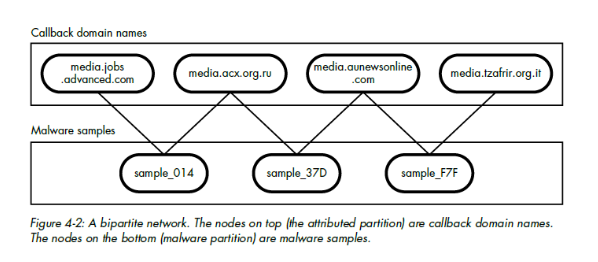
4.Identifying Attack Campaigns Using Malware Networks:Kötü Amaçlı Yazılım Ağlarını Kullanarak Saldırı Kampanyalarını Tanımlama

Kötü amaçlı yazılım ağ analizi malware network , kötü amaçlı yazılım veri kümelerini değerli tehdit istihbaratına dönüştürebilir, düşman saldırı kampanyalarını , yaygın kötü amaçlı yazılım  taktiklerini ve kötü amaçlı yazılım örnekleri kaynaklarını ortaya çıkarabilir. Bu bölüm sonunda malwarelardan tehdit istihbaratını çıkarmakla ilgili olduğu için ağ analizinni teorisinin temelleri, malwarelar arasındaki ilişkileri belirlemek için görselleştirmeleri kullanmanın yolları, veri analizi ve görselleştirme için python ve çeşitli açık kaynak araç takımları kullanıalrak kötü amaçlı yazılım ağlarından istihbarat nasıl oluşturulur, görselleştirilir ve çıkarılır, Gerçek dünyadaki kötü amaçlı yazılım veri kümelerindeki saldırı kampanyalarını ortaya çıkarmak ve analiz etmek için tüm bu bilgiler nasıl bir araya getirilir. bunlara bakalım.

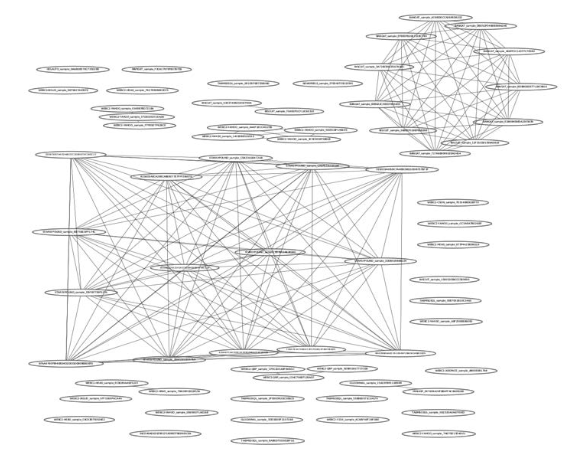
->Nodes and Edges:Düğümler ve Kenarlar: Ağlar network, bağlı nesnelerin koleksiyonudur.(düğüm adı verilir). Bu düğümler arasındaki bağlantılara kenarlar denir. amaçlarımız için önemsediğimiz şey bu düğümler ve kenarlar arasındaki ara bağlantıların yapısıdır. Çünkü bu kötü amaçlı yazılımlar hakkında ayrıntılı bilgi verebilir. Kötü amaçlı yazılımları analiz etmek için ağları kullanırken, her bir kötü amaçlı yazılım dosyasını bir düğümün tanımı olarak ele alabiliriz ve ilgili ilişkileri (paylaşılan kod veya ağ davranışı gibi) bir uç tanımı olarak ele alabiliriz. Benzer kötü amaçlı yazılım dosyaları kenarları paylaşır ve bu nedenle zorla yönlendirilmiş ağları uyguladığımızda birlikte kümelenir (bunun tam olarak nasıl çalıştığını daha sonra göreceksiniz)

İki taraflı ağ, bipartite networks düğümleri iki bölüme (gruplara) bölünebilen ağdır ve hiçbir bölüm dahili bağlantılar içermemektedir. Bu tür ağlar, kötü amaçlı yazılım örnekleri arasında paylaşılan öznitelikleri göstermek için kullanılabilir.

Gördüğünüz gibi, bu kadar basit bir görselleştirme bile önemli bir zeka parçasını ortaya çıkarır: kötü amaçlı yazılım örneklerinin paylaşılan geri arama sunucularına dayanarak, sample\_014'ün muhtemelen sample\_37D ile aynı saldırgan tarafından konuşlandırıldığını tahmin edebiliriz. Ayrıca, sample\_37D ve sample\_F7F'nin muhtemelen aynı saldırganı paylaştığını ve bu sample\_014 ve sample\_F7F'nin muhtemelen aynı saldırganı paylaştığını tahmin edebiliriz, çünkü sample\_37D ile bağlantılılar (ve aslında, Şekil 4-2'de gösterilen örneklerin hepsi aynı "APT1" Çinli saldırgan grubu).



  Şekil 4-3, daha önce atıfta bulunulan tüm Çin APT1 veri kümesinin paylaşılan geri arama sunucularının öngörülen ağını göstermektedir. shared-callback servers



Şekil 4-3: APT1 veri kümesinden gelen kötü amaçlı yazılım örneklerinin bir projeksiyonu, yalnızca en az bir sunucuyu paylaşıyorlarsa kötü amaçlı yazılım örnekleri arasındaki bağlantıları gösterir.

İki büyük küme clusters , iki farklı saldırı kampanyasında campaigns kullanıldı.

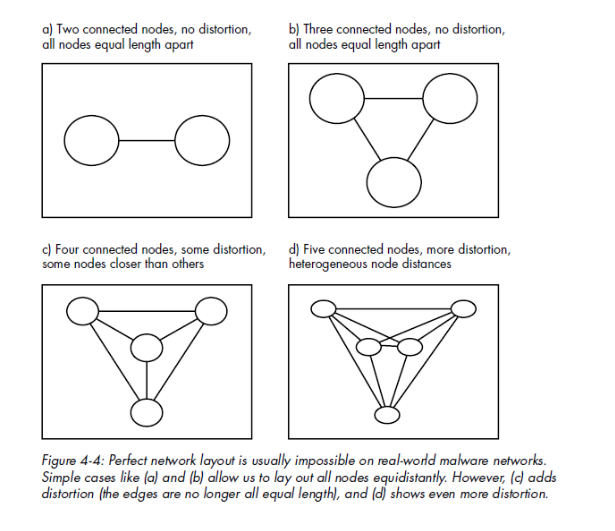
Şekil 4-3'te görebileceğiniz gibi, iki büyük gruplama mevcuttur (sol orta alandaki büyük kare küme ve sağ üst alandaki dairesel küme), bunlar daha fazla incelemenin ardından yürütülen iki farklı kampanyaya karşılık gelir.

-->Visualizing Malware Networks:Kötü Amaçlı Yazılım Ağlarını Görselleştirme

ağ görselleştirmede en büyük sorun onları nasıl yerleştireceğimize karar vermektir. bunun için en ideal yol onları birbirlerinden görsel uzaklıkları ağdaki en kısa yol mesafesi ile orantılı olacak şekilde koordinat boşluğuna yerleştirmektir.Başka bir deyişle, birbirinden iki atlama uzaklıkta olan düğümler, birbirinden yaklaşık iki inç uzakta olabilir ve üç atlama uzaklığındaki düğümler yaklaşık üç inç uzakta olabilir. Bunu yapmak, benzer düğüm kümelerini gerçek ilişkilerine göre doğru bir şekilde görselleştirmemize olanak tanır. Ancak bir sonraki bölümde göreceğiniz gibi, özellikle üçten fazla düğümle çalışırken bunu başarmak genellikle zordur.

\*The Distortion Problem:Bozulma Sorunu

bu ağ düzeni sorununu mükemmel bir şekilde çözmek çoğu zaman imkansızdır. bu bozulma minimize edebiliriz.

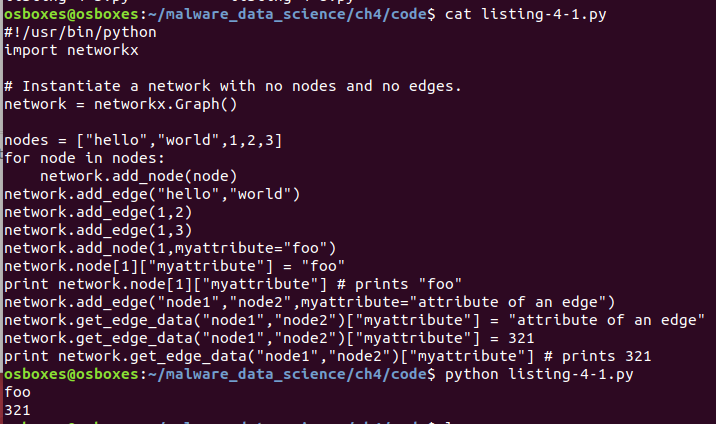


\*Force-Directed Algorithms:Kuvvet Yönlendirmeli Algoritmalar:

Yerleşim bozulmasını en iyi şekilde en aza indirmek için, bilgisayar bilimcileri genellikle kuvvet yönlendirmeli düzen algoritmaları kullanır.  Kuvvet yönelimli algoritmalar, yay benzeri kuvvetlerin ve manyetizmanın fiziksel simülasyonlarına dayanır.

-->Building Networks with NetworkX:

artık temel bir anlayışa sahip olduğumuza göre açık kaynaklı NetworkX :Python ağ analizi kitaplığını GraphViz açık kaynak ağ görselleştirme araç setini kullanarak malwarelar arasındaki ilişkilerin ağlarını nasıl oluşturacağımıza bakalım:



network = networkx.Graph() ile ağ oluşturursun.

add\_edge ()  ile kenar eklersin.

NetworkX, özellikleri hem düğümlere hem de kenarlara kolayca eklememizi sağlar. Bir düğüme bir öznitelik eklemek (ve bu özniteliğe daha sonra erişmek için), düğümü ağa eklediğinizde özniteliği bir anahtar sözcük bağımsız değişkeni olarak ekleyebilirsiniz, örneğin:network.add\_node(1,myattribute="foo")

Daha sonra bir öznitelik eklemek için, aşağıdaki sözdizimini kullanarak ağın düğüm sözlüğüne erişin:network.node[1]["myattribute"] = "foo"

Ardından düğüme erişmek için düğüm sözlüğüne erişin:

print network.node[1]["myattribute"] # prints "foo"

Düğümlerde  olduğu gibi, burada gösterildiği gibi, başlangıçta kenarları eklediğinizde anahtar sözcük bağımsız değişkenlerini kullanarak kenarlara nitelikler ekleyebilirsiniz:

network.add\_edge("node1","node2",myattribute="attribute of an edge")

Benzer şekilde, burada gösterildiği gibi, edge  sözlüğünü kullanarak bir ağa eklendikten sonra kenarlara öznitelikler ekleyebilirsiniz:

network.edge["node1"]["node2"]["myattribute"] = "attribute of an edge"

Kenar sözlüğü, Liste 4-2'de gösterildiği gibi, ilk olarak hangi düğüme başvurduğunuz konusunda endişelenmenize gerek kalmadan, düğüm niteliklerine tam tersi şekilde erişmenizi sağlaması açısından güzeldir.

network.edge["node1"]["node2"]["myattribute"] = 321

print network.edge["node2"]["node1"]["myattribute"] # prints 321

yani hem atama işlemini yapabilir hemde atadığını tekrar yazdırma işlemini yapabilirsin.

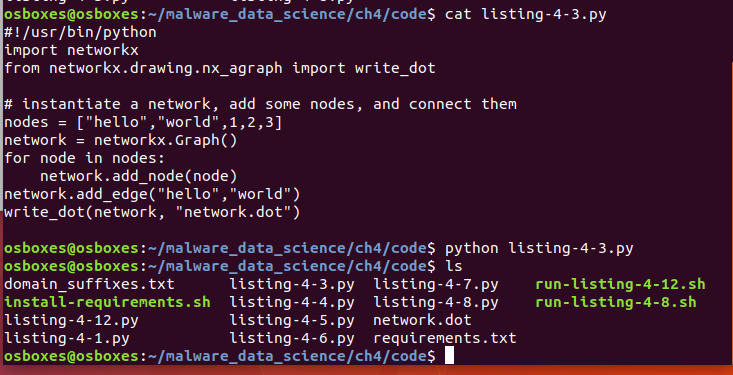
diske kaydetme işlemini için NetworkX write\_dot () işlevini çağırmanız yeterlidir.

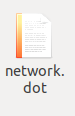
->Network Visualization with GraphViz: GraphViz ile Ağ Görselleştirme:

GraphViz, ağlarınızı görselleştirmek için mevcut en iyi komut satırı paketidir.Her GraphViz komut satırı aracı .dot biçiminden .png ye çevirir.

$ <toolname> <dotfile> -T png –o <outputfile.png>

\*Fdp kuvvet yönelimli force-directed grafik oluşturucu, bir GraphViz ağ görselleştirme aracıdır:$ fdp apt1callback.dot –T png –o apt1callback.png





14.PNG

GraphViz araçları, ağlarınızın çizilme şeklini ayarlamak için kullanabileceğiniz birçok parametre içerir. Bu parametrelerin çoğu, aşağıdaki formatta –G komut satırı bayrağı kullanılarak ayarlanır:

G<parametername>=<parametervalue>

Özellikle kullanışlı olan iki parametre örtüşme ve eğrilerdir. GraphViz'e herhangi bir düğümün birbiriyle çakışmasına izin vermemesini söylemek için örtüşmeyi false olarak ayarlayın. Ağlarınızdaki kenarları takip etmeyi kolaylaştırmak için GraphViz'e düz çizgiler yerine eğri çizgiler çizmesini söylemek için splines parametresini kullanın. Aşağıda, GraphViz'de örtüşme ve eğri parametrelerini ayarlamanın bazı yolları verilmiştir. Düğümlerin çakışmasını önlemek için aşağıdakileri kullanın:

$ <toolname> <dotfile> -Goverlap=false -T png -o <outputfile.png>

Ağ okunabilirliğini iyileştirmek için kenarları eğri çizgiler (spline) olarak çizin:

$ <toolname> <dotfile> -Gsplines=true -T png -o <outputfile.png>

Ağın okunabilirliğini artırmak için kenarları eğri çizgiler (spline'lar) olarak çizin ve düğümlerin görsel olarak örtüşmesine izin vermeyin:

$ <toolname> <dotfile> -Gsplines=true –Goverlap=false -T png -o <outputfile.png>

Sadece bir parametreyi birbiri ardına listelediğimizi unutmayın:

Gsplines = true –Goverlap = false (sıralama önemli değildir), ardından -T png -o <outputfile.png> gelir.

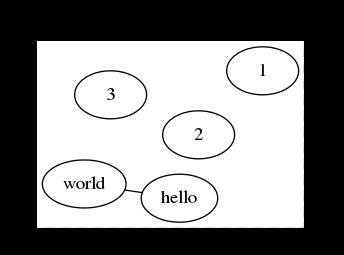
-->GraphViz araçlarından bazıları :

\*fdp:

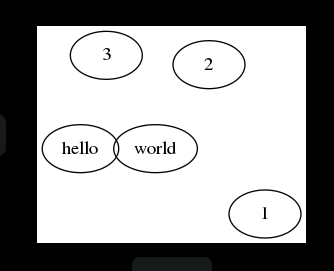
500 den az düğüme sahip malware network oluştururken fdp kısa sürede yapar. ancak fazla düğümlerde yavaşlar.

$ fdp callback\_servers\_malware\_projection.dot -T png -o fdp\_servers.png –Goverlap=false

şeklinde deneyebilirsiniz.



\*sfdp:



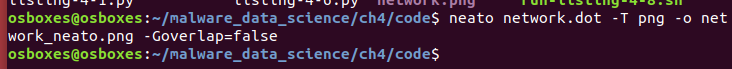
17.PNG

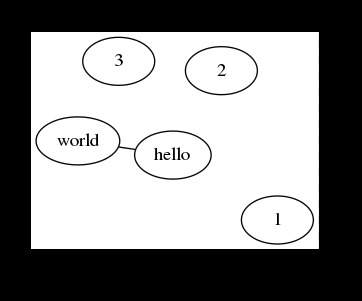
fdp ile hemen hemen aynı yaklaşımı kullanır , ancak daha iyi ölçeklenir çünkü düğümlerin yakınlıklarına göre süper düğümlerle birleştirildiği, kabalaştırmalar coarsening  olarak bilinen bir basitleştirme hiyerarşisi oluşturur.Sfdp aracı, kabalaştırmalarını tamamladıktan sonra, grafiğin çok daha az düğüm ve kenara sahip birleştirilmiş sürümlerini düzenler ve bu da yerleşim sürecini önemli ölçüde hızlandırır. Bu şekilde sfdp, ağdaki en iyi konumları bulmak için daha az hesaplama gerçekleştirebilir. Sonuç olarak, sfdp tipik bir dizüstü bilgisayarda on binlerce düğümü yerleştirebilir ve bu da onu çok büyük kötü amaçlı yazılım ağlarını yerleştirmek için açık ara en iyi algoritma yapar. Ancak bu ölçeklenebilirliğin bir bedeli vardır: sfdp, bazen fdp'deki aynı boyutlu ağların düzenlerinden daha az net olan düzenler üretir.

Gördüğünüz gibi, her bir kümenin üzerinde biraz daha fazla gürültü var, bu da neler olduğunu görmeyi biraz daha zorlaştırıyor. Bu ağı oluşturmak için, bu kitaba eşlik eden verilerin ch4 dizinini girin ve ardından Şekil 4-6'da gösterilen sfdp\_servers.png görüntü dosyasını oluşturmak için aşağıdaki kodu girin:

$ sfdp callback\_servers\_malware\_projection.dot -T png -o sfdp\_servers.png –Goverlap=false

\*neato:





Neato aracı, nesneleri ideal konumlara itmeye yardımcı olmak için tüm düğümler arasında (bağlı olmayan düğümler dahil) simüle edilmiş yaylar oluşturan farklı bir kuvvet yönelimli ağ düzeni algoritmasının GraphViz uygulamasıdır, ancak ek hesaplama pahasına. Neato'nun belirli bir ağ için en iyi düzeni ne zaman üreteceğini bilmek zor.Şekilde,APT1 paylaşılan geri arama sunucusu ağında neato düzeninin nasıl göründüğünü gösterir.

Gördüğünüz gibi, bu durumda neato, fdp ve sfdp tarafından üretilenlere benzer bir ağ düzeni üretir. Bununla birlikte, bazı veri kümeleri için, neato'nun daha iyi veya daha kötü bir düzen ürettiğini göreceksiniz - sadece veri kümenizle denemeniz ve görmeniz gerekir. Neato'yu denemek için, bu kitaba eşlik eden verilerin ch4 dizininden aşağıdakileri girin; bu, Şekil 4-7'de gösterilen neato\_servers.png ağ görüntü dosyasını oluşturmalıdır:

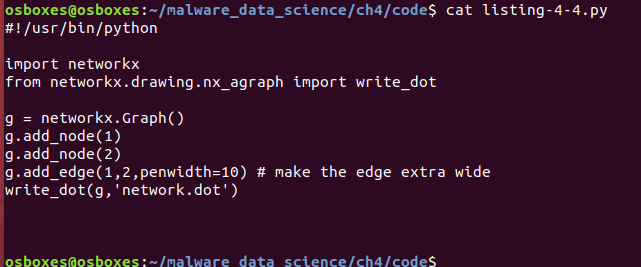
$ neato callback\_servers\_malware\_projection.dot -T png -o neato\_servers.png –Goverlap=false

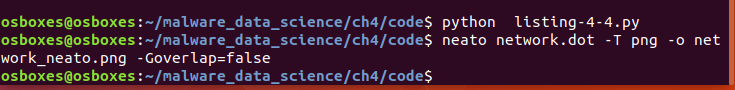
->Adding Visual Attributes to Nodes and Edges:Düğümlere ve Kenarlara Görsel Nitelikler Ekleme:

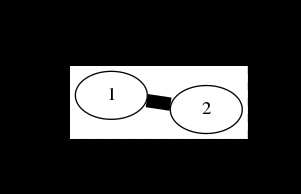
Genel ağ düzeninize karar vermenin ötesinde, tek tek düğümlerin ve kenarların nasıl işleneceğini belirtmek faydalı olabilir. Örneğin, iki düğüm arasındaki bağlantının gücüne göre kenar kalınlığını ayarlamak veya düğüm rengini her kötü amaçlı yazılım örnek düğümünün ilişkili olduğu tehlikeye göre ayarlamak isteyebilirsiniz; bu, kötü amaçlı yazılım kümelerini daha iyi görselleştirmenize olanak tanır. NetworkX ve GraphViz, basitçe bir öznitelik kümesine değerler atayarak düğümlerin ve kenarların görsel özniteliklerini belirlemenize izin vererek bunu yapmayı kolaylaştırır. .

\*Edge Width:

GraphViz'in düğümlerin etrafına çizdiği sınırın genişliğini veya kenarlar için çizdiği çizgiyi ayarlamak için, Liste 4-4'te gösterildiği gibi düğümlerin ve kenarların penwidth özniteliğini istediğiniz bir sayıya ayarlayabilirsiniz.



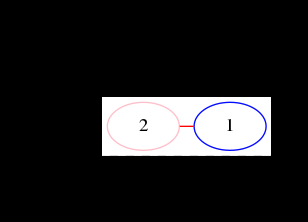


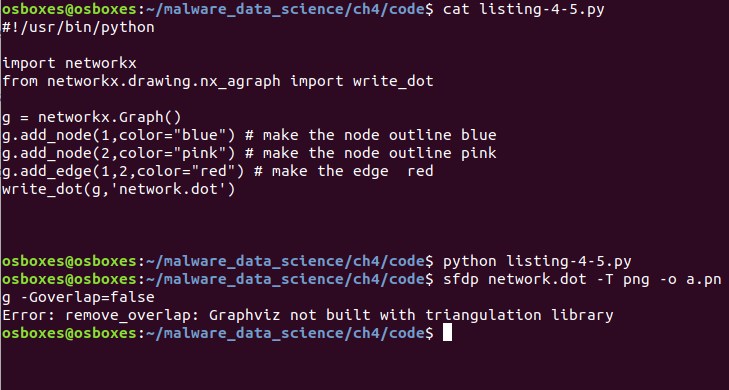


Şekilde görebileceğiniz gibi, 10'luk bir penwidth çok kalın bir kenarla sonuçlanır.

\*Node and Edge Color:

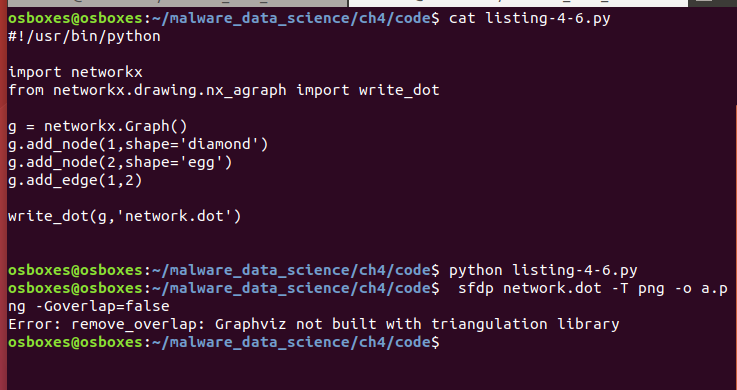
Bir düğümün kenarlığının veya kenarının rengini ayarlamak için renk özelliğini kullanımına bakalım.

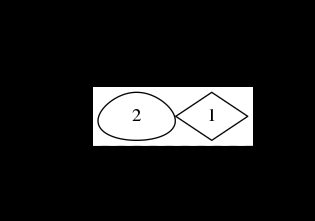
Kullanabileceğiniz tam bir renk listesi için http://www.graphviz.org/doc/ info / colours.html adresine bakın.



\*Node Shape:Düğüm Şekli

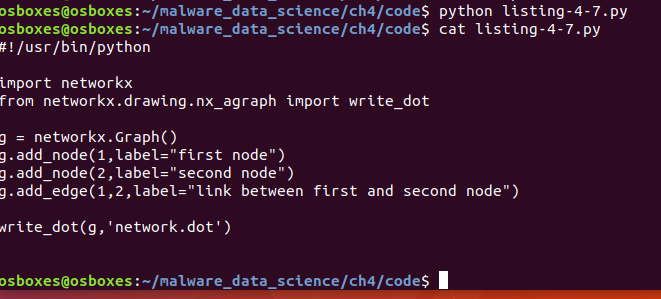
Bir düğümün şeklini ayarlamak için, http://www.GraphViz.org/doc/info/s apes.html adresinde tanımlandığı gibi, bir şekil belirten bir dizeyle şekil özniteliğini kullanın. Yaygın olarak kullanılan değerler kutu, elips, daire, egg, diamond, üçgen, beşgen ve altıgendir. 4-6 listesi, bir düğümün şekil özniteliğinin nasıl ayarlanacağını gösterir.

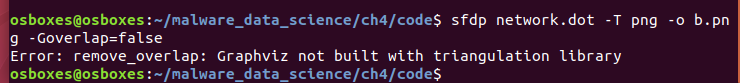


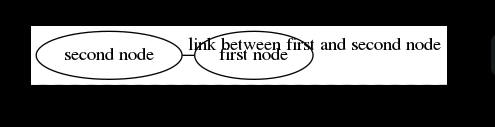


\*Text Labels:

Son olarak, GraphViz ayrıca etiket özniteliğiyle düğümlere ve kenarlara etiket eklemenize izin verir. Düğümler, atanan kimliklerine göre otomatik olarak etiketlense de (örneğin, 123 olarak eklenen bir düğümün etiketi 123 olacaktır), label= <etiketim özniteliğim> kullanarak etiketleri belirtebilirsiniz. Düğümlerden farklı olarak, kenarlar varsayılan olarak etiketlenmez, ancak labelözelliğini kullanarak bunlara etiketler atayabilirsiniz.



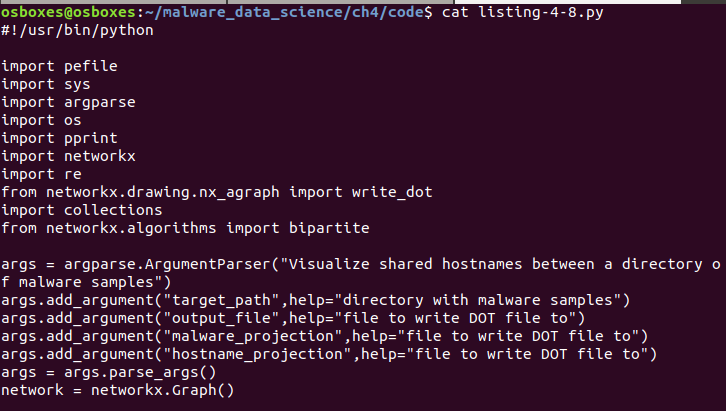


Artık düğümlerin ve kenarların temel özelliklerini nasıl kullanacağınızı bildiğinize göre,

ağları sıfırdan oluşturmaya hazırsınız.

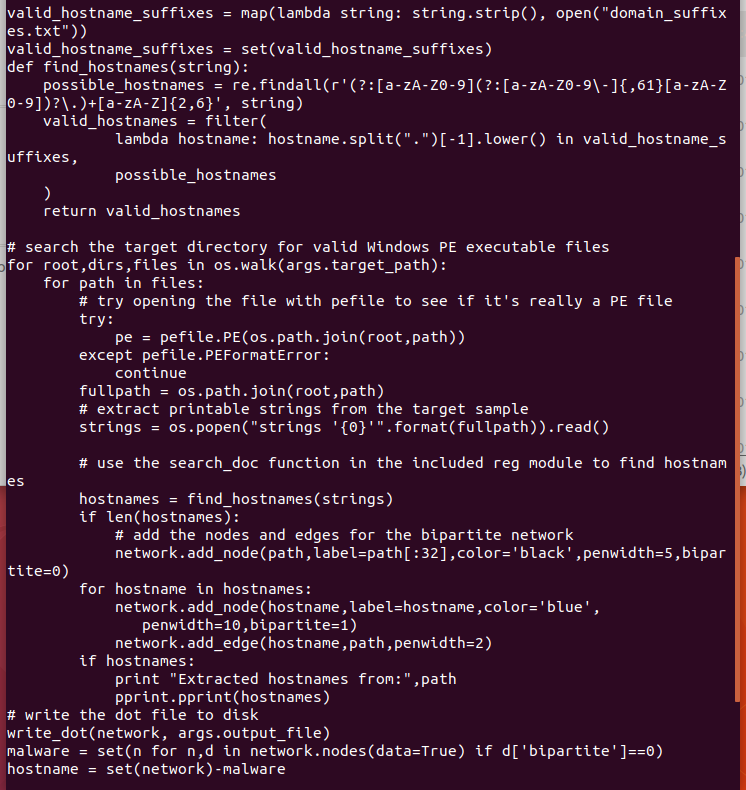
-->Building Malware Networks:

Aşağıdaki program, geri arama etki alanı adlarını kötü amaçlı yazılım dosyalarından çıkarır ve ardından kötü amaçlı yazılım örneklerinden oluşan iki taraflı bir ağ oluşturur. Daha sonra, hangi kötü amaçlı yazılım örneklerinin ortak geri arama sunucularını paylaştığını göstermek için ağın bir çizimini gerçekleştirir ve yaygın kötü amaçlı yazılım örnekleri tarafından hangi geri arama sunucularının çağrıldığını göstermek için başka bir çizim gerçekleştirir. Son olarak, program üç ağı GraphViz ile görselleştirilebilmeleri için dosyalar olarak kaydeder. Kodun tamamı, bu kitaba eşlik eden verilerde ch4 / callback\_server\_network.py dosya yolunda bulunabilir. Gerekli modülleri içe aktararak nasıl başlayacağınızı gösterir.

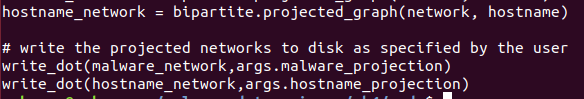


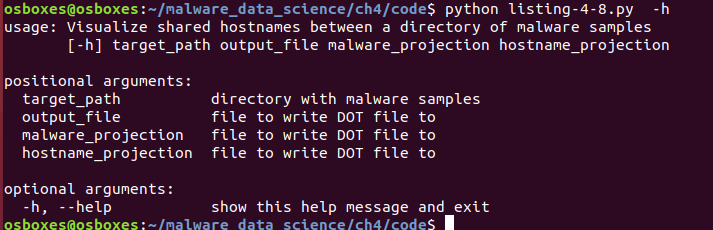
İçe aktardığımız gerekli modüllerden en önemlileri, hedef PE ikili dosyalarını ayrıştırmak için kullandığımız pefile PE ayrıştırma modülü  ve kötü amaçlı yazılım öznitelik ağını oluşturmak için kullandığımız networkx kitaplığıdır . Sonra komut satırı argümanlarını ayrıştırıyoruz.

Bu argümanlar arasında target\_path  (analiz ettiğimiz kötü amaçlı yazılımın bulunduğu dizine giden yol), output\_file  (tüm ağı yazdığımız yol), malware\_projection (indirgenmiş bir grafiğin sürümü ve hangi kötü amaçlı yazılım örneklerinin öznitelikleri paylaştığını gösterir) ve resource\_projection (grafiğin azaltılmış bir sürümünü yazdığımız ve kötü amaçlı yazılım örneklerinde hangi özniteliklerin birlikte görüldüğünü gösterdiğimiz yol). Artık programın özüne geçmeye hazırız.  Program için bir ağ oluşturma koduna bakalım.



Önce networkx.Graph () yapıcısını çağırarak yeni bir ağ oluşturuyoruz. Ardından stringlerden  ana bilgisayar adlarını çıkaran find\_hostnames () işlevini tanımlarız. Bu fonksiyonun ne yaptığı hakkında çok fazla endişelenmeyin. Bu aslında bir normal ifade ve etki alanlarını tanımlamak için elinden gelenin en iyisini yapmaya çalışan bir dizi filtreleme kodudur. Daha sonra, hedef dizindeki tüm dosyaları yineleyerek, pefile.PE sınıfının bunları yükleyip yükleyemediğini görerek bunların PE dosyası olup olmadığını kontrol ederiz (yoksa, dosyaları analiz etmeyiz). Son olarak, önce dosyadan  tüm yazdırılabilir dizeleri çıkararak ve ardından ana bilgisayar adı kaynakları  için stringsleri arayarak ana bilgisayar adı niteliklerini mevcut dosyadan çıkarıyoruz. Herhangi birini bulursak, bunları ağımıza düğümler olarak ekleriz ve ardından mevcut kötü amaçlı yazılım örneği için düğümden ana bilgisayar adı kaynak düğümlerine kenarlar ekleriz. Şimdi, Liste 4-11'de gösterildiği gibi programı tamamlamaya hazırız.





Ağımızı komut satırı argümanlarında  belirtilen konumda diske yazarak başlıyoruz. Ardından, kötü amaçlı yazılım ilişkilerini ve ana bilgisayar adı kaynak ilişkilerini gösteren iki azaltılmış ağı oluşturuyoruz. Bunu, önce kötü amaçlı yazılım düğümlerinin kimliklerini içeren bir Python seti ve kaynak düğümlerinin kimlikleri için başka bir Python seti oluşturarak yapıyoruz. Daha sonra kötü amaçlı yazılım ve kaynak kümeleri için projeksiyonlar almak için NetworkXspecific projected\_graph () işlevini  kullanırız ve bu ağları belirtilen konumlardaki  diske yazarız. Ve bu kadar! Dosyalara gömülü paylaşılan ana bilgisayar adı kaynakları arasındaki kötü amaçlı yazılım ilişkilerini görmek için bu programı bu kitaptaki kötü amaçlı yazılım veri kümelerinin herhangi birinde kullanabilirsiniz. Bu analiz modu aracılığıyla hangi tehdit istihbaratını toplayabileceğinizi görmek için kendi veri kümelerinizde bile kullanabilirsiniz.

!!!BUNUN UYGULAMASINA BİR BAK

network.dot malware.dot hostname.dot olmak üzere 3 tane dosyamız olacak.

->Building a Shared Image Relationship Network:Paylaşılan Bir Görüntü İlişkisi Ağı Oluşturmak:

Kötü amaçlı yazılımları, paylaşılan geri arama sunucularına göre analiz etmenin yanı sıra, bunları, paylaşılan simgeler ve diğer grafik varlıkları kullanımlarına göre de analiz edebiliriz. Örneğin, Şekil 4-12 ch4 / data / Truva atlarında bulunan Truva atları için paylaşılan görüntü analizi sonuçlarının bir bölümünü gösterir.

Tüm bu Truva atlarının arşiv dosyaları gibi göründüğünü ve çalıştırılabilir olsalar bile aynı arşiv dosyası simgesini (şeklin ortasında gösterilir) kullandıklarını görebilirsiniz. Kullanıcıyı oyun oynama çabalarının bir parçası olarak tamamen aynı görüntüyü kullanmaları, muhtemelen aynı saldırgandan geldiklerini gösterir. Bunu, hepsine aynı aile adını (ArchSMS) atayan Kaspersky antivirüs motoru aracılığıyla kötü amaçlı yazılım örneklerini çalıştırarak doğruladım.

Daha sonra, kötü amaçlı yazılım örnekleri arasındaki paylaşılan görüntü ilişkilerini görmek için size  gösterilen görselleştirme türünü nasıl oluşturacağınızı göstereceğim. Görüntüleri kötü amaçlı yazılımdan çıkarmak için, image\_network.py programını oluşturmak için wrestool'a  dayanan yardımcı kitaplık görüntülerini kullanıyoruz. Wrestool'un Windows çalıştırılabilir dosyalarından görüntüleri çıkardığını hatırlayın.

Kodun  gösterilen ilk bölümünden başlayarak, paylaşılan bir görüntü ağı oluşturma sürecini inceleyelim.

KOD  Liste 4-12'de g

Program az önce tartıştığımız ana bilgisayar adı grafik programına çok benzer .İlk önce pefile ve networkx dahil olmak üzere bir dizi modülü içe aktarır. Ancak burada, hedef kötü amaçlı yazılım örneklerinden grafiksel varlıkları çıkarmak için kullandığımız ExtractImages yardımcı sınıfını tanımlıyoruz. Ardından program, tüm hedef kötü amaçlı yazılım ikili dosyalarını yinelediğimiz bir döngüye girer .

Artık döngümüzde olduğumuza göre, ExtractImages sınıfını kullanarak hedef kötü amaçlı yazılım ikili dosyalarından grafik varlıkları ayıklamanın zamanı geldi (bu, Bölüm 1'de tartışılan icoutils programlarının etrafına bir sarmalayıcıdır). 4-13 listesi, kodun bunu yapan kısmını gösterir.

İlk olarak, ExtractImages sınıfına hedef kötü amaçlı yazılım ikili dosyasına giden yolu iletiriz ve ardından ortaya çıkan örneğin work () yöntemini  çağırırız. Bu, ExtractImages sınıfının kötü amaçlı yazılım görüntülerini sakladığı geçici bir dizin oluşturmasına ve ardından görüntüler sınıfı özniteliğinde her görüntü hakkında verileri içeren bir sözlüğü depolamasına neden olur. Artık ExtractImages'dan çıkarılan görüntülerin listesine sahip olduğumuza göre, üzerinde yineliyoruz , karmasını daha önce görmediysek bir görüntü için yeni bir ağ düğümü oluşturuyoruz  ve şu anda işlenen kötü amaçlı yazılım örneğini ağdaki görüntüsü oluşturulur.

Artık içerdikleri görüntülerle bağlantılı kötü amaçlı yazılım örnekleri ağımızı oluşturduğumuza göre, Liste 4-14'te gösterildiği gibi grafiği diske yazmaya hazırız.

İlk olarak, tüm ağı diske  yazıyoruz ve ardından iki projeksiyonu (kötü amaçlı yazılım için projeksiyon ve burada kaynak olarak adlandırdığımız görüntüler için projeksiyon) diske yazıyoruz. İmage\_network.py'yi, bu kitaptaki kötü amaçlı yazılım veri kümelerinin herhangi birindeki grafik varlıkları analiz etmek veya seçtiğiniz kötü amaçlı yazılım veri kümelerinden istihbarat çıkarmak için kullanabilirsiniz.

Özet:

bipartite iki taraflı ağ

Bu bölümde, kendi kötü amaçlı yazılım veri kümelerinizde paylaşılan öznitelik analizi yapmak için gerekli araçları ve yöntemleri öğrendiniz. Özellikle, ağların, iki taraflı ağların ve iki taraflı ağ projeksiyonlarının kötü amaçlı yazılım örnekleri arasındaki sosyal bağlantıları belirlemeye nasıl yardımcı olabileceğini, ağ düzeninin neden ağ görselleştirme için merkezi olduğunu ve zorla yönlendirilmiş ağların nasıl çalıştığını öğrendiniz. Ayrıca, Python ve NetworkX gibi açık kaynaklı araçları kullanarak kötü amaçlı yazılım ağlarını nasıl oluşturacağınızı ve görselleştireceğinizi de öğrendiniz. 5. Bölümde, örnekler arasında paylaşılan kod ilişkilerine dayalı olarak kötü amaçlı ağların nasıl oluşturulacağını öğreneceksiniz.

UYGULAMASINI YAPAMADIM