**1.Basic Static Malware Analysis~Temel Statik Kötü Amaçlı Yazılım Analizi**

Statik analiz bir programın dosyasının disassembled kodunun, grafik görüntülerini , yazdırılabilir dizelerin strings ve diğer disk üzerindeki kaynakların analiz edilmesiyle gerçekleşir. Programı çalıştırmadan reverse engineering tersine mühendisliği ifade eder.

Statik analiz tekniklerinin kendi eksiklikleri olsa da, çok çeşitli kötü amaçlı yazılımları anlamamıza yardımcı olabilirler. Dikkatli tersine mühendislik yoluyla, kötü amaçlı yazılım ikili dosyalarının saldırganlara bir hedefi ele geçirdikten sonra sağladığı faydaların yanı sıra saldırganların virüs bulaşmış bir makineye saldırılarını gizleme ve sürdürme yöntemlerini daha iyi anlayabileceksiniz. Göreceğiniz gibi, bu bölüm açıklamaları ve örnekleri birleştiriyor. Her bölüm bir statik analiz tekniğini tanıtır ve ardından gerçek dünya analizindeki uygulamasını gösterir.

ilk olarak çoğu windows programı tarafından kullanılan portable executable (PE) dosya formatını açıklacak ve sonra pythonda olan popüler pefile kütüphanesi ile gerçek dünya kötü amaçlı yazılım ikili dosyasını incelemek için nasıl kullanacağına bakalım. daha sonra içe aktarma analizi, grafik görüntü analizi ve dizi analizi gibi teknikleri açıklayacak. describe techniques import analysis , graphical image analysis and strings analysis

her durumda analiz tekniğini gerçek dünyadaki kötü amaçlı yazılımlara uygulamak için açık kaynaklı araçları nasıl kullanacağını gösterecek. Son olarak, bölümün sonunda, kötü amaçlı yazılımların kötü amaçlı yazılım analistleri için hayatı zorlaştırabileceği yolları tanıtacağım ve bu sorunları azaltmanın bazı yollarını tartışacağım.Bu bölümdeki örneklerde kullanılan kötü amaçlı yazılım örneğini bu kitabın verilerinde / ch1 dizini altında bulacaksınız. Bu bölümde tartışılan teknikleri göstermek için, vahşi ortamda yaygın olarak görülen kötü amaçlı yazılım türlerine bir örnek olarak deneysel kullanım için oluşturulmuş bir Internet Relay Chat (IRC) botu olan ircbot.exe'yi kullanıyoruz. Bu nedenle program, bir IRC sunucusuna bağlıyken hedef bilgisayarda yerleşik kalacak şekilde tasarlanmıştır. İrcbot .exe bir hedefi ele geçirdikten sonra, saldırganlar hedef bilgisayarı IRC aracılığıyla kontrol edebilir ve hedefin fiziksel konumunun video beslemelerini yakalamak ve gizlice çıkarmak için bir web kamerasını açmak, masaüstünün ekran görüntülerini almak, çıkarmak gibi eylemlerde bulunmalarına izin verebilir. hedef makinedeki dosyalar vb. Bu bölüm boyunca, statik analiz tekniklerinin bu kötü amaçlı yazılımın yeteneklerini nasıl ortaya çıkarabileceğini göstereceğim.

\*The Microsoft Windows Portable Executable Format:Microsoft Windows Taşınabilir Yürütülebilir Biçim:

Statik kötü amaçlı yazılım analizi gerçekleştirmek için, .exe, .dll ve .sys dosyaları gibi modern Windows program dosyalarının yapısını tanımlayan ve verileri saklama yöntemlerini tanımlayan Windows PE biçimini anlamanız gerekir. PE dosyaları x86 talimatlarını, görüntüler ve metin gibi verileri ve bir programın çalışması için ihtiyaç duyduğu meta verileri içerir.

PE biçimi formatı  başlangıçta aşağıdakileri yapmak için tasarlanmıştır:

Windows'a bir programın belleğe nasıl yükleneceğini anlatın PE biçimi, bir dosyanın hangi parçalarının belleğe yüklenmesi gerektiğini ve nereye yükleneceğini açıklar. Ayrıca, Windows'un program kodunun neresinde bir programın yürütülmesini başlatması gerektiğini ve hangi dinamik olarak bağlantılı kod kitaplıklarının belleğe yüklenmesi gerektiğini de söyler.

Çalışan bir programın yürütülmesi sırasında kullanabileceği tedarik ortamı (veya kaynaklar) Bu kaynaklar, GUI iletişim kutularında veya konsol çıktısında bulunanlar gibi karakter dizilerinin yanı sıra görüntüleri veya videoları içerebilir.

Dijital kod imzaları gibi güvenlik verileri sağlayın: Windows, kodun güvenilir bir kaynaktan geldiğinden emin olmak için bu tür güvenlik verilerini kullanır.

PE formatı, tüm bunları Şekil 1-1'de gösterilen yapı  structs serisinden yararlanarak gerçekleştirir.

8-> .reloc section (memory translations )

7-> .rsrc section (strings, images, ..)

6->  .idata section (imported libraries)

5-> .text section (program code)

4-> Section headers Bölüm başlıkları

3-> Optional headers İsteğe bağlı başlıklar

2-> PE header

1-> DOS header

8 e doğru increasing file offsets artan dosya ofsetleri

Şekilde gösterildiği gibi, PE biçimi, işletim sistemine programın belleğe nasıl yükleneceğini anlatan bir dizi başlık içerir. Ayrıca, gerçek program verilerini içeren bir dizi bölüm içerir. Windows bölümleri belleğe, bellek uzaklıkları diskte göründükleri yere karşılık gelecek şekilde yükler. PE başlığından başlayarak bu dosya yapısını daha ayrıntılı inceleyelim. 1980'lerden kalma Microsoft DOS işletim sisteminin bir kalıntısı olan ve yalnızca uyumluluk nedenleriyle sunulan DOS başlığıyla ilgili tartışmayı atlayacağız.

\*The PE Header:

Şekil 1-1'in altında, DOS başlığı 1'in üzerinde gösterilen, bir programın ikili kod, görüntüler, sıkıştırılmış veriler ve diğer program öznitelikleri gibi genel özniteliklerini tanımlayan PE başlığı 2'dir. Ayrıca bize bir programın 32 bitlik sistemler için mi yoksa 64 bit sistemler için mi tasarlandığını söyler. PE başlığı, kötü amaçlı yazılım analistine temel ancak yararlı bağlamsal bilgiler sağlar. Örneğin, üstbilgi header , kötü amaçlı yazılım yazarının dosyayı derlediği zamanı verebilen bir zaman damgası alanı içerir. Bu, kötü amaçlı yazılım yazarları bu alanı sahte bir değerle değiştirmeyi unuttuğunda meydana gelir, ki bu genellikle yaparlar.

\*The Optional Header:

İsteğe bağlı başlık 3, adından da anlaşılacağı gibi, günümüzün PE çalıştırılabilir programlarında aslında her yerde bulunur. Programın yüklendikten sonra çalıştırdığı ilk talimatı ifade eden PE dosyasındaki programın giriş noktasının konumunu tanımlar. Ayrıca, PE dosyasını, Windows alt sistemini, program hedeflerini (Windows GUI veya Windows komut satırı gibi) ve programla ilgili diğer üst düzey ayrıntıları yüklerken Windows'un belleğe yüklediği verilerin boyutunu da tanımlar. Bu başlıktaki bilgiler tersine mühendisler için çok değerli olabilir çünkü bir programın giriş noktası onlara tersine mühendisliğe nereden başlayacaklarını söyler.

\*Section Headers:

Bölüm başlıkları 4, bir PE dosyası içinde bulunan veri bölümlerini açıklar. Bir PE dosyasındaki bir bölüm, işletim sistemi bir programı yüklediğinde belleğe eşlenecek veya programın belleğe nasıl yüklenmesi gerektiğiyle ilgili talimatlar içerecek bir veri yığınıdır. Başka bir deyişle, bir bölüm, bellekte bitişik bir bayt dizisi haline gelecek veya işletim sistemini yükleme işleminin bazı yönleri hakkında bilgilendirecek olan diskteki bir bayt dizisidir. Bölüm başlıkları ayrıca Windows'a bölümlere hangi izinleri vermesi gerektiğini, örneğin program yürütülürken program tarafından okunabilir, yazılabilir veya çalıştırılabilir olması gerektiğini söyler. Örneğin, x86 kodunu içeren .text bölümü tipik olarak okunabilir ve çalıştırılabilir olarak işaretlenir, ancak program kodunun yürütme sırasında yanlışlıkla kendi kendini değiştirmesini önlemek için yazılamaz. .Text ve .rsrc gibi bir dizi bölüm Şekil 1-1'de gösterilmektedir. PE dosyası yürütüldüğünde bunlar belleğe eşlenir. .Reloc bölümü gibi diğer özel bölümler, belleğe eşlenmez. Bu bölümleri de tartışacağız. Şekil 1-1'de gösterilen bölümlerin üzerinden geçelim.

-The .text Section:

Her PE programı, bölüm başlığında yürütülebilir olarak işaretlenmiş en az bir x86 kodu bölümü içerir; bu bölümler hemen hemen her zaman .text 5 olarak adlandırılır. Bölüm 2'de program sökme ve tersine mühendislik gerçekleştirirken .text bölümündeki verileri parçalarına ayıracağız.

-The .idata Section:

İçe aktarma  import olarak da adlandırılan .idata bölümü 6, dinamik olarak bağlantılı kitaplıkları ve işlevlerini listeleyen İçe Aktarma Adres Tablosunu (IAT) içerir. IAT, analiz için bir PE ikili programına ilk yaklaşırken incelenecek en önemli PE yapıları arasındadır, çünkü bir programın yaptığı kitaplık çağrılarını ortaya çıkarır ve bu da kötü amaçlı yazılımın üst düzey işlevselliğine ihanet edebilir.highlevel functionality

-The Data Sections:

Bir PE dosyasındaki veri bölümleri, fare imleci görüntüleri, düğme dış görünümleri, ses ve bir program tarafından kullanılan diğer ortamlar gibi öğeleri depolayan .rsrc, .data ve .rdata gibi bölümleri içerebilir. Örneğin, Şekil 1-1'deki .rsrc bölümü 7, bir programın metni dizeler olarak işlemek için kullandığı yazdırılabilir karakter dizilerini içerir.

.Rsrc (kaynaklar) bölümündeki bilgiler kötü amaçlı yazılım analistleri için hayati öneme sahip olabilir, çünkü bir PE dosyasındaki yazdırılabilir karakter dizelerini, grafik görüntüleri ve diğer varlıkları inceleyerek dosyanın işlevselliği hakkında önemli ipuçları elde edebilirler. Sayfa 7'deki "Kötü Amaçlı Yazılım Görüntülerini İnceleme" bölümünde, kötü amaçlı yazılım ikili dosyalarının kaynaklar bölümlerinden grafik görüntüleri çıkarmak için icoutils araç setini (icotool ve wrestool dahil) nasıl kullanacağınızı öğreneceksiniz. Ardından, "Kötü Amaçlı Yazılım Dizelerini İnceleme" sayfa 8'de, kötü amaçlı yazılım kaynakları bölümlerinden yazdırılabilir dizeleri nasıl çıkaracağınızı öğreneceksiniz.

-The .reloc Section:

Bir PE ikili kodu konumdan bağımsız değildir, yani amaçlanan bellek konumundan yeni bir bellek konumuna taşınırsa doğru şekilde çalışmayacaktır. .Reloc bölümü 8, kodun kırılmadan taşınmasına izin vererek bunun üstesinden gelir. Windows işletim sistemine, kod taşınmışsa, kodun hala düzgün çalışması için bir PE dosyasının kodundaki bellek adreslerini çevirmesini söyler. Bu çeviriler genellikle bir bellek adresinden bir ofset eklemeyi veya çıkarmayı içerir. Bir PE dosyasının .reloc bölümü kötü amaçlı yazılım analizinizde kullanmak isteyeceğiniz bilgileri içerebilse de, bu kitapta daha fazla tartışmayacağız çünkü odak noktamız kötü amaçlı yazılımlara makine öğrenimi ve veri analizi uygulamak değil yer değiştirmelere bakmayı içeren sert tersine mühendislik.

\*Dissecting the PE Format Using pefile:Pefile Kullanarak PE Formatını Kesmek:

Ero Carerra tarafından yazılan ve sürdürülen pefile Python modülü, PE dosyalarını incelemek için endüstri standardı bir kötü amaçlı yazılım analiz kitaplığı haline geldi. Bu bölümde ircbot.exe dosyasını incelemek için pefile'ı nasıl kullanacağınızı göstereceğim. İrcbot.exe dosyası bu kitaba eşlik eden sanal makinede ~ / malware\_data\_science / ch1 / data dizininde bulunabilir. 1-1 listesi, ircbot.exe'nin geçerli çalışma dizininizde olduğunu varsayar.

-->pip install pefile ile indirelim

Şimdi, Python'u başlatmak, pefile modülünü içe aktarmak ve pefile kullanarak ircbot.exe PE dosyasını açmak ve ayrıştırmak için Liste 1 1'deki komutları kullanın.

--> import pefile

pe=pefile.PE("ircbot.exe")

PE dosyalarını, niteliklerini inceleyebilmemiz için ayrıştırır. PE yapıcısını çağırarak, bu örnekte ircbot.exe olan belirtilen PE dosyasını yükler ve ayrıştırırız. Dosyamızı yükleyip ayrıştırdığımıza göre, ircbot.exe’nin PE alanlarından bilgi almak için Liste 1-2’deki kodu çalıştırın.

# based on Ero Carrera's example code (pefile library author)

for section in pe.sections:

print (section.Name, hex(section.VirtualAddress),

hex(section.Misc\_VirtualSize), section.SizeOfRawData )

-->OUTput:

('.text\x00\x00\x00', (1)'0x1000', (2)'0x32830', (3)207360)

('.rdata\x00\x00', '0x34000', '0x427a', 17408)

('.data\x00\x00\x00', '0x39000', '0x5cff8', 10752)

('.idata\x00\x00', '0x96000', '0xbb0', 3072)

('.reloc\x00\x00', '0x97000', '0x211d', 8704)

Liste 1-3'te görebileceğiniz gibi, PE dosyasının beş farklı bölümünden veri aldık: .text, .rdata, .data, .idata ve .reloc. Çıktı, çekilen her PE bölümü için bir tane olmak üzere beş tuple olarak verilir. Her satırdaki ilk giriş PE bölümünü tanımlar. (Basitçe C tarzı boş dize sonlandırıcılar olan \ x00 boş bayt serisini göz ardı edebilirsiniz.) Kalan alanlar bize her bölümün belleğe yüklendikten sonra bellek kullanımının ne olacağını ve yüklendikten sonra bellekte nerede bulunacağını söyler. .

Örneğin, 0x1000 (1), bu bölümlerin yükleneceği temel sanal bellek adresidir. Bunu bölümün temel bellek adresi olarak düşünün. Sanal boyut alanındaki 0x32830 2, yüklendikten sonra bölüm için gereken bellek miktarını belirtir. Üçüncü alandaki 207360 3, bölümün bu bellek parçası içinde alacağı veri miktarını temsil eder.

Bir programın bölümlerini ayrıştırmak için pefile kullanmaya ek olarak, bir ikilinin yükleyeceği DLL'leri ve bu DLL'ler içinde isteyeceği işlev çağrılarını listelemek için de kullanabiliriz. Bunu, bir PE dosyasının IAT'sini boşaltarak yapabiliriz. 1-4 listesi, ircbot.exe için IAT'yi dökmek üzere pefile'ın nasıl kullanılacağını gösterir.

-->python

pe = pefile.PE("ircbot.exe")

for entry in pe.DIRECTORY\_ENTRY\_IMPORT:

print entry.dll

for function in entry.imports:

print '\t',function.name

-->output kısa:KERNEL32.DLL

GetLocalTime

ExitThread

CloseHandle

WriteFileυ

CreateFileAϖ

ExitProcess

CreateProcessAω

GetTickCount

GetModuleFileNameA

--snip--kırpmak

Liste 1-5'te görebileceğiniz gibi, bu çıktı kötü amaçlı yazılım analizi için değerlidir çünkü kötü amaçlı yazılımın bildirdiği ve başvuracağı zengin bir işlev dizisini listeler. Örneğin, çıktının ilk birkaç satırı bize kötü amaçlı yazılımın WriteFile (1) kullanarak dosyalara yazacağını, CreateFileA çağrısını (2) kullanarak dosyaları açacağını ve CreateProcessA (3) kullanarak yeni işlemler oluşturacağını söyler. Bu, kötü amaçlı yazılım hakkında oldukça temel bilgiler olsa da, kötü amaçlı yazılımın davranışını daha ayrıntılı olarak anlamak için bir başlangıçtır.

\*Examining Malware Images:Kötü Amaçlı Yazılım Görüntüleri İnceleme:

kötü amaçlı yazılım üzerindeki .rsrc ile yazılımı anlamak için bakalım . Örneğin, kötü amaçlı yazılım ikili dosyaları, genellikle Word belgeleri, oyun yükleyicileri, PDF dosyaları vb. Ayrıca, kötü amaçlı yazılımda, ağ saldırı araçları ve saldırganlar tarafından güvenliği ihlal edilmiş makinelerin uzaktan kontrolü için çalıştırılan programlar gibi, saldırganların ilgisini çeken programları öneren görüntüler de bulabilirsiniz.Hatta cihatçıların masaüstü simgelerini, kötü görünümlü siberpunk çizgi film karakterlerinin resimlerini ve Kalaşnikof tüfek resimlerini içeren ikili dosyalar bile gördüm.Örnek görüntü analizimiz için, güvenlik şirketi Mandiant'ın Çin devletinin sponsorluğundaki bir bilgisayar korsanlığı grubu tarafından hazırlanmış olarak tanımladığı bir kötü amaçlı yazılım örneğini ele alalım. Bu kötü amaçlı yazılım örneğini, bu bölümün veri dizininde fakepdfmalware.exe adı altında bulabilirsiniz. Bu örnek, kullanıcıları, aslında kötü niyetli bir PE yürütülebilir dosya olsa da, bunun bir Adobe Acrobat belgesi olduğunu düşünmeleri için kandırmak için bir Adobe Acrobat simgesi kullanır.

Fakepdfmalware.exe ikili dosyasından görüntüleri Linux komut satırı aracı wrestool kullanarak ayıklamadan önce, çıkaracağımız görüntüleri tutmak için bir dizin oluşturmamız gerekir. 16 numaralı liste tüm bunların nasıl yapılacağını gösterir.

KOMUTLAR:

mkdir images

$ wrestool –x fakepdfmalware.exe –output=images

$ icotool –x –o images images/\*.ico

Çıkarılan görüntüleri tutacak bir dizin oluşturmak için ilk olarak mkdir resimlerini kullanırız. Daha sonra, fakepdfmalware.exe'den / images klasörüne görüntü kaynaklarını (-x) ayıklamak için wrestool kullanıyoruz ve ardından Adobe .ico simge formatındaki herhangi bir kaynağı çıkartmak (-x) ve png grafiklerine dönüştürmek (-o) için standart resim görüntüleme araçlarını kullanarak bunları görüntüleyebiliriz. Sisteminizde wrestool kurulu değilse, http: // www .nongnu.org / icoutils / adresinden indirebilirsiniz. Yürütülebilir hedefteki görüntüleri PNG formatına dönüştürmek için wrestool'u kullandıktan sonra, bunları en sevdiğiniz resim görüntüleyicide açabilmeli ve çeşitli çözünürlüklerde Adobe Acrobat simgesini görebilmelisiniz. Buradaki örneğimin gösterdiği gibi, PE dosyalarından görüntüleri ve simgeleri çıkarmak nispeten basittir ve kötü amaçlı yazılım ikili dosyaları hakkında ilginç ve yararlı bilgileri hızlı bir şekilde ortaya çıkarabilir. Benzer şekilde, daha fazla bilgi için yazdırılabilir dizeleri kötü amaçlı yazılımlardan kolayca çıkarabiliriz ve bunu daha sonra yapacağız. yazdırılabilir dize strings

\*Examining Malware Strings:Kötü Amaçlı Yazılım Dizelerini İnceleme

Dizeler, bir program ikili dosyası içindeki yazdırılabilir karakter dizileridir. Kötü amaçlı yazılım analistleri, içinde neler olup bittiğini hızlı bir şekilde anlamak için genellikle kötü amaçlı bir örnekteki dizelere güvenir. Bu dizeler genellikle web sayfalarını ve dosyaları, IP adreslerini ve kötü amaçlı yazılımın hangi adreslere bağlandığını size söyleyen ana bilgisayar adlarını ve benzerlerini indiren HTTP ve FTP komutları gibi şeyler içerir. Bazen dizeleri yazmak için kullanılan dil bile bir kötü amaçlı yazılım ikili dosyasının menşe ülkesini ima edebilir, ancak bu sahte olabilir.Kötü niyetli bir ikili dosyanın amacını leetspeak'te açıklayan bir dizede metin bile bulabilirsiniz. Dizeler ayrıca bir ikili dosya hakkında daha fazla teknik bilgi ortaya çıkarabilir. Örneğin, onu oluşturmak için kullanılan derleyici, ikilinin yazıldığı programlama dili, gömülü komut dosyaları veya HTML vb. Hakkında bilgi bulabilirsiniz. Kötü amaçlı yazılım yazarları tüm bu izleri gizleyebilse, şifreleyebilse ve sıkıştırabilse de, gelişmiş kötü amaçlı yazılım yazarları bile çoğu zaman en azından bazı izleri açığa çıkararak kötü amaçlı yazılımları analiz ederken dize dökümlerini incelemeyi özellikle önemli hale getirir.

NOT: yani gizleniyormuş genelde ama unuttukları  için bakmak gerekir.

\*Using strings Program:

Bir dosyadaki tüm dizeleri görüntülemenin standart yolu, aşağıdaki sözdizimini kullanan komut satırı araç dizelerini kullanmaktır:

$ strings filepath | less

Bu komut bir dosyadaki tüm dizeleri terminale satır satır yazdırır. "|" Ekleniyor sonunda daha az, dizelerin terminal boyunca kaymasını engeller. Varsayılan olarak, strings komutu minimum 4 bayt uzunluğa sahip tüm yazdırılabilir dizeleri bulur, ancak komutlar kılavuz sayfasında listelendiği gibi farklı bir minimum uzunluk ayarlayabilir ve çeşitli diğer parametreleri değiştirebilirsiniz. Ben sadece 4 varsayılan minimum dizi uzunluğunu kullanmanızı tavsiye ederim, ancak minimum dizi uzunluğunu –n seçeneğini kullanarak değiştirebilirsiniz. Örneğin, dizeler –n 10 dosya yolu yalnızca minimum 10 bayt uzunluğundaki dizeleri çıkarır.

bu bayt işini anlamadım -n 10 diyince ilk baştan 10 kısmı getirmiyor mu yoksa bayt o kadar olanı tamamlayan stringlerimi baştan sonra yazıyor ??

\*Analyzing Your strings Dump: Dizelerinizin dökümünü analiz etme

Artık bir kötü amaçlı yazılım programının yazdırılabilir dizelerini attığımıza göre, asıl zorluk dizelerin ne anlama geldiğini anlamaktır. Örneğin, bu bölümde daha önce pefile kitaplığını kullanarak incelediğimiz ircbot.exe için dizeleri ircbotstring.txt dosyasına döktüğümüzü varsayalım:

$ strings ircbot.exe > ircbotstring.txt

İrcbotstring.txt dosyasının içeriği binlerce satırlık metin içerir, ancak bu satırlardan bazıları dışarı çıkmalıdır. Örneğin, Liste 1-7, DOWNLOAD kelimesiyle başlayan dize dökümünden çıkarılan bir grup satırı gösterir.

[DOWNLOAD]: Bad URL, or DNS Error: %s.

[DOWNLOAD]: Update failed: Error executing file: %s.

[DOWNLOAD]: Downloaded %.1fKB to %s @ %.1fKB/sec. Updating.

[DOWNLOAD]: Opened: %s.

--snip--

[DOWNLOAD]: Downloaded %.1f KB to %s @ %.1f KB/sec.

[DOWNLOAD]: CRC Failed (%d != %d).

[DOWNLOAD]: Filesize is incorrect: (%d != %d).

[DOWNLOAD]: Update: %s (%dKB transferred).

[DOWNLOAD]: File download: %s (%dKB transferred).

[DOWNLOAD]: Couldn't open file: %s.

Liste 1-7: Kötü amaçlı yazılımın saldırgan tarafından belirtilen dosyaları bir hedef makineye indirebileceğine dair kanıt gösteren dizeler çıktısı

Bu satırlar, ircbot.exe'nin bir saldırgan tarafından belirtilen dosyaları hedef makineye indirmeyi deneyeceğini gösterir. Başka birini analiz etmeyi deneyelim. Liste 1-8'de gösterilen dize dökümü, ircbot.exe'nin saldırgandan gelen bağlantılar için hedef makinede dinleyen bir web sunucusu görevi görebileceğini belirtir.

(1) GET

(2) HTTP/1.0 200 OK

Server: myBot

Cache-Control: no-cache,no-store,max-age=0

pragma: no-cache

Content-Type: %s

Content-Length: %i

Accept-Ranges: bytes

Date: %s %s GMT

Last-Modified: %s %s GMT

Expires: %s %s GMT

Connection: close

HTTP/1.0 200 OK

(3) Server: myBot

Cache-Control: no-cache,no-store,max-age=0

pragma: no-cache

Content-Type: %s

Accept-Ranges: bytes

Date: %s %s GMT

Last-Modified: %s %s GMT

Expires: %s %s GMT

Connection: close

HH:mm:ss

ddd, dd MMM yyyy

application/octet-stream

text/html

Liste 1-8: Kötü amaçlı yazılımın, saldırganın bağlanabileceği bir HTTP sunucusuna sahip olduğunu gösteren dizeler çıktısı

1-8 arası liste, ircbot.exe tarafından bir HTTP sunucusu uygulamak için kullanılan çok çeşitli HTTP standart levhalarını gösterir. Bu HTTP sunucusunun, saldırganın, kurbanın masaüstünün ekran görüntüsünü alıp saldırgana geri gönderme komutu gibi komutlar vermek için HTTP aracılığıyla bir hedef makineye bağlanmasına izin vermesi muhtemeldir. Giriş boyunca HTTP işlevselliğinin kanıtlarını görüyoruz. Örneğin, GET yöntemi (1) bir internet kaynağından veri ister. HTTP / 1.0 200 OK (2) satırı, bir HTTP ağ işlemiyle her şeyin yolunda gittiğini belirten 200 durum kodunu döndüren bir HTTP dizesidir ve Sunucu: myBot (3), HTTP sunucusunun adının myBot olduğunu belirtir, ircbot.exe'nin yerleşik bir HTTP sunucusuna sahip olduğu bir bilgi. Tüm bu bilgiler, belirli bir kötü amaçlı yazılım örneğini veya kötü amaçlı kampanyayı anlamak ve durdurmak için yararlıdır. Örneğin, bir kötü amaçlı yazılım örneğinin, bağlandığınızda belirli dizeleri üreten bir HTTP sunucusuna sahip olduğunu bilmek, virüslü ana bilgisayarları belirlemek için ağınızı taramanıza olanak tanır.

\*Summary : özet

Bu bölümde, bir kötü amaçlı yazılım programını çalıştırmadan incelemeyi içeren statik kötü amaçlı yazılım analizine ilişkin üst düzey bir genel bakış elde edeceksiniz. Windows .exe ve .dll dosyalarını tanımlayan PE dosya biçimini öğrendiniz ve gerçek bir kötü amaçlı yazılım olan ircbot.exe ikili dosyasını incelemek için Python kitaplık dosyasını nasıl kullanacağınızı öğrendiniz. Kötü amaçlı yazılım örneklerinden daha fazla bilgi elde etmek için görüntü analizi ve dizi analizi gibi statik analiz tekniklerini de kullandınız. Bölüm 2, kötü amaçlı yazılımdan kurtarılabilen derleme kodunu analiz etmeye odaklanarak statik kötü amaçlı yazılım analizi tartışmamıza devam ediyor.