Zararlı PDF Analizi

written by Mert SARICA | 24 Ocak, 2012

Eskiden art niyetli olsun veya olmasın bilgisayar korsanlarının hedef sistemlere sızmalarının arkasında yatan başlıca sebepler; hacktivism gayesiyle dünyaya mesaj verme, finansal fayda sağlamak amacıyla finansal bilgilere erişme, sistemde tespit edilen güvenlik zafiyetleri konusunda sistem yöneticilerini uyarma, kin ve nefret güdüleriyle hareket ederek hedef sistemlere zarar vermekti. Ancak zaman değişti ve uluslar, bilgisayar korsanlarından oluşan kendi siber kuvvetlerini oluşturarak hackingi bir silah olarak diğer devletlere karşı kullanmaya başladılar.

Siber savaşın yaşandığı günümüzde çoğunlukla izlenen strateji zarardan çok kalıcı olarak hedef sistemlerde barınmaya çalışma ve olabildiğince hassas bilgilere ulaşma yönünde oluyor ve çoğunlukla arkasında ulusların olduğuna inanılan bu sızmalara Advanced persistent threat (APT) adı veriliyor. Her ne kadar APT tehditlerinin arkasında çoğunlukla çok iyi organize olmuş bir grubun, ulusun olduğuna ve sızmalarda ileri seviye zararlı yazılımlardan ve sofistike araçlardan faydalandıkları düşünülse de aslında RSA vakasında olduğu gibi sahte bir e-posta hazırlama becerisi, ofis dokümanlarını açmak için kullanılan bir uygulamada Fuzzer ile güvenlik zafiyeti keşfetme becerisi, bu zafiyeti istismar edecek kod parçasını (exploit) hazırlama becerisi ve son olarak kaynak kodlarına internette rastlayabileceğiniz bir aracı özelleştirme (örnek: modifiye edilmiş poison ivy) becerisi yeterli olmaktadır. Hele ki günümüzde bu işlerin nasıl yapılabileceğini anlatan sayısız makale olduğu düşünüldüğünde sıradan bir kurum için bile bu tehditin gerçekleşme olasılığı oldukça yüksek seviyelere çıkmaktadır.

Çoğunlukla ABD Savunma Sanayii şirketlerini hedef alan Çin'li grubun, sistemlere sızmak ve gizli belgeleri çalmak için kullandıkları Sykipot adındaki zararlı yazılımı bulaştırmak için 2007 yılından bu yana şirket çalışanlarına, zafiyete sahip istismar kodu içeren ofis dokümanları gönderdikleri ve bu yöntemin bir çok organize suç örgütü tarafından da kullanıldığı düşünüldüğünde genel kabul görmüş güvenlik kontrollerinin (ips, antivirüs, vs.) yetersiz olduğu anlaşılmaktadır. Durum böyle olunca da kullanıcılardan gelen ihbarlar doğrultusunda olası zararlı kod içeren ofis dokümanlarının analiz edilmesi bir kurum için kaçınılmazdır.

Zararlı doküman analizinde analiz edilecek dosya Microsoft ofis dokümanı ise amaç, zararlı kod içerebilecek VB makro kodu, OLE verisi, kabuk kodu, PE dosyasını tespit etmektir. Bunun için OfficeMalScanner ve OffVis araçlarından faydalanabilirsiniz.

Eğer analiz edilecek dosya PDF ise bu defa amaç, zararlı kod içerebilecek Javascript kodunu tespit etmektir. Bunun için de pdf-parser.py, peepdf ve Origami gibi araçlardan faydalanabilirsiniz.

Örnek olarak malware.pdf adında zararlı kod içeren bir pdf dosyasını analiz

etmek istersek yapacağımız ilk iş ayrı ayrı bu iş için tasarlanmış araçları indirmek yerine zararlı yazılım analizi gerçekleştirmek için özel olarak hazırlanmış ve bir çok aracı üzerinde barındıran REMnux sanal işletim sistemini indirip kullanmaktır.

Kullanmaya başladığım bu işletim sisteminde, masaüstüne kopyaladığımız malware.pdf dosyasını peepdf aracı ile analiz etmek için **peepdf.py -i malware.pdf** komutunu yazarak bu pdf dosyası üzerinde javascript kodu olup olmadığını kontrol edebiliriz.

Görüldüğü üzere peepdf aracı bize 4. objede (Object with JS code) javascript kodu olduğu bilgisini vermektedir. **js_beautify object 4** komutunu yazarak pdf dosyası içinde yer alan javascript kodunu düzgün bir biçimde görüntüleyebiliriz.

Analiz neticesinde oluşturulan kabuk kodununun (shellcode) Collab.CollectEmailInfo fonksiyonuna gönderiliyor olması sonucunda bu pdf dosyasının içinde 2008 yılından kalma Adobe PDF v8.1.1 ve önceki sürümlerinde bulunan zafiyeti istismar eden istismar kodunun (exploit) bulunduğunu öğrenmiş oluyoruz.

Kimi zaman bu örnekte olduğu gibi pdf dosyası içinde yer alan istismar kodunun hangi zafiyeti istismar ettiğini anlamak bu kadar kolay olmayabilir. Bu gibi durumlarda javascript kodu içinde yer alan kabuk kodunu **set shellcode "kabuk kodu"** komutu ile tanımladıktan sonra **js_unescape variable shellcode** komutu ile analiz edilebilir hale getirebilir, **sctest variable shellcode** komutu ile kabuk kodunun çalışmasını simüle edebilir, shellcode2exe aracı ile yürütülebilir programa (executable) çevirerek immunity debugger ile dinamik olarak analiz edebilirsiniz.

Bir sonraki yazıda görüşmek dileğiyle yeni yılın herkese güvenli günler dilerim.