, kötü amaçlı yazılımın ne yaptığını belirlememize yardımcı olur. Eğer zaten bir sistem için iyi bir temeliniz varsa, bu araçları kullanarak enfekte olduğunu düşündüğünüz bir üretim sisteminde değişiklik olup olmadığını karşılaştırabilirsiniz. Şüpheli bir alan varsa, bu alanı daha fazla araştırmak isteyebilirsiniz. Bu, tehdit avcılığı ve olay müdahalesi yaparken gerçekten değerli bir yöntemdir.

Bu süreçte temel konsept, önce meşru süreçlerin ne olduğunu anlamak, böylece şüpheli olanları tanımlayabilmektir. Şimdi, tipik bir Windows sisteminde hangi süreçlerin meşru kabul edildiğini ve bunların dışındaki herhangi bir şeyin şüpheli olarak etiketlenmesi gerektiğini ele alalım.

İlk olarak, sistemin PID 0 olan Sistem Boşta Süreci ve PID 4 olan Sistem Süreci, her zaman bu PID'lere sahip olacaktır ve bu süreçler meşru olarak kabul edilir. Bunlar, ilk kullanıcı modu işlemi olan Oturum Yöneticisi Alt Sistemi'ni (smss.exe) başlatan çekirdek düzeyinde ikili dosyalardır. Process Explorer veya Windows'taki İşlemci sekmesine baktığınızda, bu iki sürecin her zaman orada olduğunu göreceksiniz.

Bir diğer önemli süreç ise csrss.exe olan İstemci Sunucusu Çalışma Zamanı Alt Sistemi'dir. Bu süreç, düşük seviyeli Windows işlevlerini yönetir ve birden fazla örneği çalışabilir. Bu süreçlerin %SystemRoot%\System32 dizininden çalıştırıldığından ve bir ebeveyni olmadığından emin olunmalıdır. Eğer bu süreçler ebeveynli olarak görünüyorsa, kötü amaçlı yazılım olma ihtimalleri vardır.

winnit.exe, hizmetleri ve sürücüleri yöneten bir süreçtir ve yalnızca tek bir işlem olarak çalışmalıdır. Eğer Process Explorer'da birden fazla örneğini görürseniz, bu bir sorun olabilir. Services.exe ise arka plan hizmetlerini yöneten bir süreçtir ve genellikle kötü amaçlı yazılımlar tarafından taklit edilmeye çalışılır. Bu sürecin yalnızca bir örneği çalışmalı ve WINNIT'in bir çocuğu olmalıdır.

lsass.exe (Yerel Güvenlik Yetkilisi Alt Sistemi) ve winlogon.exe (WINLOGON) gibi süreçler de, belirli ebeveyn-çocuk ilişkilerine sahip olmalıdır. lsass.exe, winnit.exe'nin bir çocuğu olmalıdır, aksi halde şüpheli olabilir. winlogon.exe, kullanıcı masaüstüne erişimi yönetir ve yalnızca bir örneği olmalıdır.

userinit.exe, oturum açma işlemi sırasında explorer.exe'yi başlatan bir süreçtir ve genellikle oturum açma işlemi sırasında kısaca görünmelidir. Eğer bu süreç, oturum açtıktan uzun süre sonra hala çalışıyorsa, bu şüpheli bir durum olarak değerlendirilmelidir.

explorer.exe, tipik bir kullanıcı kabuğudur ve kullanıcı tarafından başlatılan tüm işlemler için ebeveyn olmalıdır. Eğer explorer.exe dışında başka bir süreç, oturum açmış bir kullanıcı tarafından başlatılmış gibi görünüyorsa, bu şüpheli bir durumdur.

Bir sürecin şüpheli olup olmadığını belirlemek için sekiz farklı yol vardır:

1. Tanınmayan bir işlem adı şüpheli olabilir. Bu durumda, internete bakarak bu ismin bilinen bir işlem olup olmadığını kontrol etmelisiniz.
2. Meşru bir sisteme çok benzeyen ama hafifçe farklı görünen işlem adları (örneğin, scvhost vs. svchost) şüpheli olabilir.
3. Rastgele oluşturulmuş gibi görünen karışık bir adı olan işlemler şüphelidir.
4. Simge, sürüm bilgisi, açıklama veya şirket adı olmayan süreçler şüpheli olabilir.
5. Microsoft gibi tanınmış bir şirketten gelen ancak imzasız bir işlem şüpheli olabilir.
6. Dijital imzası olan ancak tanımlanan yayımcıyla eşleşmeyen bir işlem şüpheli olabilir.
7. Temel bir Windows işlemine ait olmayan bir ebeveyn-çocuk ilişkisi şüpheli olabilir.
8. Paketlenmiş veya sıkıştırılmış bir süreç şüpheli olabilir.

Şüpheli bir işlem belirledikten sonra, bu işlemle ilgili şu sorulara yanıt aramalısınız:

1. İşlem, kayıt defteriyle veya dosya sistemiyle nasıl etkileşime giriyor?
2. Bu süreci kim başlattı?
3. Bu işlem sistem klasöründen mi yoksa geçici bir klasörden mi başlatıldı?
4. Bu süreç hangi dosyaları manipüle ediyor?
5. Bu işlem, sistem yeniden başlatıldığında kendini geri yüklüyor mu?
6. Bu süreç bir sistem ayrıcalığı yapıyor mu veya hizmet engelleniyor mu?
7. Bu işlem ağ ile etkileşime giriyor mu?

Bu soruların yanıtları, bir işlemin şüpheli olup olmadığını ve daha fazla araştırılması gerekip gerekmediğini belirlemenize yardımcı olacaktır.

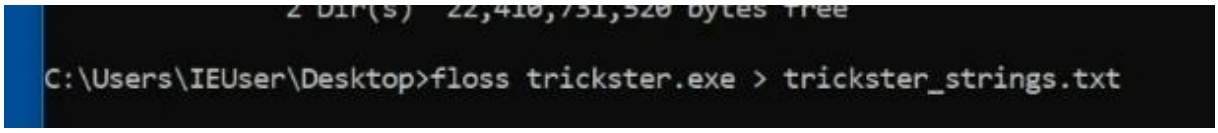
Makine öğrenimi ve yapay zeka gibi teknolojiler, bu tür analizleri otomatikleştirebilir, ancak bir analist olarak bu süreçleri manuel olarak da yapabilmeniz önemlidir. Bu, neye baktığınızı anlamanıza ve güvenlik tehditlerini etkili bir şekilde tespit etmenize olanak tanır.

6. Malware Analysis demo

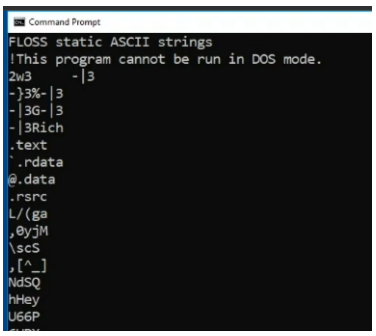
Bu bölümde zararlı yazılımın nasıl yapıldığını ve analiz edildiğini adım adım inceleyeceğiz. Bu süreç, tersine mühendislerin kullandığı araçlar ve tekniklerin bir tanıtımını içeriyor.

Statik Analiz

İlk olarak, dinamik analize geçmeden önce, zararlı yazılımın statik analizini yapmamız gerekiyor. Bu işlem için komut istemini açıyoruz. Komut isteminde dizinleri masaüstüne değiştirdikten sonra, masaüstünde bulunan Trickster adlı zararlı yazılımı çıkarmamız gerekiyor. Bu işlem için 7Zip kullanarak dosyayı çıkardık . Trickster'ı masaüstünde erişilebilir hale getirdik.



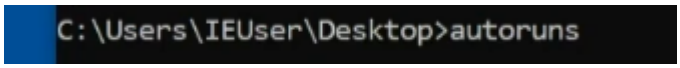
Ardından, Floss adlı bir program kullanarak Trickster'ın içindeki ASCII dizilerini çıkarmak amacıyla ikili dosyayı analiz ettik. Bu işlem, trickster.exe dosyasındaki ASCII karakterine benzeyen metinleri çıkartarak, trickster_strings.txt adlı bir metin dosyasına aktardı.



Metin dosyasını inceleyerek, zararlı yazılımın şifreli olduğunu ve statik analizle IP adresleri, ana bilgisayar adları, e-posta adresleri gibi anlamlı bilgiler elde edilemediğini fark ettik. Bu, zararlı yazılımın şifreleme kullanarak gerçek kaynağını gizlediğini gösteriyordu.

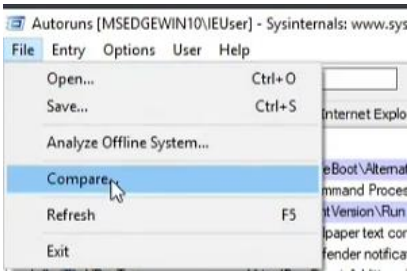
Dinamik Analiz

Trickster'ın şifreli olması nedeniyle, dinamik analiz yöntemlerine geçtik. Bu aşamada, zararlı yazılımın çalıştırılması ve sistem üzerinde yaptığı değişikliklerin izlenmesi amaçlanıyor. İlk adım olarak, sistemimizin bilinen iyi bir temelini oluşturduk. Bu temel, sistemin kötü amaçlı yazılım bulaşmamış durumdaki halini gösteriyor.



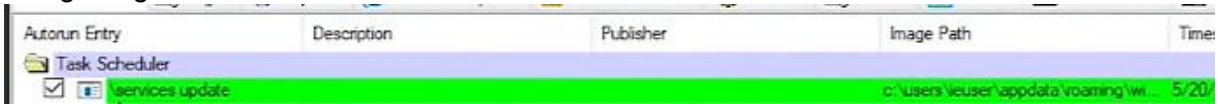
Bu işlemi gerçekleştirmek için Otomatik Çalıştırmalar (Autoruns) adlı bir program kullandık ve sistemi tarayarak, sistem bileşenlerinin mevcut durumunu kaydettik. Bu temel dosyayı "baseline.autorun" olarak masaüstüne kaydettik.

Ardından, "Trickster" adlı zararlı yazılımı çalıştıracağız. Bunu yaparken yönetici olarak çalıştırmak için sağ tıklayın ve "Yönetici olarak çalıştır" seçeneğini seçin. Bu işlem, makinenize Trickster'ı bulaştıracaktır. Onaylamak için "Evet" tuşuna basın ve Trickster çalışmaya başlayacak.



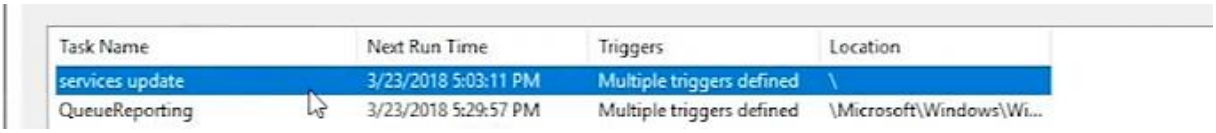
Şimdi, zararlının sistemde neler yaptığını görmek için Autoruns tekrar açacağız. Yenile düğmesine tıklayarak sistemi tekrar tarayın. Bu işlem, sistemde nelerin değiştiğini tespit edecektir. Ardından, mevcut sistem durumu ile kaydettiğimiz temel dosya olan "Baseline.arn" arasında karşılaştırma yapacağız. Bu karşılaştırma, Trickster'ın yaptığı değişiklikleri ortaya çıkaracaktır.

Karşılaştırmanın ardından, Trickster'ın tek yaptığı şeyin bir görev zamanlayıcı eklemek olduğunu göreceksiniz.

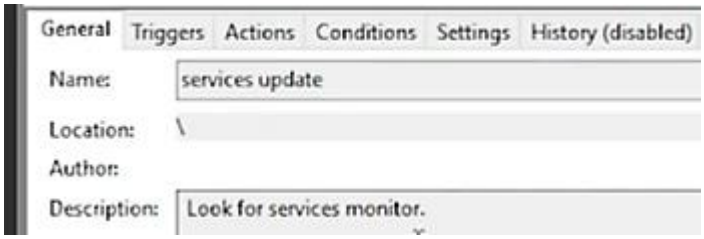


Bu görev, "Hizmet Güncellemesi" adında ve "C:\Users\ieuser\AppData\Roaming\winappusjdltufs.exe" dosyasını çalıştırıyor. Dosyanın ismi oldukça garip olduğundan, bu dosyaya kısaca "u.exe" olarak atıfta bulunacağız.

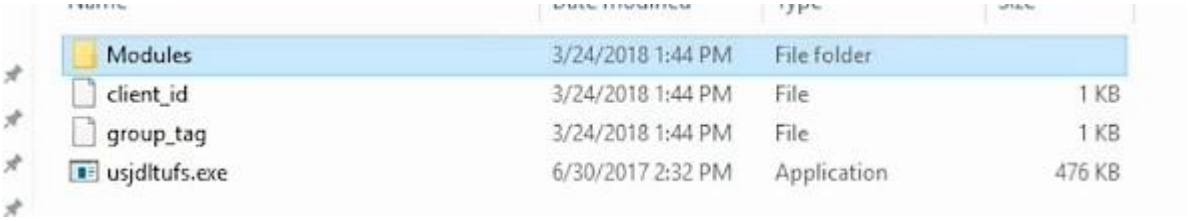
Görev zamanlayıcısında bu görevi inceleyin. Bu görev, belirli tetikleyicilere bağlı olarak çalıştırılacak. Örneğin, bir kullanıcı sisteme giriş yaptığında veya her gün belirli bir saatte bu görev aktif hale gelecektir. Ayrıca, her üç dakikada bir çalışacak şekilde ayarlanmıştır. Ancak, bu dosyanın ne yaptığını henüz bilmiyoruz.



Trickster'ın her üç dakikada bir "u.exe" dosyasını çalıştırmasını istediğini biliyoruz. Tahminimize göre, bu dosya bir komuta ve kontrol sunucusuna bağlanarak makineyi bulaştırdığını ve emirler için hazır olduğunu bildiriyor olabilir.



Zamanlayıcı tetiklendikten sonra, görev arka planda çalıştırılacak ve bu işlemi görmek için süreç izleyicisi veya ağ trafiği analiz aracını kullanmamız gerekecek. Sisteminize bulaşan dosyanın izlerini takip etmek için önce C sürücüsündeki "u.exe" dosyasını bulmamız gerekiyor. Bu dosya, kullanıcının Uygulama Verileri klasöründe yer alıyor ve makinenize ne gibi zararlar verebileceğini tespit etmek için analiz edilecek.



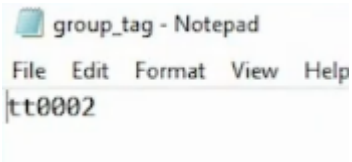
Zararlı yazılımın birinci aşama bir damlalık (dropper) olduğu ve daha fazla kötü amaçlı yazılım indirmediği tespit edildi. Ancak, bu dosyanın kötü niyetli olabileceğini göz ardı etmemek gerekir.

Kararlaştırılan yöntem ne olursa olsun, burada iki metin dosyası bulunuyor: istemci kimliği ve grup etiketi.

Öncelikle müşteri kimliğini açacağım ve Not Defteri ile inceleyeceğim. İstemci kimliği, bu

benim ana bilgisayar adım olan 'MSEdgeWin10'dur. Bu, benzersiz bir tanımlayıcı veya seri numarasıdır ve kötü amaçlı yazılımın yaptığı da budur. Bu yazılım, tüm bu makinelere bağlanacak ve bu bilgilerle bir şeyler yapmaya çalışacaktır. Eğer bu makineleri bir botnet'e dahil ediyorsa, hangi makinelerin ne olduğunu bilmesi gerekir. Fidye yazılımı yapacaksa, kim olduğunuzu bilmesi gerekir ki ödeme yaptığınızda makinenizi serbest bırakabilsin veya

bırakmasın; bu, saldırganın işini nasıl yapacağına bağlıdır. Ama yine de, sizi tanımlamanın bir yoluna ihtiyaçları var ve bu da istemci kimliğidir. Bu sadece bir seri numarasıdır.



Ardından, bir de grup etiketi var. Grup etiketini Not Defteri'nde açtığımızda, sadece 'TT0002' dediğini görüyoruz. Bu, bu grup için sadece bir alfa sayısal koddur; makinenizin bu gruba dahil olduğunu gösterir. Eğer bir botnet'in parçası olacaksanız, belki 100 makineyi, 1.000 makineyi veya bir seferde 10.000 makineyi bir DDoS saldırısı gerçekleştirmek için satmak isteyebilirsiniz. Bu makineleri bir araya getirmenin bir yoluna ihtiyaç vardır ve bu grup etiketi de bu amaca hizmet eder.

Bu, bu iki dosya ve modülün sabit diskimize yaptığı işlemdir, ancak henüz kötü niyetli bir şey yapmamıştır çünkü bu, sadece birinci aşama dropper'dır.

Yapılan analiz sonucu, üç farklı uzlaşma göstergesi tespit edildi:

1. Trickster.exe dosyasının hash değeri

2. u.exe dosyasının oluşturulması ve yerleştirilmesi

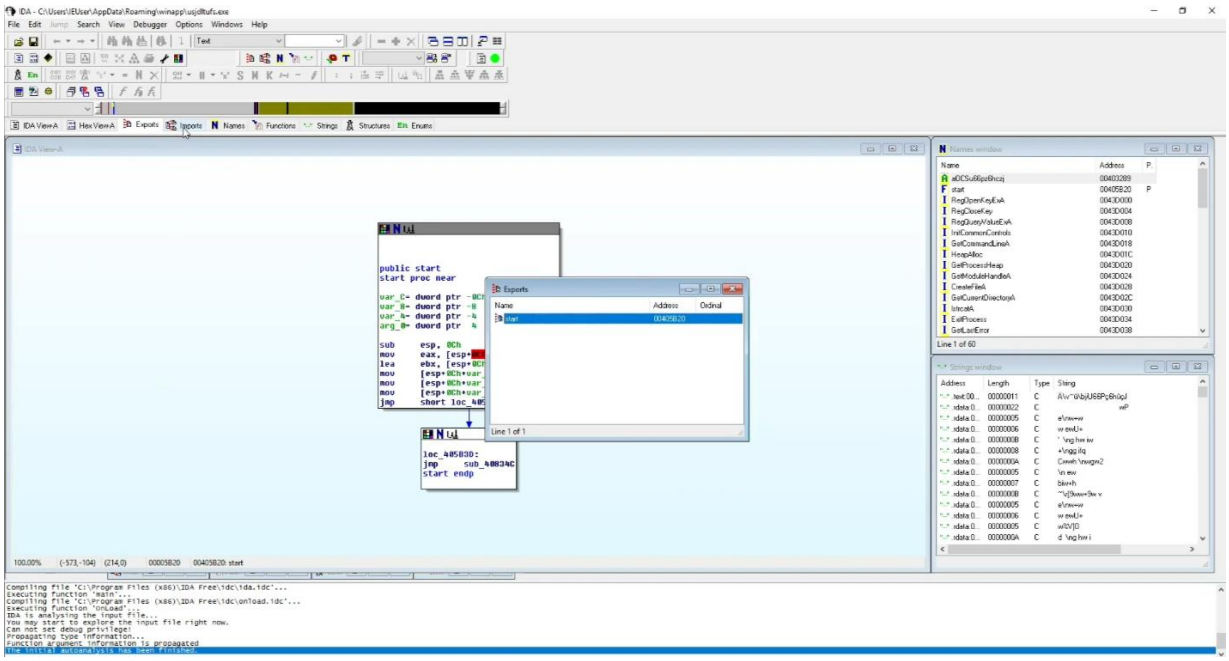
3. İstemci kimliği ve grup etiketi dosyalarının oluşturulması

Bu noktada, Trickster'ın daha derinlemesine bir analizini yapmak için statik analize geçebiliriz. Bunun için u.exe dosyasını IDA gibi bir kod çözücüye yükleyip detaylı bir analiz gerçekleştireceğiz.

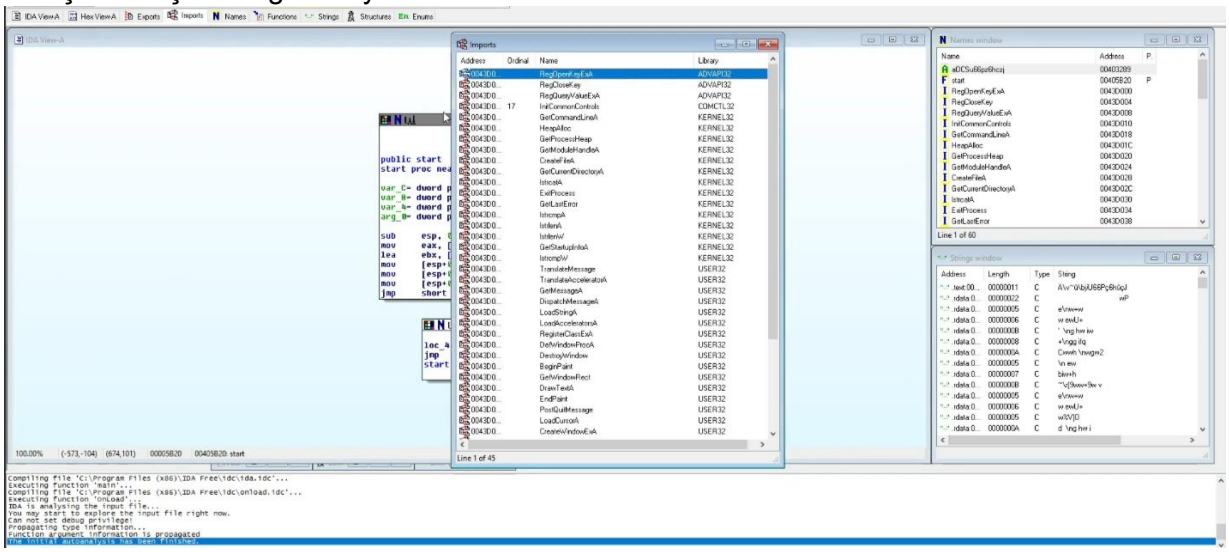
IDA programını kullanarak, u.exe dosyasını analiz edeceğiz ve programın hangi işlemleri çağırdığını göreceğiz. Kayıt defteri anahtarlarını açma, kapatma ve sorgulama gibi işlevleri çağırabildiğini görebiliriz. Ancak, bu bilgiler henüz tam anlamıyla ne olduğunu anlamamıza yeterli değil.

Şimdi, burada IDA'nın sol taraftaki A görünümünde dikkat ettiyseniz, programın bize montaj kodunu verdiğini göreceksiniz. Bu kod, IDA'nın programı en iyi şekilde analiz ederek, ayrıştırıp tahminler yapmasına olanak tanır.

IDA'da bizim için asıl önemli olan şey, ithalat ve ihracatın ne olduğudur, yani bu programın ne kullandığıdır. Şimdi, exports göz atacağım, ama gerçekten hiçbir şey görmüyoruz.



İnports baktığımızda ise, programın çekirdekten ve Windows API'lerinden çağırdığı tüm farklı süreçleri ve işlevleri görebiliyoruz.

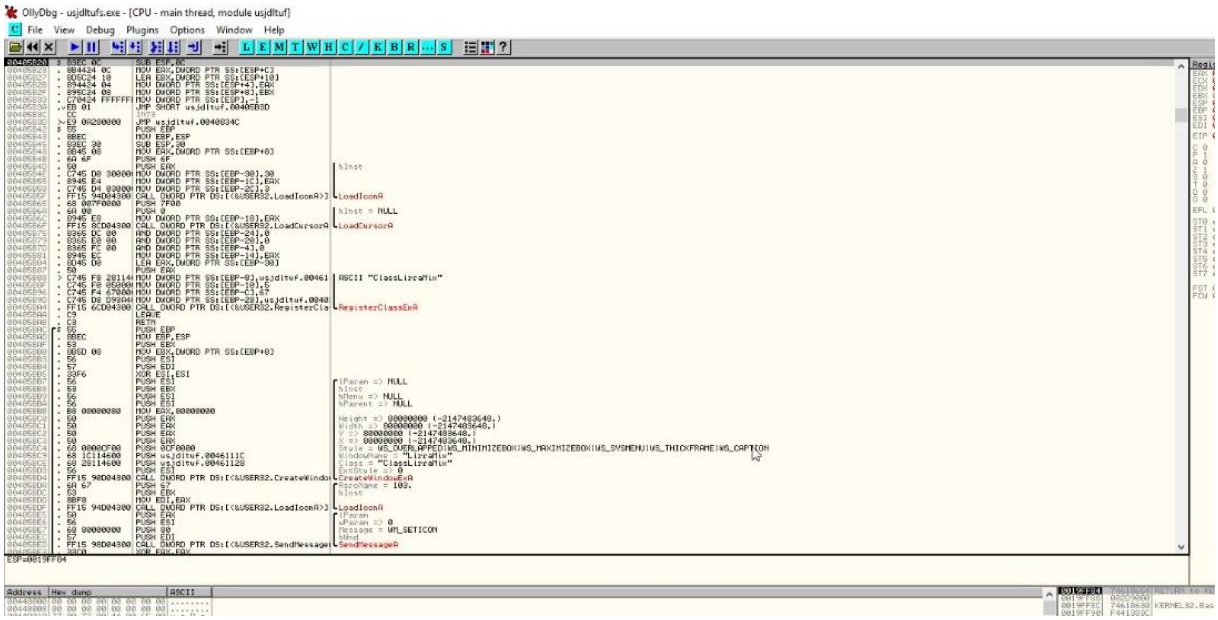


Bu sayede, programın ne tür işlevleri çağırdığını ve kayıt defteri anahtarlarını nasıl açıp kapattığını, sorguladığını fark ediyoruz. Süreç monitöründe de bu programın kayıt defteriyle birçok şey yaptığını görebiliyoruz. Ayrıca dosya oluşturabildiğini, geçerli bir dizin aldığını, komut satırı argümanlarını işleyip komutları çalıştırabildiğini, dizeleri yükleyip sınıfları kaydedebildiğini ve daha birçok farklı şey yapabildiğini tespit ediyoruz.

Bu aşamada program hakkında biraz bilgi edinmiş oluyoruz, fakat eğer montaj dilini okuyamıyorsanız ve adım adım ilerleyemiyorsanız, bu bilgi giriş düzeyinde kötü amaçlı yazılım analizi için yeterli olmayacaktır. Eğer bu programı tamamen tersine mühendislik etmek isterseniz, çok daha derinlemesine bilgi gerektiren bir yol izlemeniz gerekir ki bu, bu sınıfta ele almayacağımız bir konudur.

Şimdi IDA'dan çıkalım ve programı bir hata ayıklayıcı ile çalıştıralım. Bir hata ayıklayıcı, bir programı adım adım çalıştırmamıza olanak tanır, her seferinde bir talimat geçerek. Bu

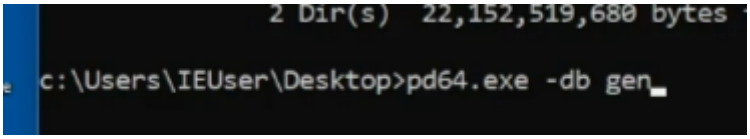
yüzden dosyamızı OllyDbg'ın içinde açacağım. Bunun için C sürücüsüne gideceğim, kullanıcıya, uygulama verilerine, dolaşıma ineceğim ve sonra WinApp'e gideceğim. U.exe'yi açtığımda, tüm montaj dilimiz burada olacak ve kaynak koda dönüştürülerek adım adım çalıştırılabilir.



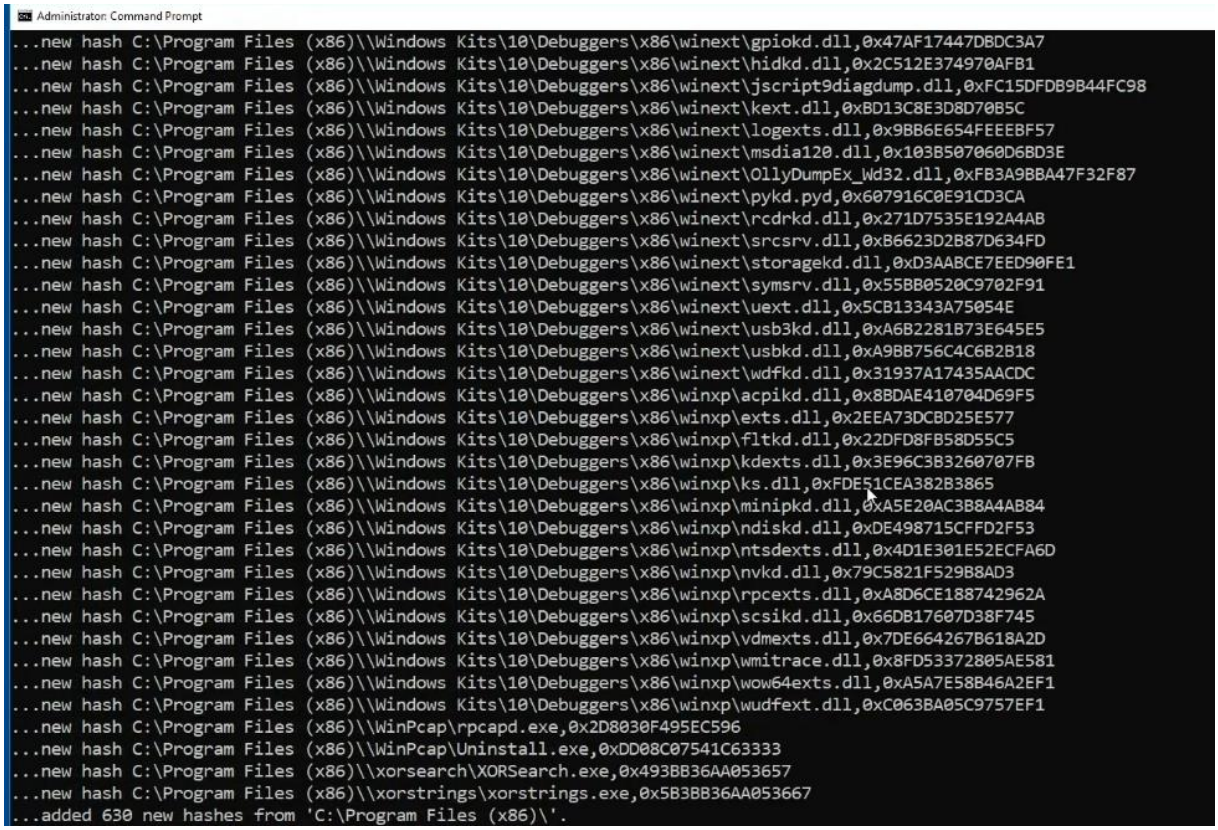
Ancak, birçok kötü amaçlı yazılım kendini adım adım çalıştırılmaktan korur. Trickster ve u.exe'nin bu kötü amaçlı yazılımlardan biri olup olmadığını görelim. Programı adım adım çalıştırmaya çalıştığımızda, çok hızlı çalışır ve aniden sona erer. Bu, kötü amaçlı yazılımın bir hata ayıklayıcıda çalıştırıldığını tespit ettiğini gösterir. Bu durumda, kodu çıkarabilmek ve analiz edebilmek için daha karmaşık bir yöntem kullanmamız gerekecektir.

Bu işlemi hafızada yakalamak ve analiz etmek için, temiz bir sanal makine (VM) ile tekrar başlamalıyız. Antivirüsümüzü ve güvenlik duvarımızı kapattık, Trickster dosyasını açtık. Ancak, Process Dump adlı bir yazılım parçası indirmemiz gerekiyor. Edge tarayıcımızı açarak split-code.com/processdump.html adresine gideceğiz. Buradan Windows 2.1 için 32 bit veya 64 bit sürümünü indirebiliriz. 64 bit işletim sistemi kullandığımız için bu sürüm bizim için uygun olacak. Dosyayı masaüstümüze açıp, PD64'ü masaüstümüze bırakacağız ve kullanıma hazır hale getireceğiz.

Ardından, bir komut istemi açacağız, ancak bu sefer üzerine sağ tıklayıp yönetici olarak çalıştıracağız. Komut istemimizde masaüstü dizinine gideceğiz, C:/users/ieuser/desktop altında olacak. Burada DIR komutunu çalıştırdığımızda Trickster ve PD64'ü göreceğiz.

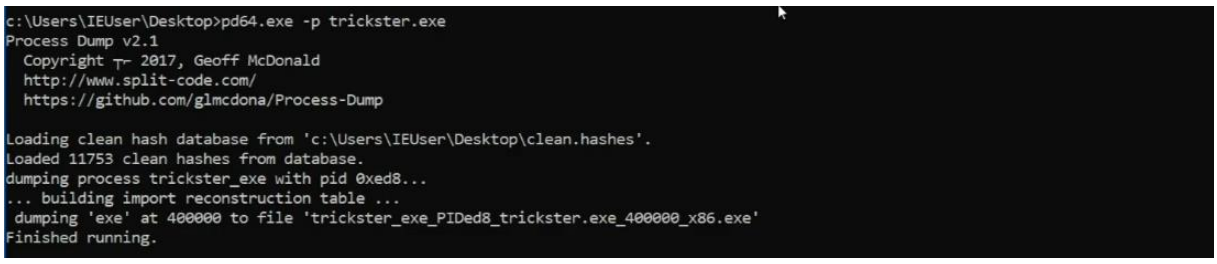


PD64'ü çalıştırarak şu anda çalışmakta olan tüm işlemlerin veritabanını oluşturacağız. Bu, sistemin hızına bağlı olarak beş ila on dakika sürecektir.



Trickster'ı yakalamak için atacağımız bir sonraki adım, `PD64.exe -P` komutunu yazmaktır ve ardından çalışacak olan dosya `trickster.exe` olacak. Ancak, enter tuşuna basmadan önce Trickster'ı oluşturmalıyız, çünkü şu anda enter tuşuna basarsak, hiçbir şey bulamayacak çünkü Trickster henüz mevcut değil.

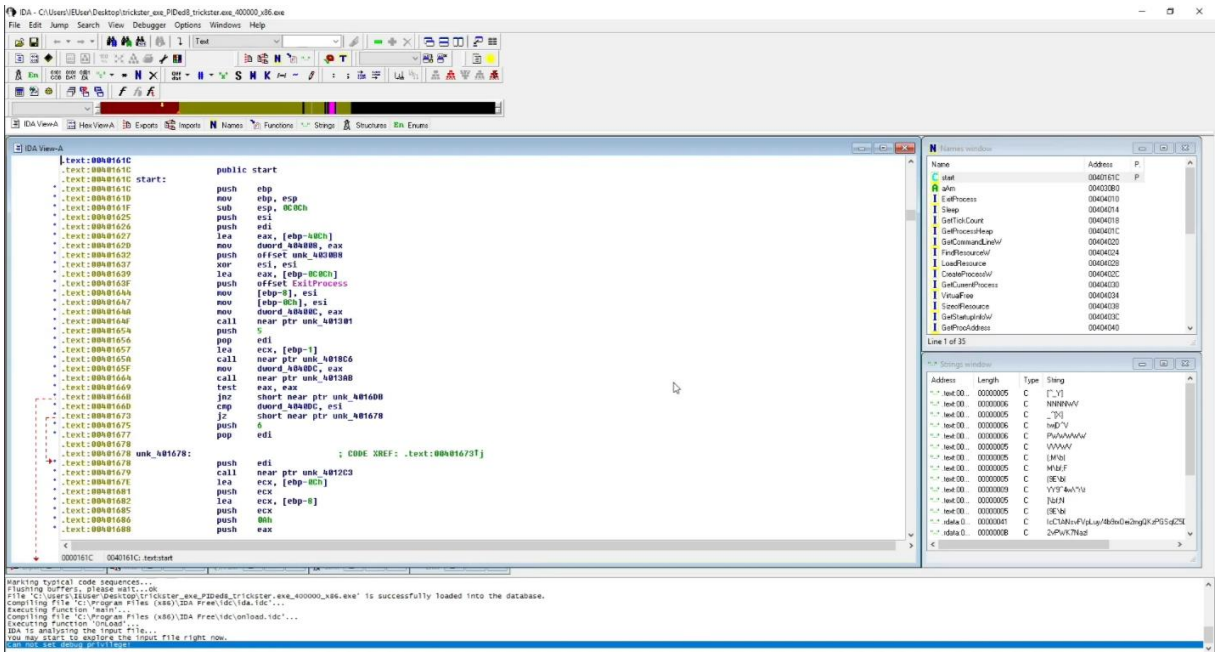
Öncelikle Trickster'ı çalıştırmak için sağ tıklayıp "Yönetici olarak çalıştır" seçeneğini seçmeliyiz. Ardından "Evet" düğmesine basar basmaz, komut istemine geçip enter tuşuna basarak süreci yakalayacağız. "Evet"e tıklayın ve ardından enter tuşuna basın.

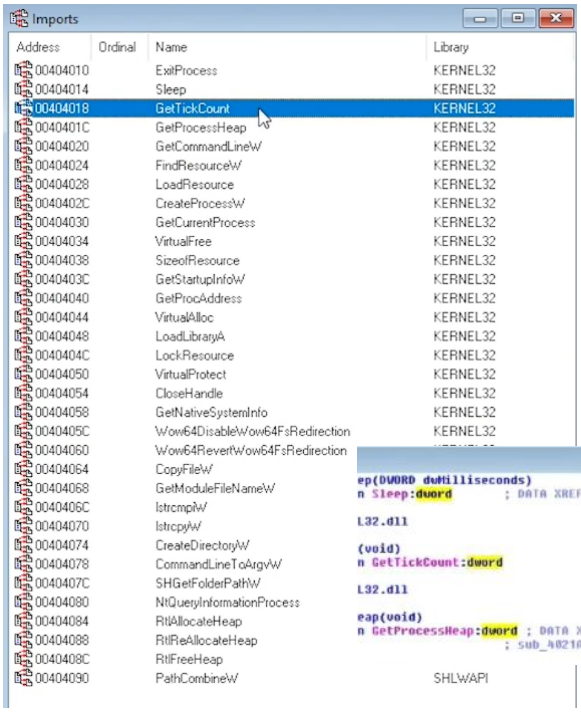


Bunun sonucunda, exe dosyasını şu dosya adıyla göreceksiniz: `trickster.exe`. Bu büyük ve uzun dosya adı artık masaüstünüzde görünecek. İşte o kötü amaçlı yazılım parçası, bellekten çıkarılmış ve masaüstünde.

Bir sonraki adım, bu dosyayı IDA'da açmaktır. Windows simgesine gidip IDA'yı başlatın, gerekli bilgileri görüntülemek istemediğinizi belirterek şartlar ve koşulları kabul edin, ardından yeni bir proje oluşturun ve iptal tuşuna basın. Masaüstündeki `trickster.exe` dosyasını seçin ve IDA'ya bırakın. Tamam ve onay tuşlarına basarak işlemi tamamlayın.

Artık bellekten yakaladığımız bu dosya elimizde. Şimdi, bu dosyada ilgimizi çeken herhangi bir şey olup olmadığını inceleyebiliriz. Bu adım, IDA'da ilk kez bakmamızdan daha verimli olmalı.





Devam edip import tablosunu inceleyelim. Burada daha önce sahip olmadığımız bazı yeni fonksiyonlar var, örneğin `sleep` ve `GetTickCount`. `GetTickCount`, işlemlerin sayısını hesaplayan bir sayaç işlevi görür. `Sleep` ise belirli bir süre boyunca beklemeyi sağlar. Eğer `Sleep` fonksiyonunu çift tıklarsanız, milisaniyeler üzerinden belirli bir süre bekleyeceğini görebilirsiniz. Buradaki `push 1388H`, yaklaşık beş saniyeye denk gelir.

Neden beş saniye bekliyor olabilir? Programın, dosya ve klasörleri oluşturup çalıştırabilmesi için zamana ihtiyaç duyduğu için. Program başlar, bekler, klasörü oluşturur, bekler, programı çalıştırır ve tekrar bekler.

Bu yeni fonksiyonlar, bir sonraki adımda her şeyin hazır olup olmadığını tespit etmek için kullanılır. Eğer montaj kodunu okuyabilirseniz, bu kodun ne yaptığını daha iyi anlayabilirsiniz.

Ancak, bu bölüm genel bir fikir edinmeniz içindir. Asıl amaç, kötü amaçlı yazılımın nasıl analiz edileceğine dair temel bir anlayış kazandırmaktır.

7. EDR Configuration

Uç nokta algılama ve yanıtı yapılandırırken, yanlış pozitiflerinizi azaltmak önemlidir. Çünkü bu kurallar, yanlış pozitifler üretebilir ve analistlerin dikkatini dağıtarak gerçek tehditleri tanımlamayı zorlaştırabilir.

Bunu yapmanın yollarından biri, kötü amaçlı yazılım olarak düşündüğümüz bir örneği topluluktaki diğerleriyle paylaşmaktır. Farklı topluluklarla ve endüstri portallarıyla işbirliği yaparak, bu bilgileri paylaşabilir ve tehdit istihbaratını artırabiliriz. Bu, daha iyi imzalar geliştirmemize ve dolayısıyla yanlış pozitifleri azaltmamıza yardımcı olacaktır.

Bu amaçla kullanabileceğimiz araçlardan biri VirusTotal'dır. VirusTotal, yüklediğiniz dosyayı 70'in üzerinde farklı antivirüs tarayıcısı, URL ve alan adı kara listeye alma hizmeti ile tarar. Ayrıca, çalışma içeriğinden sinyaller çıkarmak için birçok diğer araca sahiptir. VirusTotal, kötü amaçlı yazılımlar hakkında daha fazla bilgi edinmemizi sağlar ve bu bilgileri antivirüs şirketleriyle paylaşır, böylece daha iyi imzalar geliştirilir ve daha hızlı güncellenir.

Ancak, VirusTotal hakkında bir not var: Dosyanızı yüklemeden önce kuruluşunuzun bunu kabul ettiğinden emin olmalısınız. Çünkü eğer kötü amaçlı yazılım bulaşmış bir dosyayı VirusTotal'a gönderirseniz, saldırganlar dosyanın bilindiğini fark edebilir ve tekniklerini değiştirebilirler. Bu, ödüllerle karşılaştırıldığında bir risk teşkil eder.

Başka bir şey olarak, kötü amaçlı yazılım örneklerinizi antivirüs şirketinize veya siber tehdit istihbaratı satıcınıza gönderebilirsiniz. FireEye, Symantec veya Microsoft gibi satıcılar, örneklerinizi daha iyi kurallar oluşturmak ve hem sizi hem de diğer müşterileri korumak için kullanabilirler.

Kuruluşunuzun, özel kötü amaçlı yazılım imzaları veya algılama kuralları oluşturması gerekebilir. Bu genellikle belirli durumlarda yapılır ve kuruluşunuzun türüne bağlıdır. Çoğu kuruluş özel imzalar oluşturmak zorunda kalmaz, ancak kullanım durumunuza bağlı olarak, böyle bir gereksiniminiz olabilir.

Özel imzaların diğer programlarla uyumlu çalışmasını sağlamak için belirli bir şemayı takip etmelisiniz. Yaygın bir şema, Kötü Amaçlı Yazılım Özelliği Numaralandırma ve Karakterizasyon Şeması (MAEC) olarak bilinir. MAEC, kötü amaçlı yazılımlarla ilgili yapılandırılmış bilgileri paylaşmak için standartlaştırılmış bir dildir ve STIX ve TAXII ile uyumludur.

Bir diğer araç ise Yara'dır. Yara, Windows üzerinde çalışan çok platformlu bir programdır ve Linux ile Mac OS X'te de kötü amaçlı yazılım örneklerini tanımlamak ve sınıflandırmak için kullanılabilir. Yara, belirli dize kombinasyonlarını eşleştirmek için Yara kuralları oluşturur. Bu kurallar, ikili dosyalar, günlük dosyaları, paket yakalamalar veya e-postalar gibi veri kaynakları içinde belirli dizeleri arar.

Yara kural setleri, belirli bir kötü amaçlı yazılım parçasını tanımlamak için oluşturulur. Örneğin, 2015'te oluşturulmuş bir Yara kural setinde "File Backdoor exe" adlı bir kural bulunabilir. Bu kurallar, kötü amaçlı yazılımın belirli dizelerini arar ve diğer kopyaları tespit etmek için kullanılır.

Sınav için, kendi Yara kurallarınızı yazmak zorunda değilsiniz. Ancak, bu bilgiyi gerçek dünyada tehdit avcılığı yaparken kullanabilirsiniz. Yara, arama yaparak kötü amaçlı yazılımı tespit etmenize yardımcı olur ve birçok profesyonel, ağdan alınan verileri Yara ile tarayarak tehditleri doğrulamaya çalışır.

8. Block Lists and Allow Lists

Engelleme Listesi (Block List):

- Engelleme listesi, bilinen kötü uygulamaların, hizmetlerin veya trafiklerin sistemlere erişimini engelleyen bir güvenlik yapılandırmasıdır.

- Bir engelleme listesi, örneğin, bir gece kulübüne girmek için listeye bakılması gibi çalışır; listede adı olmayan kişiler içeri girebilir.

- Kötü IP adreslerini engellemek için kullanılabilir, ancak yanlış pozitifler riski vardır. Yanlış pozitifler, meşru trafiğin engellenmesine neden olabilir.

- Bilinmeyen kötü IP adreslerini yakalama riski de vardır çünkü engelleme listesi neyin kötü olduğunu bilmeye dayanır.

İzin Verme Listesi (Allow List):

- İzin verme listesi, sadece bilinen ve güvenilir uygulamaların veya hizmetlerin sistemlere erişmesine izin verir.

- Listeye dahil olmayan her şey engellenir. Bu, belirli güvenilir sitelere erişimin sağlanmasını ancak diğer her şeyin engellenmesini sağlar.

- İzin verme listeleri, bir geri dönüş duruşu olarak etkili olabilir; örneğin, bir saldırı sırasında tüm trafiği kesip sadece önceden belirlenen güvenilir sitelere izin verebilirsiniz.

- Ancak, izin verilenler listesi çok kısıtlayıcı olabilir ve güncel tutmak zor olabilir. Örneğin, küçük bir değişiklik (web sitenizin adresinin değişmesi gibi) erişim sorunlarına yol açabilir.

Yürütme Kontrolleri:

- Yürütme kontrolleri, sistemlerde hangi yazılımların çalıştırılabileceğini belirler.

- Windows'ta, Yazılım Kısıtlama Politikaları (SRP) ve AppLocker gibi yöntemler kullanılabilir. SRP, belirli dizinlerden ve hash'lerden uygulama çalıştırmayı kontrol ederken, AppLocker daha ayrıntılı yönetim sağlar.

- Linux'ta, SELinux ve AppArmor gibi güvenlik modülleri yürütme kontrolü sağlar.

Konfigürasyon Yönetimi:

- Engelleme ve izin verme listeleri için konfigürasyon yönetimi önemlidir. Yapılan değişiklikler, kullanıcı istekleri, iş operasyonları ve olay tepkilerine göre yönetilmelidir.

- Büyük değişiklikler yapmadan önce risk değerlendirmesi ve iş etki analizi yapılmalıdır.