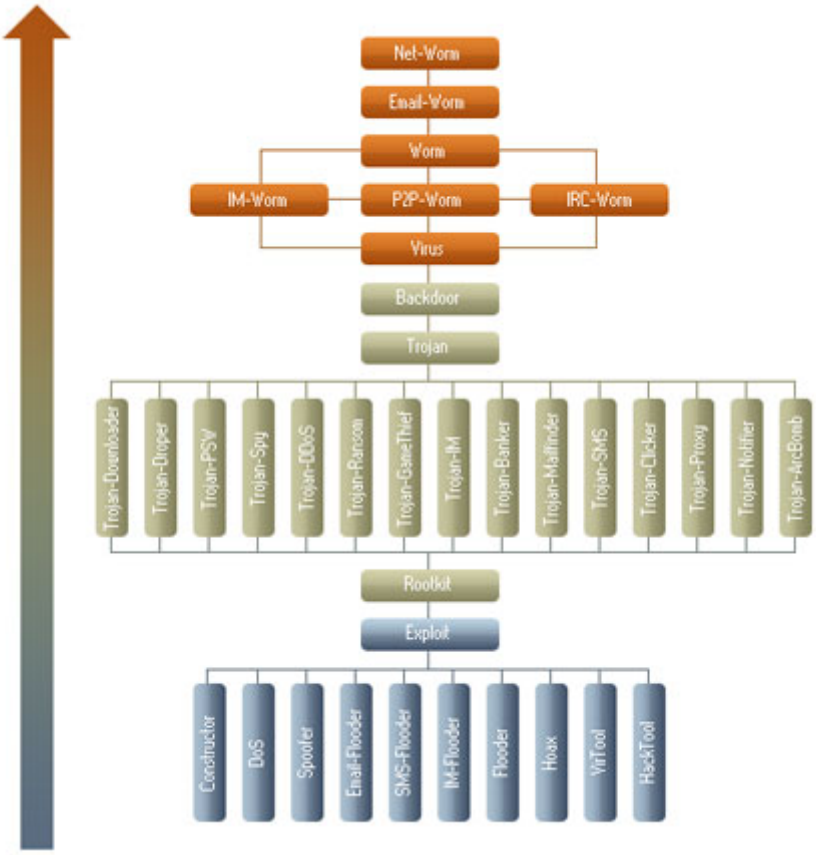
Gün 2

Not defteri:	Zararlı Yazılım Analizine Giriş		
Oluşturulan:	26.01.2020 09:40	**Güncellen...**	26.01.2020 16:53
Yazar:	Özlem Körpe		

Malware Threat Level

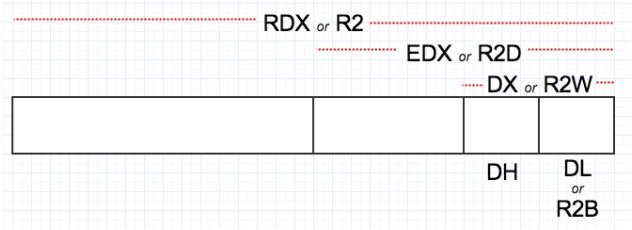


Cmder: Açık kaynaklı bir terminal emülatörüdür.

Not: Lisanslama programları şifreleri exe içerisinde doğrudan yerleştirmez. Yazılım çalıştırıldıktan sonra şifreler oluşturulur. Böylece çalıştırılmadığı sürece kod incelendiğinde kullanıcı adı, parola görüntülenemez. Bu gibi durumlarda yazılım debug edilmelidir. Böylece kullanıcı adı ve parolalar oluşturulup exe içerisinde yerini alır. Yine de şifre açık bir şekilde string olarak görüntülenmeyebilir, gizlenmesi macıyla farklı fonksiyonlardan geçirilmiş olabilir. Kod işleyişi incelenmelidir.

CrackMe: https://crackmes.one/lasts

Register Partitions



Anti Detection Teknikleri

Zararlı yazılımlar analiz edilmelerini zorlaştırmak ve güvenlik önlemlerini atlatabilmek için bazı yöntemler bulundurabilir.

Obfuscation

Türkçe karşılığı gizlemek olan bir işlemdir. Kodun anlaşılabilirliğini azaltmayı amaçlar. Böylece analizi ve tespit edilmesini zorlaştırır. Örneğin okunabilir kaynak kodlarının karmaşık değişken isimleriyle değiştirilebilir. Hello world gibi birkaç satırlık bir program bile satırlarca koda dönüştürülebilir.

Encoding

Kodlama verileri bir formdan diğerine dönüştürme işlemidir

> **Base64 Encoding**: Kodlama sırasında 3 baytlık veriler 6 bitlik dörtlü gruplara dağıtılırlar. Her bir 6 bitlik grup 0 ile 63 arasında bir sayı oluşturur (26=64). En çok kullanılan tekniktir. Genellikle düşük seviyedeki zararlı yazılımlar tarafından kullanılır.

Code Encryption: Tersine mühendislik girişimlerini zorlaştırmak amaçlı programların kaynak kodlarını çeşitli kriptografik algoritmalar kullanarak şifreler. Packer/Crypter yazılımlarının kullanılmasındaki ana fikir tersine mühendislikten korunmak istenen programların kaynak kodlarının çeşitli şifreleme ve sıkıştırma algoritmaları kullanılarak paketlenmesidir.

https://www.cynet.com/blog/a-guide-to-malware-detection-techniques-av-ngav-and-beyond/

NanoCore RAT Analysis

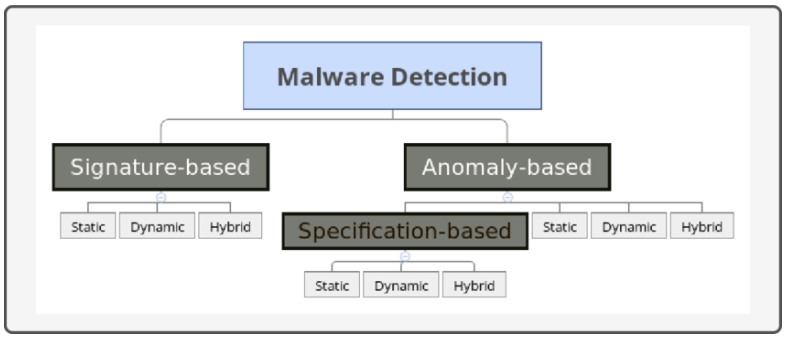
Very capable RAT, heavy obfuscation, persistent

1. Hangi dille yazıldığına bakmak için CFF Explorer kullanılabilir. İncelendiğinde C# ile geliştirildiği görülür. C# ile geliştirildiği için dnspy kullanılabilir.
2. Dnspy ile classlar incelenir. RAT'ın nasıl çalıştığı, nasıl kalıcılık sağladığı, nasıl yetki yükselttiği anlaşılmaya çalışılır.
3. Procexp programı admin yetkisiyle çalıştırılır. Sistemde çalışan processler, açılan portlar görüntülenebilir.
4. Procmon: Başlar başlamaz sistem üzerindeki tüm processleri taramaya başlar, bu yüzden durdurulduktan sonra filtreler kullanılarak tarama yapılır.

Mutex nedir ?

İşletim sisteminin eş zamanlı çalıştırdığı iş parçacıkları proses olarak adlandırılır. Prosesler de kendi içlerinde eş zamanlı çalışan iş parçacıklarına sahip olabilir ve bunlara da Thread denir.
Farklı prosesler arasında ya da aynı proses içinde yer alan Thread'ler arasında zaman zaman senkronizasyon ihtiyacı duyulur çünkü bu Thread'ler görevlerini yerine getirebilmek için işletim sistemi tarafından sağlanan veya prosesin kendisinin tuttuğu paylaşılan bir kaynağa erişmek isterler. Örnek olarak Thread'lerin bir log dosyasına veya ekrana log yazmak istedikleri durumu ele alalım. İki Thread aynı anda log dosyasına yazmaya çalışırsa dosyaya yazılan loglar birbirine karışacak ve okunmaz hale gelecektir. Burada Thread'leri log dosyasına yazma konusunda sıraya sokacak bir mekanizmaya ihtiyaç vardır.
Mutex (MUTual EXclusion)'ler tam da bu mekanizmayı sağlamak için tasarlanmıştır. Mutex'ler uygulamanın yazıldığı dil ve Runtime tarafından sağlanan basit veri yapılarıdır. Farklı Thread'ler tarafından paylaşılan her bir kaynak için kaynağa olan erişimi düzenlemek üzere bir Mutex yaratılır. Paylaşılan kaynağa erişim yapılan kod bölgesi Critical Section olarak adlandırılır. Kaynakla işi olan Thread, Mutex'in sahipliğini almaya (Acquire) çalışır. Mutex o anda başka bir Thread tarafından tutulmuyorsa Thread Mutex'i alır, Critical Section'a girerek ilgili kaynağı kullanır. Diğer durumda, yani Mutex o anda başka bir Thread tarafından kullanılıyor ise, ikinci Thread işlemci tarafından beklemeye alınır. Mutex'i tutan Thread Critical Section'ı bitirip Mutex'i bırakırken, halihazırda Mutex'in bırakılmasını bekleyen Thread uyandırılır ve Mutex'in sahipliğini alarak Critical Section'a girer ve paylaşılan kaynağa erişim sağlar.
https://medium.com/@gokhansengun/semaphore-mutex-ve-spinlock-nedir-ve-ne-i%C5%9Fe-yarar-ba552a17c03

Malware Detection



Yara: Zararlı yazılım analistlerinin kötü amaçlı yazılım örneklerini tanımlamasına ve sınıflandırmasına yardımcı olmayı amaçlayan bir araçtır. YARA ile metinsel veya ikili kalıplara dayalı olarak körü amaçlı yazılım ailelerinin açıklamaları oluşturulabilir. Her açıklama, yani kural, bir dizi string değerlerinden ve eşleşme mantığını belirleyen bir ifadeden oluşur. Yara kuralı hazırlanırken Visual Studio'nun yara eklentisi kullanılabilir. Böylece syntax hataları vb. kolayca görülebilir.

```
rule rogues
{
    meta:
        created  = "09/11/2017 00:00:00"
        modified = "09/11/2017 17:12:07"
        author   = "Metallica"
        Vendor   = "PC Smart Cleanup"
    strings:
        $textstring1 = "smart" ascii wide nocase
        $textstring2 = "cleanup" ascii wide nocase
        $textstring3 = "longrun" ascii wide nocase
    condition:
        $textstring1 and @textstring2 and $textstring3
}
```

Meta taginin altında kural ile ilgili açıklama bulununur.
Strings tagi altında taranan hafızada bulunmak istenen değerler bulunur. Dosyanın içinde o string geçiyor mu kontrol eder.
Condition kuralın hangi durumda match olacağını tanımlayan kısımdır.

Example Yara Rule : Hidden Tear Ransomware

```
rule Hidden-tear {

    meta:
        description = "HiddenTear Ransomware by Utku Sen"
        in_the_wild = true
        threat_level = 8

    strings:
        $s1 ="SendPassword"
        $s2 = "get_MachineName"
        $s3 = "AES_Encrypt"
        $s4= /\\obj\\(Release|Debug)\\hidden-tear\.pbd /

    condition:
        ($s4) or (3 of them)
}
```

filesize< 2 MB vb kurallar yazılabilir.

Example Yara Rule : Nanocore RAT

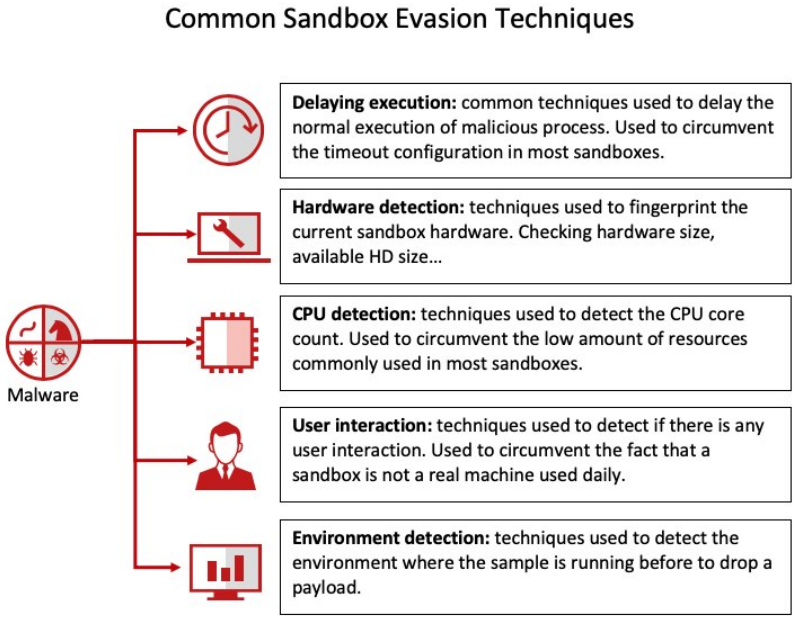
```
rule nanocore : rat {
```

```
    meta:
        description = "a rat"
        in_the_wild = true
        threat_level = 3

    strings:
        $kur1 = "#=qAM4ZJ3aDwBm_a3IkqHxLmjdKzHIQbFeE9thLHux2o6g="
        $kur2 = "#=qvA35ZDPTM3VgF89oJb9AmWFE4pqnIDYGjeV5H4uvblU="
        $kur3 = "#=qruARjy_8oZkz3lsHPGxBMA=="
        $kur4 = "GDelegate5"
        $kur5 = "NanoCore Client.exe"
        $kur6 = "Resolved hostname '{0}' to '{1}'"

    condition:
        any of them
```

Snort: IP ağlarında gerçek zamanlık trafik analizi ve paket günlüğü gerçekleştirebilen açık kaynaklı bir ağ saldırı önleme sistemidir. Protokol analizi, içerik arama/eşleştirme gerçekleştirebilir ve buffer overflow zaafiyetleri, gizli oırt taramaları, CGI saldırıları, SMB probları ve çok daha fazlası gibi çeşitli saldırı ve probları tespit etmek için kullanılır.

Automated Analysis Platforms
https://www.virustotal.com/
https://app.any.run/
https://www.hybrid-analysis.com/
https://www.joesandbox.com/
https://cape.contextis.com/

Sandbox Evasion Techniques



https://www.slideshare.net/bgasecurity/sandbox-atlatma-teknikleri-ve-neriler