

NMAP BÖLÜM 6-10

6. Optimizing Nmap Performance (Bölüm 6. Nmap Performansını Optimize Etme)

- Introduction (İçindekiler)
- Scan Time Reduction Techniques (Tarama Süresini Azaltma Teknikleri)
 - Omit Non-critical Tests (Kritik Olmayan Testleri Atlayın)
 - Optimize Timing Parameters (Zamanlama Parametrelerini Optimize Edin)
 - Separate and Optimize UDP Scans (UDP Taramalarını Ayırın ve Optimize Edin)
 - Upgrade Nmap (Nmap'i Yükseltin)
 - Execute Concurrent Nmap Instances (Eşzamanlı Nmap Örnekleri Yürütün)
 - Scan From a Favorable Network Location (Uygun Bir Ağ Konumundan Tarayın)
 - Increase Available Bandwidth and CPU Time (Kullanılabilir Bant Genişliğini ve CPU Süresini Artırın)
- Coping Strategies for Long Scans (Uzun Taramalar için Başa Çıkma Stratejileri)
 - Use a Multi-stage Approach (Çok Aşamalı Bir Yaklaşım Kullanın)
 - Estimate and Plan for Scan Time (Tarama Süresini Tahmin Edin ve Planlayın)
- Port Selection Data and Strategies (Port Seçimi Verileri ve Stratejileri)
- Low-Level Timing Controls (Düşük Seviyeli Zamanlama Kontrolleri)
- Timing Templates ([T](#)) (Zamanlama Şablonları (-T))
- Scanning 676,352 IP Addresses in 46 Hours (46 Saatte 676.352 IP Adresi Tarama)

Introduction (İçindekiler)

En yüksek Nmap geliştirme önceliklerimden biri her zaman performans olmuştur. Yerel ağımdaki bir ana bilgisayarın varsayılan taraması (nmap <ana bilgisayar adresi>) saniyenin beşte biri kadar sürüyor. Bu, göz kırpmak için ancak yeterli bir süre, ancak yüzlerce veya binlerce ana bilgisayarı taradığınızda eklenir. Dahası, UDP taraması ve sürüm tespiti gibi belirli tarama seçenekleri tarama sürelerini önemli ölçüde artırabilir. Belirli güvenlik duvarı yapılandırmaları, özellikle de yanıt hızı sınırlaması da öyle. Nmap bu taramaları hızlandırmak için paralellik ve birçok gelişmiş algoritma kullanırken, kullanıcı Nmap'in nasıl çalışacağı üzerinde nihai kontrole sahiptir. Uzman kullanıcılar, zaman kısıtlamalarını karşılaşırken yalnızca önemsedikleri bilgileri elde etmek için Nmap komutlarını dikkatlice oluştururlar.

Nmap performansı yüksek bir öncelik olsa da, doğruluk daha da önemlidir. Rakip tarayıcıların yazarları, tarayıcılarının tüm bir B sınıfı adres alanını taramasının yalnızca dört saniye sürdüğü hakkında yüksek profilli konferans sunumları yaptılar. Bu tarayıcıların yazılması aslında öbensizdir, çünkü tüm tikanıklık kontrolü ve paket kaybı algılama algoritmalarını atırlar, sadece sistemin üretebileceği veya kablonun taşıyabileceği kadar hızlı prob paketleri püskürten sıkı bir döngü bırakırlar. Bu tür tarayıcılar genellikle durum bilgisi olmayan

(stateless) olarak tanıtırlılar, yani problemleri izlemek ve yeniden iletmek için gereken kodu da ihmal etmişlerdir. Nmap'in saniyede en az 1.000 paket göndemesini istemek için --min-rate 1000 ve zaman aşımına uğramış problemlerin yeniden iletimini devre dışı bırakmak için --max-retries 0 gibi bayraklar ekleyerek Nmap ile benzer bir davranış elde edebilirsiniz. Yine de bunu nadiren öneririm. Paketlerin yüzde doksan dokuzu bir sonraki yönlendirici tarafından düşürülebilir ve tarayıcı aradaki farkı asla bilemez.

Scanrand gibi ölçülmemiş paket patlatma tarayıcıları bazı durumlarda kullanışlıdır, ancak Nmap çok daha muhafazakar ve doğru bir yol izler. Nmap ilk başta hedef ağların en kötüsünü (yüksek gecikme ve paket kaybı) varsayar, daha sonra bunu güvenli bir şekilde yapabileceğini gösteren istatistikleri topladıkça hızlanır. Bu otomatik olarak gerçekleşirken, bir yönetici ağla ilgili ipuçlarını Nmap'e ileterek öğrenme sürecini hızlandırabilir. Böyle bir ipucuna örnek olarak --max-rtt-timeout 200ms verilebilir, bu da Nmap'in bir hedef ana bilgisayar probuna verilen yanıtların 200 milisaniye içinde geleceğini varsaymasını sağlar.

Bu bölüm ilk olarak tarama sürelerini iyileştirmek için üst düzey metodolojileri tartısmaktadır. Daha sonra doğruluğu etkilemeden Nmap'i hızlandırmak için zamanlama şablonlarının ve düşük seviyeli kontrollerin nasıl kullanıldığı anlatılmaktadır. Bölüm, Mayo Clinic'ten Jack Mogren'in 676.352 IP'lik ağında tarama süresini neredeyse bir haftadan 46 saatte nasıl indirdiğini anlatan bir eğitimle sona eriyor. Tarayıcı performansının büyük önemi göz önüne alındığında, bu bölüm kısa görünebilir. Bunun nedeni bölümün üst düzey genel tarama performansı ipuçlarına odaklanması, belirli tarama tekniklerini optimize etmeye yönelik ipuçlarının ise bu tekniklerin ele alındığı bu kitaba yayılmış olmasıdır.

Scan Time Reduction Techniques (Tarama Süresini Azaltma Teknikleri)

Uzun tarama süreleri için ideal çözüm bu süreleri azaltmaktadır. Bu bölüm bunu yapmak için birçok üst düzey ipucu sunmaktadır. Hayattaki birçok durumun aksine, Nmap komut satırınızı ayarlamak büyük bir fark yaratabilir. Honda Accord'unuzu bir kahve kutusu egzoz ucu, üç ayak yüksekliğinde bir spoyler ve büyük kırmızı bir "tip R" etiketi ile modifiye etmek 0-100 süresini çok fazla kısaltmayacaktır. Yine de "46 Saatte 676.352 IP Adresi Tarama" adlı bölümde Jack Mogren'in Nmap komut satırına birkaç çıkartma (yani seçenek) ekleyerek Nmap çalışma süresini nasıl günlerce kısalttığı anlatılıyor.

Omit Non-critical Tests (Kritik Olmayan Testleri Atlayın)

Asla kaldırımdan ayrılmadığınızda ya da market alışverişinden fazlasını taşımadığınızda bir Hummer satın almanın elektronik eşdeğeri, nispeten önemsiz miktarda bilgi elde etmek için yoğun ve kapsamlı bir Nmap taraması başlatmaktadır. Bir ev ağında ana bilgisayar başına birkaç saniye harcamak nadiren önemlidir, ancak büyük işletmeler için günlük WAN taramalarını uygulanamaz hale getirebilir. Aşağıdaki liste, en korkunç sorunlardan başlayarak ve ardından ileri düzey kullanıcıların bile sıklıkla unuttuğu daha ince optimizasyonlarla yaygın aşırı tarama hatalarından kaçınmanın yollarını detaylandırmaktadır.

Yalnızca hangi ana bilgisayarların çevrimiçi olduğunu belirlemeniz gerekiğinde bağlantı noktası taramasını (-sn) atlayın.

Bazı insanlar nmap <hostname> komutunu kullanarak bir ana bilgisayarın çevrimiçi olup olmadığını belirler. Bu işe yarasa da, aşırıdır. Nmap, ana bilgisayarın açık olduğunu belirlemek için dört paket, ardından ana bilgisayarı port taraması için en az 1.000 paket gönderecektir. Tüm çevrimiçi ana bilgisayarları ya da belirli bir ana bilgisayarı bulmak için tüm ağ bu şekilde tarandığında sorun daha da büyür.

Bağlantı noktası taramasıyla zaman kaybetmek yerine, tek bilmek istediğiniz hangi ana bilgisayarların açık olduğu veya MAC adreslerinin ne olduğu olduğunda ping taraması yapmak için -sn belirtin.

Taranan bağlantı noktası sayısını sınırlayın.

Varsayılan olarak, Nmap en yaygın 1.000 portu tarar. Duyarlı makinelerden oluşan hızlı bir ağda bu, ana bilgisayar başına saniyenin bir kısmını alabilir. Ancak Nmap, hız sınırlamasıyla veya yanıt vermeden prob paketlerini düşüren güvenlik duvarlarıyla karşılaşlığında önemli ölçüde yavaşlamalıdır. UDP taramaları bu nedenlerden dolayı acı verici derecede yavaş olabilir. Yine de açık portların büyük çoğunluğu sadece birkaç yüz port numarasına düşer. Varsayılan 1.000 port yerine sadece 100 portu tararsanız port taraması yaklaşık 10 kat daha hızlı olacaktır. F (hızlı tarama) seçeneğiyle sadece en popüler 100 bağlantı noktasını tarayabilir, --top-ports ile en sık açık olan bağlantı noktalarının rastgele bir sayısını belirtebilir veya -p ile özel bir bağlantı noktası listesi sağlayabilirsiniz.

Gelişmiş tarama türlerini (-sC, -sV, -O, --traceroute ve -A) atlayın.

Bazı insanlar düzenli olarak -A Nmap seçeneğini belirtirler, bu da onlara işlerini verir. Nmap'in varsayılan port taramasının yanı sıra işletim sistemi tespiti, sürüm tespiti, komut dosyası taraması (NSE) ve traceroute yapmasına neden olur. Sürüm tespiti olağanüstü faydalı olabilir, ancak büyük bir taramayı da tıkayabilir. NSE de öyle. Zamanınız kısıtlı olduğunda, büyük ölçekli taramada -sC ve -sV'yi her zaman atlayabilir ve daha sonra gerekiğinde bunları tek tek bağlantı noktalarında gerçekleştirebilirsiniz.

İşletim sistemi tespiti sürüm tespiti kadar yavaş değildir, ancak yine de çevrimiçi ana bilgisayar başına kolayca 5-10 saniye sürebilir. Bu olmadan bile, genellikle bir LAN'daki ada, açık bağlantı noktalarına ve MAC adresine dayanarak işletim sistemini tahmin edebilirsiniz. Ve çoğu durumda işletim sistemi umurunuzda olmayabilir. Bu yüzden -O sadece gerekiği kadar kullanım için başka bir adaydır. Bir uzlaşma olarak, --osscan-limit --max-ostries 1 belirtebilirsiniz, bu da Nmap'e eşleşmeyen işletim sistemi algılama girişimlerini yeniden denememesini ve ayrıca en az bir açık TCP bağlantı noktası ve bir kapalı TCP bağlantı noktası olmayan tüm çevrimiçi ana bilgisayarlara karşı işletim sistemi algılamasını atlamasını söyler. OS tespiti bu tür ana bilgisayarlara karşı zaten doğru değildir.

Gerekli olmadığına DNS çözünürlüğünü kapatmayı unutmayın.

Varsayılan olarak, Nmap çevrimiçi olduğu tespit edilen her ana bilgisayara karşı ters-DNS çözümlemesi gerçekleştirir. Ping adımını -Pn ile atlarsanız veya -R belirtirseniz, tüm ana bilgisayarlara karşı yapılır. Bu, ana bilgisayar DNS kütüphaneleri her seferinde bir IP aramak için kullanıldığından büyük bir darboğazdı.

Nmap artık sorguları hızlandırmak için hızlı bir paralel ters-DNS sistemine sahip olsa da, yine de önemli miktarda zaman alabilirler. Veriye ihtiyacınız olmadığından -n seçeneği ile devre dışı bırakın. Çok sayıda ana bilgisayara karşı yapılan basit taramalarda (ping taramaları gibi), DNS'nin devre dışı bırakılması bazen tarama süresini %20 veya daha fazla azaltabilir. DNS zamanı, binlerce bağlantı noktasını araştıran veya sürüm algılama gibi yoğun özelliklerini kullanan daha kapsamlı taramalarda önemli bir faktör değildir. Nmap ana makinesinin isim çözümlemesini yapmasını istiyorsanız (gethostbyaddr fonksiyonunu kullanarak), --system-dns seçeneğini belirtin. Bunu yapmak taramaları önemli ölçüde yavaşlatabilir.

Optimize Timing Parameters (Zamanlama Parametrelerini Optimize Etme)

Nmap, tarama etkinliğini kontrol etmek için ipuçları ve kurallar sağlamak üzere düzinelere seçenek sunar. Bunlar -T seçeneği ("Zamanlama Şablonları (-T)" adlı bölümde açıklanmıştır) tarafından sağlanan yüksek seviye zamanlama agresiflik seviyelerinden "Düşük Seviye Zamanlama Kontrolleri" adlı bölümde açıklanan daha ince taneli kontrollere kadar uzanır. Hatta ikisini birlestirebilirsiniz. Bu seçenekler özellikle Nmap'in kendi zamanlama tahminlerini belirlemek için çok az yanıt aldığı yüksek oranda filtrelenmiş ağları tararken kullanışlıdır. Tarama süresi genellikle güvenli bir şekilde yarıya indirilebilir. Bu seçeneklerin çoğu, duyarlı ana bilgisayarlara dolu yerel bir LAN'a karşı çok az etkiye sahip olacaktır, çünkü Nmap bu durumda en uygun değerleri kendisi belirleyebilir.

Separate and Optimize UDP Scans (UDP Taramalarını Ayırın ve Optimize Edin)

UDP portlarının taraması önemlidir çünkü birçok savunmasız hizmet bu protokolü kullanır, ancak UDP taramalarının zamanlama özellikleri ve performans gereksinimleri TCP taramalarından çok farklıdır. Son derece

yaygın olan ve UDP taramalarını TCP'den çok daha sık etkileyen ICMP hata hızı sınırlaması özellikle endişe vericidir.

Bu nedenlerden dolayı, Nmap -sSU gibi seçeneklerle bunu yapmayı desteklese de, performans kritik olduğunda TCP ve UDP taramalarını birleştirmeyi önermiyorum. Genellikle her protokol için ayrı komut satırları gerektiren farklı zamanlama bayrakları istersiniz. "UDP Taramalarını Hızlandırmaya" başlıklı bölüm, UDP tarama performansını artırmak için değerli püf noktaları ve gerçek hayattan örnekler sunar.

Upgrade Nmap (Upgrade Nmap)

Nmap performansının düşük olduğuna dair raporları incelediğimde, rapor sahibinin uzun yıllardır güncel olmayan eski bir sürümü kullandığını tespit ettiğim birçok vaka oldu. Nmap'in en yeni sürümleri önemli algoritmik iyileştirmelere, hata düzeltmelerine, yerel ağ ARP taraması gibi performans artırıcı özelliklere ve daha fazlasına sahiptir. Performans sorunlarına ilk müdahale, Nmap sürümünüzü (nmap -V çalıştırın) <https://nmap.org> adresinde bulunan en son sürümle karşılaşmak olmalıdır. Gerekirse yükseltin. Hala yeterince hızlı değilse, bu bölümdeki diğer teknikleri deneyin.

Execute Concurrent Nmap Instances (Eşzamanlı Nmap Örnekleri Yürütme)

Bazı insanlar her biri bir hedefe karşı paralel olarak birçok kopya çalıştırarak Nmap'i hızlandırmaya çalışır. Örneğin, Nessus tarayıcısı bunu varsayılan olarak yapardı. Bu genellikle Nmap'in tüm ağa karşı çalışmasına izin vermekten çok daha az verimli ve yavaştır. Nmap'in ihtiyaçlarına göre özelleştirilmiş kendi paralelleştirme sistemi vardır ve Nmap büyük bir grubu taradığında ağ güvenilirliği hakkında bilgi edindikçe hızlanabilir. Ayrıca, sadece bir B sınıfını taramak için işletim sisteminden 65.536 ayrı Nmap örneğini çatallamasını istemek önemli bir ek yüktür. Nmap'in dzinelerce kopyasının paralel olarak çalışması da her örnek nmap-services ve nmap-os-db gibi veri dosyalarının kendi kopyasını yüklediğinden bellek tüketimine neden olur.

Tek ana bilgisayarlı Nmap taramalarını paralel olarak başlatmak kötü bir fikir olsa da, genel hız genellikle taramayı birkaç büyük gruba bölgerek ve bunları eşzamanlı olarak çalıştırarak iyileştirebilir. Yine de aşırıya kaçmayın. Beş veya on Nmap işlemi iyidir, ancak aynı anda 100 Nmap işleminin başlatılması önerilmez. Çok fazla eşzamanlı Nmap işlemi başlatmak kaynak çekimine yol açar. Bir başka eşzamanılık türü de Nmap'i aynı anda farklı ana bilgisayarlardan çalıştırmaktadır. Ağlarınızın her birindeki yerel ana bilgisayarları cron (ya da Windows'ta At) ile aynı anda taramaya başlatabilir, ardından sonuçları merkezi bir veri sunucusuna postalayabilirsiniz. Avustralya ağınızı ABD'den taramak, o ağdaki yerel bir makineden taramaktan daha yavaş olacaktır. ABD'deki makinenin uzaktaki ağa ulaşmak için ekstra güvenlik duvarlarından geçmesi gerekiyorsa fark daha da büyük olacaktır.

Scan From a Favorable Network Location (Uygun Bir Ağ Konumundan Tarama)

Kısıtlayıcı güvenlik duvarları beş saniyelik bir taramayı birkaç saatlik bir angaryaya dönüştürebilir. Bazı İnternet rotalarıyla ilişkili gecikme ve paket kaybı da yardımcı olmaz. Nmap'i hedef ağın yerel ana bilgisayar(lar)ından çalıştırabileceğiniz, bunu yapın. Elbette amaç ağı harici bir saldırganın göreceği şekilde görmek ya da güvenlik duvarını test etmekse, harici tarama gereklidir. Öte yandan, dahili ağın taranması ve güvenliğinin sağlanması, dahili tehditlere ve güvenlik duvarını aşan kurnaz saldırganlara karşı kritik öneme sahip olan derinlemesine savunma sağlar (bkz. Bölüm 10, Güvenlik Duvarlarını ve Saldırı Tespit Sistemlerini Tespit Etme ve Yıkma).

Ters DNS çözümlemesi yaparken, özellikle de çok yüklü bir yerel ad sunucunuz varsa, daha az meşgul bir ad sunucusu kullanmak veya doğrudan yetkili ad sunucularını sorgulamak yardımcı olabilir. Bu kazanç genellikle azdır ve yalnızca tekrarlanan veya çok büyük taramalar için yapmaya değerdir. Elbette, bazen ad sunucularını seçmek için performans dışı nedenler de olabilir.

Increase Available Bandwidth and CPU Time (Kullanılabilir Bant Genişliğini ve CPU Süresini Artırın)

Bazen mevcut bant genişliğini veya CPU gücünü artırarak Nmap tarama sürelerini iyileştirebilirsiniz. Bu, yeni bir veri hattı veya CPU yükleyerek ya da bu kaynaklar için rekabet eden eşzamanlı çalışan uygulamaları

durdurarak yapılabilir. Örneğin, The Matrix Reloaded'in korsan torrentini indirerek DSL hattınızı eşzamanlı olarak doyurursanız Nmap daha yavaş çalışacaktır.

Nmap'in kendi tıkanıklık kontrol algoritmaları tarafından kısıtlanması, CPU'ya bağlı olmaktan veya mevcut yerel bant genişliği ile sınırlı olmaktan çok daha yaygındır. Bu kontroller ağ taşmasını önlemeye ve doğruluğu artırmaya yardımcı olur. CPU gücünü ve yerel bant genişliğini artırmak, Nmap'in bu tür kendi kendini sınırlamasına yardımcı olmaz; bunun yerine zamanlama seçeneklerinin ayarlanması gereklidir. CPU yükünü Unix'te top veya Windows'ta Görev Yöneticisi gibi bir uygulama ile izleyerek Nmap'in CPU kısıtlaması olup olmadığını test edebilirsiniz. CPU'nuz zamanının çoğunu boşta geçiriyorsa, yükselme yapmak pek yardımcı olmayacağından emin olun. Nmap'in bant genişliği kullanımını test etmek için, verbose modunda (-v) çalıştırın. Nmap daha sonra Örnek 6.1'de gösterdiği gibi gönderilen ve alınan bayt sayısını ve yürütme süresini raporlayacaktır.

Örnek 6.1. Yerel 100 Mbps ethernet ağı üzerinden bant genişliği kullanımı

```
# nmap -v -n -p- sec.titan.net
Starting Nmap ( https://nmap.org )
[10 lines cut]
Nmap scan report for 192.168.0.8
Not shown: 65534 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 00:1A:6B:C1:33:37 (USI)

Nmap done: 1 IP address (1 host up) scanned in 2.20 seconds
Raw packets sent: 65536 (2.884MB) | Rcvd: 65536 (2.621MB)
```

Saniye başına bit cinsinden ortalama bant genişliği kullanımını elde etmek için bayt değerlerini sekize çarpın ve yürütme süresine bölün. Örnek 6.1'de, Nmap 2.20 saniyede 2,621,000 bayt almıştır (Nmap 1,000,000 baytı bir MB olarak kabul eder). Yani alma trafiği yaklaşık 9,5 Mbps idi (gönderme hızı 10,5 Mbps idi). Bu nedenle 100 Mbps ethernet bağlantısı muhtemelen Nmap'i kısıtlamıyor ve gigabit ethernet'e yükseltmek de pek yardımcı olmayacağından emin olun.

Bazı tüketici geniş bant cihazları ve diğer ekipmanlar, küçük paket boyutu (genellikle Nmap boş başlıklar gönderir) bant genişliğini düşük tutsa da, Nmap tarafından gönderilen paketlerin hızını kaldırmakta zorlanır. Örnek 6.1'de, "Yerel 100 Mbps ethernet ağı üzerinden bant genişliği kullanımı", Nmap saniyede yaklaşık 30.000 paket göndermiş ve benzer sayıda paket almıştır. Bu kadar yüksek paket hızları düşük kaliteli cihazlarda soruna neden olabilir. Bu durumda, hem gönderme hem de alma paket sayılarının 65.536 olduğunu görüyoruz, bu da taranan bağlantı noktası sayısı (65.535) artı ilk ARP ping probu için birdir. Bu nedenle Nmap yeniden iletişim gerektiren herhangi bir paket düşüşüyle karşılaşmamıştır. Bu da yine ağ ekipmanının sınırlayıcı bir faktör olmadığını göstermektedir-Nmap muhtemelen CPU'ya bağlıydı.

Coping Strategies for Long Scans (Uzun Taramalar İçin Başa Çıkma Stratejileri)

Bir taramayı hızlandırmak için tarama seçeneklerini optimize etmek size uzun bir yol ettirebilir, ancak doğruluğu korurken ve rakip ağ akışlarına adil davranırken Nmap'in ne kadar hızlı çalışabileceğinin bir sınırı vardır. Binlerce ana bilgisayar, tüm 65K bağlantı noktaları, UDP veya sürüm algılamayı içeren büyük taramalar, optimizasyondan sonra bile muhtemelen biraz zaman alacaktır. Bu bölüm, bu uzun taramalarla başa çıkmak için güçlü stratejiler sunmaktadır.

Use a Multi-stage Approach (Çok Aşamalı Bir Yaklaşım Kullanın)

Kapsamlı bir güvenlik denetimi, her bir protokol için 65.536 bağlantı noktasının tümünün UDP ve TCP taramasını içermelidir, genellikle bir makinenin açık olması ancak yoğun bir şekilde filtrelenmesi durumunda -Pn ile. Ancak bu port numaralarının 100'den azı yaygın olarak kullanılır ve çoğu ana bilgisayar orta düzeyde ana bilgisayar bulma seçenekleriyle yanıt verir. Bu nedenle, önce bilinen çevrimiçi ana bilgisayarlarda popüler bağlantı

noktalarını hızlı bir şekilde taramak için -F belirtin. Bu, arka planda sürüm ve işletim sistemi algılama ile tüm TCP ve UDP bağlantı noktalarının büyük -Pn taramasını başlatırken çevrimiçi ana bilgisayarları ve açık bağlantı noktalarının çoğunu analiz etmenizi sağlar. Hızlı taramayı hızlandırmak için kısa yol seçenekleri "Kritik Olmayan Testleri Atla" adlı bölümde ele alınmıştır. Yavaş tarama tamamlandığında, yeni keşfedilen ana bilgisayarları veya bağlantı noktalarını bulmak için önceki sonuçlarla karşılaşır.

Estimate and Plan for Scan Time (Tarama Süresini Tahmin Edin ve Planlayın)

Çoğu durumda, uzun taramaların en sinir bozucu yönü ne zaman tamamlanacağı hakkında hiçbir fikre sahip olmamaktır. Nmap, verbose modu (-v) etkinleştirildiği sürece düzenli tarama süresi tahminleri sağladığı için artık eskisinden daha yardımcıdır.

Örnek 6.2. Tarama süresini tahmin etme

```
# nmap -T4 -sS -p0- -iR 500 -n --min-hostgroup 100 -v
Starting Nmap ( https://nmap.org )
Initiating SYN Stealth Scan against 29 hosts [65536 ports/host] at 23:27
[...]
SYN Stealth Scan Timing: About 0.30% done; ETC: 09:45 (10:15:45 remaining)
```

Örnek 6.2 bize SYN taramasının 29 ana bilgisayarı taramak için muhtemelen on saat on sekiz dakika (23:27 - 9:45) sürecekğini göstermektedir. Dolayısıyla, Nmap'in ağı taramak için harcayacağı toplam süre, ana bilgisayar başına 21 dakika ile çevrimiçi ana bilgisayar sayısını çarpılarak kabaca tahmin edilebilir. Eğer sürüm tespiti veya UDP de yapılyorsa, bunlar için zamanlama tahminlerini de izlemeniz gerekecektir.

Diğer bir seçenek de Nmap'in ilk ana bilgisayar grubunu taramayı tamamen tamamlamasını beklemektir. Daha sonra bu grubun büyülüğü için geçen süreyi tüm hedef ağını büyülüğü üzerinden tahmin etmektir. Bu daha basittir çünkü tek tek tarama bileşenleri hakkında endişelenmenize gerek yoktur. Tahminlerinizi hedef IP alanı boyutuna karşı tamamlanan hedef IP adreslerinin sayısına dayandırmak yanlıılı olabilir, çünkü çevrimiçi ana bilgisayarlar bu IP alanına nadiren eşit olarak dağıtilır. Genellikle IP alanının başlangıcına yakın yerlerde kümeler halinde bulunurlar. Bu nedenle, taramanın kendisi ana bilgisayar keşfini içeriye (yani -Pn seçeneği yoksa), daha doğru bir ölçüm, önce tüm ağı ping taraması yapmak ve ardından tahminlerinizi Nmap'in taramayı tamamladığı çevrimiçi ana bilgisayar sayısına ve ping taramasıyla çevrimiçi bulunan sayıya dayandırmaktır.

Ara sıra yapılan tahminler ayrıntılı modda otomatik olarak yazdırılsa da, <enter> tuşuna basarak her zaman geçerli tahmini isteyebilirsiniz ("Çalışma Zamanı Etkileşimi" adlı bölüme bakın). Eğer tahmin sizin zaman diliminiz içindeyse, tahmin devam ederken yapacak başka bir şey planlayabilirsiniz. Bu, her 20 dakikada bir Nmap'in tamamlayıp tamamlanmadığını kontrol etmekten daha iyidir. Nmap'in zamanında bitmeyeceğini gösteren bir tahmin daha da değerlidir. Hemen taramayı optimize etmek veya görevi uzatmak için çalışabilirsiniz. Taramanın çok yavaş olduğunu ancak son tarih geçiktan ve Nmap hala çalışıktan sonra belirlerseniz seçenekleriniz çok daha sınırlıdır.

Port Selection Data and Strategies (Liman Seçim Verileri ve Stratejileri)

Port taraması, tarama sürüm tespiti veya NSE komut dosyaları içerdiginde bile bir Nmap taramasının en çok zaman alan kısmı olabilir. Port tarama süresi kabaca taranan port sayısı ile orantılıdır, bu nedenle port sayısını azaltmak önemli bir performans artışı sağlar. Bunun dezavantajı, azaltılmış taramaların daha az kapsamlı olmasıdır, bu nedenle açık bağlantı noktalarını kaçırabilirsiniz.

Gerçek şu ki, her protokolde 65.536 port var ve bunların çoğu neredeyse hiç açık değil. Her bir TCP ve UDP portunun yaygınlığını belirlemek için bir yaz boyunca büyük ölçekli taramalar gerçekleştirdim. Sonuçlar, on milyonlarca Internet IP adresinin yanı sıra içерiden taranan kurumsal ağların taranmasından elde edilen verileri

icermektedir. Bu bölüm, taramalarınızda hız ve etkinlik arasında doğru dengeyi kurmak için güvenebileceğiniz deneysel sonuçlar sağlar.

Yüz binden fazla (toplam) TCP ve UDP portu mevcut olsa da, açık portların büyük çoğunluğu çok daha küçük bir kümeye girmektedir. Araştırmamıza göre, en iyi 10 TCP portu ve en iyi 1.075 UDP portu, protokoller için açık portların yarısını temsil etmektedir. Açık portların %90'ını yakalamak için 576 TCP portunu ve 11.307 UDP portunu taramanız gereklidir. Varsayılan olarak, Nmap istenen her tarama protokolü için en iyi 1.000 portu tarar. Bu, TCP portlarının yaklaşık %93'ünü ve UDP portlarının %49'unu yakalar. F (hızlı) seçeneği ile yalnızca ilk 100 bağlantı noktası taranır ve TCP için %78, UDP için %39 etkinlik sağlanır. Farklı sayıda bağlantı noktası belirtmek için --top-ports seçeneğine bu değeri girin. Tablo 6.1, söz konusu protokol için belirli bir etkinlik oranına ulaşmak için taramanız gereken TCP veya UDP bağlantı noktası sayısını yaklaşık olarak vermektedir.

Tablo 6.1. Çeşitli etkinlik seviyelerine ulaşmak için gerekli --top-port değerleri

Effectiveness (Etkinlik)	TCP ports required (TCP portları gereklili)	UDP ports required (UDP portları gereklili)
10%	1	5
20%	2	12
30%	4	27
40%	6	135
50%	10	1,075
60%	18	2,618
70%	44	5,157
80%	122	7,981
85%	236	9,623
90%	576	11,307
95%	1,558	13,035
99%	3,328	15,094
100%	65,536	65,536

Nmap port seçimini sizin için otomatik olarak yapabılırken (varsayılanlara güvendiğinizde veya -F veya --top-ports gibi seçenekleri kullandığınızda), portları -p ile açıkça belirtmek genellikle yararlıdır. Her iki durumda da, en sık görülen açık portlara aşina olmak önemlidir. Verilerimize göre en popüler portlar "En Popüler Portlar Nelerdir?" başlıklı bölümde açıklanmıştır.

Low-Level Timing Controls (Düşük Seviyeli Zamanlama Kontrolleri)

Nmap, tarama hızını kontrol etmek için birçok ince taneli seçenek sunar. Çoğu kişi bu seçenekleri Nmap'i hızlandırmak için kullanır, ancak Nmap'i yavaşlatmak için de yararlı olabilirler. İnsanlar bunu IDS sistemlerinden kaçmak, ağ yükünü azaltmak ve hatta ağ koşulları Nmap'in muhafazakar varsayılanının bile çok agresif olduğu kadar kötüye doğruluğu artırmak için yaparlar.

Tablo 6.2'de her bir düşük seviye zamanlama kontrol seçeneği fonksiyonlarına göre listelenmiştir. Her seçenek hakkında ayrıntılı kullanım bilgisi için "Zamanlama ve Performans" adlı bölüm okuyun. Okuyucunun "Tarama Kodu ve Algoritmalar" bölümünde açıklanan Nmap tarama algoritmalarına zaten aşina olduğu varsayılmaktadır.

Tablo 6.2. Fonksiyona göre düşük seviye zamanlama kontrolleri

Function (Fonksiyon)	Options (Seçenekler)
Ana bilgisayar grubu (eszamanlı olarak taranan ana bilgisayar grubu) boyutu	--min-hostgroup , --max-hostgroup
Paralel olarak başlatılan sonda sayısı	--min-parallelism , --max-parallelism
Prob zaman aşımı değerleri	--min-rtt-timeout , --max-rtt-timeout , --initial-rtt-timeout
İzin verilen maksimum prob yeniden iletim sayısı	--max-retries
Tüm bir ana bilgisayardan vazgeçmeden önce maksimum süre	--host-timeout
Her bir ana bilgisayara karşı her bir prob arasına yerleştirilen kontrol gecikmesi	--scan-delay , --max-scan-delay
Saniye başına gönderilen sonda paketlerinin oranı	--min-rate , --max-rate
Hedef ana bilgisayarlar tarafından RST paket yanıt oranını yenme	--defeat-rst-ratelimit

Timing Templates (T) (Zamanlama Şablonları (-T))

Önceki bölümde tartışılan ince taneli zamanlama kontrolleri güçlü ve etkili olsa da, bazı insanlar bunları kafa karıştırıcı bulmaktadır. Dahası, uygun değerleri seçmek bazen optimize etmeye çalışığınız taramadan daha fazla zaman alabilir. Bu yüzden Nmap altı zamanlama şablonu ile daha basit bir yaklaşım sunar. Bunları -T seçeneği ve numaraları (0-5) veya adları ile belirtebilirsiniz. Şablon adları paranoid (0), sinsi (1), kibar (2), normal (3), agresif (4) ve çılgın (5) şeklidindedir. İlk ikisi IDS'den kaçmak içindir. Kibar mod, daha az bant genişliği ve hedef makine kaynağı kullanmak için taramayı yavaşlatır. Normal mod varsayılandır ve bu nedenle -T3 hiçbir şey yapmaz. Agresif mod, oldukça hızlı ve güvenilir bir ağda olduğunuzu varsayıarak taramaları hızlandırır. Son olarak çılgın mod, olağanüstü hızlı bir ağda olduğunuzu veya hız için bir miktar doğruluğu feda etmeye istekli olduğunuzu varsayar.

Bu şablonlar, kullanıcının ne kadar agresif olmak istediğini belirtmesine izin verirken, Nmap'i tam zamanlama değerlerini seçmeye bırakır. Şablonlar ayrıca, şu anda ince taneli kontrol seçeneklerinin mevcut olmadığı bazı küçük hız ayarlamaları da yapar. Örneğin, -T4 TCP portları için dinamik tarama gecikmesinin 10 ms'yi aşmasını yasaklar ve -T5 bu değeri 5 ms ile sınırlar. Şablonlar ayrıntılı kontrollerle birlikte kullanılabilir ve ayrıntılı seçenekler bu belirli değerler için genel zamanlama şablonlarını geçersiz kılar. Oldukça modern ve güvenilir ağları tararken -T4 kullanmanızı öneririm. Bu seçeneği (komut satırının başında) ince taneli kontroller eklediğinizde bile saklayın, böylece etkinleştirdiği ekstra küçük optimizasyonlardan yararlanabilirsiniz.

Tablo 6.3'te zamanlama değişkenlerinin her -T değeri için nasıl değiştiği gösterilmektedir. Tüm zaman değerleri milisaniye cinsindendir.

Tablo 6.3. Zamanlama şablonları ve etkileri

	T0	T1	T2	T3	T4	T5
Name	Paranoid	Sneaky	Polite	Normal	Aggressive	Insane
min-rtt-timeout	100 ms	100 ms	100 ms	100 ms	100 ms	50 ms
max-rtt-timeout	5 minutes	15 seconds	10 seconds	10 seconds	1250 ms	300 ms
initial-rtt-timeout	5 minutes	15 seconds	1 second	1 second	500 ms	250 ms
max-retries	10	10	10	10	6	2

Initial (and minimum) scan delay (<code>--scan-delay</code>)	5 minutes	15 seconds	400 ms	0	0	0
Maximum TCP scan delay	5 minutes	15,000	1 second	1 second	10 ms	5 ms
Maximum UDP scan delay	5 minutes	15 seconds	1 second	1 second	1 second	1 second
<code>host-timeout</code>	0	0	0	0	0	15 minutes
<code>script-timeout</code>	0	0	0	0	0	10 minutes
<code>min-parallelism</code>	Dynamic, not affected by timing templates					
<code>max-parallelism</code>	1	1	1	Dynamic	Dynamic	Dynamic
<code>min-hostgroup</code>	Dynamic, not affected by timing templates					
<code>max-hostgroup</code>	Dynamic, not affected by timing templates					
<code>min-rate</code>	No minimum rate limit					
<code>max-rate</code>	No maximum rate limit					
<code>defeat-rst-ratelimit</code>	Not enabled by default					

Eğer iyi bir geniş bant ya da ethernet bağlantınız varsa, her zaman -T4 kullanmanızı tavsiye ederim. Benim zevkime göre çok agresif olsa da bazı insanlar -T5'i seviyor. İnsanlar bazen ana bilgisayarları çökertme olasılığının daha düşük olduğunu düşündükleri için veya kendilerini genel olarak kibar gördükleri için -T2'yi belirtiyorlar. Genellikle -T kibarlığının gerçekte ne kadar yavaş olduğunun farkında değillerdir. Taramaları varsayılan taramadan on kat daha uzun sürebilir. Varsayılan zamanlama seçeneklerinde (-T3) makine çökmeleri ve bant genişliği sorunları nadirdir ve bu nedenle normalde dikkatli tarayıcılar için bunu öneririm. Sürüm algılamayı atlamak, bu sorunları azaltmak için zamanlama değerleriyle oynamaktan çok daha etkilidir.

T0 ve -T1 IDS uyarılarından kaçınmak için yararlı olsa da, binlerce makineyi veya bağlantı noktasını taramak için olağanüstü uzun bir zaman alacaktır. Bu kadar uzun bir tarama için, hazır -T0 ve -T1 değerlerine güvenmek yerine ihtiyacınız olan tam zamanlama değerlerini ayarlamayı tercih edebilirsiniz.

Scanning 676,352 IP Addresses in 46 Hours (46 Saatte 676.352 IP Adresinin Taranması)

Bu hikaye Mayo Clinic'ten Jack L. Mogren tarafından gönderilmiştir. Düzenli bir Nmap tarama rejimi uygulamak ve bu büyük ajan tarama süresini bir haftadan 46 saate düşürmek için attığı adımları gösteren bir öğretici işlevi görür.

Mayo Clinic, kullanımda olan 70.000'den fazla IP adresini gösteren ARP tabloları ile nispeten büyük bir özel ağ kurmuştur. Ağ yönetimimiz eskiden üç büyük kampüste ve ülke çapında birkaç düzine uyduda fiziksel mimariyi oluşturmaya ve sürdürmeye odaklıydı. Sloganımız "İhtiyacınız mı var? Biz inşa ederiz" idi. Ağa gerçekte neyin bağlı olduğu çok az dikkate alınıyordu. Ağ yönetimi uygun bir şekilde veri girişinde sona eriyordu ve şeker çubuğu sendromundan muzdarıptı. Dışarıdan bakıldığından çitir çitir ve güvenliydi ama içi yumuşak ve çığnenebilirdi. İyi korunan sınırlarımız vardı ama çok az iç kontrolümüz vardı.

Bu tutum Ocak 2003'te Slammer solucanı (W32.SQLExp) ve türevleri çevremize girdiğinde aniden değişti. Birdenbire ağımıza neyin bağlı olduğunu bilmek çok önemli hale geldi. Slammer vakasında, MS SQL Server 2000 veya MSDE 2000 çalıştırılan tüm cihazların nerede bulunduğu ve yöneticilerin kim olduğunu bilmemiz gerekiyordu. Bu bilgilerden yoksun olduğumuz için Slammer'ı ortadan kaldırma çabamız birkaç ay sürdü.

Böylece "Ağda ne olduğunu bilme" çabası doğdu. Kulağa basit geliyor, ancak boyut, karmaşıklık ve ağ geçmişinin göz önüne alındığında, bu ileriye doğru atılmış büyük bir adım ve ağ yönetimi hizmetlerimiz için yeni bir yoldu.

Nmap bu çabada değerli bir araç olduğunu kanıtladı. Fiyatını yenemezsiniz ve açık kaynak topluluğunun gelişimine getirdiği avantajları takdir ediyorum. Özellikle işletim sistemi parmak izi ve son kullanıcılar tarafından sağlanan birçok katkı.

Nmap ile denemeler yapmaya başladım. Amacım, TCP/IP parmak izi yoluyla uzak ana bilgisayar tanımlamasını hızlı bir şekilde gerçekleştirmek için Nmap -O seçeneğini kullanarak anlamlı bir ağ envanteri oluşturmak.

IP ortamımız ve tarama platformum hakkında birkaç kelime ile başlayayım. Şu anda bir B sınıfı ve 44 C sınıfı aralığı sahibiz ve özel adres alanının çoğunu kullanıyoruz. Bu da 676.352 olası IP adresi anlamına geliyor. Taramalarımı Red Hat Linux 8.0 çalıştırın bir Compaq DL380'den gerçekleştirdim. İlk denemem, işletim sistemi algılama (-O) ve yalnızca ana bilgisayar keşfi için ICMP yankı istekleri (-PE) ile bu vanilya TCP SYN taramasıydı:

```
# nmap -O -PE -v -oX mayo.xml -iL ip_networks.txt
```

Ne yazık ki, bu o kadar yavaş ilerledi ki tüm ağımızı taramak bir hafta sürerdi. Ağımızın tüm önemli bölümlerinin en az bir T1 hattı (1.54 Mbps) ile bağlı olduğunu göz önünde bulundurarak, çığın konserve zamanlama politikasını (-T5) ekledim. Ayrıca taranan port sayısını yaklaşık 1600'den 1200'e düşüren hızlı tarama modunu (-F) ekledim[11]. Ayrıca --osscan-limit ekledim, böylece Nmap açık portu olmayan ana bilgisayarları tarayarak zaman kaybetmez. Bu, aşağıdaki komutla sonuçlandı:

```
# nmap -O -T5 -PE -F --osscan-limit -v -oX mayo.xml -iL ip_networks.txt
```

Ne yazık ki bu işlem birkaç gün sürecek gibi görünüyordu. Bu yüzden nmap-services dosyasını düzenleyerek port sayısını 270'e düşürdüm. Tarama 49 saatten biraz fazla bir sürede tamamlandı ve 66,558 aygit bulundu. Zamanlama değişkenlerini ayarlamak, verbose seçeneğini kaldırmak ve çıktıyı /dev/null'a yönlendirmek bu süreyle 46 saatे düşündü. Geriye bu son komut kaldı:

```
# nmap -O -T5 -PE -F --osscan-limit --max-rtt-timeout 100ms \
--max-parallelism 100 --min-hostgroup 100 -oX mayo.xml \
-iL ip_networks.txt
```

Bu taramayı haftalık olarak gerçekleştirmeyi ve çıktıyı XML formatında bir MS SQL veritabanına sağlamayı planlıyorum. Diğer tarama yöntemlerimiz zaten bu veritabanını besliyor ve ağda ne olduğunu bilmek şeklindeki asıl hedefimize ulaşmamıza yardımcı olacak raporlar oluşturabiliyoruz. Taramanın alt kümelerini birkaç sistem üzerinde çalıştırarak yükü dağıtmaya karar verebilirim.

[11] Nmap 4.75 veya daha yüksek sürümlerde -F taranan port sayısını 100'e düşürdüğü için daha da etkilidir.

Chapter 7. Service and Application Version Detection (Bölüm 7. Hizmet ve Uygulama Sürümü Algılama)

- Introduction (İçindekiler)
- Usage and Examples (Kullanım ve Örnekler)
- Technique Described (Açıklanan Teknik)
 - Cheats and Fallbacks (Hileler ve Geri Dönüşler)
 - Probe Selection and Rarity (Prob Seçimi ve Nadirlik)
- Technique Demonstrated (Gösterilen Teknik)
- Post-processors (Son işlemciler)
 - Nmap Scripting Engine Integration (Nmap Scripting Engine Entegrasyonu)
 - RPC Grinding (RPC Taşlama)
 - SSL Post-processor Notes (SSL Post-işlemci Notları)
- `nmap-service-probes` File Format (nmap-service-probes Dosya Formatı)
 - `Exclude` Directive (Hariç Tutma Yönergesi)
 - `Probe` Directive (Prob Yönergesi)
 - `match` Directive (match Yönergesi)
 - `softmatch` Directive (softmatch Yönergesi)
 - `ports` and `sslpports` Directives (ports ve sslports Yönergeleri)
 - `totalwaitms` Directive (totalwaitms Yönergesi)
 - `tcpwrappedms` Directive (tcpwrappedms Yönergesi)
 - `rarity` Directive (nadirlik Yönergesi)
 - `fallback` Directive (fallback Yönergesi)
 - Putting It All Together (Hepsini Bir Araya Getirme)
- Community Contributions (Topluluk Katkıları)
 - Submit Service Fingerprints (Servis Parmak İzlerini Gönder)
 - Submit Database Corrections (Veritabanı Düzeltmelerini Gönder)
 - Submit New Probes (Yeni Probları Gönder)
- SOLUTION: Find All Servers Running an Insecure or Nonstandard Application Version (ÇÖZÜM: Güvensiz veya Standart Olmayan Uygulama Sürümü Çalıştıran Tüm Sunucuları Bulun)
 - Problem (Sorun)
 - Solution (Çözüm)
 - Discussion (Tartışma)
- SOLUTION: Hack Version Detection to Suit Custom Needs, such as Open Proxy Detection (ÇÖZÜM: Açık Proxy Algılama gibi Özel İhtiyaçlara Uygun Sürüm Algılamayı Hackleyin)

- Problem (Sorun)
- Solution (Çözüm)
- Discussion (Tartışma)

Introduction (İçindekiler)

Nmap birçok şey yapsa da, en temel özelliği port taramasıdır. Nmap'i uzaktaki bir makineye yönlendirdiğinizde size 25/tcp, 80/tcp ve 53/udp bağlantı noktalarının açık olduğunu söyleyebilir. Nmap, 2.200'den fazla iyi bilinen hizmetten oluşan nmap-services veritabanını kullanarak, bu bağlantı noktalarının muhtemelen sırasıyla bir posta sunucusuna (SMTP), web sunucusuna (HTTP) ve ad sunucusuna (DNS) karşılık geldiğini bildirecektir. Bu arama genellikle doğrudur - TCP bağlantı noktası 25'i dinleyen daemonların büyük çoğunluğu aslında posta sunucularıdır. Ancak, güvenliğinizı buna bağlamamalısınız! İnsanlar garip bağlantı noktalarında hizmet çalıştırabilir ve çalıştırıbmaktadır. Belki de ana web sunucuları zaten 80 numaralı bağlantı noktasındaydı, bu nedenle hazırlama veya test sunucusu için farklı bir bağlantı noktası seçtiler. Belki de savunmasız bir hizmeti belirsiz bir portta saklanmanın "kötü hackerların" onu bulmasını engellediğini düşünüyordardır. Son zamanlarda daha da yaygın olan bir durum ise insanların portları çalıştırıbmak istedikleri hizmete göre değil, güvenlik duvarından ne geleceğine göre seçmeleridir. Büyük Microsoft IIS solucanları CodeRed ve Nimda'dan sonra ISS'ler 80 numaralı portu engellediğinde, kullanıcı orduları kişisel web sunucularını başka bir porta taşıyarak yanıt verdi. Şirketler korkunç güvenlik riskleri nedeniyle Telnet erişimini engellediğinde, kullanıcıların bunun yerine Güvenli Kabuk (SSH) portunda telnet'd çalıştırıldıklarını gördüm.

Nmap haklı olsa ve yukarıdaki varsayımsal sunucu SMTP, HTTP ve DNS sunucularını çalıştırıyor olsa bile, bu çok fazla bilgi değildir. Şirketlerinizin veya müşterilerinizin güvenlik açığı değerlendirmelerini (hatta basit ağ envanterlerini) yaparken, hangi posta ve DNS sunucularının ve sürümlerinin çalıştığını gerçekten bilmek istersiniz. Doğru bir sürüm numarasına sahip olmak, bir sunucunun hangi açıklara karşı savunmasız olduğunu belirlemeye önemli ölçüde yardımcı olur. Güvenlik düzeltmelerinin genellikle yazılımın önceki sürümlerine geri aktarıldığını unutmayın, bu nedenle bir hizmetin savunmasız olduğunu kanıtlamak için yalnızca sürüm numarasına güvenemezsınız. Yanlış negatifler daha nadirdir, ancak aptal yöneticiler savunmasız bir hizmetin sürüm numarasını değiştirerek yamalı görünmesini sağladığında ortaya çıkabilir.

Hizmet türlerini ve sürüm numaralarını belirlemenin bir başka iyi nedeni de birçok hizmetin aynı bağlantı noktası numarasını paylaşmasıdır. Örneğin, 258/tcp bağlantı noktası hem Checkpoint Firewall-1 GUI yönetim arayüzü hem de yak Windows sohbet istemcisi tarafından kullanılır. Bu, nmap-services tablosuna dayalı bir tahmini daha da az doğru yapar. Çok fazla tarama yapan herkes bilir ki genellikle kayıtlı olmayan portları dinleyen servisler de bulabilirsiniz - bunlar versiyon tespiti olmadan tam bir gizemdir. Son bir sorun da filtrelenmiş UDP portlarının genellikle basit bir port tarayıcıya açık portlarla aynı görünmesidir ("UDP Taraması (-sU)" bölümüne bakın). Ancak Nmap sürüm tespiti tarafından gönderilen hizmete özel problere yanıt verirlerse, açık olduklarından (ve genellikle tam olarak neyin çalıştığından) emin olabilirsiniz.

Hizmet taramaları bazen bir hedef hakkında hizmet türü ve sürüm numarasının ötesinde bilgiler ortaya çıkarır. Bir hizmet hakkında keşfedilen çeşitli bilgiler "info" alanında toplanır. Bu, ürün adı ve sürüm numarasını takip eden parantez içindeki VERSİYON sütununda görüntülenir. Bu alan SSH protokol numaralarını, Apache modüllerini ve çok daha fazlasını içerebilir.

Bazı hizmetler, makinelerin ters DNS ana bilgisayar adlarından şaşırıcı derecede farklı olan yapılandırılmış ana bilgisayar adlarını da bildirir. Ana bilgisayar adı alanı, bağlantı noktası tablosunu takip eden bir Hizmet Bilgisi satırında raporlanır. Küçük bir bilgi sızıntısı gibi görünse de sonuçları olabilir. Bir yıl CanSecWest güvenlik konferansında, dizüstü bilgisayarımla odamda toplantımdı. Birden ekranımın köşesindeki tcpdump penceresi

çılgına döndü ve makinemin saldırısının altında olduğunu fark ettim. Geri taradım ve alışılmadık yüksek bir portun açık olduğunu gördüm. Bağlandıktan sonra, port bir sürü ikili karakter çıktı, ancak çıktıdaki bir ASCII alanı yapılandırılmış bir alan adı verdi. Alan adı, tam olarak kimin sorumlu olduğunu bildiğim kadar küçük bir güvenlik şirketine aitti. Resepsiyona otel odasını arattım ve kutumu araştırmayı bırakmasını istedigimde çok şaşırdı.

Sürüm algılamanın keşfedebileceğinin iki alan daha işletim sistemi ve cihaz türüdür. Bunlar da Servis Bilgisi satırında rapor edilir. Burada iki teknik kullanıyoruz. Birincisi uygulama ayrıcalığıdır. Eğer bir hizmeti Microsoft Exchange olarak tanımlarsak, işletim sisteminin Windows olduğunu biliriz çünkü Exchange başka hiçbir şeyde çalışmaz. Diğer teknik ise daha taşınabilir uygulamaları platform bilgilerini ifşa etmeye ikna etmektedir. Birçok sunucu (özellikle web sunucuları) çok az ikna gerektirir. Bu tür işletim sistemi tespiti Nmap'in işletim sistemi tespit sistemini (-O) tamamlamak için tasarlanmıştır ve bazen farklı sonuçlar bildirebilir. Örneğin, bağlantı noktası yönlendirmeli bir Unix güvenlik duvarının arkasına gizlenmiş bir Microsoft Exchange sunucusunu düşünün.

Nmap sürüm tarama alt sistemi tüm bu verileri açık portlara bağlanarak ve belirli hizmetlerin anladığını problemleri kullanarak daha fazla bilgi için onları sorgulayarak elde eder. Bu, Nmap'in sadece hangi port numaralarının açık olduğundan ziyade, gerçekten neyin çalıştığını dair ayrıntılı bir değerlendirme yapmasını sağlar. Örnek 7.1 gerçek çıktıyı göstermektedir.

Örnek 7.1. Sürüm tespitinin basit kullanımı

```
# nmap -sV -T4 -F insecure.org
Starting Nmap ( https://nmap.org )
Nmap scan report for insecure.org (74.207.254.18)
Host is up (0.016s latency).
rDNS record for 74.207.254.18: web.insecure.org
Not shown: 95 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 4.3 (protocol 2.0)
25/tcp    open  smtp   Postfix smptd
80/tcp    open  http   Apache httpd 2.2.3 ((CentOS))
113/tcp   closed auth
443/tcp   open  ssl/http Apache httpd 2.2.3 ((CentOS))
Service Info: Host: web.insecure.org

Nmap done: 1 IP address (1 host up) scanned in 14.82 seconds
```

Nmap sürüm tespiti aşağıdaki gelişmiş özellikleri sunar (daha sonra tam olarak açıklanacaktır):

- Yüksek hızlı, blokajsız soketlerle paralel çalışma ve verimli ancak güçlü uygulama için tasarlanmış bir prob/eşleşme tanımı grameri.
- Yalnızca hizmet protokolünü değil, varsa uygulama adını ve sürüm numarasını belirler.
- Hem TCP hem de UDP protokollerinin yanı sıra hem metinsel ASCII hem de paketlenmiş ikili hizmetleri destekler.
- Linux, Windows, Mac OS X, FreeBSD/NetBSD/OpenBSD, Solaris ve Nmap'in çalıştığı bilinen diğer tüm platformlar dahil olmak üzere çoklu platform desteği.
- SSL tespit edilirse, Nmap OpenSSL (varsayılan) kullanarak bağlanır ve bu şifreleme katmanının arkasında hangi hizmetin dinlendiğini belirlemeye çalışır. Bu, HTTPS, POP3S, IMAPS, vb. gibi hizmetleri keşfetmesine ve sürüm ayrıntılarını sağlama konusuna olanak tanır.
- Bir SunRPC hizmeti keşfedilirse, Nmap program numarasını, adını ve sürüm numarasını bulmak için kaba kuvvet RPC öğütücüsünü başlatır.
- TCP, UDP ve TCP üzerinden SSL dahil olmak üzere IPv6 desteklenmektedir.

- Diğer yazılımlarla birlikte çalışma için Ortak Platform Numaralandırma (CPE) çıktısı (bazı bilgiler yalnızca XML çıktısında bulunur). CPE hakkında daha fazla bilgi için "Ortak Platform Numaralandırma (CPE)" bölümüne bakın.
- Topluluk katkıları: Nmap tanımadığı bir hizmetten veri alırsa, bir gönderim URL'si ile birlikte bir hizmet parmak izi yazdırılır. Bu sistem, son derece başarılı olan Nmap OS Detection parmak izi gönderme sürecinden sonra tasarlanmıştır. Yeni problemler ve düzeltmeler de gönderilebilir.
- Kapsamlı veritabanı: Nmap, ACAP, AFP ve AIM'den XML-RPC, Zebedee ve Zebra'ya kadar 180'den fazla benzersiz hizmet protokolünü kapsayan binden fazla hizmet imzasını tanır.

Usage and Examples (Kullanım ve Örnekler)

Sürüm tespitinin nasıl uygulandığına ilişkin teknik ayrıntılara girmeden önce, burada kullanımını ve yeteneklerini gösteren bazı örnekler bulunmaktadır. Sürüm algılamayı etkinleştirmek için, normalde kullandığınız Nmap bayraklarına -sV eklemeniz yeterlidir. Ya da -A seçeneğini kullanın, bu seçenek daha sonra sürüm algılamayı ve diğer Gelişmiş ve Agresif özellikleri açar. Örnek 7.2'de gösterildiği gibi gerçekten bu kadar basit.

Örnek 7.2. www.microsoft.com adresine karşı sürüm tespiti

```
# nmap -A -T4 -F www.microsoft.com
Starting Nmap ( https://nmap.org )
Nmap scan report for 80.67.68.30
(The 1208 ports scanned but not shown below are in state: closed)
PORT      STATE    SERVICE      VERSION
22/tcp    open     ssh          Akamai-I SSH (protocol 1.5)
80/tcp    open     http         AkamaiGHost (Akamai's HTTP Acceleration service)
443/tcp   open     ssl/http    AkamaiGHost (Akamai's HTTP Acceleration service)
Device type: general purpose
Running: Linux 2.1.X|2.2.X
OS details: Linux 2.1.19 - 2.2.25

Nmap finished: 1 IP address (1 host up) scanned in 19.223 seconds
```

Bu önceki tarama birkaç şeyi göstermektedir. Öncelikle, www.Microsoft.Com adresinin Akamai'nin Linux kutularından birinden sunulduğunu görmek sevindirici. Bu bölümle daha ilgili olan ise 443 numaralı bağlantı noktası için listelenen hizmetin ssl/http olmasıdır. Bu, hizmet algılamanın önce bağlantı noktasının SSL olduğunu keşfettiği, ardından OpenSSL'i yüklediği ve şifrelemenin arkasında AkamaiGHost çalışan bir web sunucusunu keşfetmek için SSL bağlantıları aracılığıyla hizmet algılamayı tekrar gerçekleştirdiği anlamına gelir. T4'ün Nmap'in daha hızlı çalışmasına (daha agresif zamanlama) neden olduğunu ve -F'nin Nmap'e yalnızca nmap-services'te kayıtlı portları taramasını söylediğini hatırlayın.

Örnek 7.3 daha uzun ve daha çeşitli bir örnektir.

Örnek 7.3. Karmaşık sürüm tespiti

```

# nmap -A -T4 localhost
Starting Nmap ( https://nmap.org )
Nmap scan report for felix (127.0.0.1)
(The 1640 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          WU-FTPD wu-2.6.1-20
22/tcp    open  ssh          OpenSSH 3.1pl1 (protocol 1.99)
53/tcp    open  domain       ISC BIND 9.2.1
79/tcp    open  finger       Linux fingerd
111/tcp   open  rpcbind     2 (rpc #100000)
443/tcp   open  ssl/http    Apache httpd 2.0.39 ((Unix) mod_perl/1.99_04-dev)
515/tcp   open  printer      CUPS 1.1
631/tcp   open  ipp          CUPS?
953/tcp   open  rrdnc?
5000/tcp  open  ssl/ftp     WU-FTPD wu-2.6.1-20
5001/tcp  open  ssl/ssh     OpenSSH 3.1pl1 (protocol 1.99)
5002/tcp  open  ssl/domain  ISC BIND 9.2.1
5003/tcp  open  ssl/finger  Linux fingerd
6000/tcp  open  X11          (access denied)
8000/tcp  open  http-proxy   Junkbuster webproxy
8080/tcp  open  http         Apache httpd 2.0.39 ((Unix) mod_perl/1.99_04-dev)
8081/tcp  open  http         Apache httpd 2.0.39 ((Unix) mod_perl/1.99_04-dev)
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux Kernel 2.4.0 - 2.5.20

Nmap finished: 1 IP address (1 host up) scanned in 42.494 seconds

```

Burada RPC hizmetlerinin nasıl ele alındığını görebilirsiniz; 111 numaralı bağlantı noktasının rpcbind sürüm 2 olduğunu belirlemek için kaba kuvvet RPC tarayıcısı kullanılır. Ayrıca 515 numaralı bağlantı noktasının hizmeti yazıcı olarak verdiği, ancak sürüm alanının boş olduğunu fark edebilirsiniz. Nmap problama yaparak servis adını belirledi, ancak başka bir şey belirleyemedi. Öte yandan, 953 numaralı bağlantı noktası hizmeti "rndc?" olarak verir. Soru işaretleri bize Nmap'in problama yoluyla servis adını bile belirleyemediğini söylüyor. Yedek olarak, nmap-services'de 953 numaralı bağlantı noktası kayıtlı olduğu için rndc'den bahsedilmiştir. Ne yazık ki, Nmap'in problemlerinin hiçbirini rndc'den herhangi bir yanıt alamadı. Eğer olsaydı, Nmap bir sonraki sürümde tanınabilmesi için bir hizmet parmak izi ve bir gönderim URL'si yazdırırırdı. Bu haliyle, Nmap özel bir sonda gerektirmektedir. Siz bu yazıyı okuduğunuzda bir tane bile mevcut olabilir. "Topluluk Katkıları" adlı bölümde kendi problemlerinizi yazmaya ilişkin ayrıntılar yer almaktadır.

Bazı hizmetlerin yalnızca sürüm numarasından çok daha fazla bilgi sağladığını da belirtmek gerekir. Yukarıdaki örnekler arasında X11'in bağlantılarına izin verip vermediği, SSH protokol numarası ve Apache modül sürümleri listesi bulunmaktadır. Hatta bazı Apache modüllerinin bu sayfaya sığması için çıktıdan kesilmesi gerekti.

Birkaç ilk yorumcu SSH ve finger gibi hizmetleri SSL üzerinden çalıştırmanın akıl sağılığını sorguladı. Bu aslında paralel SSL taramalarının gerçekten çalışmasını sağlamak için stunnel ile sadece eğlenceliydi.

Technique Described (Açıklanan Teknik)

Nmap sürüm taraması aslında oldukça basittir. Ölçeklenebilir, hızlı ve doğru olmasına rağmen mümkün olduğunda basit olacak şekilde tasarlanmıştır. Gerçekten ince ayrıntılar en iyi kaynak kodunu indirip inceleyerek keşfedilebilir, ancak kullanılan tekniklerin bir özeti aşağıda verilmiştir.

Nmap önce talimatlarınıza göre bir port taraması yapar ve ardından tüm açık veya açık/filtrelenmiş TCP ve/veya UDP portlarını servis tarama modülüne iletir. Bu portlar daha sonra paralel olarak sorgulanır, ancak burada basitlik için tek bir port açıklanmıştır.

1. Nmap, portun nmap-service-probes içinde Exclude yönergesi ile belirtildiği gibi hariç tutulacak portlardan biri olup olmadığını kontrol eder. Eğer öyleyse, Nmap "nmap-service-probes Dosya Formatı" bölümünde belirtilen nedenlerden dolayı bu portu taramayacaktır.
2. Eğer port TCP ise, Nmap porta bağlanarak başlar. Bağlantı başarılı olursa ve port açık|filtreli durumdaysa, açık olarak değiştirilir. Bu (TCP için) nadirdir, çünkü açık|filtrelenmiş portlar üreten bir TCP tarama türü (FIN taraması gibi) kullanacak kadar gizli olmaya çalışan insanlar genellikle sürüm tespiti yaparak tüm gizliliklerini bozmaktan daha iyi bilirler.
3. TCP bağlantısı yapıldıktan sonra, Nmap yaklaşık beş saniye dinler. Çoğu FTP, SSH, SMTP, Telnet, POP3 ve IMAP sunucusu dahil olmak üzere birçok yaygın hizmet, kendilerini ilk karşılaşma banner'ında tanır. Nmap bunu "NULL probe" olarak adlandırır çünkü Nmap herhangi bir probe verisi göndermeden sadece yanıtları dinler. Herhangi bir veri alınırsa, Nmap bunu nmap-service-probes dosyasındaki ("nmap-service-probes Dosya Formatı" adlı bölümde açıklanmıştır) yaklaşık 3.000 NULL prob imzasıyla karşılaşır. Eğer servis tam olarak tanımlanmışsa, o port ile işimiz bitmiştir! İmzalardaki düzenli ifade, yanıtta sürüm numaralarını seçmek için alt dize eşleşmelerini içerebilir. Bazı durumlarda, Nmap hizmet türünde bir "yumuşak eşleşme" alır, ancak sürüm bilgisi yoktur. Bu durumda, Nmap devam eder, ancak yalnızca yumuşak eşleşen hizmet türünü tanıdığı bilinen problemleri gönderir.
4. Bu nokta, Nmap'in UDP problemleri için başladığı yerdir ve yukarıda açıklanan NULL probu başarısız olursa veya yumuşak eşleşirse TCP bağlantıları burada devam eder. Gerçek şu ki, çoğu port nmap-services'de kayıtlı oldukları servis tarafından kullanıldığından, her probun en etkili olduğu düşünülen port numaralarının bir listesi vardır. Örneğin, web sunucularını (diğer birçok hizmetin yanı sıra) tanıyan GetRequest adlı prob, olası portları arasında 80-85, 8000-8010 ve 8080-8085'i listeler. Nmap sırayla (dosyada göründükleri sırayla) taranan port numarasıyla eşleşen prob(ları) çalıştırır.
 - Her prob, porta gönderilen bir prob dizesi (rastgele ASCII metni veya \xHH kaçmış ikili olabilir) içerir. Geri gelen yanıtlar, yukarıdaki NULL probu açıklamasında tartışıldığı gibi aynı türden imza düzenli ifadelerin bir listesiyle karşılaşır. NULL probunda olduğu gibi, bu testler tam eşleşme (uzak hizmet için işlemeyi sonlandırır), yumuşak eşleşme (gelecek problemleri belirli bir hizmetle eşleşenlerle sınırlar) veya hiç eşleşme olmaması ile sonuçlanabilir. Nmap'in bir eşleşmeye test etmek için kullandığı düzenli ifadelerin tam listesi, prob geri dönüş yapılmamasına bağlıdır. Örneğin, X11Probe'dan döndürülen verilerin GetRequest probu için hazırlanmış herhangi bir düzenli ifadeyle eşleşmesi pek olası değildir. Öte yandan, test edilen iki protokol yakından ilişkili olduğu için RTSPRequest gibi bir Probdan döndürülen sonuçların GetRequest için hazırlanmış bir düzenli ifadeyle eşleşmesi olasıdır. Bu nedenle, RTSPRequest sondasının GetRequest eşleşmeleri için bir geri dönüşü vardır. Daha kapsamlı bir açıklama için "Hileler ve Geri Dönüşler" adlı bölümę bakın.
 - Sürüm tespiti sırasında açık|filtreli durumda olan bir UDP portundan herhangi bir yanıt alınırsa, bu durum açık olarak değiştirilir. Bu, sürüm algılamayı, bazı yaygın güvenlik duvarı kuralları yüreklikte olduğunda taranan tüm UDP bağlantı noktalarını açık|filtreli olarak etiketlemek zorunda kalan UDP taraması için mükemmel bir tamamlayıcı yapar. UDP taramasını sürüm tespiti ile birleştirmek düz bir UDP taramasından çok daha uzun sürebilir, ancak etkili ve kullanışlı bir tekniktir. Bu yöntem "Açık ve Filtrelenmiş UDP Portlarını Ayırt Etme" başlıklı bölümde açıklanmıştır.
5. Çoğu durumda, yukarıda açıklanan NULL probu veya olası port probu (problemleri) (genellikle yalnızca bir tane vardır) hizmetle eşleşir. NULL probu bağlantısını muhtemel port probu ile paylaştığından, bu durum çoğu durumda servis tespitinin sadece bir kısa bağlantı ile yapılmasına olanak sağlar. UDP ile genellikle yalnızca bir paket gereklidir. Ancak NULL probu ve olası port prob(ları) başarısız olursa, Nmap diğer mevcut problemleri sırayla gözden geçirir. TCP durumunda, Nmap önceki problemlerin sonuçlarını bozmasını önlemek için her prob için yeni bir bağlantı kurmalıdır. Bu en kötü senaryo biraz zaman alabilir, özellikle de yavaş ağ bağlantıları ve

diğer yavaş yanıt veren hizmetler nedeniyle Nmap'in her bir probun sonuçları için yaklaşık beş saniye beklemesi gerektiğinden. Neyse ki, Nmap taramaları hızlandırmak için çeşitli otomatik teknikler kullanır:

- Nmap çoğu probu birçok servisle eşleşecek kadar genel hale getirir. Örneğin, GenericLines probu servise iki boş satır ("\\r\\n\\r\\n") gönderir. Bu, FTP, ident, POP3, UUCP, Postgres ve whois dahil olmak üzere birçok farklı hizmet türünün daemonlarıyla eşleşir. GetRequest probu daha da fazla hizmet türüyle eşleşir. Diğer örnekler arasında "help\\r\\n" ve genel RPC ve MS SMB probları bulunmaktadır.
 - Eğer bir servis softmatch yönergesi ile eşleşirse, Nmap'in sadece bu servisle eşleşebilecek problemleri denemesi gereklidir.
 - Tüm problemler eşit yaratılmamıştır! Bazıları diğerlerinden çok daha fazla hizmetle eşleşir. Bu nedenle Nmap, eşleşme olasılığı son derece düşük olan problemleri denemekten kaçınmak için nadirlik metriğini kullanır. Deneyimli Nmap kullanıcıları, "Prob Seçimi ve Nadirlik" başlıklı bölümde anlatılan --version-intensity, --version-all ve --version-light seçeneklerini kullanarak tüm problemlerin denenmesini zorlayabilir veya problemleri denemelerini varsayılandan daha da fazla sınırlayabilir.
6. Problardan biri hedef portun SSL çalıştırıp çalışmadığını test eder. Eğer öyleyse (ve OpenSSL mevcutsa), Nmap SSL üzerinden geri bağlanır ve şifrelemenin arkasında neyin dirlendiğini belirlemek için hizmet taramasını yeniden başlatır. Özel bir yönerge normal ve SSL tüneli bağlantılar için farklı olası portlara izin verir. Örneğin, Nmap 443 (HTTPS) portuna karşı bir SSL sondası ile başlamalıdır. Ancak SSL algılandıkta ve etkinleştirildikten sonra, Nmap 443 numaralı bağlantı noktasına karşı GetRequest probunu denemelidir, çünkü bu bağlantı noktasında genellikle SSL şifrelemesinin arkasında dinleyen bir web sunucusu vardır.
 7. Başka bir genel sonda RPC tabanlı hizmetleri tanımlar. Bunlar bulunduğuanda, Nmap RPC öğütücü (daha sonra ele alınacaktır) RPC program numarasını/adını ve desteklenen sürüm numaralarını kaba kuvvetle zorlamak için başlatılır. Benzer şekilde, Windows hizmetlerinin parmak izini almak için bir SMB son işlemcisi Bölüm 9, Nmap Scripting Engine'in bir parçası olarak mevcuttur.
 8. Problardan en az biri bir tür yanıt verirse, ancak Nmap hizmeti tanıyamazsa, yanıt içeriği kullanıcıya bir parmak izi şeklinde yazdırılır. Kullanıcılar gerçekle hizmetlerin dirlendiğini biliyorlarsa, "Hizmet Parmak İzlerini Gönderme" bölümünde açıklandığı gibi, Nmap'e entegrasyon için parmak izini Nmap geliştiricilerine göndermeleri teşvik edilir.

Cheats and Fallbacks (Hileler ve Geri Dönüşler)

Nmap hizmetlerin yanıt vermesi için cömert bir süre beklesede de, bazen bir uygulama NULL probuna yanıt vermekte yavaş kalabilir. Bu, bazı hizmetler tarafından gerçekleştirilen yavaş ters DNS aramaları da dahil olmak üzere çeşitli nedenlerle ortaya çıkabilir. Bu nedenle, Nmap bazen sonraki bir sondadan gelen sonuçları NULL sonda için tasarlanmış bir eşleşme satırıyla eşleştirebilir.

Örneğin, neyin dirlendiğini belirlemek için bir sunucudaki 25 numaralı bağlantı noktasını (SMTP) taradığımızı varsayıyalım. Bağlanır bağlanmaz, bu hizmet spam gönderici olarak muamele görüp görmeyeceğimizi ve hizmetin reddedilip reddedilmeyeceğini belirlemek için bir dizi DNS kara liste araması yapabilir. Bunu tamamlamadan önce, Nmap bir NULL prob yanıtını beklemekten vazgeçer ve "HELP\\r\\n" olan 25 numaralı port kayıtlı bir sonraki probu gönderir. Hizmet nihayet anti-spam kontrollerini tamamladığında, bir karşılama başlığı yazdırır. Yardım probunu okur ve Örnek 7.4'te gösterildiği gibi yanıt verir.

Örnek 7.4. NULL prob hile örneği çıktısı

```

220 hcsweb.org ESMTP Sendmail 8.12.3/8.12.3/Debian-7.1; Tue, [cut]
214-2.0.0 This is sendmail version 8.12.3
214-2.0.0 Topics:
214-2.0.0      HELO    EHLO    MAIL    RCPT    DATA
214-2.0.0      RSET    NOOP    QUIT    HELP    VRFY
214-2.0.0      EXPN    VERB    ETRN    DSN    AUTH
214-2.0.0      STARTTLS
214-2.0.0 For more info use "HELP <topic>".
214-2.0.0 To report bugs in the implementation send email to
214-2.0.0      sendmail-bugs@sendmail.org.
214-2.0.0 For local information send email to Postmaster at your site.
214 2.0.0 End of HELP info

```

Nmap bu verileri soketten okur ve Yardım yoklamasındaki hiçbir düzenli ifadenin döndürülen verilerle eşleşmediğini bulur. Bunun nedeni, Nmap'in normalde NULL probu sırasında ESMTP başlığını almayı ve orada eşleetirmeyi beklemesidir.

Bu nispeten yaygın bir senaryo olduğundan, Nmap, proba özgü satırlardan hiçbir eşleşmeyorsa, yanıtları NULL Probe eşleşme satırlarından herhangi biriyle eşleştirmeye çalışarak "hile yapar". Bu durumda, programın Sendmail, sürümün 8.12.3/8.12.3/Debian-7.1 ve ana bilgisayarının hcsweb.org olduğunu bildiren bir null eşleşme satırı mevcuttur.

NULL prob hilesi aslında daha genel bir Nmap özelliğinin özel bir örneğidir: geri dönüş yönergesi "nmap-service-probes Dosya Biçimi" adlı bölümde ayrıntılı olarak açıklanmıştır. Esasen, diğer problkardaki düzenli ifadelerle eşleştirilebilecek sonuçlarla karşılaşma olasılığı olan herhangi bir prob, bu diğer problemleri belirten bir geri dönüş yönergesine sahiptir.

Örneğin, popüler Apache web sunucusunun bazı yapılandırmalarında, Apache GetRequest ("GET / HTTP/1.0\r\n\r\n") sondasına yanıt vermeyecektir çünkü sanal ana bilgisayar adı belirtilmemiştir. Nmap yine de bu sunucuları doğru bir şekilde tanımlayabilir çünkü bu sunucular genellikle HTTPOptions probuna yanıt verir. Bu sonda, Apache'nin HTTPOptions sondalarına verdiği yanıtları tanıtmak için yeterince genel olan GetRequest düzenli ifadelerine bir geri dönüşü sahiptir.

Probe Selection and Rarity (Prob Seçimi ve Nadirlik)

Hangi problemlerin kullanılacağını belirlerken, Nmap bunların nadirliğini göz önünde bulundurur. Bu, probun yararlı veriler döndürme olasılığının bir göstergesidir. Bir probun nadirliği yüksekse, daha az yaygın olduğu kabul edilir ve denenebilme olasılığı daha düşüktür. Nmap kullanıcıları, aşağıda açıklandığı gibi sürüm taramasının yoğunluk seviyesini değiştirerek hangi problemlerin deneneceğiini belirleyebilirler. Nmap'in hangi problemlerin kullanılacağını belirlerken kullandığı kesin algoritma aşağıdaki gibidir:

1. TCP için her zaman ilk olarak NULL probu denenir.
2. Taranmakta olan portun olası port olarak listelendiği tüm problemler ("nmap-service-probes Dosya Formatı" bölümüne bakın) nmap-service-probes içinde göründükleri sırayla denenir.
3. Taramanın mevcut yoğunluk değerine eşit veya daha düşük nadirlik değerine sahip diğer tüm problemler, nmap-service-probes içinde göründükleri sırayla denenir.

Bir probun eşlestiği tespit edildiğinde, algoritma sonlandırılır ve sonuçlar raporlanır.

Nmap'in tüm problemleri (NULL probu dışında) kendileriyle ilişkili bir nadirlik değerine sahip olduğundan, bir sürüm taraması gerçekleştirirken kaç tanesinin deneneceğini kontrol etmek nispeten kolaydır. Tarama için uygun bir yoğunluk seviyesi seçmeniz yeterlidir. Yoğunluk seviyesi ne kadar yüksekse o kadar fazla prob denenecektir. Bu nedenle, çok kapsamlı bir tarama isteniyorsa, daha düşük bir yoğunluk seviyesinde yapılan bir taramadan daha uzun süre de, yüksek bir yoğunluk seviyesi uygundur. Nmap'in varsayılan yoğunluk seviyesi 7'dir, ancak Nmap farklı tarama ihtiyaçları için aşağıdaki anahtarları sağlar:

`--version-intensity <intensity level between 0 and 9>` Sürüm taramasının yoğunluk düzeyini belirtilen değere ayarlar. 0 belirtilirse, yalnızca NULL probu (TCP için) ve portu olası bir port olarak listeleyen problemler denenir. Örnek: nmap

-sV --version-intensity 3 scanme.nmap.org

--version-light Yoğunluk seviyesini 2 olarak ayarlar. Örnek: nmap -sV --version-light scanme.nmap.org

--version-all Yoğunluk seviyesini 9'a ayarlar. Tüm probalar 1 ile 9 arasında bir nadirlik seviyesine sahip olduğundan, bu tüm probaları dener. Yumuşak bir eşleşme olsa bile, tüm ek probalar denenecektir. Örnek: nmap -sV --version-all scanme.nmap.org

Technique Demonstrated (Gösterilen Teknik)

Yukarıdaki İngilizce açıklama yeterince açık değilse, Nmap komut satırınıza --version-trace (ve genellikle -d (hata ayıklama)) seçeneklerini ekleyerek nasıl çalıştığını kendiniz görebilirsiniz. Bu, hizmet taramasının tüm bağlantı ve veri okuma/yazma etkinliğini gösterir. Açıklamalı bir gerçek dünya örneği aşağıdadır.

```
# nmap -sSV -T4 -F -d --version-trace insecure.org
Starting Nmap ( https://nmap.org )
Host insecure.org (205.217.153.53) appears to be up ... good.
Initiating SYN Stealth Scan against insecure.org (205.217.153.53) at 19:53
Initiating service scan against 4 services on 1 host at 19:53
```

SYN taraması 4 açık port buldu-şimdi bunların her birine karşı paralel olarak bir hizmet taraması başlatıyoruz. NULL probu için bir TCP bağlantısı ile başlıyoruz:

```
Starting probes against new service: 205.217.153.53:22 (tcp)
NSOCK (2.0750s) TCP connection requested to 205.217.153.53:22 (IOD #1) EID 8
Starting probes against new service: 205.217.153.53:25 (tcp)
NSOCK (2.0770s) TCP connection requested to 205.217.153.53:25 (IOD #2) EID 16
Starting probes against new service: 205.217.153.53:53 (tcp)
NSOCK (2.0830s) TCP connection requested to 205.217.153.53:53 (IOD #3) EID 24
Starting probes against new service: 205.217.153.53:80 (tcp)
NSOCK (2.0860s) TCP connection requested to 205.217.153.53:80 (IOD #4) EID 32
NSOCK (2.0870s) Callback: CONNECT SUCCESS for EID 32 [205.217.153.53:80]
NSOCK (2.0870s) Read request from IOD #4 [205.217.153.53:80]
(timeout: 5000ms) EID 42
NSOCK (2.0870s) Callback: CONNECT SUCCESS for EID 24 [205.217.153.53:53]
NSOCK (2.0870s) Read request from IOD #3 [205.217.153.53:53]
(timeout: 5000ms) EID 50
NSOCK (2.0870s) Callback: CONNECT SUCCESS for EID 16 [205.217.153.53:25]
NSOCK (2.0870s) Read request from IOD #2 [205.217.153.53:25]
(timeout: 5000ms) EID 58
NSOCK (2.0870s) Callback: CONNECT SUCCESS for EID 8 [205.217.153.53:22]
NSOCK (2.0870s) Read request from IOD #1 [205.217.153.53:22]
(timeout: 5000ms) EID 66
```

Bu noktada, NULL prob bağlantıları dört hizmetin hepsine başarıyla yapılmıştır. 2 saniyeden başlar çünkü ping ve SYN taramaları bu kadar uzun sürmüştür.

```
NSOCK (2.0880s) Callback: READ SUCCESS for EID 66 [205.217.153.53:22]
(23 bytes): SSH-1.99-OpenSSH 3.1p1.
Service scan match: 205.217.153.53:22 is ssh.
Version: |OpenSSH|3.1p1|protocol 1.99|
```

SSH, bağlantı kurulduktan hemen sonra kendisini OpenSSH 3.1p1 olarak tanımlayacak kadar nazikti. Biri gitti, üçü kaldı.

```
NSOCK (2.0880s) Callback: READ SUCCESS for EID 58 [205.217.153.53:25]
(27 bytes): 220 core.lnxnet.net ESMTP..
Service scan soft match: 205.217.153.53:25 is smtp
```

Port 25'teki posta sunucusu da bize faydalı bir banner verdi. Ne tür bir posta sunucusu olduğunu bilmiyoruz, ancak 220 ile başlaması ve ESMTP kelimesini içermesi bize bunun bir posta (SMTP) sunucusu olduğunu söylüyor. Bu yüzden Nmap smtp'yi softmatch eder, yani bundan sonra sadece SMTP sunucularıyla eşleşebilen probalar denenir. Yazdırılan karakterlerin noktalarla temsil edildiğini unutmayın; bu nedenle ESMTP'den sonraki "." gerçekten "\r\n" satır sonlandırma dizisidir.

```
NSOCK (2.0880s) Read request from IOD #2 [205.217.153.53:25]
(timeout: 4996ms) EID 74
NSOCK (7.0880s) Callback: READ TIMEOUT for EID 74 [205.217.153.53:25]
NSOCK (7.0880s) Write request for 6 bytes to IOD #2 EID 83
[205.217.153.53:25]: HELP..
NSOCK (7.0880s) Read request from IOD #2 [205.217.153.53:25]
(timeout: 5000ms) EID 90
```

Nmap, sunucunun söyleyecek daha fazla şeyi olması ihtimaline karşı SMTP bağlantısını biraz daha uzun süre dinler. Okuma isteği beş saniye sonra zaman aşımına uğrar. Nmap daha sonra 25 numaralı bağlantı noktasına kayıtlı ve SMTP imzalarına sahip bir sonraki sondayı bulur. Bu prob sadece Nmap'in bağlantıya yazdığı HELP\r\n'den oluşur.

```
NSOCK (7.0880s) Callback: READ TIMEOUT for EID 50 [205.217.153.53:53]
NSOCK (7.0880s) Write request for 32 bytes to IOD #3 EID 99
[205.217.153.53:53]: .....version.bind.....
NSOCK (7.0880s) Read request from IOD #3 [205.217.153.53:53]
(timeout: 5000ms) EID 106
```

Port 53 üzerindeki DNS sunucusu hiçbir şey döndürmez. nmap-service-probes içinde 53 numaralı bağlantı noktasına kaydedilen ilk prob DNSVersionBindReq'tır ve bir DNS sunucusunu versiyon numarası için sorgular. Bu tel üzerinden gönderilir.

```
NSOCK (7.0880s) Callback: READ TIMEOUT for EID 42 [205.217.153.53:80]
NSOCK (7.0880s) Write request for 18 bytes to IOD #4 EID 115
[205.217.153.53:80]: GET / HTTP/1.0....
NSOCK (7.0880s) Read request from IOD #4 [205.217.153.53:80]
(timeout: 5000ms) EID 122
```

Port 80 NULL probu da herhangi bir veri döndüremedi. Bu prob 80 numaralı bağlantı noktasına kayıtlı olduğu için bir HTTP GET isteği gönderilir.

```
NSOCK (7.0920s) Callback: READ SUCCESS for EID 122
[205.217.153.53:80] [EOF](15858 bytes)
Service scan match: insecure.org (205.217.153.53):80 is http.
Version: |Apache httpd|2.0.39|(Unix) mod_perl/1.99_07-dev..
```

Apache büyük (15KB) bir yanıt döndürdü, bu yüzden yazdırılmadı. Bu yanıt, Nmap'in yanittan seçtiği ayrıntılı yapılandırma bilgilerini sağlamıştır. Port 80 için kayıtlı başka prob yoktur. Eğer bu başarısız olsaydı, Nmap nmap-service-probes içindeki ilk TCP probunu deneyecekti. Bu prob basitçe boş satırlar gönderir ("\\r\\n\\r\\n"). GET sondasının hizmeti karıştırması durumunda yeni bir bağlantı yapılmış olurdu.

```
NSOCK (7.0920s) Callback: READ SUCCESS for EID 106 [205.217.153.53:53]
(50 bytes): 0.....version.bind.....9.2.1
Service scan match: insecure.org (205.217.153.53):53 is domain.
Version: |ISC BIND|9.2.1|
```

53 numaralı bağlantı noktası DNS sürüm isteğiimize yanıt verdi. Yanıtın çoğu (probda olduğu gibi) ikilidir, ancak orada 9.2.1 sürümünü açıkça görebilirsiniz. Bu sonda başarısız olsaydı, 53 numaralı bağlantı noktasına kaydedilen bir sonraki sonda bir DNS sunucusu durum isteğidir (14 bayt: \0\x0C\0\0\x10\0\0\0\0\0\0). Bu yedek sondaya sahip olmak yardımcı olur çünkü bir durum isteğine sürüm numarası isteğinden çok daha fazla sunucu yanıt verir.

```
NSOCK (7.0920s) Callback: READ SUCCESS for EID 90 [205.217.153.53:25]
      (55 bytes): 214 qmail home page: http...
Service scan match: insecure.org (205.217.153.53):25 is smtp.
Version: |qmail smptd||
```

Port 25, Yardım sondasına çok yararlı bir yanıt verir. Postfix, Courier ve Exim gibi diğer SMTP sunucuları da genellikle bu sonda ile tespit edilebilir. Yanıt eşleşmezse, Nmap bu hizmetten vazgeçerdi çünkü zaten smtp'yi softmatched etmiş ve nmap-service-probes'da daha fazla SMTP probu yoktu.

```
The service scan took 5 seconds to scan 4 services on 1 host.
```

Bu hizmet tarama çalışması oldukça iyi gitti. Hiçbir hizmet birden fazla bağlantı gerektirmedi. Nmap ilk gerçek problemleri göndermeden önce Qmail ve Apache beş saniyelik NULL prob zaman aşımına ugradığı için beş saniye sürdü. İşte bu çabaların ödülü:

```
Interesting ports on insecure.org (205.217.153.53):
(The 1212 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 3.1pl1 (protocol 1.99)
25/tcp    open  smtp     qmail smptd
53/tcp    open  domain   ISC BIND 9.2.1
80/tcp    open  http     Apache httpd 2.0.39 ((Unix) mod_perl/1.99_07-dev)

Nmap finished: 1 IP address (1 host up) scanned in 7.104 seconds
```

Post-processors (Son İşlemciler)

Nmap genellikle yukarıda gösterildiği gibi servis ve sürüm bilgilerini çıkardıktan sonra bir port üzerinde çalışmayı bitirir. Ancak, Nmap'in ek çalışma gerçekleştirdiği belirli hizmetler vardır. Şu anda mevcut olan son işlemciler Nmap Scripting Engine entegrasyonu, RPC taşlama ve SSL tünellemedir. Windows SMB sorgulaması değerlendirme aşamasındadır.

Nmap Scripting Engine Integration (Nmap Scripting Engine Entegrasyonu)

Düzenli ifade tabanlı sürüm algılama yaklaşımı güçlündür, ancak her şeyi tanıyamaz. Bazı hizmetler sadece standart bir sonda gönderilerek ve yanıtla bir desen eşleştirilerek tanınamaz. Bazı hizmetler özel prob dizeleri veya karmaşık çok adımlı bir el sıkışma süreci gerektirir. Diğerleri ise bir yanını tanımak için düzenli bir ifadeden daha gelişmiş bir işlem gerektirir. Örneğin, Skype v2 hizmeti, yerleşik taşıyıcıların (DSL hatları sağlayan telefon şirketleri gibi) bu hizmeti bir rakip olarak görmesi ve abonelerinden gelen hizmeti düşürmesi veya engellemesi riski nedeniyle tespit edilmesi zor olacak şekilde tasarlanmıştır. Bu hizmeti tespit etmek için bulabildiğimiz tek yol iki farklı sondaya verilen yanıtları analiz etmektir. Benzer şekilde, kaba kuvvetle birkaç yüz farklı topluluk adı denerek daha fazla SNMP hizmetini tanıyalırdık. Bu görevlerin hiçbirini geleneksel Nmap sürüm tespiti için uygun değildir, ancak her ikisi de Bölüm 9, Nmap Scripting Engine ile gerçekleştirilmiştir. Bu nedenlerden dolayı, sürüm tespiti artık "NSE Kullanarak Sürüm Tespiti" adlı bölümde açıklandığı gibi bazı zor hizmetleri işlemek için varsayılan olarak NSE'yi çağırmaktadır.

RPC Grinding (RPC Grinding)

SunRPC (Sun Remote Procedure Call), NFS dahil birçok hizmeti uygulamak için kullanılan yaygın bir Unix protokolüdür. Nmap, yaklaşık 600 RPC programından oluşan bir nmap-rpc veritabanı ile birlikte gelir. Birçok RPC hizmeti yüksek numaralı bağlantı noktalarını ve/veya UDP taşıma protokolünü kullanır, bu da onları kötü yapılandırılmış birçok güvenlik duvarı üzerinden kullanılabılır hale getirir. RPC programları (ve altyapı kütüphanelerinin kendileri) ayrıca uzaktan istismar edilebilen ciddi güvenlik açıkları konusunda uzun bir geçmişe sahiptir. Bu nedenle ağ yöneticileri ve güvenlik denetçileri genellikle ağlarındaki RPC programları hakkında daha fazla bilgi edinmek isterler.

Portmapper (rpcbind) hizmeti (UDP veya TCP bağlantı noktası 111) mevcutsa, RPC hizmetleri Unix rpcinfo komutu ile numaralandırılabilir. Örnek 7.5 bunu varsayılan bir Solaris 9 sunucusuna karşı göstermektedir.

Örnek 7.5. RPC hizmetlerini rpcinfo ile numaralandırma

```
> rpcinfo -p ultra
    program vers proto   port
  100000  4   tcp    111  rpcbind
  100000  4   udp    111  rpcbind
  100232  10  udp   32777  sadmind
  100083  1   tcp   32775  ttdbserverd
  100221  1   tcp   32777  kcms_server
  100068  5   udp   32778  cmsd
  100229  1   tcp   32779  metad
  100230  1   tcp   32781  metamhd
  100242  1   tcp   32783  rpc.metamedd
  100001  4   udp   32780  rstatd
  100002  3   udp   32782  rusersd
  100002  3   tcp   32785  rusersd
  100008  1   udp   32784  walld
  100012  1   udp   32786  sprayd
  100011  1   udp   32788  rquotad
  100024  1   udp   32790  status
  100024  1   tcp   32787  status
  100133  1   udp   32790  nsm_addrand
  100133  1   tcp   32787  nsm_addrand
[ Dozens of lines cut for brevity ]
```

Bu örnek, ana bilgisayarların sıkılıkla birçok RPC hizmeti sunduğunu göstermektedir, bu da birinin istismar edilebilir olma olasılığını artırmaktadır. Ayrıca hizmetlerin çoğunun garip yüksek numaralı bağlantı noktalarında olduğunu (herhangi bir nedenle değişebilir) ve UDP ve TCP taşıma protokoller arasında bölündüğünü fark etmelisiniz.

RPC bilgileri çok hassas olduğundan, birçok yönetici portmapper bağlantı noktasını (111) engelleyerek bu bilgileri gizlemeye çalışır. Ne yazık ki, bu deliği kapatmaz. Nmap, aşağıdaki üç adımlı işlem aracılığıyla açık RPC portlarıyla doğrudan iletişim kurarak aynı bilgilerin tümünü belirleyebilir.

1. TCP ve/veya UDP port taraması tüm açık portları bulur.
2. Sürüm tespiti, açık bağlantı noktalarından hangilerinin SunRPC protokolünü kullandığını belirler.
3. RPC kaba kuvvet motoru, nmap-rpc'deki 600 program numarasının her birine karşı bir null komutu deneyerek her RPC portunun program kimliğini belirler. Nmap çoğu zaman yanlış tahminde bulunur ve istenen program numarasının bağlantı noktasında dinlenmediğini belirten bir hata mesajı alır. Nmap, bunlardan biri için başarı döndürülene kadar listesindeki her numarayı denemeye devam eder. Nmap, bilinen tüm program numaralarını tüketmesi durumunda veya bağlantı noktası gerçekten RPC olmadığını gösteren hatalı biçimlendirilmiş yanıtlar gönderirse pes eder.

RPC program tanımlama problemleri paralel olarak yapılır ve UDP portları için yeniden iletimler ele alınır. Bu özellik, sürüm tespiti herhangi bir RPC portu bulunduğuunda otomatik olarak etkinleştirilir. Örnek 7.6, sürüm tespitinin bir

parçası olarak yapılan doğrudan RPC taramasını göstermektedir.

Örnek 7.6. Nmap doğrudan RPC taraması

```
# nmap -F -A -sSU ultra
Starting Nmap ( https://nmap.org )
Nmap scan report for ultra.nmap.org (192.168.0.50)
(The 2171 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE VERSION
[A whole bunch of ports cut for brevity]
32776/tcp  open  Kcms_server      1 (rpc #100221)
32776/udp  open  sadmind        10 (rpc #100232)
32777/tcp  open  Kcms_server      1 (rpc #100221)
32777/udp  open  sadmind        10 (rpc #100232)
32778/tcp  open  metad          1 (rpc #100229)
32778/udp  open  cmsd           2-5 (rpc #100068)
32779/tcp  open  metad          1 (rpc #100229)
32779/udp  open  rstatd         2-4 (rpc #100001)
32780/tcp  open  metamhd        1 (rpc #100230)
32780/udp  open  rstatd         2-4 (rpc #100001)
32786/tcp  open  status          1 (rpc #100024)
32786/udp  open  sprayd         1 (rpc #100012)
32787/tcp  open  status          1 (rpc #100024)
32787/udp  open  rquotad        1 (rpc #100011)
Device type: general purpose
Running: Sun Solaris 9
OS details: Sun Solaris 9
Nmap finished: 1 IP address (1 host up) scanned in 252.701 seconds
```

SSL Post-processor Notes (SSL Son İşlemci Notları)

Teknik bölümünde tartışıldığı gibi, Nmap SSL şifreleme protokolünü tespit etme ve ardından normal sürüm tespitini gerçekleştirdiği şifreli bir oturum başlatma yeteneğine sahiptir. Daha önce tartışılan RPC ögütücüsünde olduğu gibi, uygun bir (SSL) port tespit edildiğinde SSL son işlemcisi otomatik olarak çalıştırılır. Bu durum Örnek 7.7'de gösterilmiştir.

Örnek 7.7. SSL aracılığıyla sürüm taraması

```
$ nmap -Pn -sSV -T4 -F www.amazon.com
Starting Nmap ( https://nmap.org )
Nmap scan report for 207-171-184-16.amazon.com (207.171.184.16)
(The 1214 ports scanned but not shown below are in state: filtered)
PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache Stronghold httpd 2.4.2 (based on Apache 1.3.6)
443/tcp   open  ssl/http Apache Stronghold httpd 2.4.2 (based on Apache 1.3.6)

Nmap finished: 1 IP address (1 host up) scanned in 35.038 seconds
```

Sürüm bilgisinin her iki açık bağlantı noktası için de aynı olduğuna, ancak hizmetin 80 numaralı bağlantı noktasında http ve 443 numaralı bağlantı noktasında ssl/http olduğuna dikkat edin. HTTPS'nin 443 numaralı bağlantı noktasındaki yaygın durumu sabit kodlu değildir-Nmap herhangi bir bağlantı noktasındaki SSL'yi algılayabilmeli ve Nmap'in açık metin olarak algılayıldığı herhangi bir hizmet için temel protokolü belirleyebilmelidir. Eğer Nmap SSL arkasında dinleme yapan sunucuyu tespit etmemiş olsaydı, listelenen hizmet ssl/unknown olurdu. Eğer Nmap SSL desteği ile oluşturulmamış olsaydı, listelenen hizmet sadece ssl olurdu. Bu iki durumda da sürüm alanı boş olacaktır.

Nmap için SSL desteği ücretsiz OpenSSL kütüphanesine bağlıdır. Bu kütüphanelere sahip olmayan sistemlerin bozulmasını önlemek için Linux RPM ikili dosyalarına dahil edilmemiştir. Nmap kaynak kodu dağıtıımı, bir sistemdeki OpenSSL'yi tespit etmeye ve mevcut olduğunda ona bağlanmaya çalışır. OpenSSL'i dahil etmek veya hariç tutmak için derleme işlemini özelleştirme hakkında ayrıntılar için Bölüm 2, Nmap'i Edinme, Derleme, Yükleme ve Kaldırma'ya bakın.

nmap-service-probes

File Format (nmap-service-probes Dosya Biçimi)

Uzak işletim sistemi tespitinde (-O) olduğu gibi, Nmap sürüm tespit problemlerini ve eşleşme dizelerini saklamak için düz bir dosya kullanır. Nmap ile dağıtılan nmap-services sürümü çoğu kullanıcı için yeterli olsa da, dosya biçimini anlamak ileri düzey Nmap korsanlarının algılama motoruna kendi hizmetlerini eklemelerine olanak tanır. Birçok Unix dosyası gibi, nmap-service-probes da satır odaklıdır. Bir hash (#) ile başlayan satırlar yorum olarak değerlendirilir ve ayırtıcı tarafından yok sayılır. Boş satırlar da yok sayılır. Diğer satırlar aşağıda açıklanan direktiflerden birini içermelidir. Bazı okuyucular aşağıdaki incelemeye geçmeden önce "Hepsini Bir Araya Getirmek" adlı bölümdeki örnekler göz atmayı tercih edebilirler.

Exclude Directive (Hariç Tutma Yönergesi)

Syntax: `Exclude <port specification>`

Örnekler:

```
Exclude 53,T:9100,U:30000-40000
```

Bu yönerge belirtilen portları sürüm taramasının dışında tutar. Yalnızca bir kez kullanılabilir ve dosyanın üst kısmına yakın, tüm Probe yönergelerinin üzerinde olmalıdır. Exclude yönergesi Nmap -p anahtarı ile aynı biçimde kullanır, bu nedenle aralıklar ve virgülle ayrılmış port listeleri desteklenir. Nmap ile birlikte gelen nmap-service-probes'da hariç tutulan tek port TCP port 9100 ile 9107 arasıdır. Bunlar yazıcıların dinlediği yaygın bağlantı noktalarıdır ve genellikle kendilerine gönderilen tüm verileri yazdırırlar. Bu nedenle bir sürüm algılama taraması, Nmap'in gönderdiği SunRPC istekleri, yardım ifadeleri ve X11 problemleri gibi problemlerle dolu birçok sayfa yazdırılmalarına neden olabilir.

Bu davranış, özellikle bir taramanın gizli olması amaçlandığında, genellikle istenmeyen bir durumdur. Bununla birlikte, Nmap'in bu bağlantı noktasını taramaktan kaçınma varsayılan davranış, sinsi bir kullanıcının bir hizmeti gizlemesini kolaylaştırabilir: sadece 9100 gibi hariç tutulan bir bağlantı noktasında çalıştırın ve adla tanımlanma olasılığı daha düşüktür. Port taraması yine de açık olarak gösterecektir. Kullanıcılar --allports seçeneği ile Exclude yönergesini geçersiz kılabılırler. Bu, sürüm algılamanın tüm açık portları sorgulamasına neden olur.

Probe Directive (Sonda Yönergesi)

Syntax: `Probe <protocol> <probename> <probestring> [no-payload]`

Örnekler:

```
Probe TCP GetRequest q|GET / HTTP/1.0\r\n\r\n|
Probe UDP DNSStatusRequest q|\0\0\x10\0\0\0\0\0\0\0\0\0\0|
Probe TCP NULL q||
Probe UDP Sqlping q|\x02| no-payload
```

Probe yönergesi Nmap'e çeşitli servisleri tanıtmak için hangi dizgeyi göndereceğini söyler. Daha sonra tartışılabilecek olan tüm yönergeler en son Probe deyimi üzerinde çalışır. Argümanlar aşağıdaki gibidir:

`<protocol>` Bu TCP ya da UDP olmalıdır. Nmap yalnızca taramaya çalıştığı hizmetin protokolüyle eşleşen problemleri kullanır.

`<probename>` Bu, prob için düz İngilizce bir ismidir. Hizmet parmak izlerinde hangi problemlerin yanıt verdiği tanımlamak için kullanılır.

`<probestring>` Nmap'e ne göndereceğini söyler. Bir q ile başlamalı, ardından dizeyi başlatan ve bitiren bir sınırlayıcı karakter olmalıdır. Sınırlayıcı karakterler arasında gerçekte gönderilen dize bulunur. Aşağıdaki standart kaçış karakterlerine izin verdiği için bir C veya Perl dizesine benzer şekilde biçimlendirilir: \\ \0, \a, \b, \f, \n, \r, \t, \v ve \xHH (burada H herhangi bir onaltılık basamaktır). nmap-service-probes içindeki bir Prob satırı, yukarıdaki üçüncü örnekte gösterildiği gibi boş bir prob dizesine sahiptir. Bu, birçok hizmetin gönderdiği ilk banner'ları dinleyen TCP NULL probudur. Prob dizeniz için sınırlayıcı karakteriniz (bu örneklerde |) gerekiyorsa, farklı bir sınırlayıcı seçmeniz gereklidir.

[`no-payload`] Bu anahtar sözcük, Nmap'e UDP port taraması sırasında protokole özgü bir yük olarak verilen probu kullanmaması talimatını vermek için kullanılır. Örneğin Sqlping probu bazen "MS-SQL ping girişimi" için IDS uyarılarını tetikler.

`match Directive (Maç Yönergesi)`

Syntax: `match <service> <pattern> [<versioninfo>]`

Örnekler:

```
match ftp m/^220.*Welcome to .*Pure-?FTPd (\d\S+\s*)/ p/Pure-FTPd/ v/$1/ cpe:/a:pureftpd:pure-ftpd:$1/
match ssh m/^SSH-(\d\.)+)-OpenSSH[_-](\w\.)+\r?\n/i p/OpenSSH/ v/$2/ i/protocol $1/ cpe:/a:openbsd:op
enssh:$2/
match mysql m|^x10\0\0\x01\xff\x13\x04Bad handshake$| p/MySQL/ cpe:/a:mysql:mysql/
match chargen m|@ABCDEFGHIJKLMNPQRSTUVWXYZ|
match uucp m|^login: login: $| p/NetBSD uucpd/ o/NetBSD/ cpe:/o:netbsd:netbsd/a
match printer m|^(\w\_-)+: lpd: Illegal service request\n$| p/lpd/ h/$1/
match afs m|^[\d\D]{28}\s*(OpenAFS)([\d\.\.]{3}[\^\s\0]* )\0| p/$1/ v/$2/
```

Match yönergesi Nmap'e bir önceki Probe yönergesi tarafından gönderilen dizgeye verilen yanıtlarla göre hizmetleri nasıl tanıyalacağını söyler. Tek bir Probe satırını düzinelere veya yüzlerce match deyiği takip edebilir. Verilen kalıp eşleşirse, isteğe bağlı bir sürüm belirteci uygulama adını, sürüm numarasını ve Nmap'in raporlaması için ek bilgileri oluşturur. Bu yönergenin argümanları aşağıdaki gibidir:

`<service>` Bu, basitçe kalının eşleştiği hizmet adıdır. Örnekler ssh, smtp, http veya snmp olabilir. Özel bir durum olarak, hizmet adının önüne ssl/vmware-auth gibi ssl/ ekini getirebilirsiniz. Bu durumda, hizmet SSL ile tünenmiş vmware-auth olarak saklanır. Bu, bir SSL bağlantısı yapmanın ek yükü olmadan tamamen tanınabilecek hizmetler için kullanılabilir.

`<pattern>` Bu kalıp, alınan yanıtın önceki parametrede verilen hizmetle eşleşip eşleşmediğini belirlemek için kullanılır. Biçim Perl gibidir, sözdizimi `m/[regex]/[opts]` şeklindedir. "m" Nmap'e bir eşleşme dizesinin başladığını söyler. İleri eğik çizgi (/), ikinci eğik çizgi de eşleşecek şekilde değiştirildiği sürece hemen hemen tüm yazdırılabilir karakterlerle değiştirilebilen bir sınırlayıcıdır. Regex, Perl tarzı bir düzenli ifadedir. Bu, mükemmel Perl Uyumlu Düzenli İfadeler (PCRE) kütüphanesi (<http://www.pcre.org>) tarafından mümkün kılmıştır. Şu anda desteklenen tek seçenek, eşleşmeye büyük/küçük harf duyarsız hale getiren 'i' ve '! belirtecine satırsonu ekleyen 's' seçenekleridir. Tahmin edebileceğiniz gibi, bu iki seçenek Perl'deki ile aynı anlamlara sahiptir. Yakalanacak alt ifadeler (sürüm numaraları gibi) yukarıdaki örneklerin çoğunda gösterildiği gibi parantezlerle çevrelenir.

`<versioninfo>` `<versioninfo>` bölümü aslında birkaç isteğe bağlı alan içerir. Her alan tanımlayıcı bir harfle başlar ("hostname" için h gibi). Ardından imza yazarının seçtiği bir sınırlayıcı karakter gelir. Tercih edilen sınırlayıcı, alanın kendisinde kullanılmadığı sürece eğik çizgidir ('/'). Daha sonra alan değeri ve ardından sınırlayıcı karakter gelir. Aşağıdaki tabloda altı alan açıklanmaktadır:

Tablo 7.1. versioninfo alan biçimleri ve değerleri

Field Format (Alan biçimci)	Value Description (Değer açıklaması)
p/vendorproductname/	Satici ve genellikle hizmet adını içerir ve "Sun Solaris rexecd", "ISC BIND named" veya "Apache httpd" biçimindedir.
v/version/	Sayısal olmayan karakterler ve hatta birden fazla kelime içerebilen uygulama sürümü "numarası".
i/info/	Hemen ulaşılabilen ve yararlı olabilecek çeşitli ek bilgiler. Örnekler arasında bir X sunucusunun kimliği doğrulanmamış bağlantılara açık olup olmadığı veya SSH sunucularının protokol numarası yer alır.
h/hostname/	Bir hizmet tarafından sunulan ana bilgisayar adı (varsı). Bu, SMTP ve POP3 gibi protokoller için yaygındır ve bu ana bilgisayar adları dahili ağlar için olabileceğiinden veya doğrudan ters DNS yanıtlarından farklı olabileceğiinden yararlıdır.
o/operatingsystem/	Hizmetin üzerinde çalıştığı işletim sistemi. Bu, Nmap IP yığını tabanlı işletim sistemi tespiti tarafından bildirilen işletim sisteminden yasal olarak farklı olabilir. Örneğin, hedef IP, istekleri DMZ'deki bir Microsoft IIS sunucusuna iletmek için ağ adresi çevirisini kullanan bir Linux kutusu olabilir. Bu durumda, yoğun işletim sistemi tespiti işletim sistemini Linux olarak bildirirken, hizmet tespiti 80 numaralı bağlantı noktasını Windows olarak bildirmelidir.
d/devicetype/	Hizmetin üzerinde çalıştığı cihazın türü, "yazdırma sunucusu" veya "web kamerası" gibi bir dize. Bazı hizmetler bu bilgiyi açıklar ve daha birçok durumda çıkarılabilir. Örneğin, HP-ChaiServer web sunucusu yalnızca yazıcılarda çalışır. Aygit türlerinin tam listesi için "Aygit Türleri" başlıklı bölümde bakın.
cpe:/cpename/[a]	Hizmetin bazı yönleri için bir CPE adı. Bu birden fazla kez kullanılabilir; yalnızca hizmeti (cpe:/a adları) değil, aynı zamanda işletim sistemini (cpe:/o adları) ve donanım platformunu (cpe:/h adları) da tanımlayabilmek düşünülebilir. Sondaki egek çizgi CPE sözdiziminin bir parçası değildir ancak diğer alanların biçimyle eşleşmesi için dahil edilmiştir. CPE hakkında daha fazla bilgi için "Common Platform Enumeration (CPE)" adlı bölümde bakın.

Alanlardan herhangi biri atlanabilir. Aslında, hizmet hakkında daha fazla bilgi mevcut değilse tüm alanlar atlanabilir. Sürüm alanlarından herhangi biri \$1 veya \$2 gibi numaralandırılmış dizeler içerebilir, bunlar (Perl benzeri bir şekilde) <pattern> içinde karşılık gelen parantezli alt dizeye değiştirilir. cpe:// şablonları içinde, bu tür ikameler şu şekilde dönüştürülür: iki nokta üst üste gibi belirli karakterlerden kaçılır; boşluklar alt çizgiye dönüştürülür ve tüm karakterler küçük harf haline getirilir.

Nadir durumlarda, eklemeden önce değiştirme metnine bir yardımcı işlev uygulanabilir. Aşağıdaki tabloda mevcut üç yardımcı fonksiyon açıklanmaktadır:

Tablo 7.2. versioninfo yardımcı fonksiyonları

Helper Function (Yardımcı fonksiyon)	Function Description (İşlev Açıklaması)
\$P()	Yazdırılamayan karakterleri filtreler. Bu, W\00\0R\0K\0G\0R\0O\0U\0P\0 gibi Unicode UTF-16 kodlu dizeleri ASCII yaklaşımı WORKGROUP'a dönüştürmek için kullanılmıştır. Yazdırılabilir hale getirmek istediğiniz eşleşmenin numarasını ileterek herhangi bir versioninfo alanında kullanılabilir, şu şekilde: i/\$P(3)/.
\$SUBST()	Yazdırılmadan önce eşleşmelerde değişiklik yapar. Üç bağımsız değişken alır. Birincisi, \$1 veya \$3 gibi normal bir değiştirme değişkeninde kullandığınız gibi

	desendeki değiştirme numarasıdır. İkinci ve üçüncü bağımsız değişkenler sırasıyla bulmak ve değiştirmek istediğiniz bir alt dizeyi belirtir. Alt dizede bulunan eşleşme dizesinin yalnızca ilk örneği değil, tüm örnekleri değiştirilir. Örneğin, VanDyke VShell sshd sürüm numarasını 2_2_3_578 gibi bir biçimde verir. Bunu daha geleneksel olan 2.2.3.578 biçimine dönüştürmek için <code>v/\$SUBST(1,"_",".")/ versioninfo</code> alanını kullanınız.
\$()	Yakalanan baytlardan işaretsiz bir tamsayı çıkarır. En fazla 8 baylıklı yakalanan bir dize verildiğinde, bunları işaretsiz bir tamsayı olarak ele alacak ve ondalık biçimde dönüştürecektir. İki bağımsız değişken alır. Birincisi desendeki ikame numarasıdır. İkincisi, baytların big-endian sırada olduğunu belirtmek için ">" veya little-endian olduğunu belirtmek için "<" dizesidir.

softmatch Directive (softmatch Yönergesi)

Syntax: `softmatch <service> <pattern>`

Örnekler:

```
softmatch ftp m/^220 [-.\w ]+ftp.*\r\n$/i
softmatch smtp m|^220 [-.\w ]+SMTP.*\r\n|
softmatch pop3 m|^+OK [-[\]\(\)!/:<>@\.\w ]+\r\n$|
```

Softmatch yönergesi biçim olarak yukarıda tartışılan match yönergesine benzer. Temel fark, softmatch'ten sonra taramanın devam etmesi, ancak verilen hizmetle eşleştiği bilinen problemler sınırlı olmasıdır. Bu, daha sonra yararlı sürüm bilgileri sağlayabilecek normal ("sert") bir eşleşmenin bulunmasına olanak tanır. Bunun nasıl çalıştığını dair daha fazla ayrıntı için "Açıklanan Teknik" bölümune bakın. Argümanlar burada tanımlanmamıştır çünkü yukarıdaki match ile aynıdır, ancak hiçbir zaman `<versioninfo>` argümanı yoktur. Ayrıca match ile olduğu gibi, tek bir Probe bölümü içinde birçok softmatch deyimi bulunabilir.

ports and ssports Directives (ports ve ssports Yönergeleri)

Syntax: `ports <portlist>`

Örnekler:

```
ports 21,43,110,113,199,505,540,1248,5432,30444
ports 111,4045,32750-32810,38978
```

Bu satır Nmap'e bu prob tarafından tanımlanan servislerin genellikle hangi portlarda bulunduğu söyler. Her Prob bölümünde yalnızca bir kez kullanılmalıdır. Sözdizimi, Nmap -p seçeneği tarafından alınanın biraz basitleştirilmiş bir versiyonudur. Yukarıdaki örnekler bakın. Bunun nasıl çalıştığını dair daha fazla ayrıntı "Açıklanan Teknik" adlı bölümde yer almaktadır.

Syntax: `ssports <portlist>`

Örnekler:

ssports 443

Bu, yukarıda açıklanan 'ports' yönergesi ile aynıdır, ancak bu portlar genellikle bir hizmeti SSL ile sarmak için kullanılır. Örneğin, HTTP probu "ssports 443" bildirir ve SMTP algılama problemleri "ssports 465" satırına sahiptir, çünkü bunlar sırasıyla HTTPS ve SMTPS için standart portlardır. `<portlist>` biçimini portlar ile aynıdır. Bu istege bağlı yönerge her Prob için birden fazla görünemez.

totalwaitms Directive (totalwaitms Direktifi)

Syntax: `totalwaitms <milliseconds>`

Örnekler:

`totalwaitms 6000`

Nadiren gerekli olan bu yönerge, Nmap'in belirli bir hizmete karşı en son tanımlanan Prob'dan vazgeçmeden önce beklemesi gereken süreyi belirtir. Nmap varsayılanı genellikle iyidir.

`tcpwrappedms` Directive (`tcpwrappedms` Yönergesi)

Syntax: `tcpwrappedms <milliseconds>`

Örnekler:

`tcpwrappedms 3000`

Bu yönerge sadece Null sondası için kullanılır. Eğer bir servis bu zamanlayıcı bitmeden TCP bağlantısını kapatırsa, servis tcpwrapped olarak etiketlenir. Aksi takdirde, eşleştirme her zamanki gibi devam eder.

`rarity` Directive (Nadirlik Yönergesi)

Syntax: `rarity <value between 1 and 9>`

Örnekler:

`rarity 6`

Nadirlik yönergesi kabaca bu sondanın ne kadar seyrek olarak yararlı sonuçlar vermesinin beklenebileceğine karşılık gelir. Sayı ne kadar yüksek olursa, prob o kadar nadir kabul edilir ve bir hizmete karşı denenme olasılığı o kadar düşük olur. Daha fazla ayrıntı "Prob Seçimi ve Nadirlik" adlı bölümde bulunabilir.

`fallback` Directive (geri dönüş Yönergesi)

Syntax: `fallback <Comma separated list of probes>`

Örnekler:

`fallback GetRequest,GenericLines`

Bu isteğe bağlı yönerge, geçerli Prob bölümünde hiçbir eşleşme olmaması durumunda hangi problemlerin yedek olarak kullanılacağını belirtir. Yedekler hakkında daha fazla bilgi için "Hileler ve Yedekler" başlıklı bölümde bakınız. Geri dönüş yönergesi olmayan TCP problemleri için, Nmap önce probun kendisindeki eşleşme satırlarını dener ve ardından NULL probuna örtük bir geri dönüş yapar. Eğer geri dönüş yönergesi mevcutsa, Nmap önce probun kendisindeki eşleşme satırlarını, sonra geri dönüş yönergesinde belirtilen problemlerdeki eşleşme satırlarını (soldan sağa) dener. Son olarak, Nmap NULL probu deneyecektir. UDP için davranış, NULL probun asla denenmemesi dışında aynıdır.

Putting It All Together (Hepsini Bir Araya Getirmek)

İşte tüm bunları bir araya getiren nmap-service-probes'dan bazı örnekler (yerden tasarruf etmek için birçok satır atlanmıştır). Bu bölümde bu kadar okuduktan sonra, aşağıdakiler anlaşılmalıdır.

```
# The Exclude directive takes a comma separated list of ports.  
# The format is exactly the same as the -p switch.  
Exclude T:9100-9107  
  
# This is the NULL probe that just compares any banners given to us  
#####NEXT PROBE#####  
Probe TCP NULL q||  
# Wait for at least 5 seconds for data. Otherwise an Nmap default is used.
```

```

totalwaitms 5000
# Windows 2003
match ftp m/^220[ -]Microsoft FTP Service\r\n/ p/Microsoft ftpt/
match ftp m/^220 ProFTPD (\d\S+) Server/ p/ProFTPD/ v/$1/
softmatch ftp m/^220 [-.\w ]+ftp.*\r\n$/i
match ident m|^flock\b(\() on closed filehandle .*midentd| p/midentd/ i/broken/
match imap m|^* OK Welcome to Binc IMAP v(\d[-.\w]+)| p/Binc IMAPd/ v$1/
softmatch imap m|^* OK [-.\w ]+imap[-.\w ]+\r\n$/i
match lucent-fwadm m|^0001;2$| p/Lucent Secure Management Server/
match meetingmaker m/^xc1,$/ p/Meeting Maker calendaring/
# lopster 1.2.0.1 on Linux 1.1
match napster m|^1$| p/Lopster Napster P2P client/

Probe UDP Help q|help\r\n\r\n|
rarity 3
ports 7,13,37
match chargen m|@ABCDEFGHIJKLMNPQRSTUVWXYZ|
match echo m|^help\r\n\r\n$|

```

Community Contributions (Toplumsal Katkılar)

Bir hizmet algılama çerçevesi teknik olarak ne kadar gelişmiş olursa olsun, eşleştirilecek kapsamlı bir hizmet veritabanı olmadan neredeyse işe yaramaz. Nmap'in açık kaynak doğasının gerçekten parladığı yer burasıdır. Insecure.Org laboratuvarı geek standartlarına göre oldukça büyütür, ancak dışarıdaki makine türlerinin ve hizmetlerin küçük bir yüzdesinden fazlasını çalıştırmayı asla umut edemez. Neyse ki işletim sistemi tespit parmak izleri ile ilgili deneyimler, Nmap kullanıcılarının birlikte tüm yaygın şeylerin yanı sıra şarşıcı bir dizi tuhaf ekipmanı da çalıştığını göstermiştir. Nmap OS parmak izi veritabanı, her türlü anahtar, WAP, VoIP telefon, oyun konsolu, Unix kutuları, Windows ana bilgisayarları, yazılıclar, yönlendiriciler, PDA'lar, güvenlik duvarları vb. dahil olmak üzere binden fazla giriş içerir. Sürüm tespiti kullanıcı gönderimlerini de destekler. Nmap kullanıcıları binlerce hizmete katkıda bulunmuştur. Nmap topluluğunun bunu olağanüstü bir veritabanı haline getirmeye yardımcı olduğu üç temel yol vardır: hizmet parmak izleri, veritabanı düzeltmeleri ve yeni probalar göndermek.

Submit Service Fingerprints (Servis Parmak İzlerini Gönderin)

Bir hizmet Nmap'in bir veya daha fazla probuna yanıt verirse ve yine de Nmap bu hizmeti tanımlayamazsa, Nmap bunun gibi bir hizmet parmak izi yazdırır:

```

SF-Port21-TCP:V=3.40PVT16%D=9/6%Time=3F5A961C%r(NULL,3F,"220\x20stage\x20F
SF:TP\x20server\x20\Version\x202\.1WU(1)\+SCO-2\.6\.1\+-sec1)\x20ready\
SF:.\r\n")%r(GenericLines,81,"220\x20stage\x20FTP\x20server\x20\Version\x
SF:202\.1WU(1)\+SCO-2\.6\.1\+-sec1)\x20ready\.\r\n500\x20":\x20command\
SF:x20not\x20understood\.\r\n500\x20":\x20command\x20not\x20understood\.\
SF:r\r\n");

```

Böyle bir parmak izi alırsınız ve hedef ana bilgisayarda hangi daemon sürümünün çalıştığını bildiğinizden eminseniz, lütfen parmak izini Nmap'in size verdiği URL'den gönderin. Tüm gönderim süreci anonimdir (tanımlayıcı bilgi vermeyi seçmediğiniz sürece) ve birkaç dakikadan fazla sürmemelidir. Eğer özellikle

yardımsever hissediyorsanız, sistemi -d kullanarak tekrar tarayın (Nmap bazen bu şekilde daha uzun parmak izleri verir) ve her iki parmak izini de gönderim formundaki parmak izi kutusuna yapıştırın. Bazen insanlar dosya formatı bölümünü okur ve kendi çalışma eşleşme satırlarını gönderirler. Bu sorun değil, ancak lütfen hizmet parmak izlerini de gönderin, çünkü mevcut komut dosyaları bunları entegre etmeyi ve test etmeyi nispeten kolaylaştırır.

İlgilenenler için, yukarıdaki parmak izindeki bilgiler port numarası (21), protokol (TCP), Nmap sürümü (3.40PVT16), tarih (6 Eylül), hex cinsinden Unix zamanı ve r({<probename>}, {<responselength>}, "<responsestring>") şeklinde bir dizi prob yanıdır.

Submit Database Corrections (Veritabanı Düzeltmelerini Gönderin)

Bu, veritabanını geliştirmeye yardımcı olmanın bir başka kolay yoludur. "Windows XP üzerinde chargen" veya "FooBar FTP sunucusu 3.9.213" için gönderilen bir hizmet parmak izini entegre ederken, eşleşmenin ne kadar genel olduğunu belirlemek zordur. Solaris üzerinde chargen veya FooBar FTP 2.7 ile de eşleşecek mi? Bunu söylemenin iyi bir yolu olmadığından, eşleşmenin genelleştirilmesi gerektiğiinde insanların bildireceği umuduyla çok özel bir isim kullanılır. Nmap DB'nin bu kadar kapsamlı olmasının tek nedeni, binlerce kullanıcının her birinin yeni bilgiler göndermek için birkaç dakika harcamış olmasıdır. Bir ana bilgisayarı tararsanız ve hizmet parmak izi yanlış bir işletim sistemi, sürüm numarası, uygulama adı ve hatta hizmet türü verirse, lütfen aşağıda açıkladığı gibi bize bildirin:

En son Nmap'e yükseltme (isteğe bağlı)

Birçok Linux dağıtımı ve diğer işletim sistemleri Nmap'in eski sürümleri ile birlikte gelmektedir. Nmap sürüm algılama veritabanı neredeyse her sürümde geliştirilir, bu nedenle nmap -V çalıştırarak sürüm numaranızı kontrol edin ve ardından bunu <https://nmap.org/download.html> adresindeki en son sürümle karşılaştırın. Göründüğünüz sorun çoktan düzeltilmiş olabilir. En yeni sürümü yüklemek çoğu platformda yalnızca birkaç dakika süre ve bildirdiğiniz sürüm algılama kusurunun hala var olup olmadığına bakılmaksızın değerlidir. Ancak şu anda yükseltme yapmak için zamanınız olmasa bile, eski sürümlerden yapılan gönderimler hala değerlidir.

Neyin çalıştığını bildiğinizden kesinlikle emin olun

Geçersiz "düzeltmeler" sürüm tespit DB'sini bozabilir. Uzak makinede tam olarak ne çalıştığını emin değilseniz, lütfen göndermeden önce öğrenin.

Parmak izi oluşturma

nmap -O -Pn -sSV -T4 -d --version-trace -p<port> <hedef> komutunu çalıştırın; burada <port>, <hedef> ana bilgisayarında yanlış tanımlanan hizmeti çalıştırın bağlantı noktasıdır. Eğer hizmet TCP yerine UDP ise, -sSV yerine -sUV yazın.

Düzeltmenizi bize gönderin

Şimdi düzeltmenizi <https://insecure.org/cgi-bin/submit.cgi?corr-service> adresinden bize göndermeniz yeterli. Nmap topluluğuna katkıda bulunduğunuz ve sürüm tespitini daha da iyi hale getirmeye yardımcı olduğunuz için teşekkür ederiz!

Submit New Probes (Yeni Problar Gönderin)

Nmap'in bir hizmeti tespit edemediğini varsayılmı. Herhangi bir sondaya yanıt aldıysa, yukarıda açıkladığı gibi gönderilebilecek bir parmak izi sağılmalıdır. Peki ya yanıt yoksa ve dolayısıyla bir parmak izi mevcut değilse? Kendi probunuza oluşturun ve gönderin! Bunlar çok hoş karşılanır. Aşağıdaki adımlar süreci açıklamaktadır.

Yeni bir sürüm algılama probu oluşturma adımları

1. Nmap'in en son sürümünü <https://nmap.org> adresinden indirin ve tekrar deneyin. Zaten eklenmiş olduğunu öğrenmek için yeni bir prob geliştirmek için zaman harcamak biraz aptalca olacaktır. Çok fazla yeni prob oluşturmaktansa mümkünse mevcut problemleri kullanarak hizmetleri tanımak daha iyi olacağından, hiçbir

parmak izinin mevcut olmadığından emin olun. Hizmet mevcut problkardan hiçbirine yanıt vermiyorsa, başka seçenek yoktur.

2. Hizmeti tanımak için iyi bir sonda dizesine karar verin. İdeal bir prob, mümkün olduğunda çok sayıda hizmet örneğinden yanıt almalıdır ve ideal olarak yanıtlar, aralarında ayırmak için yeterince benzersiz olmalıdır. Protokolü çok iyi anlıyorsanız bu adım en kolayıdır, bu nedenle ilgili RFC'leri ve ürün belgelerini okumayı düşünün. Basit bir yaklaşım, verilen hizmet için bir istemci başlatmak ve Wireshark veya tcpdump ile ağı koklayarak veya dinleyen bir Ncat'e bağlanarak ilk el sıkışmanın nasıl yapıldığını izlemektir.
3. Uygun dizeye karar verdikten sonra, Nmap'e uygun yeni Prob satırını ekleyin ("Açıklanan Teknik" adlı bölüm ve "nmap-service-probes Dosya Formatı" adlı bölümne bakın). İlk başta herhangi bir eşleşme satırı eklemeyin, ancak bu yeni testin ilk olarak kayıtlı bağlantı noktalarına karşı yapılmasını sağlayacak bir bağlantı noktaları yönergesi tamamdır. Ardından hizmeti Nmap ile birkaç kez tarayın. Hizmetin yeni probunuza verdiği yanıt gösteren bir parmak izi almalısınız. Yeni sonda satırını ve parmak izlerini (mükemmese farklı makinelere karşı, ancak aynı daemon'a karşı birkaç tane bile farklılıklar not etmeye yardımcı olur) dev@nmap.org adresindeki Nmap geliştirme listesine gönderin. Muhtemelen daha sonra Nmap'in gelecekteki sürümlerine entegre edilecektir. Prob dizenizin doğası hakkında sağlayabileceğiniz herhangi bir ayrıntı da yardımcı olacaktır. Yalnızca ağınızda görünen özel hizmetler için, bunları genel Nmap yerine kendi nmap-service-probes'ınıza eklemek daha iyidir.

SOLUTION: Find All Servers Running an Insecure or Nonstandard Application Version **(ÇÖZÜM: Güvensiz veya Standart Olmayan Uygulama Sürümünü Çalıştıran Tüm Sunucuları Bulun)**

Problem (Problem)

Yayın bir görev, belirli bir sürümün tüm sunucularını bulmak veya hatta belirli bir özelliği karşılamak için bir dizi IP adresini taramaktır. Bu, Nmap'in sürüm tespitinin üstün olduğu bir konudur.

En popüler veritabanı uygulamalarından biri açık kaynaklı MySQL sunucusudur. MySQL, güvenilmeyen IP'lerden gelen tüm uzaktan girişlere izin vermeyecek şekilde yapılandırılabilir. Bu, uzaktan girişlerin gerekli olmadığı durumlarda iyi bir güvenlik uygulamasıdır. Bir örnek: 2005 yılında MySQL uzaktan kod çalıştırma açığı keşfedildi ve yayınlandı. Neyse ki, bir saldırganın önce oturum açılabilmesi gerekiyordu ki bu da interneti bir başka yıkıcı solucandan kurtarıyordu. Bu gibi sorunlar ve SQL girişlerinin ve şifrelerinin SQL enjeksiyon saldırısı, sezgi ve ağını iç bilgisi yoluyla sıklıkla tahmin edilebilir veya keşfedilebilir olduğu gerçeği işliğinde, uzaktan girişler mümkün olduğunda reddedilmelidir.

Bir ağ yöneticisi için sorun, güvenilmeyen IP'lerden girişlere gereksiz yere izin veren MySQL sunucularını keşfetmek ve uygun savunma önlemlerini almaktır.

Not: Bu çözüme Nmap geliştiricisi Doug Hoyte katkıda bulunmuştur.

Solution (Çözüm)

Nmap'in sürüm tespiti bu durumda işe yarar, çünkü sunucu ana bilgisayarımıza herhangi bir erişimi yasaklılığında hizmet algılama bilgisi satırına yetkisiz kelimesini ekler. Eğer 10.0.0.0/24 ağını taramak istiyorsak basit ama etkili bir strateji aşağıdaki komutu güvenilmeyen bir kaynaktan çalıştırmaktadır:

```
nmap -sV -p 3306 -oG 10.0.0-mysqls-032506.gnmap 10.0.0.0/24
```

Daha sonra, IP'mizden bağlantıları kabul eden ve varsayılan olarak girişlere izin vermeyen IP'leri bulmak için Unix grep yardımcı programını kullanabiliriz (grep'in -v anahtarı ters sonuçları belirtir ve yalnızca verilen kalıpla eşleşmeyen satırları yazdırır):

```
grep 'Ports: 3306/open/tcp//mysql' 10.0.0-mysqls-032506.gnmap | grep -v unauthorized
```

Elde edilen çıktı, uzaktan oturum açmaya izin veren MySQL sunucularını gösterir:

```
Host: 10.0.0.33 (foo.com) Ports: 3306/open/tcp//mysql//MySQL 4.1.11/
Host: 10.0.0.72 (bar.com) Ports: 3306/open/tcp//mysql//MySQL 4.0.24-standard/
Host: 10.0.0.99 () Ports: 3306/open/tcp//mysql//MySQL 4.1.11-Debian_4sarge2/
Host: 10.0.0.154 () Ports: 3306/open/tcp//mysql//MySQL 4.0.25-standard/
Host: 10.0.0.155 () Ports: 3306/open/tcp//mysql//MySQL 4.0.25-standard/
```

Discussion (Tartışma)

Bunun püf noktası, bazı MySQL protokol temellerini anlamak ve nmap-service-probes dosyasının nasıl okunacağını bilmektir. Probe ve mysql eşleşme satırları için dosyayı taramak aşağıdaki (satırı sarılmış) çıktıya yol açar:

```
$ cat /usr/local/share/nmap/nmap-service-probes | egrep '^^(Probe|match mysql)'
Probe TCP NULL q||
match mysql m|^.\0\0\0\xffj\x04.*Host .* is not allowed to connect to this
          MySQL server$/ p/MySQL/ i/unauthorized/
match mysql m|^.\0\0\0\xffj\x04Host hat keine Berechtigung, eine Verbindung
          zu diesem MySQL Server herzustellen\.| p/MySQL/
          i/unauthorized; German/
match mysql m|^.\0\0\0...Al sistema '[..w]+' non e` consentita la
          connessione a questo server MySQL$/ p/MySQL/
          i/unauthorized; Italian/
match mysql m|^.\0\0\0\xffj\x04?Host .* is blocked because of many connection
          errors\| p/MySQL/ i/blocked - too many connection errors/
match mysql m|^.\0\0\0.(3\.[..w]+)\0.*\x08\x02\0\0\0\0\0\0\0\0\$|s
          p/MySQL/ v/$1/
match mysql m|^.\0\0\0\n(3\.[..w]+)\0...\\0/s p/MySQL/ v/$1/
match mysql m|^.\0\0\0\n(4\.[..w]+)\0...\\0/s p/MySQL/ v/$1/
match mysql m|^.\0\0\0\n(5\.[..w]+)\0...\\0|s p/MySQL/ v/$1/
match mysql m|^.\0\0\0\xffj\x04'[\d.]+'.* MySQL|s p/MySQL/
Probe TCP GenericLines q|r\n|r\n|
Probe TCP GetRequest q|GET / HTTP/1.0|r\n|r\n|
Probe TCP HTTPOptions q|OPTIONS / HTTP/1.0|r\n|r\n|
...
```

mysql eşleşme satırlarının NULL probu tarafından tetiklenecek şekilde tasarlandığını görüyoruz, bu nedenle hangi sunucuların uzaktan oturum açmaya izin verdiğine belirlemek için özel problemlere gerek yoktur (bunun için "ÇÖZÜM: Açık Proxy Algılama gibi Özel İhtiyaçlara Uygun Sürüm Algılamayı Hackleme" başlıklı bölümde bakın). Bu mysql eşleşme satırlarına bakarak, uzaktan oturum açmaya izin vermeyen MySQL hizmetlerinin yetkisiz kelimesini içeren bir bilgi alıyla sonuçlanacağını keşfeliyoruz.

Hizmet türleri ve sürüm numaralarına ek olarak, sürüm algılamanın tarama hedefleri hakkında yararlı bilgiler toplayabildiği birçok durum vardır. Prob dosyası, protokol araştırması, komut dosyası kodlama, test sunucularını bulma ve hata ayıklama gibi zaman alıcı bir görevi basit bir Nmap komutuna dönüştürebilecek bu tür mücevherlerle doludur. Sürüm tespitinin bazen ortaya çıkarabileceğinin birkaç ilginç bilgi vardır:

- SSH protokol sürümleri
- CVS pserver'in düzgün yapılandırılıp yapılandırılmadığı
- Popüler eşler arası dosya paylaşım istemcileri tarafından kullanılan kullanıcı adları
- Bir X sunucusunun bağlantıları kabul edip etmediği
- Bir çok hizmetin dil ve diğer yerleştirme parametreleri

- Hedefin CPU'sunun kelime boyutu
- eggdrop gibi popüler IRC botlarının yapılandırılmış bot adları
- İnternet haber (NNTP) sunucularında gönderime izin verilip verilmemiği

Nmap kullanıcı topluluğu ve onların hizmet parmak izi gönderimleri sayesinde sürüm algılama veritabanı sürekli olarak büyümekte ve geliştirilmektedir. Bu çözüm, Nmap'in hizmet algılama yeteneklerinin araştırılmasının birçok farklı soruna nasıl zarif, bazen de açık olmayan çözümler sağlayabileceğinin iyi bir örneğidir.

SOLUTION: Hack Version Detection to Suit Custom Needs, such as Open Proxy Detection (ÇÖZÜM: Sürüm Tespitini Açık Proxy Tespiti gibi Özel İhtiyaçlara Uyacak Şekilde Hackleyin)

Problem

Herhangi bir ağın güvenliğini sağlayanın önemli bir parçası, tehlikeli ana bilgisayarları tanımlamaktır. Nmap'in hizmet algılama sistemi bunu yapmanın esnek ve güvenilir bir yoludur. Yazılımın savunmasız sürümlerini belirlemeye, yanlış yapılandırılmış sunucuları bulmaya ve daha fazlasına yardımcı olabilir. Ancak bazen hizmetleri stok sürüm taramasının cesaret edemeyeceği şekilde kötüye kullanmaya çalışmak, gerçekten savunmasız olup olmadıklarını belirlemenin en iyi yoludur.

Açık proxy'ler, güvenilmeyen ana bilgisayarlardan gelen istekleri kendi seçikleri sunuculara körük körüğe aktaran sunuculardır. Saldırganların bunları bir ağ içinde çalıştırması birçok nedenden dolayı son derece tehlikeli olabilir:

- Ağınızdan geliyormuş gibi görünen saldırılar başlatmak
- Sizden bant genişliği veya diğer ağ hizmetlerini çalmak
- Kuruluşunuz içindeki ayrıcalıklarını daha da artırmak için dahili bir müşteri gibi davranışmak

Bu, özellikle açık proxy'lerden yararlanmaya çalışmak için sürüm algılamayı kesmek için iyi bir motivasyon sağlar. Nmap'in normal proxy eşleştirme satırlarını kullanarak hangi portların proxy olduğunu muhtemelen haritalayabiliriz, ancak bir uygulamanın savunmasız olduğunu kanıtlamanın en iyi ve tek gerçek yolu, onu gerçekten kendiniz istismar etmektir.

Not: Bu çözüme Nmap geliştiricisi Doug Hoyte katkıda bulunmuştur.

Solution (Çözüm)

Yapacağımız ilk şey nmap-service-probes dosyasını kopyalamaktır, böylece geçici bir kopya üzerinde çalışabilirez:

```
mkdir ~/proxydetect  
cp /usr/local/share/nmap/nmap-service-probes ~/proxydetect
```

Daha sonra Nmap'i geçici olarak geçici dosyamızı kullanmaya zorlamak istiyoruz:

```
export NMAPDIR=$HOME/proxydetect
```

Şimdi dosyaya bir prob ve eşleşme satırı eklememiz gerekiyor, bu yüzden favori editörünüza açın ve aşağıdaki metni nmap-service-probes kopyanızı yerleştirin. Bunu koymak için iyi bir yer, NULL probundaki tüm eşleşme satırlarından sonra, ancak bir sonraki Prob satırından (GenericLines) hemen öncedir.

```

Probe TCP ProxyProbe q|GET https://insecure.org/ HTTP/1.1\r\nHost: insecure.org\r\n\r\n
rarity 1
ports 1-65535
totalwaitms 20000
match proxy m|^HTTP/1.[01] 200 OK|r?n.*TITLE>Insecure.O|s p/Open HTTP Proxy!!/

```

Şimdi Nmap, taranan tüm portları proxy olarak değerlendirerek insecure.org'dan HTTP indirme talebinde bulunmaya çalışacaktır. Açık proxy içeren ağların taramalarında aşağıdakileri görmeye başlayacağız:

PORT	STATE	SERVICE	VERSION
80/tcp	open	proxy	Open HTTP Proxy!!

Discussion (Tartışma)

Probumuzun yerleşimi, düşük nadirlik değeri ve geniş port aralığı, özel probumuzun hizmet taramasında çok kısa sürede denenmesini sağlamaya yardımcı olur, böylece GetRequest gibi diğer problk, aktif probumuzu kullanma şansımız olmadan önce bunu bir proxy olarak tanımlamaz.

Ayrıca Nmap'in bu probun zaman aşımına uğraması için daha uzun süre beklemesini sağlamak için bir totalwaitms yönergesi kullandık. Bu gerekli olabilir çünkü sadece proxy ile aramızdaki bağlantının gecikmesi ve güvenilmezliği ile değil, aynı zamanda proxy ile talep ettiğimiz sayfayı içeren sunucu (insecure.org) arasındaki bağlantının gecikmesi ve güvenilmezliği ile de uğraşıyoruz.

HTTP'ye ek olarak birçok başka protokolün de proxy'lenebileceğini unutmayın. Sürüm tespiti, FTP, POP3, IMAP ve SMTP dahil olmak üzere birçoğu için proxy'leri tanımlayacaktır. SOCKS proxy'leri, proxy'nin yapılandırdığı kimlik doğrulama seçeneklerilarındaki bilgileri belirleyen özel eşleşme satırlarına sahiptir. Bu çözümde yaptığımız gibi, genellikle özel prob dosyalarını kullanarak bu tür proxy'lerin açık olup olmadığını anlamak için sürüm algılamayı kullanabiliyoruz. Ancak, daha karmaşık testler muhtemelen en iyi NSE betikleri ile yapılır.

Chapter 8. Remote OS Detection (Bölüm 8. Uzak İşletim Sistemi Algılama)

- Introduction (Giriş)
 - Reasons for OS Detection (İşletim Sistemi Algılama Nedenleri)
 - Determining vulnerability of target hosts (Hedef ana bilgisayarların güvenlik açığını belirleme)
 - Tailoring exploits (Terzilik açıkları)
 - Network inventory and support (Ağ envanteri ve desteği)
 - Detecting unauthorized and dangerous devices (Yetkisiz ve tehlikeli cihazların tespit edilmesi)
 - Social engineering (Sosyal mühendislik)
- Usage and Examples (Kullanım ve Örnekler)
- TCP/IP Fingerprinting Methods Supported by Nmap (Nmap Tarafından Desteklenen TCP/IP Parmak izi Yöntemleri)
 - Probes Sent (Probes Sent)
 - Sequence generation (SEQ , OPS , WIN , and T1) (Sekans oluşturma (SEQ, OPS, WIN ve T1))

- ICMP echo (`IE`) (ICMP yankısı (IE))
- TCP explicit congestion notification (`ECN`) (TCP açık tıkanıklık bildirimi (ECN))
- TCP (`T2` – `T7`)
- UDP (`U1`)
- Response Tests (Yanıt Testleri)
 - TCP ISN greatest common divisor (`GCD`) (TCP ISN en büyük ortak bölen (GCD))
 - TCP ISN counter rate (`ISR`) (TCP ISN sayaç oranı (ISR))
 - TCP ISN sequence predictability index (`SP`) (TCP ISN sıra tahmin edilebilirlik endeksi (SP))
 - IP ID sequence generation algorithm (`TI` , `CI` , `II`) (IP ID dizisi oluşturma algoritması (TI, CI, II))
 - Shared IP ID sequence Boolean (`SS`) (Paylaşılan IP Kimliği sırası Boolean (SS))
 - TCP timestamp option algorithm (`TS`) (TCP zaman damgası seçeneği algoritması (TS))
 - TCP options (`O` , `O1-O6`) (TCP seçenekleri (O, O1-O6))
 - TCP initial window size (`W` , `W1` – `W6`) (TCP başlangıç pencere boyutu (W, W1-W6))
 - Responsiveness (`R`) (Duyarlılık (R))
 - IP don't fragment bit (`DF`) (IP parçalama biti (DF))
 - Don't fragment (ICMP) (`DFI`) (Parçalama (ICMP) (DFI))
 - IP initial time-to-live (`T`) (IP ilk yaşam süresi (T))
 - IP initial time-to-live guess (`TG`) (IP ilk zaman-canlı tahmin (TG))
 - Explicit congestion notification (`CC`) (Açık tıkanıklık bildirimi (CC))
 - TCP miscellaneous quirks (`Q`) (TCP çeşitli tuhaflıklar (Q))
 - TCP sequence number (`S`) (TCP sıra numarası (S))
 - TCP acknowledgment number (`A`) (TCP onay numarası (A))
 - TCP flags (`F`) (TCP bayrakları (F))
 - TCP RST data checksum (`RD`) (TCP RST veri sağlama toplamı (RD))
 - IP total length (`IPL`) (IP toplam uzunluğu (IPL))
 - Unused port unreachable field nonzero (`UN`) (Kullanılmayan bağlantı noktası ulaşılamaz alanı sıfır olmayan (UN))
 - Returned probe IP total length value (`RIPL`) (İade edilen prob IP toplam uzunluk değeri (RIPL))
 - Returned probe IP ID value (`RID`) (İade edilen prob IP kimlik değeri (RID))
 - Integrity of returned probe IP checksum value (`RIPCK`) (İade edilen prob IP sağlama toplamı değerinin bütünlüğü (RIPCK))
 - Integrity of returned probe UDP checksum (`RUCK`) (İade edilen prob UDP sağlama toplamının bütünlüğü (RUCK))
 - Integrity of returned UDP data (`RUD`) (İade edilen UDP verilerinin bütünlüğü (RUD))
 - ICMP response code (`CD`) (ICMP yanıt kodu (CD))

- IPv6 fingerprinting (IPv6 parmak izi)
 - Probes Sent (Problar Gönderimi)
 - Sequence generation (`S1 – S6`) (Dizi oluşturma (S1-S6))
 - ICMPv6 echo (`IE1`)
 - ICMPv6 echo (`IE2`)
 - Node Information Query (`NI`) (Dügüm Bilgisi Sorusu (NI))
 - Neighbor Solicitation (`NS`) (Komşu İsteği (NS))
 - UDP (`U1`)
 - TCP explicit congestion notification (`TECN`) (TCP açık tıkanıklık bildirimi (TECN))
 - TCP (`T2 – T7`)
 - Feature extraction (Özellik çıkarma)
 - List of all features (Tüm özelliklerin listesi)
 - Differences from IPv4 (IPv4'ten Farklılıklar)
- Fingerprinting Methods Avoided by Nmap (Nmap Tarafından Kaçınılan Parmak İzi Yöntemleri)
 - Passive Fingerprinting (Pasif Parmak İzi)
 - Exploit Chronology (İstismar Kronolojisi)
 - Retransmission Times (Yeniden İletim Süreleri)
 - IP Fragmentation (IP Parçalanması)
 - Open Port Patterns (Açık Bağlantı Noktası Kalıpları)
 - Retired Tests (Emekli Testler)
- Understanding an Nmap Fingerprint (Nmap Parmak İzini Anlama)
 - Decoding the Subject Fingerprint Format (Denek Parmak İzi Formatının Şifresini Çözme)
 - Decoding the `SCAN` line of a subject fingerprint (Bir deneğin parmak izinin TARAMA satırının kodunun çözülmesi)
 - Decoding the Reference Fingerprint Format (Referans Parmak İzi Formatının Kodunu Çözme)
 - Free-form OS description (`Fingerprint` line) (Serbest biçimli işletim sistemi açıklaması (Parmak izi satırı))
 - Device and OS classification (`Class` lines) (Cihaz ve işletim sistemi sınıflandırması (Sınıf çizgileri))
 - CPE name (`CPE` lines) (CPE adı (CPE hatları))
 - Test expressions (Test ifadeleri)
 - IPv6 fingerprints (IPv6 parmak izleri)
- Device Types (Cihaz Türleri)
- OS Matching Algorithms (İşletim Sistemi Eşleştirme Algoritmaları)
 - IPv4 matching (IPv4 eşleştirme)
 - IPv6 matching (IPv6 eşleştirme)

- Dealing with Misidentified and Unidentified Hosts (Yanlış Tanımlanmış ve Tanımlanmamış Ev Sahipleri ile Başa Çıkma)
 - When Nmap Guesses Wrong (Nmap Yanlış Tahmin Yaptığında)
 - When Nmap Fails to Find a Match and Prints a Fingerprint (Nmap Bir Eşleşme Bulamadığında ve Parmak İzi Çıkardığında)
 - Modifying the `nmap-os-db` Database Yourself (nmap-os-db Veritabanını Kendiniz Değiştirme)
- SOLUTION: Detect Rogue Wireless Access Points on an Enterprise Network (ÇÖZÜM: Kurumsal Ağdaki Sahte Kablosuz Erişim Noktalarını Tespit Etme)
 - Problem (Sorun)
 - Solution (Çözüm)
 - WAP Characteristics (WAP Özellikleri)

Introduction (Giriş)

Güvenlik denetimi veya envanter/yönetim için bir ağı araştırırken, genellikle tanımlanan makinelerin çiplak IP adreslerinden daha fazlasını bilmek istersiniz. Bir yazılıcı keşfetmeye verdığınız tepki, bir yönlendirici, kablosuz erişim noktası, telefon PBX'i, oyun konsolu, Windows masaüstü veya Unix sunucusu bulmaya verdığınız tepkiden çok farklı olabilir. Daha ince taneli tespit (Mac OS X 10.4'ü 10.3'ten ayırt etmek gibi), belirli kusurlara karşı güvenlik açığını belirlemek ve bu güvenlik açıkları için etkili istismarları uyarlamak için kullanışlıdır.

Kısmen saldırganlar için değeri nedeniyle, birçok sistem tam doğası ve işletim sistemi yapılandırması hakkında sıkı sıkıya gizlidir. Neyse ki Nmap, çeşitli TCP/IP problemlerine nasıl yanıt verdiklerine bağlı olarak binlerce farklı sistemi tanımlamak için büyük bir sezgisel veri tabanı içerir. Başka bir sistem (sürüm tespitinin bir parçası) cihaz türünü ve işletim sistemi ayrıntılarını belirlemek için açık TCP veya UDP bağlantı noktalarını sorgular. Bu iki sistemin sonuçları bağımsız olarak raporlanır, böylece 80 numaralı bağlantı noktasını bir Windows IIS sunucusuna yönlendiren bir Checkpoint güvenlik duvarı gibi kombinasyonları tanımlayabilirsiniz.

Nmap 1998 yılından beri işletim sistemi tespitini desteklese de, bu bölümde 2006 yılında piyasaya sürülen 2. nesil sistem anlatılmaktadır.

Reasons for OS Detection (İşletim Sistemi Algılama Nedenleri)

Bir ağdaki temel işletim sistemi ve cihaz türlerini keşfetmenin bazı faydalari açık olsa da, diğerleri daha belirsizdir. Bu bölümde, bu ekstra bilgileri keşfetmek için duyduğum en önemli nedenleri listeliyorum.

Determining vulnerability of target hosts (Hedef ana bilgisayarların güvenlik açığını belirleme)

Bazen mevcut bir hizmetin belirli bir güvenlik açığına karşı duyarlı olup olmadığını veya yamanmış olup olmadığını uzaktan belirlemek çok zordur. Uygulama sürüm numarasını almak bile her zaman yardımcı olmaz, çünkü işletim sistemi dağıticıları genellikle sürüm numarasını değiştirmeden güvenlik düzeltmelerini geri taşırlar. Bir güvenlik açığının gerçek olduğunu doğrulamanın en kesin yolu onu istismar etmektir, ancak bu hizmeti çökertme riski taşırlar ve hizmetin yamalı olduğu ortaya çıkarsa boş harcanan saatlere, hatta günlerce süren sinir bozucu istismar çabalarına yol açabilir.

İşletim sistemi tespiti bu yanlış pozitifleri azaltmaya yardımcı olabilir. Örneğin, yamalanmamış Sun Solaris 7 ile 9 üzerindeki Rwho arka plan programı uzaktan istismar edilebilir (Sun uyarısı #57659). Güvenlik açığını uzaktan belirlemek zordur, ancak bir hedef sistemin Solaris 10 çalıştığını tespit ederek bunu ekarte edebilirsiniz.

Bunu bir pen-tester yerine bir sistem yöneticisinin bakış açısından ele alırsak, #57659 uyarısı çıktığında büyük bir Sun mağazası işlettiğinizi düşünün. Kötü adamlardan önce yamalanması gereken makineleri bulmak için tüm ağınızı işletim sistemi algılama ile tarayın.

Tailoring exploits (Terzilik açıkları)

Bir hedef sistemde bir güvenlik açığı keşfettikten sonra bile, işletim sistemi tespiti bu açığın istismar edilmesinde yardımcı olabilir. Arabellek taşmaları, biçim dizesi istismarları ve diğer birçok güvenlik açığı genellikle hedef işletim sistemi ve donanım mimarisine uyacak şekilde oluşturulmuş ofsetler ve montaj yükleri ile özel olarak uyarlanmış kabuk kodu gerektirir. Bazı durumlarda, yalnızca bir deneme hakkınız vardır çünkü kabuk kodunu yanlış girerseniz hizmet çöker. Önce işletim sistemi algılamayı kullanın, aksi takdirde bir FreeBSD sunucusuna Linux kabuk kodu gönderebilirsiniz.

Network inventory and support (Ağ envanteri ve desteği)

Özel olarak hazırlanmış bir format dizesi istismarı yoluyla root'u kırmak kadar heyecan verici olmasa da, ağınızda neyin çalıştığını takip etmek için birçok idari neden vardır. IRIX destek sözleşmesini bir yıl daha yenilemeden önce, bu tür makineleri hala kullanan biri olup olmadığını görmek için tarama yapın. Bir envanter, BT bütçelemesi ve tüm şirket ekipmanlarının hesaba katılmasını sağlamak için de yararlı olabilir.

Detecting unauthorized and dangerous devices (Yetkisiz ve tehlikeli cihazların tespit edilmesi)

Mobil cihazların ve ucuz emtia ağ ekipmanlarının yaygınlaşmasıyla birlikte şirketler, çalışanların ağlarını istenmeyen şekillerde genişlettiklerini giderek daha fazla fark ediyor. Çalışanlar, korunan şirket ağını otoparktaki ya da yakındaki binalardaki potansiyel saldırganlara açlıklarını fark etmeden (ya da umursamadan) çalışma odalarına 20 dolarlık bir kablosuz erişim noktası (WAP) kurabiliyorlar. WAP'lar o kadar tehlikeli olabilir ki "ÇÖZÜM: Kurumsal Ağdaki Sahte Kablosuz Erişim Noktalarını Tespit Etme" başlıklı bölümde gösterildiği gibi Nmap'in bunları tespit etmek için özel bir kategorisi vardır. Kullanıcılar ayrıca güvensiz ve/veya solucan bulaşmış dizüstü bilgisayarları şirket ağına bağlayarak sistem yöneticilerini sıkıntıya sokabiliyorlar. Düzenli tarama, inceleme ve kontrol altına alma için yetkisiz cihazları tespit edebilir.

Social engineering (Sosyal Mühendislik)

Bir başka olası kullanım da sosyal mühendisliktir. Diyelim ki bir hedef şirketi tariyorsunuz ve Nmap "Datavoice TxPORT PRISM 3000 T1 CSU/DSU 6.22/2.06" rapor ediyor. Datavoice desteği gibi davranışarak hedefi arayabilir ve PRISM 3000 ile ilgili bazı sorunları tartışabilirisiniz. Onlara büyük bir güvenlik açığını duyurmak üzere olduğunuzu, ancak önce değerli müşterilere yama sağlayacağınızı söyleyin. Bazı saf yöneticiler, yalnızca Datavoice'tan yetkili bir mühendisin CSU/DSU'ları hakkında bu kadar çok şey bilebileceğini varsayıyorlar. Elbette onlara gönderdiğiniz yama, ağlarını koklamak ve izlemek için size uzaktan erişim sağlayan bir Truva atıdır. Bunu denemeden önce tespit doğruluğu ve doğrulama tavsiyeleri için bu bölümün geri kalanını okuduğunuzdan emin olun. Hedef sistemi yanlış tahmin ederseniz ve polis çağrırlarsa, bu hücre arkadaşlarınıza anlatacağınız utanç verici bir hikaye olacaktır.

Usage and Examples (Kullanım ve Örnekler)

İşletim sistemi algılamanın iç işleyişi oldukça karmaşıktır, ancak kullanımı en kolay özelliklerden biridir. Tarama seçeneklerinize -O eklemeniz yeterlidir. İşletim sistemiyle ilgili daha fazla ayrıntı için -v ile ayrıntı düzeyini de artırmak isteyebilirisiniz. Bu Örnek 8.1'de gösterilmiştir.

Örnek 8.1. Verbosity (-O -v) ile işletim sistemi algılama

```

# nmap -O -v scanme.nmap.org
Starting Nmap ( https://nmap.org )
Nmap scan report for scanme.nmap.org (74.207.244.221)
Not shown: 994 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
80/tcp    open     http
646/tcp   filtered ldp
1720/tcp  filtered H.323/Q.931
9929/tcp  open     nping-echo
31337/tcp open     Elite
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.39
OS details: Linux 2.6.39
Uptime guess: 1.674 days (since Fri Sep  9 12:03:04 2011)
Network Distance: 10 hops
TCP Sequence Prediction: Difficulty=205 (Good luck!)
IP ID Sequence Generation: All zeros

Read data files from: /usr/local/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 5.58 seconds
Raw packets sent: 1063 (47.432KB) | Rcvd: 1031 (41.664KB)

```

O -v seçeneklerinin eklenmesi Nmap'in aşağıdaki ekstra satır öğelerini oluşturmasına neden oldu:

Cihaz tipi ⇒ Tüm parmak izleri, yönlendirici, yazıcı, güvenlik duvarı veya (bu durumda olduğu gibi) genel amaçlı gibi bir veya daha fazla üst düzey cihaz türüyle sınıflandırılır. Bunlar "Cihaz ve işletim sistemi sınıflandırması (Sınıf çizgileri)" adlı bölümde daha ayrıntılı olarak açıklanmaktadır. Birkaç cihaz türü gösterilebilir, bu durumda "Cihaz Türü: yönlendirici|güvenlik duvarı" örneğinde olduğu gibi boru sembolüyle ayrılırlar.

Koşmak ⇒ Bu alan aynı zamanda "Cihaz ve İşletim Sistemi sınıflandırması (Sınıf çizgileri)" adlı bölümde açıklanan İşletim Sistemi sınıflandırma şemasıyla da ilgilidir. İşletim sistemi ailesini (bu durumda Linux) ve varsa işletim sistemi neslini (2.6.X) gösterir. Birden fazla işletim sistemi ailesi varsa, bunlar virgülle ayrılır. Nmap işletim sistemi nesillerini belirli bir seçeneğe indirgeyemediğinde, seçenekler boru sembolü ('|') ile ayrılır. Örnekler arasında OpenBSD 3.X, NetBSD 3.X|4.X ve Linux 2.4.X|2.5.X|2.6.X bulunur.

Eğer Nmap kısa bir şekilde yazdırırmak için çok fazla işletim sistemi ailesi bulursa, bu satırı atlayacaktır. Mükemmel eşleşme olmadığından, Nmap alanı Çalışıyor (SADECE TAHMİN) olarak değiştirir ve her aday aile adından sonra parantez içinde bir doğruluk yüzdesi (%100 mükemmel eşleşmedir) ekler. Hiçbir parmak izi yakın eşleşme değilse, satır atlanır.

OS CPE ⇒ Bu, mevcut olduğunda işletim sisteminin Ortak Platform Numaralandırma (CPE) temsilini gösterir. Ayrıca donanım türünün CPE gösterimine de sahip olabilir. İşletim sistemi CPE'si cpe:/o ile başlar ve donanım CPE'si cpe:/h ile başlar. CPE hakkında daha fazla bilgi için "Ortak Platform Numaralandırma (CPE)" başlıklı bölüme bakın.

İşletim sistemi ayrıntıları ⇒ Bu satır eşleşen her parmak izi için ayrıntılı açıklama verir. Aygit türü ve Çalışıyor satırları bir bilgisayar tarafından ayrıştırılması kolay olan önceden tanımlanmış numaralandırılmış listelerden gelirken, işletim sistemi ayrıntıları satır raporu okuyan bir insan için yararlı olan serbest biçimli veriler içerir. Bu, belirli bir parmak izine özgü daha kesin sürüm numaralarını, cihaz modellerini ve mimarileri içerebilir. Bu örnekte, eşleşen tek parmak izi Linux 2.6.20-1 (Fedora Core 5) idi. Birden fazla tam eşleşme olduğunda, bunlar virgülle ayrılır. Herhangi bir mükemmel eşleşme yoksa, ancak bazı yakın tahminler varsa, alan Agresif OS tahminleri olarak yeniden adlandırılır ve parmak izlerinin ardından parantez içinde her eşleşmenin ne kadar yakın olduğunu belirten bir yüzde gösterilir.

Çalışma süresi tahmini ⇒ İşletim sistemi tespitinin bir parçası olarak, Nmap arkayaarkaya birkaç SYN/ACK TCP paketi alır ve bir zaman damgası seçeneği için başlıklarını kontrol eder. Birçok işletim sistemi bunun için önyükleme sırasında sıfırdan başlayan ve ardından saniyede iki kez gibi sabit bir oranda artan basit bir sayaç kullanır. Nmap birkaç yanıt bakarak mevcut değerleri ve artış oranını belirleyebilir. Basit doğrusal ekstrapolasyon önyükleme süresini belirler. Zaman damgası algoritması işletim sistemi tespiti için de kullanılır

("TCP zaman damgası seçeneği algoritması (TS)" adlı bölüme bakın) çünkü farklı sistemlerdeki artış oranı 2 Hz ile 1.000 Hz arasında değişir.

Çalışma zamanı tahmini bir "tahmin" olarak etiketlenmiştir çünkü çeşitli faktörler bunu tamamen yanlış yapabilir. Bazı işletim sistemleri zaman damgası sayacını sıfırdan başlatmaz, ancak rastgele bir değerle başlatır, bu da sıfıra ekstrapolasyonu anlamsız hale getirir. Sıfırdan başlayan basit bir sayaç kullanan sistemlerde bile, sayaç sonunda taşar ve etrafını sarar. 1.000 Hz sayaç artış hızıyla, sayaç kabaca her 50 günde bir sıfırlanır. Dolayısıyla 102 gündür açık olan bir ana bilgisayar sadece iki gündür açık görünecektir. Bu uyarılara rağmen, çalışma süresi tahmini çoğu işletim sistemi için çoğu zaman doğrudur, bu nedenle mevcut olduğunda yazdırılır, ancak yalnızca ayrıntılı modda. Eğer hedef SYN/ACK paketlerinde sıfır ya da hiç zaman damgası seçeneği vermezse ya da hiç cevap vermezse uptime tahmini atlanır. Nmap zaman damgası artış oranını ayırt edemiyorsa veya şüpheli görünüyorsa (30 yıllık bir çalışma süresi gibi) satır da atlanır.

Ağ Mesafesi ⇒ İşletim sistemi tespit testlerinden birinin yan etkisi, Nmap'in hedef ana bilgisayar ile arasında kaç yönlendirici olduğunu hesaplamasına olanak tanır. Localhost'u tararken mesafe sıfırdır ve aynı ağ segmentindeki bir makine için birdir. Yol üzerindeki her ilave yönlendirici atlama sayısına bir ekler. Bu örnekte Ağ Mesafesi satırı yazdırılmamıştır, çünkü Nmap hesaplanmadığında (ilgili proba yanıt verilmediğinde) satırı atlar.

TCP Sıra Tahmini ⇒ TCP başlangıç sıra numarası üretimi zayıf olan sistemler kör TCP sahtekarlık saldırılara karşı savunmasızdır. Başka bir deyişle, bu sistemlere tam bağlantı kurabilir ve farklı bir IP adresini taklit ederek veri gönderebilirsiniz (ancak alamazsınız). Hedefin günlükleri sahte IP'yi gösterecektir ve aralarındaki herhangi bir güven ilişkisinden yararlanabilirisiniz. Bu saldırı, insanların güvenilir IP adreslerinden herhangi bir şifre olmadan hesaplarına giriş yapmak için rlogin'i yaygın olarak kullandığı doksanlı yılların ortalarında çok popülerdi. Kevin Mitnick'in Aralık 1994'te Tsutomu Shimomura'nın bilgisayarlarına girmek için bu saldırıyı kullandığı iddia edilmektedir.

İyi haber şu ki, artık neredeyse hiç kimse rlogin kullanmıyor ve birçok işletim sistemi RFC 1948 tarafından önerildiği gibi öngörülemeyen başlangıç sıra numaralarını kullanacak şekilde düzeltildi. Bu nedenlerden dolayı, bu satır yalnızca ayrıntılı modda yazdırılır. Ne yazık ki, birçok satıcı hala savunmasız işletim sistemleri ve cihazlar göndermektedir. Sabit olanlar bile genellikle uygulamada farklılık gösterir, bu da onları işletim sistemi algılama amaçları için değerli kılar. Sınıf, hedef tarafından kullanılan ISN oluşturma algoritmasını tanımlar ve zorluk, sistemin kör IP sahteciliğini ne kadar zorlaştırdığına dair kabaca bir tahmidir (0 en kolay olanıdır). Parantez içindeki yorum zorluk endeksine dayanır ve Önemsiz şakadan Kolay, Orta, Zorlu, Meydan okumaya değer ve son olarak İyi şanslar! Sıra testleri hakkında daha fazla ayrıntı "TCP ISN en büyük ortak bölen (GCD)" adlı bölümde verilmiştir.

rlogin ailesi çoğunlukla geçmişin bir kalıntısı olsa da, zeki saldırganlar kör TCP sahtekarlığı için hala etkili kullanımlar bulabilirler. Örneğin, sahte HTTP isteklerine izin verir. Sonuçları görmezsiniz, ancak sadece URL'nin (POST veya GET isteği) dramatik yan etkileri olabilir. Sahtecilik, saldırganların kimliklerini gizlemelerine, başka birini suçlamalarına veya IP adresi kısıtlamalarından yararlanmalarına olanak tanır.

IP Kimliği dizisi oluşturma ⇒ Birçok sistem farklı olmadan IP paketlerindeki 16 bitlik düşük ID alanını nasıl oluşturduklarına bağlı olarak trafik seviyeleri hakkında hassas bilgiler verir. Bu, diğer sistemlere karşı bir port taramasını taklit etmek ve "TCP Boş Tarama (-sl)" adlı bölümde tartışılan diğer yaramaz amaçlar için kötüye kullanılabilir. Bu alan Nmap'in ayırt edebildiği kimlik oluşturma algoritmasını tanımlar. Bunları nasıl sınıflandırdığı hakkında daha fazla bilgi "IP ID dizisi oluşturma algoritması (TI, CI, II)" adlı bölümde mevcuttur. Birçok sistemin iletişim kurduğu her ana bilgisayar için farklı bir IP ID alanı kullandığını unutmayan. Bu durumda, boşta tarama gibi saldırılara karşı güvenli olmalarına rağmen savunmasız görünebilirler (Artımlı sınıfını göstermek gibi). Bu nedenle ve sorun nadiren kritik olduğu için, IP Kimliği sırası oluşturma satırı yalnızca ayrıntılı modda yazdırılır. Nmap OS tespiti sırasında yeterli yanıt almazsa, tüm satırı atlayacaktır. Bir ana bilgisayarın boşta tarama zombisi olmaya karşı savunmasız olup olmadığını test etmenin en iyi yolu -sl ile test etmektir.

TCP parmak izi OS tespiti için güçlü bir yöntem olsa da, ipuçları için açık portları sorgulamak da bir başka etkili yaklaşımındır. Microsoft IIS gibi bazı uygulamalar yalnızca tek bir platformda çalışır (buyle onu ele verir), diğer birçok uygulama ise platformlarını aşırı ayrıntılı banner mesajlarında ifşa eder. sV seçeneğinin eklenmesi, bu ipuçlarını (diğerlerinin yanı sıra) aramak üzere eğitilmiş olan Nmap sürüm algılamasını etkinleştirir. Örnek 8.2'de, Nmap bir FTP sunucusundan platform ayrıntılarını yakalamaktadır.

Örnek 8.2. İşletim sistemini tespit etmek için sürüm taramasını kullanma

```
# nmap -sV -O -v 129.128.X.XX
Starting Nmap ( https://nmap.org )
Nmap scan report for [hostname] (129.128.X.XX)
Not shown: 994 closed ports
PORT      STATE    SERVICE      VERSION
21/tcp    open     ftp          HP-UX 10.x ftptd 4.1
22/tcp    open     ssh          OpenSSH 3.7.1p1 (protocol 1.99)
111/tcp   open     rpc          SunRPC
445/tcp   filtered microsoft-ds
1526/tcp  open     oracle-tns  Oracle TNS Listener
32775/tcp open     rpc          SunRPC
No exact OS matches for host
TCP Sequence Prediction: Class=truly random
                          Difficulty=9999999 (Good luck!)
IP ID Sequence Generation: Incremental
Service Info: OS: HP-UX
```

Bu örnekte, "No exact OS matches for host" satırı TCP/IP parmak izinin tam bir eşleşme bulamadığı anlamına gelir. Neyse ki, birkaç satır aşağıdaki Servis Bilgisi alanı işletim sisteminin HP-UX olduğunu açıklamaktadır. Eğer birden fazla işletim sistemi tespit edilmişse (bu, portları birkaç farklı makineye yönlendiren NAT ağ geçidi kutularında olabilir), alan işletim sistemleri olur ve değerler virgülle ayrılır. Hizmet Bilgisi satırı, sürüm taraması sırasında bulunan ana bilgisayar adlarını ve cihaz türlerini de içerebilir. Ancak bu bölümün odak noktası TCP/IP parmak izidir, çünkü sürüm tespiti Bölüm 7, Hizmet ve Uygulama Sürüm Tespiti'nde ele alınmıştır.

İki etkili işletim sistemi tespit yöntemi mevcutken hangisini kullanmalısınız? En iyi cevap genellikle her ikisidir. Başka bir ana bilgisayardaki bir uygulamaya yönlendirme yapan bir proxy güvenlik duvarı gibi bazı durumlarda, yanıtlar meşru olarak farklı olabilir. TCP/IP parmak izi proxy'yi tanımlarken, sürüm taraması genellikle proxy'li uygulamayı çalıştırın sunucuya tespit edecektir. Proxy veya port yönlendirme söz konusu olmadığından bile, her iki teknigin de kullanılması faydalıdır. Eğer sonuçlar aynı çıkarsa, bu sonuçları daha güvenilir kılar. Eğer çok farklı sonuçlar çıkarsa, ikisinden birine güvenmeden önce neler olup bittiğini belirlemek için daha fazla araştırma yapın. İşletim sistemi ve sürüm tespiti birlikte çok iyi gittiğinden, -A seçeneği her ikisini de etkinleştirir.

En az bir açık ve bir kapalı TCP portu bulunursa işletim sistemi tespiti çok daha etkili olur. osscan-limit seçeneğini ayarlayın ve Nmap bu kriteri karşılamayan ana bilgisayarlara karşı işletim sistemi algılamayı bile denemeyecektir. Bu, özellikle birçok ana bilgisayara karşı -Pn taramalarında önemli ölçüde zaman kazandırabilir. Yine de --osscan-limit seçeneğinin herhangi bir etkisi olması için işletim sistemi algılamayı -O (veya -A) ile etkinleştirmeniz gereklidir.

Başka bir işletim sistemi algılama seçeneği --osscan-guess'dir. Nmap mükemmel bir işletim sistemi eşleşmesi tespit edemediğinde, bazen yakın eşleşmeleri olasılık olarak sunar. Nmap'in varsayılan olarak bunu yapması için eşleşmenin çok yakın olması gereklidir. Bu seçeneği (veya eşdeğer --fuzzy seçeneğini) belirtirseniz, Nmap daha agresif bir şekilde tahmin edecektir. Nmap hala kusurlu bir eşleşme bulunduğuunda size söyle ve her tahmin için güven seviyesini (yüzde) görüntüler.

Nmap bir hedefe karşı işletim sistemi tespiti gerçekleştirdiğinde ve mükemmel bir eşleşme bulamadığında, genellikle denemeyi tekrarlar. Varsayılan olarak, Nmap koşullar işletim sistemi parmak izi gönderimi için uygunsa beş kez, koşullar o kadar iyi değilse iki kez dener. max-os-tries seçeneği bu maksimum işletim sistemi algılama deneme sayısını değiştirmenizi sağlar. Bunu düşürmek (genellikle 1'e) Nmap'i hızlandırır, ancak potansiyel olarak işletim sistemi tanımlayabilecek yeniden denemeleri kaçırırsınız. Alternatif olarak, koşullar

uygun olduğunda daha fazla yeniden denemeye izin vermek için yüksek bir değer ayarlanabilir. Bu, Nmap OS veritabanına gönderme ve entegrasyon için daha iyi parmak izleri oluşturmak dışında nadiren yapılır.

Nmap'in hemen hemen diğer tüm bölmelerinde olduğu gibi, sonuçlar nihayetinde hedef makinenin kendisinden gelir. Nadiren de olsa, sistemler zaman zaman Nmap'i şarşırtmak ya da yanılmak için yapılandırılır. Nmap işletim sistemi tespiti kandırmak için özel olarak birkaç program bile geliştirilmiştir ("İşletim Sistemi Sahteciliği" bölümüne bakın). Yapabileceğiniz en iyi şey, bir ağı keşfetmek için çok sayıda keşif yöntemi kullanmak ve bunlardan hiçbirine güvenmemektir.

TCP/IP parmak izi, hedefin IP yiğini hakkında ayrıntılı bilgi toplamayı gerektirir. TTL bilgisi gibi en sık kullanılan sonuçlar, elde edildiklerinde Nmap çıktısına yazdırılır. IP ID dizisi oluşturma ve TCP dizisi tahmin zorluğu gibi biraz daha az ilgili bilgiler yalnızca ayrıntılı modda yazdırılır. Ancak Nmap'in topladığı tüm IP yiğini ayrıntılarını istiyorsanız, bunları konu parmak izi adı verilen kompakt bir biçimde bulabilirsiniz. Nmap bazen bir ana bilgisayarını tanımadığında bunu yazdırır (kullanıcı gönderme amaçları için). Ayrıca (-d) ile hata ayıklamayı etkinleştirerek Nmap'i bunu yazdırmaya zorlayabilirsiniz (normal, etkileşimli ve XML biçimlerinde). Daha sonra bunu yorumlamak için "Nmap Parmak İzini Anlamak" adlı bölüm okuyun.

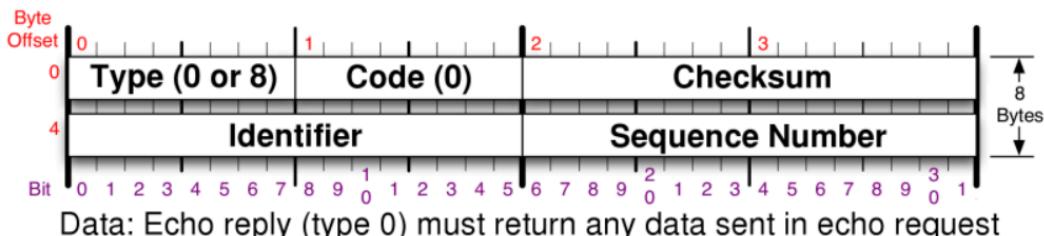
TCP/IP Fingerprinting Methods Supported by Nmap (Nmap Tarafından Desteklenen TCP/IP Parmak İzi Yöntemleri)

Nmap OS parmak izi, hedef makinenin bilinen açık ve kapalı portlarına 16 adede kadar TCP, UDP ve ICMP probu göndererek çalışır. Bu probalar, standart protokol RFC'lerindeki çeşitli belirsizliklerden yararlanmak için özel olarak tasarlanmıştır. Daha sonra Nmap yanıtları dinler. Bu yanıldakilerin düzinelere özellik analiz edilir ve bir parmak izi oluşturmak için birleştirilir. Her sonda paketi izlenir ve yanıt yoksa en az bir kez yeniden gönderilir. Tüm paketler rastgele bir IP ID değerine sahip IPv4'tür. Açık bir TCP portuna yapılan probalar, böyle bir port bulunamazsa atlanır. Kapalı TCP veya UDP portları için, Nmap önce böyle bir portun bulunup bulunmadığını kontrol edecektir. Eğer yoksa, Nmap sadece rastgele bir port seçecektir ve en iyisini umacaktır.

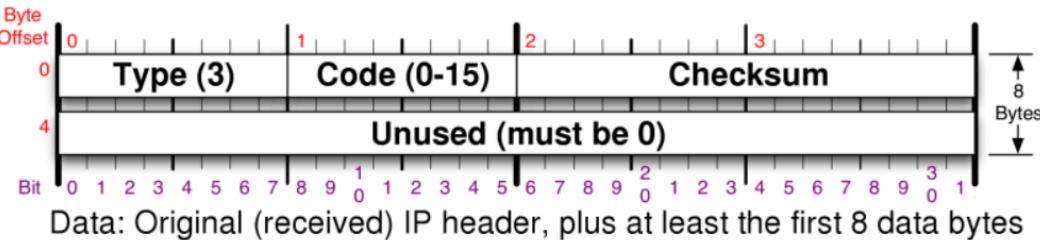
Aşağıdaki bölümler oldukça tekniktir ve Nmap OS tespitinin gizli işleyişini ortaya koymaktadır. Nmap bunu anlamadan da etkili bir şekilde kullanılabilir, ancak bu materyal uzak ağları daha iyi anlamana ve belirli anomalilikleri tespit edip açıklamanıza yardımcı olabilir. Ayrıca, bazı teknikler oldukça havalıdır. Acelesi olan okuyucular "Yanlış Tanımlanmış ve Tanımlanmamış Ana Bilgisayarlarla Başa Çıkma" adlı bölüm atlayabilirler. Ancak TCP açık tikanıklık bildirimi, ayrılmış UDP başlık bitleri, başlangıç sıra numaraları, sahte bayraklar ve Noel ağacı paketleri ile bir yolculuğa hazır olanlar için: okumaya devam edin!

En iyilerimiz bile zaman zaman paket başlık alanları ve bayrakları için bayt uzaklıklarını unutur. Hızlı başvuru için, IPv4, TCP, UDP ve ICMP başlık düzenleri "TCP/IP Referansı" adlı bölümde bulunabilir. ICMP yanıt isteği ve hedefe ulaşılamıyor paketleri için düzel Şekil 8.1 ve Şekil 8.2'de gösterilmektedir.

Şekil 8.1. ICMP yanıt isteği veya yanıtı başlık düzeni



Şekil 8.2. ICMP hedefe ulaşılamıyor başlık düzeni



Probes Sent

Bu bölümde, TCP/IP parmak izinin bir parçası olarak Nmap tarafından gönderilen her IP probu açıklanmaktadır. Aşağıdaki bölümde açıklanan Nmap yanıt testlerine ve TCP seçeneklerine atıfta bulunur.

Sequence generation (SEQ , OPS , WIN , and T1) (Sekans oluşturma (SEQ, OPS, WIN ve T1))

Bu dört test yanıt satırını oluşturmak için altı TCP probu serisi gönderilir. Problar tam olarak 100 milisaniye arayla gönderilir, böylece toplam süre 500 ms olur. Tespit ettiğimiz bazı sıra algoritmaları (ilk sıra numaraları, IP ID'leri ve TCP zaman damgaları) zamana bağlı olduğundan tam zamanlama önemlidir. Bu zamanlama değeri, yaygın 2 Hz TCP zaman damgası dizilerini güvenilir bir şekilde tespit edebilmemiz için 500 ms olarak seçilmiştir.

Her prob, uzak makinede tespit edilen bir açık porta giden bir TCP SYN paketidir. Sıra ve onay numaraları rastgeledir (ancak Nmap'in yanıtları ayırt edebilmesi için kaydedilir). Tespit doğruluğu prob tutarlılığı gerektirir, bu nedenle kullanıcı --data-length ile bir tane talep etse bile veri yükü yoktur.

Bu paketler kullandıkları TCP seçenekleri ve TCP pencere alanı değeri açısından farklılık gösterir. Aşağıdaki listede altı paketin tümü için seçenekler ve değerler verilmiştir. Listelenen pencere alanı değerleri pencere ölçeklendirmesini yansitmaz. EOL, birçok koklama aracının varsayılan olarak göstermediği seçenekler listesinin sonu seçeneğidir.

- Paket #1: pencere ölçüği (10), NOP, MSS (1460), zaman damgası (TSval: 0xFFFFFFF; TSecr: 0), SACK izinli. Pencere alanı 1'dir.
- Paket #2: MSS (1400), pencere ölçüği (0), SACK izinli, zaman damgası (TSval: 0xFFFFFFF; TSecr: 0), EOL. Pencere alanı 63'tür.
- Paket #3: Zaman damgası (TSval: 0xFFFFFFF; TSecr: 0), NOP, NOP, pencere ölçüği (5), NOP, MSS (640). Pencere alanı 4'tür.
- Paket #4: SACK izinli, Zaman Damgası (TSval: 0xFFFFFFF; TSecr: 0), pencere ölçüği (10), EOL. Pencere alanı 4'tür.
- Paket #5: MSS (536), SACK izinli, Zaman Damgası (TSval: 0xFFFFFFF; TSecr: 0), pencere ölçüği (10), EOL. Pencere alanı 16'dır.
- Paket #6: MSS (265), SACK izinli, Zaman Damgası (TSval: 0xFFFFFFF; TSecr: 0). Pencere alanı 512'dir.

Bu testlerin sonuçları dört sonuç kategorisi satırı içerir. Bunlardan ilki olan SEQ, sonda paketlerinin sıra analizine dayalı sonuçları içerir. Bu test sonuçları GCD, SP, ISR, TI, II, TS ve SS'dir. Bir sonraki satır olan OPS, her bir prob için alınan TCP seçeneklerini içerir (test adları O1'den 06'ya kadardır). Benzer şekilde, WIN satırı prob yanıtları için pencere boyutlarını içerir (W1'den W6'ya kadar adlandırılmıştır). Bu problemlerle ilgili son satır olan T1, paket #1 için çeşitli test değerlerini içerir. Bu sonuçlar R, DF, T, TG, W, S, A, F, O, RD ve Q testleri içindir. Bu testler sadece ilk prob için raporlanmıştır çünkü her prob için neredeyse her zaman aynıdır.

ICMP echo (IE)

IE testi, hedefe iki ICMP yanıt isteği paketi göndermeyi içerir. İlkinde IP DF biti ayarlanmış, hizmet türü (TOS) bayt değeri sıfır, kod dokuz (sıfır olması gereği halde), sıra numarası 295, rastgele bir IP Kimliği ve ICMP istek

tanımlayıcısı ve veri yükü için 120 bayt 0x00 vardır.

İkinci ping sorusu benzerdir, ancak dört TOS (IP_TOS_RELIABILITY) kullanılır, kod sıfırdır, 150 bayt veri gönderilir ve ICMP istek kimliği ve sıra numaraları önceki sorgu değerlerinden bir artırılır.

Bu problemlerin her ikisinin sonuçları R, DFI, T, TG ve CD testlerini içeren bir IE satırında birleştirilir. R değeri yalnızca her iki probun da yanıt vermesi durumunda doğrudur (Y). T ve CD değerleri yalnızca ilk probun yanıtı içindir, çünkü farklı olmaları pek olası değildir. DFI, bu özel çift probu ICMP durumu için özel bir testtir.

Bu ICMP problemleri, paylaşılan IP Kimliği sıra numarası testinin geçerli sonuçlarını sağlamak için TCP sıra problemlerinden hemen sonra gelir ("Paylaşılan IP Kimliği sıra Boolean (SS)" adlı bölüme bakın).

TCP explicit congestion notification (ECN) (TCP açık tıkanıklık bildirimi (ECN))

Bu sonda, hedef TCP yiğinında açık tıkanıklık bildirimi (ECN) desteğini test eder. ECN, yönlendiricilerin paketleri düşürmek zorunda kalmadan önce tıkanıklık sorunlarını bildirmelerine olanak tanıyarak İnternet performansını artırmaya yönelik bir yöntemdir. RFC 3168'de belgelenmiştir. Nmap bunu ECN CWR ve ECE tıkanıklık kontrol bayraklarının da ayarlandığı bir SYN paketi göndererek test eder. İlgisiz (ECN ile) bir test için, acil bayrağı ayarlanmamış olsa bile 0xF7F5 acil alan değeri kullanılır. Onay numarası sıfırdır, sıra numarası rastgelemdir, pencere boyutu alanı üçtür ve CWR bitinden hemen önce gelen ayrılmış bit ayarlanmıştır. TCP seçenekleri WScale (10), NOP, MSS (1460), SACK izinli, NOP, NOP'dur. Prob açık bir porta gönderilir.

Bir yanıt alınırsa, R, DF, T, TG, W, O, CC ve Q testleri gerçekleştirilecek ve kaydedilir.

TCP (T₂ – T₇)

T2'den T7'ye kadar olan altı testin her biri bir TCP prob paketi gönderir. Bir istisna dışında, her durumda TCP seçenekleri verileri (onaltılık olarak) 03030A0102040109080AFFFFFF000000000402'dir. Bu 20 bayt, pencere ölçüği (10), NOP, MSS (265), Zaman Damgası (TSval: 0xFFFFFFF; TSecr: 0) ve ardından izin verilen SACK'e karşılık gelir. Bunun istisnası, T7'nin 10 yerine 15'lik bir Pencere ölçüği değeri kullanmasıdır. Her bir probun değişken özellikleri aşağıda açıklanmıştır:

- T2, IP DF biti ayarlanmış ve pencere alanı 128 olan bir TCP null (bayrak ayarı yok) paketini açık bir bağlantı noktasına gönderir.
- T3, SYN, FIN, URG ve PSH bayrakları ayarlanmış ve pencere alanı 256 olan bir TCP paketini açık bir bağlantı noktasına gönderir. IP DF biti ayarlanmamıştır.
- T4, açık bir bağlantı noktasına IP DF ve 1024 pencere alanı ile bir TCP ACK paketi gönderir.
- T5, kapalı bir bağlantı noktasına IP DF'siz ve pencere alanı 31337 olan bir TCP SYN paketi gönderir.
- T6, kapalı bir bağlantı noktasına IP DF ve 32768 pencere alanı ile bir TCP ACK paketi gönderir.
- T7, FIN, PSH ve URG bayrakları ayarlanmış ve pencere alanı 65535 olan bir TCP paketini kapalı bir bağlantı noktasına gönderir. IP DF biti ayarlanmamıştır.

Bu durumların her birinde, parmak izine R, DF, T, TG, W, S, A, F, O, RD ve Q testlerinin sonuçlarını içeren bir satır eklenir.

UDP (u₁)

Bu prob, kapalı bir porta gönderilen bir UDP paketidir. 'C' (0x43) karakteri veri alanı için 300 kez tekrarlanır. IP ID değeri, bunu ayıramamıza izin veren işletim sistemleri için 0x1042 olarak ayarlanır. Eğer port gerçekten kapalıysa ve herhangi bir güvenlik duvarı yoksa, Nmap karşılığında bir ICMP port ulaşılamaz mesajı almayı bekler. Bu yanıt daha sonra R, DF, T, TG, IPL, UN, RIPL, RID, RIPCK, RUCK ve RUD testlerine tabi tutulur.

Response Tests (Yanıt Testleri)

Bir önceki bölümde Nmap tarafından gönderilen probalar açıklanmıştır, bu bölümde ise yanıtlar üzerinde yapılan testler açıklanarak bulmaca tamamlanmaktadır. Kısa isimler (DF, R ve RIPCK gibi) nmap-os-db parmak izi veritabanında yer kazanmak için kullanılan isimlerdir. Tüm sayısal test değerleri, aksi belirtilmekçe, baştaki sıfırlar olmadan onaltılık gösterimde verilmiştir. Testler kabaca parmak izlerinde göründükleri sırayla belgelenmiştir.

TCP ISN greatest common divisor (GCD) (TCP ISN en büyük ortak bölen (GCD))

SEQ testi, hedef makinenin açık bir portuna altı TCP SYN paketi gönderir ve SYN/ACK paketlerini geri toplar. Bu SYN/ACK paketlerinin her biri 32 bitlik bir başlangıç sıra numarası (ISN) içerir. Bu test, hedef ana bilgisayarın bu değerleri artırdığı en küçük sayıyı belirlemeye çalışır. Örneğin, birçok ana bilgisayar (özellikle eski olanlar) ISN'yi her zaman 64.000'in katları olarak artırır.

Bunu hesaplamanın ilk adımı, sonda yanıtları arasında bir farklar dizisi oluşturmaktır. İlk eleman 1. ve 2. sonda yanıtı ISN'leri arasındaki farktır. İkinci eleman 2. ve 3. yanıtlar arasındaki farktır. Nmap altı probun hepsine yanıt alırsa beş eleman vardır. Sonraki birkaç bölüm bu diziye atıfta bulunduğuundan, bu diziyi diff1 olarak adlandıracağız. Bir ISN bir öncekinden daha düşükse, Nmap hem ikinci değeri elde etmek için ilk değerden çıkarması gereken değerlerin sayısına hem de sayması gereken değerlerin sayısına bakar (32 bitlik sayacı sıfıra geri sarmak dahil). Bu iki değerden küçük olanı diff1'de saklanır. 0x20000 ile 0x15000 arasındaki fark 0xB000'dir. 0xFFFFF00 ile 0xC000 arasındaki fark 0xC0FF'dir. Bu test değeri daha sonra tüm bu elemanların en büyük ortak bölenini kaydeder. Bu GCD aynı zamanda SP sonucunu hesaplamak için de kullanılır.

TCP ISN counter rate (ISR) (TCP ISN sayaç oranı (ISR))

Bu değer, döndürülen TCP başlangıç sıra numarası için ortalama artış oranını bildirir. Her iki ardışık prob yanıtı arasında bir fark alındığını ve daha önce tartışılan diff1 dizisinde saklandığını hatırlayın. Bu farkların her biri, onları oluşturan iki probun gönderilmesi arasında geçen süreye (saniye cinsinden - genellikle yaklaşık 0,1 olacaktır) bölünür. Sonuç, saniye başına ISN sayacı artış oranlarını içeren seq_rates adını vereceğimiz bir dizidir. Dizinin her diff1 değeri için bir elemanı vardır. Dizi değerlerinin ortalaması alınır. Bu ortalama birden küçükse (örneğin sabit bir ISN kullanılıyorsa), ISR sıfırdır. Aksi takdirde ISR, bu ortalama değerin ikili logaritmasının (log tabanı-2) sekiz katıdır ve en yakın tam sayıya yuvarlanır.

TCP ISN sequence predictability index (SP) (TCP ISN sıra tahmin edilebilirlik endeksi (SP))

ISR testi ilk sıra numarası artışlarının ortalama oranını ölçerken, bu değer ISN değişkenliğini ölçer. Kabaca, bilinen altı prob yanıtı dizisinden bir sonraki ISN'yi tahmin etmenin ne kadar zor olacağını tahmin eder. Hesaplama, önceki bölümde tartışılan fark dizisini (seq_rates) ve GCD değerlerini kullanır.

Bu test yalnızca en az dört yanıt görüldüğünde gerçekleştirilir. Daha önce hesaplanan GCD değeri dokuzdan büyükse, daha önce hesaplanan seq_rates dizisinin elemanları bu değere bölünür. Daha küçük GCD değerleri için bölme işlemi yapmayız çünkü bunlar genellikle şansa bağlıdır. Daha sonra ortaya çıkan değerler dizisinin standart sapması alınır. Sonuç bir veya daha az ise SP sıfırdır. Aksi takdirde, sonucun ikili logaritması hesaplanır, ardından sekiz ile çarpılır, en yakın tam sayıya yuvarlanır ve SP olarak saklanır.

Lütfen bu testin yalnızca işletim sistemi tespiti amacıyla yapıldığını ve hedef ISN üreticisinin tam kapsamlı bir denetimi olmadığını unutmayın. Yüksek bir SP değeriyle bile kolay tahmin edilebilirliğe yol açan birçok algoritma zayıflığı vardır.

IP ID sequence generation algorithm (TI , CI , II) (IP ID dizisi oluşturma algoritması (TI, CI, II))

Yanıtların IP başlık kimliği alanını inceleyen üç test vardır. TI, TCP SEQ problarına verilen yanıtlarla dayanır. CI, kapalı bir porta gönderilen üç TCP probuna verilen yanıldan elde edilir: T5, T6 ve T7. II, iki IE ping probuna verilen ICMP yanıldan gelir. TI için, testin dahil edilmesi için en az üç yanıt alınmalıdır; CI için en az iki yanıt gereklidir; ve II için her iki ICMP yanıtı da alınmalıdır.

Bu testlerin her biri için, hedefin IP kimliği oluşturma algoritması aşağıdaki algoritmaya göre sınıflandırılır. Testler arasındaki küçük farklılıklar belirtilmiştir. Fark değerlerinin sayacın sarılabileceğini varsayıdığını unutmayın. Dolayısıyla, 65.100 IP kimliği ile 700 değeri arasındaki fark 1.136'dır. 2.000 ile 1.100 arasındaki fark ise 64.636'dır. İşte hesaplama detayları:

1. Tüm kimlik numaraları sıfır ise, testin değeri Z'dir.
2. IP Kimliği dizisi en az 20.000 artarsa, değer RD (rastgele) olur. Bu sonuç II için mümkün değildir çünkü bunu destekleyecek yeterli örnek yoktur.
3. Tüm IP ID'leri aynıysa, test hex cinsinden bu değere ayarlanır.
4. İki ardışık ID arasındaki farklardan herhangi biri 1.000'i aşyorsa ve 256'ya eşit olarak bölünemiyorsa, testin değeri RI (rastgele pozitif artışlar) olur. Fark 256'ya eşit olarak bölünebiliyorsa, bu RI sonucuna neden olmak için en az 256.000 olmalıdır.
5. Tüm farklar 256'ya bölünebiliyorsa ve 5.120'den büyük değilse, test BI (kırık artış) olarak ayarlanır. Bu, IP kimliğinin ağ bayt sırası yerine ana bilgisayar bayt sırasına göre gönderildiği Microsoft Windows gibi sistemlerde olur. Sorunsuz çalışır ve herhangi bir RFC ihlali değildir, ancak saldırganlar için yararlı olabilecek ana bilgisayar mimarisi ayrıntılarını verir.
6. Tüm farklar ondan küçükse, değer I (artımlı) olur. Diğer ana bilgisayarlardan gelen trafik sıra boşluklarına neden olabileceğiinden, burada ona kadar farka izin veriyoruz (sıralı sıralama gerektirmek yerine).
7. Önceki adımlardan hiçbirü üretim algoritmasını tanımlamazsa, test parmak izinden çıkarılır.

Shared IP ID sequence Boolean (ss) (Paylaşılan IP Kimliği sırası Boolean (SS))

Bu Boolean değeri, hedefin IP ID dizisini TCP ve ICMP protokolleri arasında paylaşıp paylaşmadığını kaydeder. Altı TCP IP ID değerimiz 117, 118, 119, 120, 121 ve 122 ise ve ICMP sonuçlarımız 123 ve 124 ise, her iki dizinin de artımlı olmakla kalmayıp aynı dizinin parçası olduğu açıktır. Öte yandan, TCP IP ID değerleri 117-122 ise ancak ICMP değerleri 32,917 ve 32,918 ise, iki farklı dizi kullanılıyor demektir.

Bu test yalnızca II RI, BI veya I ise ve TI aynı ise dahil edilir. SS dahil edilirse, sekans paylaşılıyorsa sonuç S, paylaşılmıyorsa O (diğer) olur. Bu belirleme aşağıdaki algoritma ile yapılır:

avg, son TCP sıra yanıtı IP kimliği eksi ilk TCP sıra yanıtı IP kimliği olsun ve prob numaralarındaki farka bölünsün. 1 numaralı sonda 10.000 IP kimliği döndürürse ve 6 numaralı sonda 20.000 döndürürse, avg $(20.000 - 10.000) / (6 - 1)$ olacaktır, bu da 2.000'e eşittir.

İlk ICMP yanıt yanıtı IP Kimliği, son TCP sıra yanıtı IP Kimliği artı ortalamanın üç katından küçükse, SS sonucu S olur.

TCP timestamp option algorithm (ts) (TCP zaman damgası seçeneği algoritması (TS))

TS, bir dizi sayıyı nasıl ürettiğine bağlı olarak hedef işletim sistemi özelliklerini belirlemeye çalışan başka bir testtir. Bu test, SEQ probleme verilen yanılardaki TCP zaman damgası seçeneğine (varsayı) bakar. Yankılanan TSect (son dört bayt) değeri yerine TSval (seçeneğin ilk dört baytı) değerini inceler. Her ardışık TSval arasındaki farkı alır ve bunu Nmap'in bu yanıtları oluşturan iki probu göndermesi arasında geçen süreye böler. Ortaya çıkan değer, saniye başına zaman damgası artış oranını verir. Nmap, ardışık tüm problemler üzerinden saniye başına ortalama artışları hesaplar ve ardından TS'yi aşağıdaki gibi hesaplar:

1. Yanıtlardan herhangi birinin zaman damgası seçeneği yoksa, TS U (desteklenmiyor) olarak ayarlanır.
2. Zaman damgası değerlerinden herhangi biri sıfırsa, TS 0 olarak ayarlanır.
3. Saniye başına ortalama artışlar 0-5.66, 70-150 veya 150-350 aralıklarına denk geliyorsa, TS sırasıyla 1, 7 veya 8 olarak ayarlanır. Bu üç aralık, birçok ana bilgisayar tarafından kullanılan 2 Hz, 100 Hz ve 200 Hz frekanslarına karşılık geldiği için özel muamele görür.

4. Diğer tüm durumlarda, Nmap saniye başına ortalama artışların ikili logaritmasını en yakın tam sayıya yuvarlayarak kaydeder. Çoğu ana bilgisayar 1.000 Hz frekans kullandığından, A yaygın bir sonuçtır.

TCP options (**O** , **O1-O6**) (TCP seçenekleri (O, O1-O6))

Bu test, bir paketteki TCP başlık seçeneklerini kaydeder. Orijinal sıralamayı korur ve ayrıca seçenek değerleri hakkında bazı bilgiler sağlar. RFC 793 belirli bir sıralama gerektirmeden, uygulamalar genellikle benzersiz sıralamalarla ortaya çıkar. Bazı platformlar tüm seçenekleri uygulamamaktadır (elbette bunlar istege bağlıdır). Tüm bu permutasyonları, uygulamaların kullandığı farklı seçenek değerlerinin sayısı ile birleştirildiğinde, bu test gerçek bir bilgi hazinesi sağlar. Bu testin değeri, kullanılan seçenekleri temsil eden bir karakter dizisidir. Birkaç seçenek, karakterden hemen sonra gelen bağımsız değişkenleri alır. Desteklenen seçenekler ve bağımsız değişkenlerin tümü Tablo 8.1'de gösterilmektedir.

Tablo 8.1. O test değerleri

Option Name (Seçenek Adı)	Character (Karakter)	Argument (if any) (Argüman (varsız))
Seçenekler Listesinin Sonu (EOL)	L	
İşlem yok (NOP)	N	
Maksimum Segment Boyutu (MSS)	M	Değer eklenir. Birçok sistem ilgili probda kullanılan değeri yankılar.
Pencere Ölçeği (WS)	W	Gerçek değer eklenir.
Zaman Damgası (TS)	T	T'nin ardından sırasıyla TSval ve TSecr değerlerini temsil eden iki ikili karakter gelir. Alan sıfır ise karakterler 0, aksi takdirde 1'dir.
Seçici ACK'ya izin verilir (SACK)	S	

Örnek olarak, M5B4NW3NNT11 dizesi, paketin MSS seçeneğini (0x5B4 değeri) ve ardından bir NOP içерdiği anlamına gelir. Ardından üç değerine sahip bir pencere ölçüği seçeneği ve ardından iki NOP daha gelir. Son seçenek bir zaman damgasıdır ve iki alanı da sıfır değildir. Bir yanıtta TCP seçeneği yoksa, test var olur ancak değer dizesi boş olur. Hiçbir prob döndürülmediyse, test atlanır.

Bu test genellikle O olarak adlandırılırken, sıra oluşturma amacıyla gönderilen altı prob özel bir durumdur. Bunlar özel OPS test hattına eklenir ve hangi sonda paketiyle ilgili olduklarını ayırt etmek için O1'den O6'ya kadar olan isimleri alırlar. "O", "seçenekler" anlamına gelir. Farklı isimlere rağmen, O1'den O6'ya kadar olan her test diğer O testleriyle tamamen aynı şekilde işlenir.

TCP initial window size (**W** , **W1 – W6**) (TCP initial window size (**W** , **W1 – W6**))

Bu test basitçe alınan paketin 16 bitlik TCP pencere boyutunu kaydedeler. Oldukça etkilidir, çünkü en az bir işletim sisteminin gönderdiği bilinen 80'den fazla değer vardır. Bir dezavantajı ise bazı işletim sistemlerinin kendi başlarına bir düzineden fazla olası değere sahip olmasıdır. Bu, bir işletim sistemi tarafından kullanılan tüm olası pencere boyutlarını toplayana kadar yanlış negatif sonuçlara yol açar.

Bu test genel olarak W olarak adlandırılsa da, dizi oluşturma amacıyla gönderilen altı prob özel bir durumdur. Bunlar özel bir WIN test satırına eklenir ve W1'den W6'ya kadar olan isimleri alır. Bazı işletim sistemlerinin farklı bir pencere boyutu bildirmesine neden olan TCP MSS seçenek değerlerinde farklılık gösterdiklerinden, pencere boyutu tüm sıra numarası problemleri için kaydedilir. Farklı isimlere rağmen, her test tamamen aynı şekilde işlenir.

Responsiveness (**R**) (Duyarlılık (R))

Bu test basitçe hedefin belirli bir proba yanıt verip vermediğini kaydedeler. Olası değerler Y ve N'dir. Yanıt yoksa, test için kalan alanlar atlanır.

Bu testle ilgili bir risk, bir güvenlik duvarı tarafından düşürülen problemleri içerir. Bu, konu parmak izinde R=N'ye yol açar. Ancak nmap-os-db'deki referans parmak izi, hedef işletim sistemi genellikle yanıt veriyorsa R=Y

olabilir. Böylece güvenlik duvarı doğru işletim sistemi tespitini engelleyebilir. Bu sorunu azaltmak için referans parmak izleri genellikle düşürülme olasılığı en yüksek olan IE ve U1 problemlerinden R=Y testini çıkarır. Buna ek olarak, Nmap bir hedef için kapalı bir TCP portu eksikse, denediği port yanıt vermiyor olsa bile T5, T6 veya T7 testleri için R=N ayarlamayacaktır. Sonuçta, kapalı bir bağlantı noktasının olmaması, hepsinin filtrelenmiş olmasından kaynaklanıyor olabilir.

IP don't fragment bit (DF) (IP parçalama biti (DF))

IP başlığı, yönlendiricilerin bir paketi parçalamasını yasaklayan tek bir bit içerir. Paket yönlendiricilerin işleyemeyeceği kadar büyüğse, paketi bırakmak zorunda kalırlar (ve ideal olarak "hedefe ulaşamıyor, parçalama gerekli" yanıtını döndürürler). Bu test, bit ayarlanmışsa Y, ayarlanmamışsa N kaydedeler.

Don't fragment (ICMP) (DFI) (Parçalama (ICMP) (DFI))

Bu, özel IE problemleri için kullanılan DF testinin basitçe değiştirilmiş bir sürümürdür. Gönderilen iki ICMP yanıt isteği probu için parçalama bitinin sonuçlarını karşılaştırır. Tablo 8.2'de sıralanan dört olası değeri vardır.

Tablo 8.2. DFI test değerleri

Value	Description (Açıklama)
N	Ping yanıtlarının hiçbirinde DF biti ayarlanmamıştır.
S	Her iki yanıt da probun DF değerini yansıtır.
Y	Yanıt DF bitlerinin her ikisi de ayarlanır.
O	Geriye kalan diğer bir kombinasyon - her iki yanıtta da DF biti değiştirilmiştir.

IP initial time-to-live (T) (IP ilk yaşam süresi (T))

IP paketleri, bir yönlendirciden her geçiklerinde azaltılan time-to-live (TTL) adlı bir alan içerir. Eğer bu alan sıfıra ulaşrsa, paketin atılması gereklidir. Bu, paketlerin sonsuza kadar döngüye girmesini önler. İşletim sistemleri hangi TTL ile başladıkları konusunda farklılık gösterdiğinde, işletim sistemi tespiti için kullanılabilir. Nmap, U1 probuna gelen ICMP port ulaşamıyor yanıtını inceleyerek hedeften kaç atlama uzakta olduğunu belirler. Bu yanıt, hedef tarafından alınan, zaten azaltılmış TTL alanı da dahil olmak üzere orijinal IP paketini içerir. Bu değeri gönderdiğimiz TTL'den çıkararak makinenin kaç atlama uzakta olduğunu öğreniriz. Nmap daha sonra bu ICMP prob yanıt paketi gönderildiğinde ilk TTL'nin ne olduğunu belirlemek için bu atlama mesafesini prob yanıt TTL'sine ekler. Bu ilk TTL değeri parmak izinde T sonucu olarak saklanır.

TTL gibi sekiz bitlik bir alan asla 0xFF'den büyük değerler tutamasa da, bu test bazen 0x100 veya daha yüksek değerlerle sonuçlanır. Bu, bir sistem (kaynak, hedef veya aradaki bir sistem olabilir) TTL'yi bozduğunda veya başka bir şekilde doğru şekilde azaltmadığında meydana gelir. Asimetrik rotalar nedeniyle de meydana gelebilir.

Nmap, atlama mesafesi sıfır (localhost taraması) veya bir (aynı ağ segmentinde) olduğunda sistem arayüzünden ve yönlendirme tablolarından da öğrenebilir. Bu değer, Nmap kullanıcı için atlama mesafesini yazdırıldığından kullanılır, ancak T sonuç hesaplaması için kullanılmaz.

IP initial time-to-live guess (TG) (IP ilk yaşam süresi tahmini (TG))

Nmap'in U1 probuna yanıt alamaması nadir değildir, bu da Nmap'in bir hedefin kaç atlama uzakta olduğunu öğrenmesini engeller. Güvenlik duvarları ve NAT cihazları istenmeyen UDP paketlerini engellemeyi sever. Ancak ortak TTL değerleri birbirinden oldukça uzak olduğundan ve hedefler nadiren 20 atlamanadan daha uzak olduğundan, Nmap yine de oldukça iyi bir tahmin yapabilir. Çoğu sistem 32, 60, 64, 128 veya 255 başlangıç TTL değerine sahip paketler gönderir. Dolayısıyla yanıtta alınan TTL değeri 32, 64, 128 veya 255 arasından bir sonraki değere yuvarlanır. 60 bu listede yer almaz çünkü 64'ten güvenilir bir şekilde ayırt edilemez. Zaten nadiren görülür. Ortaya çıkan tahmin TG alanında saklanır. Bu TTL tahmin alanı, gerçek TTL (T) değeri keşfedilmişse bir konu parmak izine yazdırılmaz.

Explicit congestion notification ([cc](#)) (Açık tıkanıklık bildirimi (CC))

Bu test yalnızca ECN probu için kullanılır. Bu prob, CWR ve ECE tıkanıklık kontrol bayraklarını içeren bir SYN paketidir. Yanıt SYN/ACK alındığında, bu bayraklar Tablo 8.3'te açıklandığı gibi CC (tıkanıklık kontrolü) test değerini ayarlamak için incelenir.

Tablo 8.3. CC test değerleri

Value	Description (Açıklama)
Y	Yalnızca ECE biti ayarlanmıştır (CWR değil). Bu ana bilgisayar ECN'yi destekler.
N	Bu iki bitten hiçbirini ayarlanmamıştır. Hedef ECN'yi desteklemiyor.
S	Her iki bit de ayarlanır. Hedef ECN'yi desteklemez, ancak ayrılmış bir bit olduğunu düşündüğü biti geri gönderir.
O	Bu iki bitin kalan tek kombinasyonu (diğer).

TCP miscellaneous quirks ([Q](#)) (TCP çeşitli tuhaflıklar (Q))

Bu, birkaç uygulamanın TCP yığınında sahip olduğu iki tuhaflığı test eder. Birincisi, TCP başlığındaki ayrılmış alanın (başlık uzunluğundan hemen sonra) sıfır olmamasıdır. Bunun özellikle ECN testine yanıt olarak gerçekleşmesi muhtemeldir, çünkü bu test sondada ayrılmış bir biti ayarlar. Bu bir pakette görülürse, Q dizesine bir "R" kaydedilir.

Nmap'in test ettiği diğer bir tuhaflık, URG bayrağı ayarlanmadığında sıfır olmayan bir acil işaretçi alanı değeridir. Bu, özellikle sıfır olmayan bir acil alan ayarlayan ECN probuna yanıt olarak görülmesi muhtemeldir. Bu görüldüğünde Q dizesine bir "U" eklenir.

Q dizesi her zaman alfabetik sırada oluşturulmalıdır. Hiçbir tuhaflık yoksa, Q testi boştur ancak yine de gösterilir.

TCP sequence number ([S](#)) (TCP sıra numarası (S))

Bu test TCP başlığındaki 32 bitlik sıra numarası alanını inceler. Diğer bazı testlerin yaptığı gibi alan değerini kaydetmek yerine, bu test, yanıtı ortaya çıkarılan probdan gelen TCP onay numarasıyla nasıl karşılaştırıldığını inceler. Daha sonra Tablo 8.4'te gösterildiği gibi uygun değeri kaydeden.

Tablo 8.4. S test değerleri

Value	Description (Açıklama)
Z	Sıra numarası sıfırdır.
A	Sıra numarası, sondadaki onay numarasıyla aynıdır.
A+	Sıra numarası, sondadaki onay numarası artı bir ile aynıdır.
O	Sıra numarası başka bir şeydir (diğer).

TCP acknowledgment number ([A](#)) (TCP onay numarası (A))

Bu test S ile aynıdır, ancak yanındaki onay numarasının ilgili probdaki sıra numarasıyla nasıl karşılaştırıldığını test eder. Dört olası değer Tablo 8.5'te verilmiştir.

Tablo 8.5. A test değerleri

Value	Description (Açıklama)
Z	Onay numarası sıfırdır.
S	Onay numarası, sondadaki sıra numarasıyla aynıdır.
S+	Onay numarası, sondadaki sıra numarası artı bir ile aynıdır.
O	Onay numarası başka bir şeydir (diğer).

TCP flags (F) (TCP bayrakları (F))

Bu alan yanittaki TCP bayraklarını kaydeder. Her harf bir bayrağı temsil eder ve TCP paketinde olduğu gibi aynı sırayla oluşurlar (soldaki yüksek bitten düşük olanlara doğru). Yani AS değeri ACK ve SYN bitlerinin ayarlandığını gösterirken SA değeri geçersizdir (yanlış sıra). Olası bayraklar Tablo 8.6'da gösterilmektedir.

Tablo 8.6. F testi değerleri

Character (Karakter)	Flag name (Bayrak adı)	Flag byte value (Bayrak bayt değeri)
E	ECN Echo (ECE)	64
U	Urgent Data (URG)	32
A	Acknowledgment (ACK)	16
P	Push (PSH)	8
R	Reset (RST)	4
S	Synchronize (SYN)	2
F	Final (FIN)	1

TCP RST data checksum (RD) (TCP RST veri sağlama toplamı (RD))

Bazı işletim sistemleri sıfırlama paketlerinde hata mesajları gibi ASCII verilerini döndürür. Buna RFC 1122'nin 4.2.2.12 bölümünden açıkça izin verilmektedir. Nmap bu tür verilerle karşılaşlığında, bir CRC32 sağlama toplamı gerçekleştirir ve sonuçları bildirir. Veri olmadığında, RD sıfır olarak ayarlanır. Sıfırlama paketlerinde veri döndürebilen birkaç işletim sisteminden bazıları HP-UX ve Mac OS X'ten önceki Mac OS sürümleridir.

IP total length (IPL) (IP toplam uzunluğu (IPL))

Bu test bir IP paketinin toplam uzunluğunu (oktet olarak) kaydeder. Yalnızca U1 testi tarafından ortaya çıkarılan bağlantı noktası ulaşılamıyor yanıtı için kullanılır. Bu uzunluk uygulamaya göre değişir, çünkü minimum RFC 1122 gereksinimini karşıladıkları sürece orijinal probdan ne kadar veri dahil edeceklerini seçmelerine izin verilir. Bu gereklilik, orijinal IP başlığını ve en az sekiz bayt veriyi içermektir.

Unused port unreachable field nonzero (UN) (Kullanılmayan bağlantı noktası ulaşılamaz alanı sıfır değil (UN))

Bir ICMP bağlantı noktası ulaşılamaz mesaj başlığı sekiz bayt uzunluğundadır, ancak yalnızca ilk dört bayt kullanılır. RFC 792 son dört baytin sıfır olması gerektiğini belirtir. Birkaç uygulama (çoğunlukla ethernet anahtarları ve bazı özel gömülü cihazlar) bunu yine de ayarlar. Bu son dört baytin değeri bu alana kaydedilir.

Returned probe IP total length value (RIPL) (Döndürülen sonda IP toplam uzunluk değeri (RIPL))

ICMP port ulaşılamaz mesajlarının (U1 probuna yanıt olarak gönderildiği gibi) kendilerini oluşturan IP başlığını içermesi gereklidir. Bu başlık alındığı gibi geri gönderilmelidir, ancak bazı uygulamalar IP işleme sırasında yaptıkları değişiklikler nedeniyle bozuk bir sürümü geri gönderir. Bu test basitçe döndürülen IP toplam uzunluk değerini kaydeder. Doğru değer olan 0x148 (328) döndürülürse, gerçek değer yerine G (iyi) değeri kaydedilir.

Returned probe IP ID value (RID) (Döndürülen prob IP Kimliği değeri (RID))

U1 probu 0x1042 statik IP ID değerine sahiptir. Bağlantı noktasına ulaşılamıyor mesajında bu değer döndürülürse, bu test için G değeri saklanır. Aksi takdirde, döndürülen tam değer saklanır. Solaris gibi bazı sistemler, Nmap'in gönderdiği ham IP paketleri için IP ID değerlerini değiştirir. Bu gibi durumlarda, bu test atlanır. Bazı sistemlerin, özellikle HP ve Xerox yazıcılarının baytları çevirdiğini ve bunun yerine 0x4210 döndürdüğünü tespit ettiğimizde.

Integrity of returned probe IP checksum value (RIPCK) (Döndürülen prob IP sağlama toplamı değerinin bütünlüğü (RIPCK))

IP sağlama toplamı, bir bağlantı noktasına ulaşılamıyor mesajında döndürüldüğünde aynı kalmasını bekleyemediğimiz bir değerdir. Sonuçta, aktarım sırasında her ağ atlaması, TTL azaldıkça sağlama toplamını değiştirir. Bununla birlikte, aldığımız sağlama toplamı çevreleyen IP paketiyle eşleşmelidir. Eğer eşleşirse, bu test için G (iyi) değeri saklanır. Dönen değer sıfırsa, Z saklanır. Aksi takdirde sonuç I (geçersiz) olur.

Integrity of returned probe UDP checksum ([RUCK](#)) (Dönen sonda UDP sağlama toplamının bütünlüğü (RUCK))

UDP başlık sağlama toplamı değeri tam olarak gönderildiği gibi döndürülmelidir. Eğer öyleyse, bu test için G kaydedilir. Aksi takdirde, gerçekte döndürulen değer kaydedilir.

Integrity of returned UDP data ([RUD](#)) (Dönen UDP verilerinin bütünlüğü (RUD))

Bu test, geri dönen (muhtemelen kesilmiş) UDP yükünün bütünlüğünü kontrol eder. Tüm yük baytları beklenen 'C' (0x43) ise veya yük sıfır uzunluğa kadar kesilmişse G kaydedilir; aksi takdirde I (geçersiz) kaydedilir.

ICMP response code ([CD](#)) (ICMP yanıt kodu (CD))

Bir ICMP yanıt yanıtı (sıfır tipi) paketinin kod değerinin sıfır olması gereklidir. Ancak bazı uygulamalar, özellikle yanıt isteği sıfır olmayan bir koda sahipse (IE testlerinden birinin yaptığı gibi) yanlışlıkla başka değerler gönderir. İki prob için yanıt kodu değerleri Tablo 8.7'de açıklandığı gibi bir CD değerinde birleştirilir.

Tablo 8.7. CD test değerleri

Value	Description (Açıklama)
Z	Her iki kod değeri de sıfırdır.
S	Her iki kod değeri de ilgili probdaki ile aynıdır.
<NN>	Her ikisi de aynı sıfır olmayan sayıyı kullandığında, burada gösterilir.
O	Başka herhangi bir kombinasyon.

IPv6 fingerprinting (IPv6 parmak izi)

Nmap, IPv6 için özelleşmiş benzer ancak ayrı bir işletim sistemi algılama motoruna sahiptir. Yüksek düzeyde, teknik aynıdır: probalar gönderin, yanıtları toplayın ve yanıt kümesini bir veritabanıyla eşleştirin. Farklılıklar, kullanılan belirli probarda ve bunların eşleştirilme biçimindedir.

IPv6 işletim sistemi algılama tipki IPv4 gibi kullanılır. Sadece -6 ve -O seçeneklerini birlikte kullanın. Örneğin, nmap -6 -O <hedef>.

Probes Sent

IPv6 işletim sistemi tespiti, IPv4 işletim sistemi tespitiyle aynı probaların çoğunu kullanır. İşletim sistemlerini ayırt etme gücünün çoğu TCP gibi daha yüksek katmanlı protokollerden gelir, ancak IPv6'ya özgü birkaç yeni algılama özelliği vardır.

Her durumda, IPv6 akış etiketi, bunu ayarlamamıza izin veren platformlarda 0x12345'tir. Bunu yapmayan platformlarda (göndermek için Ethernet kullanılmadığında Linux olmayan Unix platformlarını içerir), akış etiketi 0 olacaktır. Bu yanıtları etkileyebileceğinden, akış etiketinin değeri işletim sistemi parmak izlerinin EXTRA alanına kaydedilir. NS probu dışında, atlama sınırları rastgele belirlenir.

Toplamda en fazla 18 sonda gönderilebilir. Bunlar aşağıdaki sırayla gönderilir.

Sequence generation ([S1 – S6](#)) (Dizi oluşturma (S1-S6))

Bunlar IPv4 algılamasında gönderilen T1 sonda koleksiyonu ile aynı altı sondadır. Paket içeriklerinin dokümantasyonu için "Sıra oluşturma (SEQ, OPS, WIN ve T1)" adlı bölüme bakın. Bu altı prob zamanlama ölçümleri için 100 ms arayla gönderilir.

Hedefin açık bir bağlantı noktası yoksa S1-S6 problemleri atlanır.

ICMPv6 echo (IE1)

Bu aşağı yukarı sıradan bir ICMPv6 yanıt isteğidir. Tür 128 (Yankı İsteği) ve kod 9'dur, ancak 0 olması gereklidir. ICMPv6 Kimliği 0xabcd ve sıra numarası 0'dır. Veri yükü 120 sıfır bayttır. Yalnızca dolgu içeren bir Hop-By-Hop uzatma başlığı vardır.

ICMPv6 echo (IE2)

Bu, türü 128 (Yankı İsteği) ve kodu 0 olan bir yanıt isteğidir. ICMPv6 Kimliği 0xabcd ve sıra 1'dir. Veri yükü yoktur.

Bu sondayı ilginç kıلان şey, içerdiği hatalı uzantı başlıklarıdır. Bu sırayla toplamda dört tane var:

Hop-By-Hop

Hedef Seçenekleri

Routing

Hop-By-Hop

Bu başlıklar hatalıdır: Hedef Seçenekleri dışında hiçbir başlığın birden fazla görünmemesi gereklidir ve Hop-by-hop seçeneklerinin yalnızca ilk konumda görünmesi gereklidir. Testlerimizde, hiçbir işletim sistemi bunu meşru bir yanıt isteği olarak değerlendirdi. Bununla birlikte, farklı ICMPv6 hataları ile yanıt verirler.

Node Information Query (NI) (Düğüm Bilgisi Sorgusu (NI))

RFC 4620, bir hedefe ana bilgisayar adlarını, IPv4 adreslerini ve IPv6 adreslerini sormaya izin veren Düğüm Bilgisi Sorguları adı verilen ICMPv6 mesajlarını tanımlar. NI probunun türü 139 (ICMP Düğüm Bilgisi Sorgusu) ve kodu 0'dır (özneden bir IPv6 adresi olduğunu gösterir). Qtype 4'tür (IPv4 Adresleri). A bayrağı (tüm tek noktaya yayın adreslerini döndür) bayrağı ayarlanır ve başka hiçbir bayrak ayarlanmaz. Nonce "\x01\x02\x03\x04\x05\x06\x07\x0a" sabit dizesine ayarlanır.

IPv4 adresleri istenmesine rağmen, bazı işletim sistemleri bunun yerine bir DNS adı döndürür.

Neighbor Solicitation (NS) (Komşu İsteği (NS))

NS sondası, hedefin donanım adresini sorar gibi bir Komşu İsteği sorgusu gönderir. Tip 135 ve kod 0'dır. --ttl ayarı ne olursa olsun atlama sınırı her zaman 255 olarak ayarlanır; RFC 2461 ana bilgisayarların başka türlü yanıt vermesini yasaklar. Tüm bayraklar 0 olarak ayarlanır.

Bu sonda yalnızca aynı alt ağdaki ana bilgisayarlara gönderilir.

UDP (U1)

Varsa, kapalı bir bağlantı noktasına bir UDP paketi gönderilir. Veri yükü 300 'C' (0x43) bayt olarak ayarlanır. Bu sonda, bir ICMPv6 Bağlantı Noktasına Ulaşılım yok iletisi ortaya çıkarmak için tasarlanmıştır.

TCP explicit congestion notification (TECN) (TCP açık tıkanıklık bildirimi (TECN))

Bu, IPv4'teki ECN probu ile aynıdır. ECE ve CWR bayrakları da ayarlanmış olan açık bir bağlantı noktasına gönderilen bir SYN paketidir. Acil bayrağı ayarlanmamış olsa bile 0xF7F5 acil alan değeri kullanılır. Onay numarası sıfırdır, sıra numarası rastgeledir ve pencere boyutu alanı üçtür. TCP seçenekleri WScale (10), NOP, MSS (1460), SACK izinli, NOP, NOP'dur.

TCP (T2 – T7)

Bunlar, "TCP (T2-T7)" adlı bölümde açıklanan IPv4 tespitindeki T2-T7 problemlerine karşılık gelir. Numaralandırma 1 yerine 2'den başlar çünkü altı sıralama probu IPv4'te topluca "T1" olarak bilinir (IPv6 için S1-S6 olarak yeniden adlandırılmıştır).

Feature extraction (Özellik çıkarma)

Yanıtlar alındıktan sonra, bunlardan çeşitli veri parçaları çıkarılır. Makine öğrenimi literatüründe bu veri parçaları "özellikler" olarak bilinir. Özelliklere örnek olarak şunlar verilebilir: IPv6 atlama sınırı, ICMPv6 türü ve kodu ve ilk TCP seçeneğinin kodu. (Nmap terminolojisinde bunlar sırasıyla IPV6_HOPLIMIT, ICMPV6_TYPE ve TCP_OPT_0 olarak bilinir). Bazı özellikler doğrudan yanıt paketlerinden çıkarılır ve bazıları birkaç paket üzerinde bir hesaplama yapmanın sonucudur (TCP_ISR, TCP başlangıç sıra numarası sayaç oranı gibi).

Değeri belirlenemeyen tüm özellikler (örneğin, hiç alınmamış bir yanıtın özellikleri) -1 olarak ayarlanır. Özellikler tek boyutlu büyük bir özellik vektörüne yerleştirilir. Daha sonra her biri, eğitim verilerimizden tahmin edilen ölçek parametreleri kullanılarak yaklaşık olarak [0, 1] aralığına yerleştirmek için ölçeklendirilir ve çevrilir.

List of all features (Tüm özelliklerin listesi)

TCP_ISR TCP ISN sayaç oranı. Bu, 100 ms arayla gönderilen S1-S6 sıra problemlerinden elde edilir. Ardışık dizi yanıtları arasındaki farklar toplanır, ardından bu toplam ilk ve son sonda arasında geçen süreye bölünür.

Aşağıdaki özellikler her yanıt için tekrarlanır, bu nedenle örneğin tam nitelikli bir özellik adı S1.PLEN olabilir.

PLEN IPv6 Yük Uzunluğu alanı

TC IPv6 Trafik Sınıfı alanı

HLIM IPv6 Atlama Sınırı alanının orijinal değerine ilişkin bir tahmin

Aşağıdaki özellikler her TCP yanıt için tekrarlanır. Tam özellik adı T2.TCP_WINDOW olabilir.

TCP_WINDOW TCP pencere boyutu

TCP_FLAG_F, **TCP_FLAG_S**, **TCP_FLAG_R**, **TCP_FLAG_P**, **TCP_FLAG_A**, **TCP_FLAG_U**, **TCP_FLAG_E**, **TCP_FLAG_C**

TCP bayrakları. Her bayrak 0 veya 1 değerine sahip bir özellik haline gelir.

TCP_FLAG_RES8, **TCP_FLAG_RES9**, **TCP_FLAG_RES10**, **TCP_FLAG_RES11**

Bunlar TCP başlığının ayrılmış bölümünün dört bitidir. RFC 3540 TCP_FLAG_RES8'i nonce sum (NS) biti olarak tanımlar.

TCP_OPT_0, **<...>**, **TCP_OPT_16** İlk 16 TCP seçeneği için kodları yazın.

TCP_OPTLEN_0, **<...>**, **TCP_OPTLEN_16** İlk 16 TCP seçeneğinin uzunlukları.

TCP_MSS Varsa, ilk MSS seçeneğinin değeri.

TCP_SACKOK SACK-permitted seçeneği mevcutsa 1, aksi takdirde 0.

TCP_WSCALE Varsa, ilk Pencere Ölçeği seçeneğinin değeri.

Differences from IPv4 (IPv4'ten Farklılıklar)

IPv6 parmak izleri IPv4 parmak izlerinden biraz farklı görünür. Paket özelliklerinin ayrıstırılmış bir listesi yerine, gönderme ve alma süreleriyle birlikte paket içeriğinin hex dökümünden oluşurlar. Ayrıntılar için "Nmap Parmak İzini Anlamak" adlı bölüme bakın.

IPv6 eşleştirme algoritması oldukça farklıdır. Bir parmak izi listesiyle basit bir karşılaştırma yapmak yerine lojistik regresyon adı verilen bir makine öğrenimi algoritması kullanır. "IPv6 eşleştirme" adlı bölümde algoritmanın bir açıklaması bulunmaktadır.

Fingerprinting Methods Avoided by Nmap (Nmap Tarafından Kaçınılan Parmak İzi Yöntemleri)

Nmap diğer tüm programlardan çok daha fazla işletim sistemi tespit teknlığını desteklemektedir ve her zaman yeni fikirler duymak isteriz. Lütfen bunları tartışmak için Nmap geliştirme listesine (nmap-dev) gönderin. Ancak bazı yöntemler vardır ki sadece iyi bir uyum değildir. Bu bölüm en ilginç olanlardan bazılarını detaylandırmaktadır. Nmap tarafından desteklenmeseler de, bazıları bulguları doğrulamak veya daha fazla ayrıntı öğrenmek için Nmap ile birlikte kullanışlıdır.

Passive Fingerprinting (Pasif Parmak İzi)

Pasif parmak izi, Nmap tarafından gerçekleştirilen aktif parmak izi ile aynı tekniklerin çoğunu kullanır. Aradaki fark, pasif bir sistemin basitçe ağı koklaması ve trafiğini gözlemediği ana bilgisayarları fırsatçı bir şekilde sınıflandırmasıdır. Bu, aktif parmak izinden daha zordur, çünkü kendi özel problemlerini tasarlamak yerine her türlü iletişim kabul etmeniz gereklidir. Bu değerli bir tekniktir, ancak Nmap gibi temelde aktif bir araca ait değildir. Neysse ki, Michał Zalewski mükemmel p0f pasif işletim sistemi parmak izi aracını yazdı. Ayrıca mevcut Nmap OS parmak izi testlerinden birkaçını da tasarladı. Diğer bir seçenek ise hem aktif hem de pasif parmak izini destekleyen GomoR tarafından geliştirilen SinFP'dir.

Exploit Chronology (İstismar Kronolojisi)

TCP/IP parmak izi, farklı işletim sistemlerini ayırt etmek için iyi çalışır, ancak aynı işletim sisteminin farklı sürümlerini tespit etmek zahmetli olabilir. Şirketin yiğinini ayırt edebileceğimiz bir şekilde değiştirmesi gereklidir. Neysse ki, birçok işletim sistemi satıcısı en son standartlara uymak için sistemlerini düzenli olarak güncelliyor. Peki ya güncellemeyenler? Çoğu en azından eninde sonunda istismar edilebilir yiğin hatalarını düzeltir. Ve bu düzeltmeleri uzaktan tespit etmek kolaydır. Önce bir kara saldırısı, gözyaşı daması, ölüm pingi, SYN seli veya WinNuke gibi istismar yükünü gönderin. Her seferinde bir saldırısı gönderin, ardından hemen sistemle tekrar iletişime geçmeyi deneyin. Aniden yanıt vermezse, işletim sistemini düzeltmeye birlikte gönderilmeyen sürümlere kadar daraltmış olursunuz.

Dikkat : İşletim sistemi tespit paketinizin bir parçası olarak hizmet redi (DoS) açıklarını kullanıyorsanız, bu testleri en son gerçekleştirmeyi unutmayın.

Retransmission Times (Yeniden İletim Süreleri)

TCP uygulamaları, paketleri yeniden iletmeden önce tam olarak ne kadar bekledikleri konusunda önemli bir hareket alanına sahiptir. Ring ve Cron-OS kavram kanıtlama araçları bu durumdan faydalananın kullanılabilir. Açık bir bağlantı noktasına bir SYN paketi gönderirler, ardından aldığı SYN/ACK paketini bir ACK (bağlantıyı tamamlamak için) veya bir RST (bağlantıyı sonlandırmak için) ile onaylamak yerine yok sayarlar. Hedef ana bilgisayar SYN/ACK'i birkaç kez daha yeniden gönderir ve bu araçlar beklemenin her alt saniyesini takip eder. Bu teknikten gerçekten de bazı bilgiler elde edilebilse de, yamayı Nmap'e dahil etmemenin birkaç nedeni var:

- Genellikle, sisteminizin aldığı SYN/ACK'ye bir RST paketiyle yanıt vermesini önlemek için kaynak ana bilgisayar güvenlik duvarı kurallarını değiştirmeyi gerektirir. Bunu taşınabilir bir şekilde yapmak zordur. Ve kolay olsa bile, birçok kullanıcı uygulamaların güvenlik duvarı kurallarıyla oynamasından hoşlanmaz.
- Çok yavaş olabilir. Yeniden iletişim birkaç dakika sürebilir. Bu, ilk etapta çok fazla bilgi vermeyen bir test için beklemek için uzun bir süre.
- Hatalı olabilir çünkü paket düşüşleri ve gecikme (gerçek dünya ortamlarında beklemeniz gereken) sahte sonuçlara yol açabilir.

Bu nedenleri burada sıraladım çünkü bunlar önerilen diğer bazı işletim sistemi tespit yöntemleri için de geçerlidir. Yeni testler eklemeyi çok isterim, ancak bunlar hızlı olmalı ve az sayıda paket gerektirmelidir. Ana

bilgisayar güvenlik duvarı ile uğraşmak kabul edilemez. Sürüm tarama sisteminin bir parçası olarak işletim sistemi tespiti için yapılsa da, yiğin parmak izi için tam TCP bağlantıları yapmaktan kaçınmaya çalışıyorum.

IP Fragmentation (IP Parçalanması)

IP parçalanması karmaşık bir sistemdir ve uygulamalar hatalar ve tutarsızlıklarla doludur. Olası testler, üst üste binen parçaların nasıl birleştirildiğini veya birleştirme zaman aşımalarını inceleyebilir. Nmap için bu testlerden kaçınılabilir çünkü birçok güvenlik duvarı ve diğer hat içi cihazlar ağ geçitlerinde trafiği birleştirir. Böylece Nmap gerçek hedef ana bilgisayar yerine güvenlik duvarının parmak izini alabilir. Buna ek olarak, bazı işletim sistemlerinde parçaların gönderilmesi zordur. Linux 2.6 çekirdekleri, göndermeye çalıştığınız parçaları sıraya koyma ve iletimden önce bunları kendisi birleştirme eğilimindedir.

Open Port Patterns (Açık Liman Kalıpları)

Hedef ana bilgisayar işletim sistemi genellikle sadece açık olan portlara bakılarak tahmin edilebilir. Microsoft Windows makinelerinde genellikle 135 ve 139 numaralı TCP portları açıktır. Windows 2000 ve daha yenileri de 445 numaralı bağlantı noktasını dinler. Bu arada, 22 (ssh) ve 631 (Internet Yazdırma Protokolü) numaralı bağlantı noktalarında hizmet çalıştırın bir makine muhtemelen Unix çalıştırıyor.

Bu sezgisel yöntem genellikle faydalı olsa da Nmap için yeterince güvenilir değildir. Port kombinasyonları güvenlik duvari kuralları tarafından gizlenebilir ve çoğu ana protokol birden fazla platformda kullanılabilir. OpenSSH sunucuları Windows üzerinde çalıştırılabilir ve "Windows SMB" portları bir Unix makinesinde çalışan Samba tarafından kullanılabilir. Port yönlendirme sorunu daha da karmaşık hale getirmektedir. Microsoft IIS çalıştırıyor gibi görünen bir makine, 80 numaralı bağlantı noktasını bir Windows makinesine yönlendiren bir Unix güvenlik duvarı olabilir.

Bu nedenlerden dolayı, Nmap TCP/IP yiğini parmak izi tespiti sırasında açık port numaralarını dikkate almaz. Ancak Nmap, işletim sistemi ve aygit türü bilgilerini ayrı ayrı keşfetmek için sürüm algılama bilgilerini kullanabilir (bkz. Bölüm 7, Hizmet ve Uygulama Sürümü Algılama). İşletim sistemi algılama ve sürüm algılama tarafından keşfedilen işletim sistemi algılama sonuçlarını ayrı tutarak, Nmap bir Windows web sunucusuna TCP bağlantı noktasını yönlendirmesi kullanan bir Checkpoint güvenlik duvarını zarif bir şekilde işleyebilir. Yiğin parmak izi sonuçları "Checkpoint Firewall-1" olmalıdır, sürüm tespiti ise işletim sisteminin Windows olduğunu göstermelidir. Sürüm algılama imzalarının yalnızca küçük bir kısmının işletim sistemi ve cihaz türü bilgilerini içerdigini unutmayın; bu alanları yalnızca uygulama bilgileri açıkladığında veya yalnızca bir işletim sistemi veya cihaz türünde çalıştığından doldurabiliriz.

Retired Tests (Emekli Testler)

Bir zamanlar Nmap tarafından gerçekleştirilen, ancak işletim sistemlerini ayırt etmede yardımcı olmadıkları ve yalnızca veritabanında yer kapladıkları tespit edildiği için kullanılmadan kaldırılan bazı testler vardır. IE satırındaki iki test kaldırılmıştır: DLI, geri dönen paketlerdeki veri yükünün uzunluğunu kontrol etti ve SI, ICMP sıra numaralarını kontrol etti. Gönderilen değerlerden farklı bir değere rastlanmamıştır. U1 hattında, RUL testi geri dönen UDP paketinin uzunluğunu kontrol etti. Bu, 1.700'den fazla vakadan sadece birinde gönderilen paketten farklıydı. Bu testler Mart 2009'da kaldırılmıştır.

Diğer testler fazla ayrımcı oldukları için kaldırılmıştır; tespit doğruluğuna zarar veren yanlış farklılıkların ölçülmesine neden olmuşlardır. Bunların her ikisi de yanıtlardaki TOS (hizmet türü) alanıyla ilgiliydi. TOS bunu U1 probu için, TOSI ise IE için yapmıştır. Test değerleri işletim sistemleri arasında yasal olarak farklılık gösterse de, TOS bir ara ana bilgisayar tarafından değiştirildiği için genellikle yanlış farklılıklar kaydediliyordu. Bu testler Ekim 2008'de kaldırıldığından genel olarak daha iyi sonuçlar elde edildi.

Understanding an Nmap Fingerprint (Nmap Parmak İzini Anlama)

Nmap bellekte bir parmak izi sakladığında, kullanıcıların farkında bile olmaması gereken veri yapılarında bir nitelikler ve değerler ağaç kullanır. Ancak, bir makine tanımlanamadığında Nmap'in kullanıcılar için yazdırabileceği ASCII kodlu özel bir sürüm de vardır. Bu serileştirilmiş parmak izlerinin binlercesi de Nmap her çalıştığından (işletim sistemi algılama etkinken) nmap-os-db veritabanından geri okunur. Parmak izi formatı, insan kavrayışı ve kısalık arasında bir uzlaşmadır. Format o kadar kısalır ki birçok deneyimsiz kullanıcıya satır gürültüsü gibi görünür, ancak bu belgeyi okuyanlar parmak izlerini kolaylıkla deşifre edebilmelidir. Aynı genel yapıya sahip olmalarına rağmen aslında iki tür parmak izi vardır. Nmap'in okuduğu bilinen işletim sistemlerinin parmak izleri referans parmak izleri olarak adlandırılırken, Nmap'in bir sistemi taradıktan sonra görüntülediği parmak izi ise konu parmak izidir. Referans parmak izleri biraz daha karmaşıktır, çünkü diğer testler için yalnızca tek bir olası değere izin verirken, çok güvenilir olmayan testlere boş alan ekleyerek (veya atlayarak) tüm bir işletim sistemi sınıfına uyacak şekilde uyarlanabilirler. Referans parmak izlerinde işletim sistemi ayrıntıları ve sınıflandırmaları da bulunmaktadır. Söz konusu testler daha basit olduğu için önce onları açıklayacağız.

Decoding the Subject Fingerprint Format (Denek Parmak İzi Formatının Şifresini Çözme)

Nmap bir ana bilgisayarda işletim sistemi parmak izi çalışması gerçekleştirir ve umut verici koşullara rağmen (hedefte erişilebilir hem açık hem de kapalı bağlantı noktaları bulmak gibi) mükemmel bir işletim sistemi eşleşmesi elde edemezse, Nmap, Nmap'in ilgili olduğunu düşündüğü tüm test sonuçlarını gösteren bir konu parmak izi yazdırır ve ardından kullanıcıdan verileri [Nmap.Org](#)'a göndermesini ister. Nmap'in ilgili prob yanıtlarının alınmaması gibi yararlı sonuçları olmadığından testler gösterilmez. SCAN adlı özel bir satır, parmak izi gönderimlerini nmap-os-db'ye entegre etmek için yararlı bağlam sağlayan tarama hakkında ekstra ayrıntılar (Nmap sürüm numarası gibi) verir. Tipik bir konu parmak izi Örnek 8.3'te gösterilmiştir.

Örnek 8.3. Tipik bir denek parmak izi

```
OS:SCAN(V=5.05BETA1%D=8/23%OT=22%CT=1%CU=42341%PV=N%DS=0%DC=L%G=Y%TM=4A91CB  
OS:90%P=i686-pc-linux-gnu)SEQ(SP=C9%GCD=1%ISR=CF%TI=Z%CI=Z%II=I%TS=A)OPS(O1  
OS:=M400CST11NW5%O2=M400CST11NW5%O3=M400CNNT11NW5%O4=M400CST11NW5%O5=M400  
CS  
OS:T11NW5%O6=M400CST11)WIN(W1=8000%W2=8000%W3=8000%W4=8000%W5=8000%W6=8000)  
OS:ECN(R=Y%DF=Y%T=40%W=8018%O=M400CNNSNW5%CC=N%Q=)T1(R=Y%DF=Y%T=40%S=O%A  
=S+  
OS:%F=AS%RD=0%Q=)T2(R=N)T3(R=Y%DF=Y%T=40%W=8000%S=O%A=S+%F=AS%O=M400CST11N  
W  
OS:5%RD=0%Q=)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=4  
0%W  
OS:=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=  
0%Q=)  
OS:T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%  
U  
OS:N=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)
```

Şimdi bu parmak izine bakabilir ve her şeyin ne anlamına geldiğini hemen anlayabilirsiniz. Eğer öyleyse, bu bölümü atlayabilirsiniz. Ancak ben hiç böyle bir tepki görmedim. Birçok kişi muhtemelen bir tür arabellek taşması veya sonlandırmamış dize hatasının Nmap'in kendilerine çöp veri püskürtmesine neden olduğunu düşünüyor. Bu bölüm, bilgileri çözmenize yardımcı olur, böylece bu makineye karşı kör TCP sıra tahmini saldırılarının orta derecede zor olduğunu hemen söyleyebilirsiniz, ancak iyi bir boşta tarama (-sl) zombi yapabilir. Bu parmak izini anlamanın ilk adımı satır kaydirmayı düzeltmektir. Testlerin hepsi bir araya getirilir ve her satır 71 karaktere sarılır. Daha sonra OS: her satırın başına eklenerek uzunluk 74 karaktere çıkarılır. Bu,

parmak izlerinin kesilip Nmap parmak izi gönderme formuna yapıştırılmasını kolaylaştırır ("Nmap Eşleşme Bulamadığında ve Parmak İzi Yazdırdığında" başlıklı bölüme bakın). Ön ekin kaldırılması ve kelime kaydırmanın düzeltilmesi (her satır bir sağ parantezle bitmelidir) Örnek 8.4'teki temizlenmiş versiyona yol açar.

Örnek 8.4. Temizlenmiş bir özne parmak izi

```
SCAN(V=5.05BETA1%D=8/23%OT=22%CT=1%CU=42341%PV=N%DS=0%DC=L%G=Y%TM=4A91CB9
0%
P=i686-pc-linux-gnu)
SEQ(SP=C9%GCD=1%ISR=CF%TI=Z%CI=Z%II=I%TS=A)
OPS(O1=M400CST11NW5%O2=M400CST11NW5%O3=M400CNNT11NW5%
O4=M400CST11NW5%O5=M400CST11NW5%O6=M400CST11)
WIN(W1=8000%W2=8000%W3=8000%W4=8000%W5=8000%W6=8000)
ECN(R=Y%DF=Y%T=40%W=8018%O=M400CNNSNW5%CC=N%Q=)
T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=N)
T3(R=Y%DF=Y%T=40%W=8000%S=O%A=S+%F=AS%O=M400CST11NW5%RD=0%Q=)
T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)
IE(R=Y%DFI=N%T=40%CD=S)
```

Bu hala dünyanın en sezgisel formatı olmasa da (kısa tutmak zorundaydık), format artık çok daha net. Her satır bir kategoridir, örneğin sıra oluşturma testleri için SEQ, belirli TCP probundan elde edilen sonuçlar için T3 ve iki ICMP yanğı probuyla ilgili testler için IE.

Her test adının ardından, ayrı testler için sonuçları çevreleyen bir çift parantez bulunur. Testler <testname>=<value> biçimini alır. Tüm olası kategoriler, testler ve değerler "Nmap Tarafından Desteklenen TCP/IP Parmak İzi Yöntemleri" adlı bölümde açıklanmıştır. Her test çifti bir yüzde sembolü (%) ile ayrılmıştır. Test değerleri boş olabilir ve eşittir işaretinin hemen ardından bir yüzde sembolüne veya kategori sonlandırıcı sağ paranteze yol açabilir. Örneğimizin T4'ündeki "O=%RD=0%Q=)" dizesi bu boş testlerden ikisini göstermektedir. Boş bir test değeri başka bir boş değerle eşleşmelidir, bu nedenle bu boş TCP quirks Q değeri, Q'nun RU olarak ayarlandığı bir parmak iziyle eşleşmez.

Bazı durumlarda, sadece değeri değil tüm testler eksiktir. Örneğin, örnek parmak izimizdeki T2'de W (TCP penceresi), S (sıra numarası), A (onay numarası), T (TTL) veya TG (TTL tahmini) testleri yoktur. Bunun nedeni, içerdeği tek test ve değer olan R=N'nin T2 probu için hiçbir yanıt döndürülmediği anlamına gelmesidir. Bu nedenle bir pencere değeri veya sıra numarası eklemek pek anlamlı olmayacağından, Benzer şekilde, Nmap çalıştırılan sisteme iyi desteklenmeyen testler atlanır. Örnek olarak, Solaris üzerinde iyi çalışmayan RID (ICMP paketinde dönen IP ID alanı) testi verilebilir, çünkü bu sistem Nmap'in gönderdiği ID alanını bozma eğilimindedir. Sonuçsuz olan testler (TI, CI ve II testleri için IP ID dizisini tespit edememek gibi) de ihmal edilmiştir.

Decoding the `SCAN` line of a subject fingerprint (Bir deneğin parmak izinin TARAMA satırının kodunun çözülmesi)

TARAMA satırı, konu parmak izinde özel bir durumdur. Hedef sistemi tanımlamak yerine, bu testler taramanın çeşitli koşullarını tanımlar. Bunlar Nmap.Org'a gönderilen parmak izlerini entegre etmemize yardımcı olur. Bu satırdaki testler şunlardır:

- Nmap sürüm numarası (V).

- Ay/gün biçiminde tarama tarihi (D).
- Tarama için kullanılan açık ve kapalı TCP portları (hedefte) (OT ve CT). Çoğu testin aksine, bunlar ondalık formatta yazdırılır. Nmap açık veya kapalı bir port bulamadıysa, test boş bir değerle dahil edilir (Nmap muhtemelen kapalı bir port tahmin etse ve oraya bir prob gönderse bile).
- Kapalı UDP bağlantı noktası (CU). Bu CT ile aynıdır, ancak UDP içindir. Taramaların çoğu UDP içermediğinden, bu testin değeri genellikle boştur.
- Özel IP alanı (PV), hedef 10.0.0.0/8, 172.16.0.0/12 veya 192.168.0.0/16 özel ağlarında (RFC 1918) ise Y'dır. Aksi takdirde N olur.
- Ağ uzaklıği (DS) hedeften ağ atlama uzaklığıdır. Hedef localhost ise 0, bir ethernet ağına doğrudan bağlıysa 1 veya Nmap tarafından keşfedildiyse tam mesafedir. Mesafe bilinmiyorsa, bu test atlanır.
- Mesafe hesaplama yöntemi (DC), ağ mesafesinin (DS) nasıl hesaplandığını gösterir. Bu değerleri alabilir: Localhost için L (DS=0); doğrudan alt ağ bağlantısı için D (DS=1); U1 OS algılama probuna verilen ICMP yanıtına dayalı bir TTL hesaplaması için I; ve traceroute atlama sayısı için T. Bu test, ara makineler TTL'yi değiştirdiğinde ICMP TTL hesaplamasının yanlış olması mümkün olduğu için mevcuttur; gerçekten doğrudan bağlı olan bir ana bilgisayar ile sadece yanlış bir hesaplama olabilecek bir ana bilgisayarı ayırt eder.
- İyi sonuçlar (G), koşullar ve sonuçlar bu parmak izini [Nmap.Org](#)'a göndermek için yeterince iyi görünüyorsa Y'dır. Aksi takdirde N'dir. Hata ayıklamayı etkinleştirerek (-d) veya aşırı ayrıntı vererek (-vv) zorlamadığınız sürece, G=N parmak izleri Nmap tarafından yazdırılmaz.
- Hedef MAC öneki (M), hedef MAC adresinin satıcı adına karşılık gelen ilk altı hex basamağıdır. Baştaki sıfırlar dahil değildir. Hedef aynı ethernet ağı üzerinde değilse (DS=1) bu alan atlanır.
- İşletim sistemi tarama süresi (TM) Unix time_t biçiminde (onaltılık olarak) sağlanır.
- Nmap'in hangi platform için derlendiği P alanında belirtilir.

Decoding the Reference Fingerprint Format (Referans Parmak İzi Formatının Kodunu Çözme)

Nmap bir hedefi tarayarak bir özne parmak izi oluşturduğunda, bu verileri nmap-os-db veritabanındaki binlerce referans parmak iziyle eşleştirmeye çalışır. Referans parmak izleri başlangıçta bir veya daha fazla konu parmak izinden oluşturulurlar ve bu nedenle birçok ortak noktaya sahiptir. Eşleştirmeyi kolaylaştırmak ve elbette temsil ettiğleri işletim sistemlerini tanımlamak için biraz ekstra bilgiye sahiptirler. Örneğin, az önce incelediğimiz konu parmak izi, Örnek 8.5'teki referans parmak izinin temelini oluşturabilir.

Örnek 8.5. Tipik bir referans parmak izi

```
Fingerprint Apple Mac OS X Server 10.2.8 (Jaguar) (Darwin 6.8, PowerPC)
Class Apple | Mac OS X | 10.2.X | general purpose
CPE cpe:/o:apple:mac_os_x:10.2.8
SEQ(SP=FB-111%GCD=1-6%ISR=104-10E%TI=1%II=1%SS=S%TS=1)
OPS(O1=M5B4NW0NNT11%O2=M5B4NW0NNT11%O3=M5B4NW0NNT11%O4=M5B4NW0NNT11%O5=M5
B4NW0NNT11%O6=M5B4NNT11)
WIN(W1=8218%W2=8220%W3=8204%W4=80E8%W5=80F4%W6=807A)
ECN(R=Y%DF=Y%T=3B-45%TG=40%W=832C%O=M5B4NW0%CC=N%Q=)
T1(R=Y%DF=Y%T=3B-45%TG=40%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=N)
T3(R=Y%DF=Y%T=3B-45%TG=40%W=807A%S=O%A=S+%F=AS%O=M5B4NW0NNT11%RD=0%Q=)
T4(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
```

```
T5(R=Y%DF=N%T=3B-45%TG=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T7(R=Y%DF=N%T=3B-45%TG=40%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)
U1(DF=N%T=3B-45%TG=40%IPL=38%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=0%RUD=G)
IE(DFI=S%T=3B-45%TG=40%CD=S)
```

Bazı farklılıklar hemen göze çarpmaktadır. Satır kaydırma işlemi yapılmamıştır çünkü bu sadece gönderim süreci için önemlidir. SCAN satırı da kaldırılmıştır, çünkü bu bilgi genel hedef işletim sistemi özelliklerinden ziyade belirli bir tarama örneğini tanımlamaktadır.

Muhtemelen bu referans parmak izi için yeni olan Parmak İzi, Sınıf ve CPE satırlarını da fark etmişsinizdir. Daha ince bir değişiklik ise bazı bireysel test sonuçları kaldırılırken diğerlerinin mantıksal ifadelerle geliştirilmiş olmasıdır.

Free-form OS description ([Fingerprint](#) line) (Serbest biçimli işletim sistemi açıklaması (Parmak izi satırı))

Parmak izi satırı ilk olarak bir belirteç görevi görür, böylece Nmap yeni bir parmak izi yüklemeye başlayacağını bilir. Her parmak izinin yalnızca bir satırı vardır. Parmak izi belirtecinden (ve bir boşluktan) hemen sonra, bu parmak izi tarafından temsil edilen işletim sisteminin (sistemlerinin) metinsel bir açıklaması gelir. Bunlar serbest biçimli İngilizce metinlerdir ve bir makine ayırtıcılarından ziyade insan yorumlaması için tasarlanmıştır. Bununla birlikte, Nmap satıcı, ürün adı ve ardından sürüm numarasını içeren tutarlı bir biçimde bağlı kalmaya çalışır. Daha önce tartışılan sürüm numarası aralıkları ve virgülle ayrılmış alternatifler bu alanda bulunabilir. İşte bazı örnekler:

```
Fingerprint HP LaserJet printer (4050, 4100, 4200, or 8150)
Fingerprint Sun Solaris 9 or 10 (SPARC)
Fingerprint Linux 2.6.22 - 2.6.24
Fingerprint Microsoft Windows Server 2003 SP1
Fingerprint Microsoft Windows XP Professional SP1
Fingerprint Minolta Di550 laser printer
```

İdeal bir dünyada, her farklı işletim sistemi tam olarak tek bir benzersiz parmak izine karşılık gelirdi. Ne yazık ki işletim sistemi sağlayıcıları hayatı bizim için bu kadar kolaylaştırmıyor. Aynı işletim sistemi sürümü, kullanılan ağ sürücülerine, kullanıcı tarafından yapılandırılabilir seçeneklere, yama seviyelerine, işlemci mimarisine, mevcut RAM miktarına, güvenlik duvarı ayarlarına ve daha fazlasına bağlı olarak farklı parmak izi verebilir. Bazen parmak izleri fark edilebilir bir neden olmaksızın farklılık gösterir. Referans parmak izi formatı küçük farklılıklarla başa çıkmak için bir ifade sözdizimine sahip olsa da, büyük farklılıklar keşfedildiğinde aynı işletim sistemi için birden fazla parmak izi oluşturmak genellikle tercih edilir.

Bir işletim sistemi için genellikle birden fazla parmak izine ihtiyaç duyulduğu gibi, bazen tek bir parmak izi birkaç sistemi tanımlar. İki sistem her bir test için aynı sonuçları veriyorsa, Nmap'in her ikisini de olasılık olarak sunmaktan başka seçeneği yoktur. Bu genellikle birkaç nedenden dolayı meydana gelir. Bunlardan biri, satıcıların IP yığınlarında önemli bir değişiklik yapmadan işletim sistemlerinin yeni bir sürümünü yayinallyabilmeleridir. Belki sistemin başka yerlerinde önemli değişiklikler yapmışlardır ya da belki de çok az şey yapmışlardır ancak "yükseleme" satarak bir sürü para kazanmak istiyorlardır. Bu durumlarda, Nmap genellikle Apple Mac OS X 10.4.8 - 10.4.11 veya Sun Solaris 9 veya 10 gibi bir aralık yazdırır.

Yinelenen parmak izlerinin bir başka nedeni de ortak bir işletim sistemini paylaşan gömülü cihazlardır. Örneğin, bir satıcıya ait bir yazıcı ve başka bir satıcıya ait bir ethernet anahtarı arasında üçüncü bir satıcıya ait gömülü bir işletim sistemini paylaşıyor olabilir. Birçok durumda, cihazlar arasındaki ince farklar yine de ayırt edilmelerini sağlar. Ancak bazen Nmap, Cisco 1200 serisi WAP, HP ProCurve 2650 anahtarı veya Xerox Phaser 7400N veya 8550DT yazıcı gibi bir grup olasılığı listelemelidir.

Çok sayıda satıcının aynı OEM cihazı kendi marka ve model numarasıyla özel olarak etiketlediği durumlar da vardır. Burada da Nmap sadece olasılıkları listelemelidir. Ancak bunları ayırt etmek daha az önemlidir çünkü hepsi temelde aynı cihazdır.

TIP: Nmap tarafından yazdırılan açıklama (Parmak İzi satırından gelen) sizin için yeterince bilgilendirici değilse, nmap-os-db'de parmak izinin üzerindeki yorumlarda daha ayrıntılı bilgi bulunabilir. Bölüm 14, Nmap Veri Dosyalarını Anlama ve Özelleştirme'de açıkladığı gibi sisteminizde kurulu olarak bulabilir veya <https://svn.nmap.org/nmap/nmap-os-db> adresinden en son sürümeye bakabilirsiniz. Nmap'in size verdiği tam işletim sistemi açıklamasını arayın. Tam olarak aynı tanıma sahip birkaç Parmak İzi satırı olabileceğini unutmayın, bu nedenle hepsini incelemeniz gerekebilir. Ya da her eşleşmenin satır numarasını gösteren Nmap XML çıktısını kullanın.

Device and OS classification (`Class` lines) (Cihaz ve işletim sistemi sınıflandırması (Sınıf çizgileri))

Parmak İzi açıklaması, Nmap çıktısını doğrudan okuyan analistler için harika çalışsa da, birçok kişi Nmap'i diğer komut dosyalarından ve uygulamalardan çalıştırır. Bu uygulamalar işletim sistemi bilgilerini işletim sistemine özgü güvenlik açıklarını kontrol etmek veya sadece güzel bir grafik veya rapor oluşturmak için kullanabilir.

Bu amaçlar için daha yapılandırılmış bir OS sınıflandırma sistemi mevcuttur. Birden fazla eşleşme olduğunda da kullanışlıdır. Sadece kısmi bir parmak izi alırsınız (belki de hedefte hiç açık port bulunamadı, bu yüzden birçok testin atlanması gerekiyordu), nmap-os-db veritabanındaki düzinelere farklı parmak iziyle eşleşebilir. Tüm bu parmak izleri için ayrıntıları yazdırmak bir karmaşa olurdu. Ancak işletim sistemi sınıflandırması sayesinde Nmap ortak noktaları bulabilir. Tüm eşleşmeler Linux olarak sınıflandırırsa, Nmap hedefin bir Linux kutusu olduğunu yazdıracaktır.

Her parmak izinin bir veya daha fazla Sınıf satırı vardır. Her biri iyi tanımlanmış dört alan içerir: satıcı, işletim sistemi ailesi, işletim sistemi nesli ve cihaz türü. Alanlar boru simbolü (|) ile ayrılmıştır.

Satıcı, bir işletim sistemi veya cihaz üreten şirketdir. Örnekler Apple, Cisco, Microsoft ve Linksys'dir. Kontrol eden bir satıcı olmayan OpenBSD ve Linux gibi topluluk projeleri için, OS aile adı satıcı sütunu için tekrarlanır. OS ailesi Windows, Linux, IOS (Cisco yönlendiriciler için), Solaris ve OpenBSD gibi ürünleri içerir. Ayrıca açıklanmayan işletim sistemlerini kullanan anahtarlar, geniş bant yönlendiriciler ve yazıcılar gibi yüzlerce cihaz bulunmaktadır. Altta yatan işletim sistemi net olmadığından, gömülü kullanılır.

İşletim sistemi nesli, işletim sisteminin daha ayrıntılı bir tanımıdır. Linux nesilleri 2.4.X ve 2.6.X'i içerirken, Windows nesilleri 95, 98, Me, 2000, XP ve Vista'yı içerir. FreeBSD 4.X ve 5.X gibi nesiller kullanır. Nesillere ayırmadığımız belirsiz işletim sistemleri için (veya işletim sistemi sadece gömülü olarak listelendiğinde), bu alan boş bırakılır.

Cihaz türü yönlendirici, yazıcı veya oyun konsolu gibi geniş bir sınıflandırmadır ve bu bölümde daha önce ele alınmıştır. Linux ve Windows gibi hemen hemen her şey için kullanılabilen genel amaçlı işletim sistemleri genel amaçlı olarak sınıflandırılır.

Her alan sadece bir değer içerebilir. Bir parmak izi bu dört alanın birden fazla olası kombinasyonunu temsil ettiğinde, birden fazla Sınıf satırı kullanılır. Örnek 8.6'da bazı örnek Parmak İzi satırları ve bunlara karşılık gelen sınıflandırmalar verilmiştir.

Örnek 8.6. Bazı tipik parmak izi tanımları ve ilgili sınıflandırmalar

Fingerprint D-Link DSL-500G ADSL router
Class D-Link | embedded || broadband router

Fingerprint Linksys WRT54GC or TRENDnet TEW-431BRP WAP
Class Linksys | embedded || WAP

Class TRENDnet | embedded || WAP

Fingerprint Apple Mac OS X 10.3.9 (Panther) - 10.4.7 (Tiger)

Class Apple | Mac OS X | 10.3.X | general purpose

Class Apple | Mac OS X | 10.4.X | general purpose

Fingerprint Sony PlayStation 3 game console

Class Sony | embedded || game console

Bu örnekler yeterli değilse, Nmap'in en son sürümü tarafından tanımlanmış sınıflandırmaların bir listesi

<https://nmap.org/data/os-classes.txt> adresinde tutulmaktadır.

CPE name (CPE lines) (CPE adı (CPE hatları))

CPE satırları, Sınıf satırlarının Ortak Platform Numaralandırma eşdeğerlerini verir. Her Sınıfın ardından birkaç CPE satırı gelebilir (CPE satırı her zaman kendilerinden hemen önce gelen Sınıf satırına "aittir"). Bir CPE adının işletim sistemini, diğerinin ise donanım platformunu tanımlaması yaygındır. CPE sözdizimi ve anlamının bir açıklaması "Ortak Platform Numaralandırma (CPE)" adlı bölümde bulunabilir.

Örnek 8.7. Tipik CPE sınıflandırmaları

Fingerprint 3Com NBX 100 VoIP gateway (VxWorks)

Class 3Com | VxWorks || VoIP adapter

CPE cpe:/h:3com:nbx_100

CPE cpe:/o:3com:vxworks auto

Fingerprint Linux 2.0.33

Class Linux | Linux | 2.0.X | general purpose

CPE cpe:/o:linux:linux_kernel:2.0.33

Bazı CPE adlarını takip eden auto bayrağı CPE'nin bir parçası değildir; yalnızca bakım komut dosyaları tarafından dahili olarak bir CPE adının manuel olarak eklenmek yerine diğer bilgilerden otomatik olarak oluşturulduğunu belirtmek için kullanılır.

Test expressions (Test ifadeleri)

Test ifadelerinin bir denek ve referans parmak izi arasında değişmesi gerekmek, ancak neredeyse her zaman değişir. Referans parmak izinin, yalnızca taradığınız makine yerine belirli bir işletim sisteminin tüm örnekleriyle eşleşmesi için genellikle biraz genelleştirilmesi gereklidir. Örneğin, bazı Windows XP makineleri T1 probuna F424 Pencere boyutu döndürürken, diğerleri FAFO döndürür. Bunun nedeni kullanılan ethernet aygıt sürücüsü ya da kullanılabilir bellek miktarı olabilir. Her durumda, hangi pencere boyutu kullanılırsa kullanılsın Windows XP'yi tespit etmek istiyoruz.

Bir parmak izini genelleştirmenin bir yolu, tutarsız sonuçlar üreten testleri kaldırmaktır. Referans bir parmak izinden tüm pencere boyutu testlerini kaldırığınızda, sistemler hangi boyutu kullanırsa kullanınsın bu iz ile eşleşecektir. Dezavantajı ise bu şekilde birçok önemli bilgiyi kaybedebilmenizdir. Belirli bir sistemin gönderdiği tek Pencere boyutu F424 ve FAFO ise, 65.536 olasılığın tümüne değil, yalnızca bu iki değere izin vermek istersiniz.

Testleri kaldırma bazı durumlarda aşırıya kaçarken, bazı durumlarda yararlıdır. Bir yanıt olduğu anlamına gelen R=Y test değeri, genellikle nmap-os-db'ye eklenmeden önce U1 ve IE testlerinden kaldırılır. Bu problemler genellikle bir güvenlik duvarı tarafından engellenir, bu nedenle bir yanıtın olmaması işletim sistemi eşleşmesine karşı sayılmamalıdır.

Testlerin kaldırılması istenmediğinde, Nmap bir testin birden fazla değerle eşleşmesine izin vermek için bir ifade sözdizimi sunar. Örneğin, `W=F424|FAF0` diğerlerine izin vermeden bu iki Windows XP pencere değerine izin verir. Tablo 8.8 test değerlerinde izin verilen operatörleri göstermektedir.

Tablo 8.8. Referans parmak izi testi ifade operatörleri

Op Name (Op adı)	Symbol (Sembol)	Example (Örnek)	Description (Açıklama)
Or		O= ME MNNTNW	İlgili konu parmak izi testinin herhangi bir cümlenin değerini alması durumunda eşleşir. Bu örnekte, baştaki boru sembolü boş bir seçenek listesinin de eşleşeceği anlamına gelir.
Range	-	SP=7-A	Söz konusu parmak izinin karşılık gelen testi belirtilen aralıkta bir sayısal değer üretiyorsa eşleşir.
Greater than	>	SP>8	Söz konusu parmak izinin karşılık gelen testi, belirtilen değerden daha büyük bir sayısal değer üretiyorsa eşleşir.
Less than	<	GCD=<5	Söz konusu parmak izinin karşılık gelen testi belirtilen değerden daha küçük bir sayısal değer üretiyorsa eşleşir.

`GCD=1-6|64|256|>1024` ifadesinde olduğu gibi ifadeler operatörleri birleştirilebilir; bu ifade GCD'nin bir ile altı arasında, tam olarak 64, tam olarak 256 veya 1024'ten büyük olması durumunda eşleşir.

IPv6 fingerprints (IPv6 parmak izleri)

IPv6 sınıflandırma motoru farklı çalıştığı için farklı parmak izlerine sahiptir. Referans parmak izleri yoktur; bunun yerine önceden tanımlanmış bir dizi eğitim örneği, her özellik ve işletim sistemi sınıfı için birer tane olmak üzere büyük bir katsayı matrisi çeken bir eğitim algoritması aracılığıyla çalıştırılır. Konu parmak izleri, Örnek 8.8, "Bir IPv6 parmak izi" bölümünde gösterildiği gibi IPv4 ile aynı ASCII zırhlı formатı kullanır.

Örnek 8.8. Bir IPv6 parmak izi

```
OS:SCAN(V=5.61TEST1%E=6%D=9/27%OT=22%CT=443%CU=42192%PV=N%DS=5%DC=T%G=Y%TM
OS:=4E82908D%P=x86_64-unknown-linux-gnu)S1(P=6000{4}28063cXX{32}0016c1b002
OS:bbd213c57562f5a01212e0f8880000020404c40402080a5be177f2ff{4}01030307%ST=
OS:0.021271%RT=0.041661)S2(P=6000{4}28063cXX{32}0016c1b108d7da47c57562f6a0
OS:1212e0e9d20000020404c40402080a5be17856ff{4}01030307%ST=0.121251%RT=0.14
OS:4586)S3(P=6000{4}28063cXX{32}0016c1b21029efebc57562f7a01212e0cf63000002
OS:0404c40101080a5be178ceff{4}01030307%ST=0.221232%RT=0.268086)S4(P=6000{4
OS:}28063cXX{32}0016c1b31553d32dc57562f8a01212e0e3a40000020404c40402080a5b
OS:e1791eff{4}01030307%ST=0.321237%RT=0.340261)S5(P=6000{4}28063cXX{32}001
OS:6c1b41ae90087c57562f9a01212e0b04f0000020404c40402080a5be17982ff{4}01030
OS:307%ST=0.421246%RT=0.441253)S6(P=6000{4}24063cXX{32}0016c1b5207baa83c57
OS:562fa901212e014690000020404c40402080a5be179e6ff{4}%ST=0.521245%RT=0.541
OS:755)IE1(P=6000{4}803a3cXX{32}810927cbabcd00{122}%ST=0.565533%RT=0.59350
OS:5)U1(P=6000{3}01643a3cXX{32}0104be5300{4}6001234501341131XX{32}c1a9a4d0
OS:013482ff43{300}%ST=0.713832%RT=0.734263)TECN(P=6000{4}20063cXX{32}0016c
OS:1b62f0c74d8c57562fb80121310241c0000020404c40101040201030307%ST=0.763567
OS:%RT=0.784838)T2(P=6000{4}583a3cXX{32}0101ca5600{4}6001234500280632XX{32
OS:)c1b70016c57562fb85549cefa00000808c0d000003030a0102040109080aff{4}00{4}
OS:0402%ST=0.813012%RT=0.833344)T3(P=6000{4}583a3cXX{32}0101ca6000{4}60012
OS:34500280628XX{32}c1b80016c57562fc2b8e3db7a02b0100445f000003030a01020401
OS:09080aff{4}00{4}0402%ST=0.863293%RT=0.881198)T4(P=6000{4}14063cXX{32}00
```

```
OS:16c1b93a67fc8a00{4}500400000c7c0000%ST=0.912394%RT=0.93247)T5(P=6000{4}
OS:14063cXX{32}01bbc1ba00{4}c57562ff5014000019430000%ST=0.96164%RT=0.98347
OS:5)T6(P=6000{4}14063cXX{32}01bbc1bba9e336d500{4}50040000610e0000%ST=1.01
OS:164%RT=1.03554)T7(P=6000{4}583a3cXX{32}0101ca5800{4}6001234500280630XX{
OS:32}c1bc01bbc5756300095eb241a029ffffec59000003030f0102040109080aff{4}00{
OS:4}0402%ST=1.06173%RT=1.07961)EXTRA(FL=12345)
```

Örnek 8.9, "Temizlenmiş bir IPv6 parmak izi" bu parmak izinin paketlenmemiş halinin nasıl göründüğünü göstermektedir. Problemlerin çoğu atlanmıştır çünkü hepsi aynı biçimde sahiptir.

Örnek 8.9. Temizlenmiş bir IPv6 parmak izi

```
SCAN(V=5.61TEST1%E=6%D=9/27%OT=22%CT=443%CU=42192%PV=N%DS=5%DC=T%G=Y%TM4E8
2908D%
P=x86_64-unknown-linux-gnu)
S1(P=6000{4}28063cXX{32}0016c1b002bbd213c57562f5a01212e0f8880000020404c4040208
0a5be177f2ff{4}01030307%ST=0.021271%RT=0.041661)
EXTRA(FL=12345)
```

SCAN satırı IPv4 parmak izlerinde olduğu gibi aynı anlamda sahiptir. Sözde test E=6 bunun bir IPv6 parmak izi olduğunu gösterir.

Ardından, yanıt alınan her bir prob için bir satır vardır. Bunların her birinin içinde üç anahtar vardır:

P Paketin içeriği, hex- ve çalışma uzunluğu kodlu. İki rakamın ardından küme parantezleri içinde bir sayı geldiğinde, bu baytin belirtilen sayıda tekrarlanacağı anlamına gelir. Örneğin, 00{4} 00000000 için kısaltmadır. XX karakterleri, özel olan ve sınıflandırıcıyı eğitmek için zaten yararlı olmayan kaynak ve hedef adreslerin yerine konur.

ST Paketin gönderildiği zaman, OS algılamasının başladığı zamana göre saniye cinsinden.

RT Paketin alındığı zaman.

EXTRA satırı, yalnızca paket içeriğinden belirlenemeyen diğer bilgileri saklar. FL anahtarı tarama sırasında gönderilen akış etiketini saklar. Bazı işletim sistemlerinin akış etiketinin ayarlanması izin vermemesi ve bunun yerine her zaman 00000 göndermesi dışında, bu her zaman 12345 olacaktır.

Yukarıdaki gibi bir parmak izi işlendiğinde, aşağıda gösterildiği gibi dahili bir temsile dönüştürülür. Her bir değer, sınıflandırıcıya aktarılan özellik vektörünün bir elemanıdır. Bunlar, "Tüm özelliklerin listesi" adlı bölümde listelenen özelliklere karşılık gelir.

```
40 S1.PLEN
0 S1.TC
40 S2.PLEN
0 S2.TC
40 S3.PLEN
0 S3.TC
40 S4.PLEN
0 S4.TC
40 S5.PLEN
0 S5.TC
36 S6.PLEN
0 S6.TC
```

```
128 IE1.PLEN
  0 IE1.TC
UNKNOWN IE2.PLEN
UNKNOWN IE2.TC
UNKNOWN NS.PLEN
UNKNOWN NS.TC
  356 U1.PLEN
    0 U1.TC
<Many more lines omitted.>
```

UNKNOWN değerleri genellikle ilgili proba yanıt alınamadığı anlamına gelir. UNKNOWN değerleri sınıflandırmadan önce -1 ile eşleştirilir.

Device Types (Cihaz Türleri)

"Cihaz ve işletim sistemi sınıflandırması (Sınıf çizgileri)" bölümünde belirtildiği gibi, her referans parmak izi bir veya daha fazla cihaz türüyle sınıflandırılır. Bu liste Nmap tarafından kullanılan cihaz türlerini ve bir cihazı her bir tür olarak sınıflandırma kriterlerini içerir. Aynı kurallar sürüm tespitinde cihaz türlerini sınıflandırmak için de kullanılır; "eşleşme Yönergesi" adlı bölümdeki d// alanının açıklamasına bakın.

general purpose (genel amaçlı) ⇒ Bu kategori Linux ve Windows gibi genel amaçlı işletim sistemlerini içerir. nmap-service-probes dosyasında bu sınıf d// alanının olmamasıyla gösterilir.

bridge (Köprü) ⇒ Bir köprü iki veya daha fazla alt ağı tek bir ağıda birleştirir. Bir köprü ile bu, bir yönlendiriciden daha düşük bir seviyede gerçekleşir. Bu kategori Ethernet-seri köprüleri gibi şeyleri de içerir.

broadband router (geniş bant yönlendirici) ⇒ Bu kategorideki cihazlar bir ağı kablo, ADSL, fiber optik vb. aracılığıyla internete bağlar. Bu cihazlardan bazıları ağ adresi çevirisi, güvenlik duvarı, port yönlendirme veya diğer hizmetleri sağlar.

firewall (Güvenlik duvarı) ⇒ Güvenlik duvarı bir ağa hangi trafiğin girip çıkacağını kontrol eder. Bazlarının ek yetenekleri de vardır. Bu kategori, bir güvenlik duvarı ile birlikte gelen genel amaçlı işletim sistemlerini içermez, ancak yalnızca bir güvenlik duvarı olarak çalışmak üzere özel olarak oluşturulmuş işletim sistemi dağıtımlarını içerir.

game console (oyun konsolu) ⇒ Xbox veya PlayStation gibi bir video oyun konsolu.

hub ⇒ Bir hub, tüm trafiği yeniden yayinallyarak ağ segmentlerini birleştirir. Hub'lar, paketleri yalnızca ilgili hedeflere seçerek iletten anahtarlardan farklıdır.

load balancer (yük dengeleyici) ⇒ Gelen trafiği birden fazla cihaza dağıtarak bu cihazların üzerindeki yükü hafifleten cihaz.

media device (medya cihazı) ⇒ Bu kategori, taşınabilir müzik çalarlar, ev ses sistemleri, TV'ler ve projektörler dahil olmak üzere her türlü görsel-işitsel ekipmanı içerir.

PBX ⇒ Özel bir şube santrali veya PBX, telefon çağrılarını özel bir kuruluş içinde yönlendirir ve bunları genel telefon ağına veya VoIP'ye bağlar.

PDA ⇒ Bir el bilgisayarı. Aynı zamanda telefon olan cihazlar "telefon" kategorisine girer.

phone ⇒ VoIP telefonu olmayan ağ özellikli bir telefon. Bu kategorideki cihazlar genellikle cep telefonlarıdır.

power-device (güç-cihazı) ⇒ Kesintisiz güç kaynakları ve aşırı gerilim koruyucuları gibi çeşitli güç cihazları.

printer (Yazıcı) ⇒ Katıştırılmış baskı sunucusuna sahip yazıcılar da dahil olmak üzere ağ özellikli yazıcılar.

print server (Yazdırma Sunucusu) ⇒ Yazdırma sunucusu bir yazıcıyı ağa bağlar. Kendi yazdırma sunucusunu içeren yazıcılar bunun yerine "yazıcı" kategorisine girer.

proxy server (proxy sunucusu) ⇒ Web proxy'leri ve verileri önbelleğe alan veya üst düzey protokollerini anlayan diğer sunucular da dahil olmak üzere her türlü proxy.

remote management (uzaktan yönetim) ⇒ Sunucuların veya diğer ekipmanların uzaktan izlenmesini veya yönetilmesini sağlayan cihazlar.

router ⇒ Yönlendiriciler birden fazla ağı birbirine bağlar. Bir ağı genişletmek yerine paketleri farklı ağlar arasında yönlendirdikleri için hub ve anahtarlardan farklıdır.

security-misc (güvenlik-çeşitli) ⇒ "Güvenlik duvarı" kategorisine girmeyen tüm güvenlik cihazları bu kategoriye aittir. Buna saldırı tespit ve önleme sistemleri de dahildir.

specialized (uzmanlaşmış) ⇒ Her şeyi kapsayan kategori. Bir cihaz diğer kategorilerden birine girmiyorsa özelleşmiş demektir. Bu kategorideki örnekler çok çeşitlidir ve saatler, osiloskoplar, iklim sensörleri ve daha fazlasını içerir.

storage-misc (depolama-çeşitli) ⇒ Tüp desteleri ve ağa bağlı depolama cihazları gibi veri depolama cihazları.

switch ⇒ Paketleri seçici olarak yeniden yayinallyarak bir ağı genişleten bir cihaz. Anahtarlar, tüm paketleri yayinallyan hub'lardan farklıdır.

telecom-misc (Telekom Çeşitli) ⇒ Sesli posta ve ISDN sistemleri gibi PBX olmayan telefon sistemleri tarafından kullanılan cihazlar.

terminal ⇒ Birincil amacı bir terminal sunucusu veya ana bilgisayar ile doğrudan iletişim kurmak olan klavye ve monitöre sahip bir cihaz.

terminal server (Terminal Sunucusu) ⇒ Bir ağ üzerinden istemcilere terminal olanakları sağlayan bir cihaz.

VoIP adapter (VoIP Adaptör) ⇒ IP üzerinden ses (VoIP) protokollerini ile normal telefon trafiği arasında dönüşüm yapan bir cihaz. Farklı VoIP protokollerini de dönüştürebilir.

VoIP phone (VoIP Telefon) ⇒ VoIP protokolü kullanabilen bir telefon.

WAP ⇒ Kablosuz erişim noktaları bir ağa kablosuz bağlantı sunar. Çoğu 802.11b gibi radyo teknolojisi ile çalışır ancak bazıları kızıl ötesi veya başka bir şey kullanır. Kablosuz geniş bant yönlendiriciler gibi başka bir kategoriye de yerleştirilebilecek cihazlar, WAP'lar özel ağ hususları gerektirdiğinden WAP kategorisine yerleştirilir.

webcam ⇒ Resim veya video depolayan veya iletken her türlü kamera. Bu, tüketici web kameralarından güvenlik sistemi kameralarına kadar her şeyi içerir.

OS Matching Algorithms (İşletim Sistemi Eşleştirme Algoritmaları)

IPv4 matching (IPv4 eşleştirme)

Nmap'in eşlemeleri tespit etme algoritması nispeten basittir. Bir konu parmak izini alır ve nmap-os-db'deki her bir referans parmak izine karşı test eder.

Bir referans parmak izine karşı test yaparken, Nmap konu parmak izindeki (SEQ veya T1 gibi) her bir prob kategorisi satırına sırayla bakar. Referans parmak izinde bulunmayan tüm prob satırları atlanır. Referans parmak izinde eşleşen bir satır varsa, bunlar karşılaştırılır.

Bir sonda satırı karşılaştırması için Nmap, konu kategori satırındaki her bir testi (R, DF, W, vb.) sırayla inceler. Referans satırında bulunmayan tüm testler atlanır. Eşleşen bir test bulunduğuanda, Nmap PossiblePoints akümülatörünü bu teste atanan puan sayısını kadar artırır. Ardından test değerleri karşılaştırılır. Referans test boş

bir değere sahipse, konu testi yalnızca kendi değeri de boşsa eşleşir. Referans testi sadece düz bir dize veya sayı ise (işleç yok), konu testi tam olarak eşleşmelidir. Referans dizesi işaretler (|, -, > veya <) içeriyorsa, konu "Test ifadeleri" bölümünde açıklandığı gibi eşleşmelidir. Bir test eşleşirse, NumMatchPoints akümülatörü testin puan değeri kadar artırılır.

Tüm sonda hatları bir parmak izi için test edildikten sonra, Nmap NumMatchPoints'i PossiblePoints'e böler. Sonuç, söz konusu parmak izinin belirli bir referans parmak iziyle eşleşme olasılığını tanımlayan bir güven faktöründür. Örneğin, 1.00 mükemmel bir eşleşme iken 0.95 çok yakındır (%95).

Test puan değerleri nmap-os-db'de özel bir MatchPoints girdisi (yalnızca bir kez görünebilir) tarafından atanır. Bu girdi normal bir parmak izine çok benzer, ancak her test için sonuçlar sağlamak yerine, her test için puan değerleri (negatif olmayan tam sayılar) sağlar. MatchPoints yapısında listelenen testler yalnızca listelendikleri aynı testte bulunduklarında geçerlidir. Yani T1'deki W (Pencere boyutu) testi için verilen bir değer T3'teki W testini etkilemez. Bir test, 0 puan değeri atanarak etkin bir şekilde devre dışı bırakılabilir. Örnek bir MatchPoints yapısı Örnek 8.10'da verilmiştir.

Örnek 8.10. MatchPoints yapısı

```
MatchPoints
SEQ(SP=25%GCD=75%ISR=25%TI=100%CI=50%II=100%SS=80%TS=100)
OPS(O1=20%O2=20%O3=20%O4=20%O5=20%O6=20)
WIN(W1=15%W2=15%W3=15%W4=15%W5=15%W6=15)
ECN(R=100%DF=20%T=15%TG=15%W=15%O=15%CC=100%Q=20)
T1(R=100%DF=20%T=15%TG=15%S=20%A=20%F=30%RD=20%Q=20)
T2(R=80%DF=20%T=15%TG=15%W=25%S=20%A=20%F=30%O=10%RD=20%Q=20)
T3(R=80%DF=20%T=15%TG=15%W=25%S=20%A=20%F=30%O=10%RD=20%Q=20)
T4(R=100%DF=20%T=15%TG=15%W=25%S=20%A=20%F=30%O=10%RD=20%Q=20)
T5(R=100%DF=20%T=15%TG=15%W=25%S=20%A=20%F=30%O=10%RD=20%Q=20)
T6(R=100%DF=20%T=15%TG=15%W=25%S=20%A=20%F=30%O=10%RD=20%Q=20)
T7(R=80%DF=20%T=15%TG=15%W=25%S=20%A=20%F=30%O=10%RD=20%Q=20)
U1(R=50%DF=20%T=15%TG=15%TOS=0%IPL=100%UN=100%RIPL=100%RID=100%RIPCK=100%RUCK=
100%RUL=100%RUD=100)
IE(R=50%DFI=40%T=15%TG=15%TOSI=0%CD=100%SI=100%DLI=100)
```

Tüm referans parmak izleri değerlendirildikten sonra, Nmap bunları sıralar ve mükemmel eşleşmeleri yazdırır (çok fazla yoksa). Mükemmel eşleşme yoksa, ancak bazıları çok yakınsa, Nmap bunları yazdırabilir. Eğer --osscan-guess seçeneği verilirse tahminlerin yazdırılma olasılığı daha yüksektir.

IPv6 matching (IPv6 Eşleştirme)

IPv6 OS sınıflandırması, lojistik regresyon adı verilen bir makine öğrenme teknigi kullanır. Nmap bu sınıflandırmayı yapmak için LIBLINEAR kütüphanesini kullanır. Süreç, Nmap kullanıcıları tarafından gönderilen ve işletim sistemleriyle dikkatlice etiketlenen parmak izleri olan geniş bir eğitim örnekleri külliyatı ile başlar. Her eğitim örneği, çok boyutlu bir uzayda o işletim sisteminin "koordinatları" olarak düşünülebilecek bir özellik vektörü ile temsil edilir. Eğitim algoritması, her bir işletim sistemi sınıfının üyeleri ile diğer tüm sınıfların üyeleri arasında en uygun sınırı hesaplar. Daha sonra bu sınırların her birini bir vektör olarak kodlar. Her OS sınıfı için farklı bir vektör vardır.

Eşleştirme sırasında, motor bu sınır vektorlarının her birini sırayla alır ve özellik vektörü ile arasında bir nokta çarpımı hesaplar. Sonuç tek bir gerçek sayıdır. Sayı ne kadar yüksekse (ne kadar pozitifse), eşleşme olasılığı o kadar yüksektir. Negatif sayılar olası eşleşmeler değildir. Bir x sayısı, $100 / (1 + ex)$ lojistik formülü kullanılarak $[-\infty, \infty]$ aralığından $[0, 100]$ aralığına eşleştirilir. ("Lojistik regresyon" adının kaynağı budur.)

Genel olarak, en yüksek puana sahip işletim sistemi sınıfı en olası eşleşmedir, ancak daha önce hiç görülmemiş bir işletim sistemi söz konusu olduğunda, çok yüksek bir puana sahip olmak mümkündür, ancak yine de yanlış bir eşleşme olabilir. Bu nedenle ikinci bir "yenilik algılama" algoritması, gözlemlenen parmak izinin sınıfın diğer temsilcilerinden çok farklı olup olmadığını kontrol eder. Algoritma, gözlemlenen özellik vektöründen sınıfın üyelerinin özellik vektörlerinin ortalamasına olan Öklid mesafesini bulur ve her boyutta bu özelliğin varyansının tersi ile ölçeklendirilir. Daha önce görülenlere benzer özellik vektörleri düşük yeniliğe sahip olacak ve farklı olanlar yüksek yeniliğe sahip olacaktır.

En yüksek puana sahip OS sınıfı bir eşleşme olarak rapor edilir, ancak yalnızca yenilik 15'in altındaysa. Ayrıca, en yüksek iki işletim sistemi sınıfının puanları %10'dan daha az farklılık gösteriyorsa, sınıflandırma belirsiz olarak kabul edilir ve başarılı bir eşleşme değildir. Mac OS X 10.6.8'e karşı yapılan bir çalışmadan elde edilen örnek lojistik ve yenilik puanları Tablo 8.9, "Mac OS X'e karşı işletim sistemi tahminleri" bölümünde gösterilmektedir.

Tablo 8.9. Mac OS X'e karşı işletim sistemi tahminleri

Score	Novelty ()	OS class (İşletim Sistemi sınıfı)
61.05%	1.00	Apple Mac OS X 10.6.8 - 10.7.0 (Snow Leopard - Lion) (Darwin 10.8.0 - 11.0.0)
10.08%	18.04	Apple Mac OS X 10.7 (Lion) (Darwin 11.1.0)
9.97%	24.06	Apple Mac OS X 10.6.8 (Snow Leopard) (Darwin 10.8.0)
e)		
9.43%	19.26	Apple Mac OS X 10.7.2 (Lion) (Darwin 11.2.0)
5.99%	23.63	Apple Mac OS X 10.4.11 (Tiger) (Darwin 8.11.1)
2.28%	34.67	Apple iPhone mobile phone (iOS 4.2.1)
2.19%	35.07	Apple Mac OS X 10.4.7 (Panther) (Apple TV 3.0.2)
2.19%	57.63	HP ProCurve 2520G switch
2.04%	37.03	Apple Mac OS X 10.6.8 (Snow Leopard) (Darwin 10.8.0)
2.03%	68.55	Apple Mac OS X 10.6.8 (Snow Leopard) (Darwin 10.8.0)
0.59%	79.61	FreeBSD 6.1-RELEASE

IPv4'te olduğu gibi IPv6 OS veritabanını içeren ayrı bir veri dosyası yoktur. Veritabanı C++ kaynak kod dosyası FPMModel.cc'de saklanır. Bu dosya ölçekleme sabitlerini (özellik değerlerini kabaca [0, 1] aralığına yerleştirmek için kullanılır) ve yukarıda açıklanan sınır vektörlerini içerir.

Dealing with Misidentified and Unidentified Hosts (Yanlış Tanımlanmış ve Tanımlanmamış Ev Sahipleri ile Başa Çıkma)

Nmap büyük bir veritabanına sahip olsa da her şeyi tespit edemez. Nmap'in çoğu toast makinesini, buzdolabını, sandalyeyi ya da otomobili tespit etme şansı yoktur çünkü bunların IP yiğini yoktur. Yine de, sürekli genişleyen bağlı cihazlar listesi göz önüne alındığında, bunların hiçbirini göz ardı etmezdim. Nmap parmak izi DB'si çok sayıda oyun konsolu, telefon, termometre, kamera, interaktif oyuncak ve medya oynatıcı içeriyor.

Bir IP adresine sahip olmak, doğru bir parmak izini garanti etmek için gereklidir ancak yeterli değildir. Nmap yine de yanlış tahmin yapabilir veya hiç tahmin üretemeyebilir. İşte sonuçlarınızı iyileştirmek için bazı öneriler:

En son Nmap'e yükseltme ⇒ Birçok Linux dağıtımı ve diğer işletim sistemleri Nmap'in eski sürümleri ile birlikte gelmektedir. Nmap OS veritabanı neredeyse her sürümde geliştirilmektedir, bu nedenle nmap -V'yi çalıştırarak sürüm numaranızı kontrol edin ve ardından bunu <https://nmap.org/download.html> adresindeki en son sürümle karşılaşırın. En yeni sürümün yüklenmesi çoğu platformda sadece birkaç dakika sürer.

Tüm bağlantı noktalarını tarayın ⇒ Nmap belirli bir ana bilgisayara karşı işletim sistemi algılama sorunları tespit ettiğinde, uyarılar yayınılayacaktır. En yaygın olanlardan biri şudur: "Uyarı: En az 1 açık ve 1 kapalı TCP bağlantı noktası bulamadığımız için işletim sistemi algılaması ÇOK daha az güvenilir olacaktır". Bu tür bağlantı noktalarının makinede gerçekten kullanılmıyor olması mümkün değildir, ancak tüm bağlantı noktalarını taramak için taramanızı -p- ile yeniden denemek, işletim sistemi algılaması için duyarlı olan bazlarını bulabilir. Bir UDP taraması (-sU) yapmak da taramayı önemli ölçüde yavaşlatacak olsa da daha fazla yardımcı olabilir.

Daha agresif bir tahmin deneyin ⇒ Nmap yazdırılacak kadar yakın eşleşme olmadığını söylüyorsa, muhtemelen bir sorun vardır. Belki bir güvenlik duvarı ya da NAT kutusu prob ya da yanıt paketlerini değiştiriyor. Bu, bir grup testin bir işletim sistemine ait gibi göründüğü, diğer bir grubun ise tamamen farklı göründüğü karma bir duruma neden olabilir. --osscan-guess eklenmesi neyin çalıştığına dair daha fazla ipucu verebilir.

Farklı bir konumdan tarama ⇒ Paketiniz hedefine ulaşmak için ne kadar çok ağ atlamasından geçmek zorunda kalırsa, bir ağ cihazının probu veya yanıtı değiştirme (veya düşürme) olasılığı o kadar artar. NAT ağ geçitleri, güvenlik duvarları ve özellikle port yönlendirme, işletim sistemi algılamasını karıştırabilir. Paketleri farklı bir sunucu ağına yönlendiren bir yük dengeleme cihazının IP'sini tariyorsanız, "doğru" işletim sistemi algılama sonucunun ne olacağı bile net değildir.

Birçok İSS "kötü" bağlantı noktalarına giden trafiği filtreler ve diğerleri de belirli bağlantı noktalarını kendi sunucularına yönlendirmek için şeffaf proxy'ler kullanır. Hedefinizde açık olduğunu düşündüğünüz 25 veya 80 numaralı bağlantı noktası aslında İSS'nizden İSS proxy sunucularına bağlanmak için sahte olabilir. İşletim sistemi algılamasını karıştırabilecek bir başka davranış da güvenlik duvarlarının TCP sıfırlama paketlerini hedef ana bilgisayardan göstermiş gibi gösternesidir. Bu özellikle 113 numaralı bağlantı noktasından (identd) yaygındır. Hem sıfırlama sahtekarlığı hem de şeffaf proxy'ler genellikle bir hedef ağdaki her makinenin bu davranışını sergilediğini fark ederek tespit edilebilir - aksi takdirde kapalı gibi görünenler bile. Böyle bir saçmalık tespit ederseniz, sonuçlarınızı etkilememesi için bu bağlantı noktalarını taramanızdan çıkardığınızdan emin olun. Ayrıca tamamen farklı bir ağ konumundan da denemek isteyebilirsiniz. Hedefe ne kadar yakın olursanız, sonuçlar o kadar doğru olacaktır. Mükemmel bir durumda, hedefi her zaman bulunduğu aynı ağ segmentinden tararsınız.

When Nmap Guesses Wrong (Nmap Yanlış Tahmin Yaptığında)

Nmap zaman zaman yanlış olduğunu bildiğiniz bir işletim sistemi tahmini bildirir. Hatalar genellikle küçüktür (Linux 2.4.16 çalıştırılan bir makineyi "Linux kernel 2.4.8 - 2.4.15" olarak bildirmek gibi), ancak Nmap'in tamamen hatalı olduğu bildirilmiştir (web sunucunuza bir AppleWriter yazıcı olarak bildirmek gibi). Bu tür sorunlarla karşılaşığınızda (küçük veya büyük), lütfen bunları bildirin, böylece herkes faydalananabilir. Nmap DB'nin bu kadar kapsamlı olmasının tek nedeni, binlerce kullanıcının her birinin yeni bilgiler göndermek için birkaç dakika harc他已经完成。Lütfen bu talimatları izleyin:

Nmap'in yeni bir sürümüne sahip olmak ⇒ Nmap'in hangi sürümüne sahip olduğunu belirlemek için nmap -V'yi çalıştırın. Nmap'in mutlak en son sürümünü çalıştırmanız gerekmek (ancak bu ideal olacaktır), ancak sürümünüzün 4.20 veya daha yüksek olduğundan emin olun çünkü önceki sürümler tarafından üretilen eski stile değil, yalnızca ikinci nesil işletim sistemi parmak izlerine ihtiyacımız var. Nmap'in mevcut en son sürümünü <https://nmap.org/download.html> adresini ziyaret ederek öğrenebilirsiniz. Yükseltme yaparsanız, yanlış tanımlamanın zaten düzeltilmiş olduğunu görebilirsiniz.

Neyin çalıştığını bildiğinizden kesinlikle emin olun ⇒ Geçersiz "düzeltilmeler" OS DB'sini bozabilir. Uzak makinede tam olarak ne çalıştığını emin değilseniz, lütfen göndermeden önce öğrenin.

Parmak izi oluşturma ⇒ nmap -O -sV -T4 -d <hedef> komutunu çalıştırın, burada <hedef> söz konusu yanlış tanımlanmış sistemdir. Yanlış tanımlamanın hala mevcut olduğundan emin olmak için işletim sistemi algılama sonuçlarına bakın. Hedef sistemi IPv6 üzerinden tariyorsanız, -6 seçeneğini de ekleyin.

Ana işletim sistemi sonuçları için Nmap çıktısı (SADECE TAHMİN) diyorsa, sonuçların biraz yanlış olması beklenir. Bu durumda bir düzeltme göndermeyin.

Aksi takdirde, map komutunun OS fingerprint: satırını içeren sonuçlar üretmesi gerekiyor. Bunun altında parmak izi (her biri OS: ile başlayan bir dizi satır) bulunur.

İşletim sistemi algılamadan önce diğer ana bilgisayarlara karşı çalışıp çalışmadığını kontrol edin

Hedef ağ üzerinde farklı bir işletim sistemine sahip olduğunu bildiğiniz birkaç başka ana bilgisayarı taramayı deneyin. Düzgün bir şekilde algılanmazlarsa, belki de sistemler arasında paketleri bozan bazı ağ engelleri vardır.

Buraya kadar gelebildiyseniz ve hala gönderebiliyorsanız, ne mutlu size! Lütfen bilgileri <https://insecure.org/cgi-bin/submit.cgi?corr-os> adresine gönderin.

When Nmap Fails to Find a Match and Prints a Fingerprint (Nmap Bir Eşleşme Bulamadığında ve Parmak İzi Çıkardığında)

Nmap, işletim sistemi algılama koşullarının ideal görünüşünü ve yine de tam eşleşme bulamadığını tespit ettiğinde, aşağıdaki gibi bir mesaj yazdıracaktır:

```
No OS matches for host (If you know what OS is running on it, see
https://nmap.org/submit/).
TCP/IP fingerprint:
OS:SCAN(V=5.05BETA1=D=8/23%0T=22%CT=1%CU=42341%PV=N%DS=0%DC=L%G=Y%TM=4A91CB
OS:90%P=1686-pc-linux-gnu)SEQ(SP=C9%GCD=1%ISR=CF%TI=Z%CI=Z%II=I%TS=A)OPS(01
OS:=M400CST11NW5%02=M400CST11NW5%03=M400CNNT11NW5%04=M400CST11NW5%05=M400CS
OS:T11NW5%06=M400CST11)WIN(W1=8000%W2=8000%W3=8000%W4=8000%W5=8000%W6=8000)
OS:ECN(R=Y%DF=Y%T=40%W=8018%0=M400CNNSNW5%CC=N%0=)T1(R=Y%DF=Y%T=40%S=0%A=S+
OS:%F=AS%RD=0%Q=)T2(R=N)T3(R=Y%DF=Y%T=40%W=8000%S=0%A=S+F=AS%0=M400CST11NW
OS:5%RD=0%Q=)T4(R=Y%DF=Y%T=40%W=0%S=A%Z=F=R%0=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W
OS:=0%S=Z%A=S+F=AR%0=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%0=%RD=0%Q=)
OS:T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+F=AR%0=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%U
OS:N=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DF1=N%T=40%CD=S)
```

Lütfen tüm Nmap kullanıcısının faydalananabilmesi için parmak izini göndermeyi düşünün. Sadece bir ya da iki dakika sürer ve bir sonraki Nmap sürümü ile ana bilgisayarı taradığınızda o çirkin mesajı tekrar görmenize gerek kalmayabilir! Talimatlar için Nmap'in sağladığı URL'yi ziyaret etmeniz yeterlidir.

Nmap hiçbir eşleşme bulamazsa ve buna rağmen parmak izi yazdırırsa, koşullar ideal değildir. Parmak izini hata ayıklama modu veya XML çıktısı yoluyla elde etseniz bile, Nmap sizden istemedikçe (önceki örnekte olduğu gibi) lütfen göndermeyin.

Modifying the [nmap-os-db](#) Database Yourself (nmap-os-db Veritabanını Kendiniz Değiştirme)

İnsanlar genellikle bir parmak izini Nmap.Org'a göndermek yerine (veya buna ek olarak) kendilerinin entegre etmesini sorarlar. Bunun için ayrıntılı talimatlar veya komut dosyaları sunmasak da, "Bir Nmap Parmak İzini Anlamak" adlı bölümde yakından aşağı老旧后从 sona kesinlikle mümkün kılınır. Umarım bu sizin amaçlarınız için faydalı olur, ancak kendi referans parmak izi kreasyonlarınızı bize göndermenize gerek yoktur. Yalnızca web formundan ham konu parmak izi gönderimlerini entegre edebiliriz.

SOLUTION: Detect Rogue Wireless Access Points on an Enterprise Network (ÇÖZÜM: Kurumsal Ağdaki Sahte Kablosuz Erişim Noktalarını Tespit Etme)

Problem (Sorun)

Mobil cihazların ve ucuz emtia ağ ekipmanlarının yaygınlaşmasıyla birlikte şirketler, çalışanların ağlarını istenmeyen şekillerde genişlettiklerini giderek daha fazla fark etmektedir. En tehlikeli cihazlar arasında 802.11 kablosuz erişim noktaları (WAP'lar) yer almaktadır. Kullanıcılar, korunan kurumsal ağı otoparktaki veya yakındaki binalardaki potansiyel saldırganlara açlıklarını fark etmeden (veya umursamadan) dinlenme odasından çalışabilmek için odalarına 20 dolarlık bir WAP kurabilirler.

Bazı WAP kurulumları, saf kullanıcılar tarafından kurulanlardan bile daha kötüdür. Bir binanın güvenliğini ihlal etmek, bir saldırgan için uzaklardan bir ağ aracılığıyla kurumsal verilere erişmekten çok daha risklidir. Olay yerinde tutuklanma riski taşır. Bu nedenle saldırganların kompakt WAP'lar kurdukları bilinmektedir, böylece sokaktaki bir arabanın göreceli güvenliğinden istedikleri zaman ağa izinsiz girebilirler. Bir masanın altına bantlanmış ya da başka bir şekilde gizlenmiş bir WAP'in bir süre fark edilmesi pek olası değildir.

Bu çözümün odak noktası WAP'ları bulmak olsa da, aynı strateji hemen hemen her şeyi bulmak için kullanılabilir. Yeni bir yama uygulamak için tüm Cisco yönlendiricilerini ya da destek için ödeme yapmayı gerektirecek kadar sisteminiz olup olmadığını belirlemek için Solaris kutularını bulmanız gerekebilir.

Yetkisiz kablosuz cihazları bulmanın bir yolu, Kismet veya NetStumbler gibi bir kablosuz dinleyici ile alanı taramaktır. Diğer bir yaklaşım ise kablolu tarafı Nmap ile taramaktır. Şaşırtıcı olmayan bir şekilde, bu çözüm sadece ikinci yaklaşma odaklanmaktadır. Her teknik belirli WAP'ları gözden kaçırabilir, bu nedenle en iyi yaklaşım her ikisini de yapmak ve sonuçları birleştirmektir.

Solution (Çözüm)

A seçeneğini kullanarak tüm adres alanınızı tarayın. Taranan portları 1-85, 113, 443 ve 8080-8100 ile sınırlandırarak hızlandırırsınız. Bunlar çoğu WAP'ta hem açık hem de kapalı bir bağlantı noktası bulmalıdır, bu da işletim sistemi algılama doğruluğunu artırır. Ağınız birden fazla ethernet segmentine yayılıyorsa, her segmenti aynı segmentteki belirlenmiş bir makineden tarayın. Bu, taramayı hızlandırır (özellikle paralel olarak yapabileceğiniz için) ve ayrıca size her cihazın MAC adresini verir. Aynı segmentten tarama yapmak gizli cihazları tespit etmenizi de sağlar. Tüm portları filtrelenmiş bir WAP bile genellikle bir ARP isteği yanıt verecektir. Sonuçlar en azından normal ve XML formatlarında kaydedilmelidir, bu nedenle -oA kullanabilirsiniz. Bölüm 6, Nmap Performansını Optimize Etme'de açıklanan tüm performans artırıcı seçenekleri göz önünde bulundurun. Performans seçenekleri için iyi ve nispeten güvenli bir başlangıç -T4 --min-hostgroup 50 --max-rtt-timeout 1000ms --initial-rtt-timeout 300ms --max-retries 3 --host-timeout 20m --max-scan-delay 1000ms'dır. Tüm bunları aşağıdaki gibi bir komut için bir araya getirin:

```
nmap -A -oA ~/nmap-logs/wapscan -p  
1-85,113,443,8080-8100 -T4 --min-hostgroup 50 --max-rtt-timeout 1000ms  
--initial-rtt-timeout 300ms --max-retries 3 --host-timeout 20m  
--max-scan-delay 1000ms
```

<target_network>

Tarama tamamlandığında, WAP özelliklerini arayın. Birkaç yüzden daha az canlı ana bilgisayardan oluşan bir ağda, en iyi seçeneğiniz her birine ayrı ayrı bakmaktır. Daha büyük ağlar için, muhtemelen görevi otomatikleştirmeniz gerekecektir. Tek tek özelliklerin aranması grep ile yapılabilir, ancak XML çıktısını analiz eden bir Perl betiği tercih edilir. Nmap XML çıktısını ayırtmak için Nmap::Scanner ve Nmap::Parser gibi mevcut modüller sayesinde bu oldukça kolaydır. Örnekler için "Perl ile XML Çıktısını Manipüle Etme" bölümünü bakınız.

Adayların bir listesini belirledikten sonra, normal Nmap çıktı dosyasını açmak ve yanlış pozitifleri ortadan kaldırmak için her birini incelemek muhtemelen en iyisidir. Örneğin, bir Linksys cihazı, herhangi bir kablosuz

işlevi olmayan düz anahtarlarından biri olsa bile olası bir WAP olarak işaretlenebilir.

WAP'ları bulduktan sonra, sıra onları takip etmeye gelir. Bu genellikle fiziksel ethernet bağlantı noktası numaraları için bağlandıkları anahtar sorgulanarak yapılabilir.

WAP Characteristics (WAP Özellikleri)

Şimdi sıra aranacak WAP özelliklerini tartışmaya geldi. Bunları anlamak, manuel incelemeler için veya WAP bulucu komut dosyasını başka bir şey aramak için değiştirmek için yararlıdır. Örnek 8.11'deki tipik bir WAP taramasına bakarak muhtemelen bunların çoğunu hemen göreceksiniz.

Örnek 8.11. Bir tüketici WAP'ına karşı tarama sonuçları

```
# nmap -A -v wap.nmap.org
Starting Nmap ( https://nmap.org )
Nmap scan report for wap.nmap.org (192.168.0.6)
Not shown: 999 closed ports
PORT      STATE SERVICE VERSION
80/tcp    open  http    Netgear MR-series WAP (MR814; Embedded HTTPD 1.00)
MAC Address: 00:09:5B:3F:7D:5E (Netgear)
Device type: WAP
Running: Compaq embedded, Netgear embedded
OS details: WAP: Compaq iPAQ Connection Point or Netgear MR814
Service Info: Device: WAP

Nmap done: 1 IP address (1 host up) scanned in 10.90 seconds
Raw packets sent: 1783 (75.786KB) | Rcvd: 1686 (77.552KB)
```

Bu cihaz bir WAP olduğuna dair birçok bariz ipucu (Cihaz türü: WAP oldukça bariz) ve bazı daha ince ipuçları gösteriyor. Ancak WAP'ları keşfetmek her zaman o kadar kolay değildir. Bu bölümde WAP özelliklerinin bir listesi sunulmakta, en güçlü olanlardan başlanmakta ve yanlış pozitif sonuç verme olasılığı daha yüksek olan sezgisel yöntemlerle sonlandırılmaktadır. Listelenen her bir özelliğe Nmap XML çıktısında nerede bulunacağını gösteren bir XPath ifadesi eşlik etmektedir. Bu güvenlikle ilgili olduğundan, hepsini denemenizi ve yanlış pozitifleri manuel olarak kaldırmanızı öneririm.

TCP/IP parmak izi cihaz tipi ⇒ "Cihaz ve işletim sistemi sınıflandırması (Sınıf çizgileri)" bölümünde açıklandığı gibi, her referans parmak izinin kendisiyle ilişkili en az bir sınıflandırması (cihaz türünü içeren) vardır. WAP'lar çok tartışmalı olduğundan, birden fazla ürün uyabileceği durumlarda bunu kullanmaya (veya iki sınıflandırma vermeye) çalışıyoruz. Dolayısıyla D-Link DI-624 kablosuz geniş bant yönlendirici gibi cihazlar anahtar veya yönlendirici yerine WAP olarak sınıflandırılır. Cihaz türü, XML çıktısında /nmaprun/host/os/osclass/@type XPath ifadesi kullanılarak bulunabilir. (Yani, kök nmaprun öğesinin içindeki ana bilgisayar öğelerinden herhangi birinin os öğesinin osclass öğesinin tür niteliği).

TCP/IP parmak izi ayrıntıları ⇒ Kablosuz özelliğine sahip cihazların cihaz tipi WAP olarak sınıflandırılması gerekse de, emin olmak için ayrıntılı işletim sistemi açıklamasında kablosuz veya wap gibi terimleri aramakta fayda vardır. Açıklama XML çıktısında /nmaprun/host/os/osmatch/@name içindedir.

Sürüm algılama cihaz tipi ⇒ Sürüm tespiti de cihaz türlerini belirlemeye çalışır, ancak IP yiğini yerine hedefin çalışan hizmetlerinin parmak izini alarak. nmaprun/host/ports/port/service/@devicetype adresinde bulunan XML devicetype özniteliğinin WAP olup olmadığını kontrol edin. Tamamen güvenli olması için, /nmaprun/host/ports/port/service/@extrainfo alanını wap veya wireless alt dizeleri için kontrol etmek faydalı olacaktır.

Satıcı (MAC adresi, TCP/IP parmak izi ve sürüm tespitinden) ⇒ Bazı satıcılar, ofis ağlarına gizlice girme olasılığı en yüksek olan düşük maliyetli tüketici ağ cihazlarını üretme konusunda uzmanlaşmıştır. Örnekler Linksys, Netgear, Belkin, SMC, D-Link, Motorola, Trendnet, Zyxel ve Gateway'dir. Bu satıcıları MAC adresi arama (XML çıktısında /nmaprun/host/address/@vendor adresindedir), işletim sistemi algılama (XML çıktısında /nmaprun/host/os/osclass/@vendor adresindedir) veya sürüm algılama (XML çıktısında /nmaprun/host/ports/port/service/@product adresindedir) sonuçlarına göre kontrol edebilirsiniz. Satıcıyı

alanların bir alt dizesi olarak aradığınızdan emin olun, çünkü alan kuruluş türü (örn. Inc.) veya başka bilgiler içerebilir.

Bu test birçok yanlış pozitif sonuca yol açabilir. Masaüstü makinelerinize Netgear NIC'leri koymak gibi yetkili cihazlar için yoğun olarak bir satıcı kullanıyorsanız, bu satıcıyı kaldırmanız ve komut dosyasını yeniden çalıştırmanız gerekebilir.

Hostname (Ana bilgisayar adı) ⇒ Wap, kablosuz veya havaalanı gibi terimler için ana bilgisayar adlarını (ters DNS çözümlemesi) kontrol etmekten zarar gelmez. Bunlar XML çıktısında /nmaprun/host/hostnames/hostname/@name adresinde bulunabilir. Yönetici olmayan çalışanlar DNS adlarını nadiren değiştirir, ancak bu, kalem testçileri, yeni yöneticiler ve yetkili erişim noktaları arayan yeni bir ağı tarayan diğerleri için yararlı olabilir.

Chapter 9. Nmap Scripting Engine (Bölüm 9. Nmap Komut Dosyası Motoru)

- Introduction (Giriş)
- Usage and Examples (Kullanım ve Örnekler)
 - Script Categories (Komut Dosyası Kategorileri)
 - Script Types and Phases (Komut Dosyası Türleri ve Aşamaları)
 - Command-line Arguments (Komut Satırı Bağımsız Değişkenleri)
 - Script Selection (Komut Dosyası Seçimi)
 - Arguments to Scripts (Komut Dosyalarına Bağımsız Değişkenler)
 - Complete Examples (Tam Örnekler)
- Script Format (Komut Dosyası Biçimi açıklaması)
 - `description` Field (Alan kategorileri)
 - `categories` Field ()
 - `author` Field (Alan yazarı)
 - `license` Field (Alan lisansı)
 - `dependencies` Field (Alan bağımlılıkları)
 - Rules (Alan Kuralları)
 - Action (Eylem)
 - Environment Variables (Ortam Değişkenleri)
- Script Language (Komut Dosyası Dili)
 - Lua Base Language (Lua Temel Dil)
- NSE Scripts (NSE Komut Dosyaları)
- NSE Libraries (NSE Kütüphaneleri)
 - List of All Libraries (Tüm Kütüphanelerin Listesi)
 - Hacking NSE Libraries (NSE Kütüphanelerini Hackleme)

- Adding C Modules to Nselib (Nselib'e C Modülleri Ekleme)
- Nmap API (Nmap API)
 - Information Passed to a Script (Bir Komut Dosyasına Aktarılan Bilgiler)
 - Network I/O API (Ağ G/Ç API)
 - Connect-style network I/O (Bağlan-stil ağ G/Ç)
 - Raw packet network I/O (Ham paket ağ G/Ç)
 - Structured and Unstructured Output (Yapılandırılmış ve Yapılandırılmamış Çıktı)
 - Exception Handling (İstisna İşleme)
 - The Registry (Kayıt Defteri)
- Script Writing Tutorial (Komut Dosyası Yazma Eğitimi)
 - The Head (Baş)
 - The Rule (Kural)
 - The Action (Eylem)
- Writing Script Documentation (NSEDoc) (Komut Dosyası Yazma Belgeleri (NSEDoc))
 - NSE Documentation Tags (NSE Belgeleri Etiketler)
- Script Parallelism in NSE (Komut Dosyası NSE'de Paralellik)
 - Worker Threads (Çalışan İş Parçacıkları)
 - Mutexes (Muteksler)
 - Condition Variables (Koşul Değişkenleri)
 - Collaborative Multithreading (İşbirlikçi Çoklu İş Parçacığı)
 - The base thread (Temel iplik)
- Version Detection Using NSE (NSE Kullanarak Sürüm Algılama)
- Example Script: `finger` (Örnek Kod: parmak)
- Implementation Details (Uygulama Detayları)
 - Initialization Phase (Başlatma Aşaması)
 - Script Scanning (Komut Dosyası Taraması)

Introduction (Giriş)

Nmap Scripting Engine (NSE), Nmap'in en güçlü ve esnek özelliklerinden biridir. Kullanıcıların çok çeşitli ağ görevlerini otomatikleştirmek için basit komut dosyaları yazmasına (ve paylaşmasına) olanak tanır. Bu komut dosyaları daha sonra Nmap'ten beklediğiniz hız ve verimlilikle paralel olarak yürütülür. Kullanıcılar Nmap ile birlikte dağıtılan ve giderek büyüyen komut dosyalarına güvenebilir veya özel ihtiyaçlarını karşılamak için kendi komut dosyalarını yazabilirler.

NSE'yi aşağıdaki görevleri göz önünde bulundurarak çok yönlü olacak şekilde tasarladık:

Ağ keşfi ⇒ Bu Nmap'in ekmek ve tereyağıdır. Örnekler arasında hedef etki alanına dayalı whois verilerini aramak, sahipliği belirlemek için hedef IP için ARIN, RIPE veya APNIC'yi sorgulamak, açık bağlantı noktalarında identd aramaları yapmak, SNMP sorguları ve mevcut NFS/SMB/RPC paylaşımlarını ve hizmetlerini listelemek yer alır.

Daha gelişmiş sürüm algılama ⇒ Nmap sürüm tespit sistemi (Bölüm 7, Hizmet ve Uygulama Sürüm Tespit), prob ve düzenli ifade imza tabanlı eşleştirme sistemi aracılığıyla binlerce farklı hizmeti tanıyalabilir, ancak her şeyi tanıymaz. Örneğin, Skype v2 hizmetini tanımlamak için iki bağımsız sonda gerekir, bu da sürüm algılamanın üstesinden gelebilecek kadar esnek değildir. Nmap ayrıca birkaç yüz farklı topluluk adını kaba kuvvetle denerse daha fazla SNMP hizmetini tanıyalabilir. Bu görevlerin hiçbir geleneksel Nmap sürüm tespiti için uygun değildir, ancak her ikisi de NSE ile kolayca gerçekleştirilebilir. Bu nedenlerden dolayı, sürüm tespiti artık bazı zor hizmetleri yerine getirmek için varsayılan olarak NSE'yi çağırmaktadır. Bu konu "NSE Kullanarak Sürüm Tespit" başlıklı bölümde açıklanmaktadır.

Güvenlik açığı tespiti ⇒ Yeni bir güvenlik açığı keşfedildiğinde, kötü adamlar bunu yapmadan önce savunmasız sistemleri belirlemek için genellikle ağlarını hızlı bir şekilde taramak istersiniz. Nmap kapsamlı bir güvenlik açığı tarayıcısı olmasa da, NSE zorlu güvenlik açığı kontrollerinin bile üstesinden gelebilecek kadar güçlündür. Heartbleed hatası dünya çapında yüz binlerce sistemi etkilediğinde, Nmap'in geliştiricileri 2 gün içinde ssl-heartbleed tespit betiği ile yanıt verdi. Birçok güvenlik açığı tespit betiği halihazırda mevcuttur ve yazıldıkça daha fazlasını dağıtmayı planlıyoruz.

Arka kapı tespiti ⇒ Birçok saldırgan ve bazı otomatik solucanlar daha sonra yeniden giriş yapabilmek için arka kapıları bırakır. Bunlardan bazıları Nmap'in düzenli ifade tabanlı sürüm tespiti ile tespit edilebilir, ancak daha karmaşık solucanlar ve arka kapılar, güvenilir bir şekilde tespit etmek için NSE'nin gelişmiş yeteneklerini gerektirir. NSE, SMB'deki Double Pulsar NSA arka kapısını ve UnreallRCd, vsftpd ve ProFTPD'nin arka kapıları sürümlerini tespit etmek için kullanılmıştır.

Güvenlik açığı istismarı ⇒ Genel bir komut dosyası dili olarak NSE, güvenlik açıklarını bulmaktan ziyade bunlardan yararlanmak için bile kullanılabilir. Nmap'i Metasploit gibi bir istismar çerçevesine dönüştürmeyi planlamıyor olsa da, özel istismar komut dosyaları ekleme yeteneği bazı insanlar (özellikle sizma testçileri) için değerli olabilir.

Listelenen bu öğeler ilk hedeflerimizdi ve Nmap kullanıcılarının NSE için daha da yaratıcı kullanımlar bulmasını bekliyoruz.

Komut dosyaları, sürüm 5.3 olan gömülü Lua programlama dilinde yazılmıştır. Dilin kendisi Programming in Lua, Fourth Edition ve Lua 5.2 Reference Manual kitaplarında iyi bir şekilde belgelenmiştir. Lua 5.3 için güncellenen referans kılavuzu, Programming in Lua'nın ilk baskısı gibi çevrimiçi olarak da ücretsiz olarak mevcuttur. Bu mükemmel genel Lua programlama referanslarının mevcudiyeti göz önüne alındığında, bu belge yalnızca Nmap'in komut dosyası motoruna özgü yönleri ve uzantıları kapsamaktadır.

NSE, -sC seçeneği (veya özel bir komut dosyası kümesi belirtmek isterseniz --script) ile etkinleştirilir ve sonuçlar Nmap normal ve XML çıktısına entegre edilir.

Tipik bir komut dosyası taraması Örnek 9.1'de gösterilmektedir. Bu örnekte çıktı üreten hizmet betikleri, sistemin RSA ve DSA SSH anahtarlarını sağlayan ssh-hostkey ve mevcut hizmetleri listelemek için portmapper'i sorgulayan rpcinfo'dur. Bu örnekte çıktı üreten tek ana bilgisayar betiği, SMB sunucularından çeşitli bilgiler toplayan smb-os-discovery'dir. Nmap tüm bu bilgileri saniyenin üçte birinde keşfetmiştir.

Örnek 9.1. Tipik NSE çıktısı

```
# nmap -sC -p22,111,139 -T4 localhost
Starting Nmap ( https://nmap.org )
Nmap scan report for flag (127.0.0.1)
PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-hostkey: 1024 b1:36:0d:3f:50:dc:13:96:b2:6e:34:39:0d:9b:la:38 (DSA)
| 2048 77:d0:20:1c:44:1f:87:a0:30:aa:85:cf:e8:ca:4c:11 (RSA)
111/tcp   open  rpcbind
| rpcinfo:
| 100000 2,3,4  111/udp  rpcbind
| 100024 1      56454/udp status
| 100000 2,3,4  111/tcp   rpcbind
139/tcp   open  netbios-ssn

Host script results:
| smb-os-discovery: Unix
| LAN Manager: Samba 3.0.31-0.fc8
|_Name: WORKGROUP

Nmap done: 1 IP address (1 host up) scanned in 0.33 seconds
```

NSE ile ilgili 38 dakikalık bir tanıtım videosu <https://nmap.org/presentations/BHDC10/> adresinde mevcuttur. Bu sunum Fyodor ve David Fifield tarafından 2010 yılında Defcon ve Black Hat Briefinglerinde yapılmıştır.

Usage and Examples (Kullanım ve Örnekler)

NSE verimlilik için karmaşık bir uygulamaya sahip olsa da, kullanımı çarpıcı derecede kolaydır. En yaygın komut dosyalarını etkinleştirmek için -sC seçeneğini belirtmeniz yeterlidir. Ya da kategorileri, betik dosya adlarını ya da çalıştmak istediğiniz betiklerle dolu dizinlerin adını vererek çalıştırılacak kendi betiklerinizi seçmek için --script seçeneğini belirtin. Bazı komut dosyalarını --script-args ve --script-args-file seçenekleriyle argüman sağlayarak özelleştirebilirsiniz. --script-help, seçilen her bir betığın ne yaptığına dair bir açıklama gösterir. Kalan iki seçenek, --script-trace ve --script-updatedb, genellikle yalnızca betik hata ayıklama ve geliştirme için kullanılır. Komut dosyası taraması -A (agresif tarama) seçeneğinin bir parçası olarak da dahil edilmiştir.

Komut dosyası taraması normalde bir bağlantı noktası taraması ile birlikte yapılır, çünkü komut dosyaları tarama tarafından bulunan bağlantı noktası durumlarına bağlı olarak çalıştırılabilir veya çalıştırılmayabilir. -sn seçeneği ile port taraması yapmadan, sadece host keşfi yaparak script taraması yapmak mümkündür. Bu durumda sadece ana bilgisayar komut dosyaları çalıştırılabilir. Ne bir ana bilgisayar keşfi ne de bir bağlantı noktası taraması ile bir komut dosyası taraması çalıştırmak için -Pn -sn seçeneklerini -sC veya --script ile birlikte kullanın. Her konak varsayılarak ve yine de sadece konak betikleri çalıştırılacaktır. Bu teknik, whois-ip gibi yalnızca uzak sistemin adresini kullanan ve açık olmasını gerektirmeyen betikler için kullanılabilir.

Komut dosyaları bir korumalı alanda çalıştırılmaz ve bu nedenle yanlışlıkla veya kötü niyetle sisteminize zarar verebilir veya gizliliğınızı ihlal edebilir. Yazarlarına güvenmediğiniz veya komut dosyalarını kendiniz dikkatlice denetlemediğiniz sürece üçüncü taraflardan gelen komut dosyalarını asla çalıştmayın.

Script Categories (Senaryo Kategorileri)

NSE komut dosyaları ait oldukları kategorilerin bir listesini tanımlar. Şu anda tanımlı kategoriler auth, broadcast, brute, default, discovery, dos, exploit, external, fuzzer, intrusive, malware, safe, version ve vuln'dur. Kategori adları büyük/küçük harfe duyarlı değildir. Aşağıdaki listede her bir kategori açıklanmaktadır.

auth → Bu betikler hedef sistemdeki kimlik doğrulama bilgileriyle (veya bunları atlayarak) ilgilendir. Örnek olarak x11-access, ftp-anon ve oracle-enum-users verilebilir. Kimlik bilgilerini belirlemek için kaba kuvvet saldıruları kullanan komut dosyaları bunun yerine kaba kuvvet kategorisine yerleştirilir.

broadcast → Bu kategorideki betikler genellikle yerel ağıda yayın yaparak komut satırında listelenmeyen ana bilgisayarları keşfeder. Bu komut dosyalarının keşfettikleri ana bilgisayarları otomatik olarak Nmap tarama kuyruğuna eklemesine izin vermek için newtargets komut dosyası bağımsız değişkenini kullanın.

brute ⇒ Bu betikler, uzak bir sunucunun kimlik doğrulama bilgilerini tahmin etmek için kaba kuvvet saldırısını kullanır. Nmap, http-brute, oracle-brute, snmp-brute, vb. dahil olmak üzere düzinelerce protokolü kaba kuvvetle zorlamak için komut dosyaları içerir.

default ⇒ Bu betikler varsayılan kümedir ve --script ile betikleri listelemek yerine -sC veya -A seçenekleri kullanıldığında çalıştırılır. Bu kategori de diğerleri gibi --script=default kullanılarak açıkça belirtilebilir. Bir betığın varsayılan olarak çalıştırılıp çalıştırılmayacağına karar verilirken birçok faktör göz önünde bulundurulur:

Speed (Hız) ⇒ Varsayılan tarama hızlı bir şekilde tamamlanmalıdır, bu da kaba kuvvet kimlik doğrulama kırıcılarını, web örümceklerini ve tek bir hizmeti taramak için dakikalar veya saatler sürebilen diğer komut dosyalarını hariç tutar.

Usefulness (Kullanışlılık) ⇒ Varsayılan taramaların değerli ve eyleme geçirilebilir bilgiler üretmesi gereklidir. Komut dosyası yazarı bile ortalama bir ağ veya güvenlik uzmanın çıktılığını neden değerli bulacağını açıklamakta zorlanıyorsa, komut dosyası varsayılan olarak çalıştırılmamalıdır.

Verbosity ⇒ Nmap çıktısı çok çeşitli amaçlar için kullanılır ve okunabilir ve özlü olması gereklidir. Sık sık sayfalar dolusu çıktı üreten bir betik varsayılan kategorisine eklenmemelidir. Raporlanacak önemli bir bilgi olmadığından, NSE komut dosyaları (özellikle varsayılan olanlar) hiçbir şey döndürmemelidir. Belirsiz bir güvenlik açığını kontrol etmek, yalnızca bu güvenlik açığı keşfedildiğinde çıktı ürettiği sürece varsayılan olarak sorun olmamalıdır.

Reliability (Güvenilirlik) ⇒ Birçok komut dosyası, hedef ana bilgisayar veya hizmet hakkında sonuçlara ulaşmak için sezgisel yöntemler ve bulanık imza eşleştirmesi kullanır. Örnek olarak sniffer-detect ve sql-injection verilebilir. Betik genellikle yanlışsa, sıradan kullanıcıların kafasını karıştırabileceği veya yanlış yönlendirebileceği varsayılan kategoriye ait değildir. Bir betiği veya kategoriyi doğrudan belirten kullanıcılar genellikle daha ileri düzeydedir ve muhtemelen betığın nasıl çalıştığını veya en azından belgelerini nerede bulacaklarını bilirler.

Intrusiveness (Girişkenlik) ⇒ Bazı komut dosyaları çok müdahaleciidir çünkü uzaktaki sistemde önemli kaynaklar kullanırlar, sistemi veya hizmeti çökertmeleri veya uzaktaki yöneticiler tarafından saldırı olarak algılanmaları muhtemeldir. Bir komut dosyası ne kadar müdahaleci ise, varsayılan kategori için o kadar az uygundur. Varsayılan komut dosyaları da neredeyse her zaman güvenli kategorisindedir, ancak bazen sadece hafif derecede müdahaleci oldukları ve diğer faktörlerde iyi puan aldıklarımda müdahaleci komut dosyalarına varsayılan olarak izin veririz.

Privacy (Gizlilik) ⇒ Bazı betikler, özellikle de daha sonra açıklanacak olan harici kategoridekiler, doğaları gereği üçüncü taraflara bilgi ifşa eder. Örneğin, whois betiği hedef IP adresini bölgelik whois kayıtlarına ifşa etmelidir. Ayrıca, hedef SSH ve SSL anahtar parmak izlerini İnternet zayıf anahtar veritabanlarına karşı kontrol eden komut dosyaları eklemeyi de düşündük (ve buna karşı karar verdik). Bir komut dosyası ne kadar mahremiyete müdahale ederse, varsayılan kategori eklemeye için o kadar az uygun olur.

Bu kriterlerin her biri için kesin eşik değerlerimiz yoktur ve birçoğu özneldir. Bir betığın öntanımlı kategorisine yükseltildip yükseltilmeyeceğine karar verilirken tüm bu faktörler birlikte değerlendirilir. Birkaç varsayılan betik identd-owners (identd kullanarak uzak hizmetleri çalıştıran kullanıcı adını belirler), http-auth (kimlik doğrulama gerektiren web sitelerinin kimlik doğrulama şemasını ve alanını alır) ve ftp-anon (bir FTP sunucusunun anonim erişime izin vermediğini test eder).

discovery ⇒ Bu betikler genel kayıtları, SNMP etkin aygıtları, dizin hizmetlerini ve benzerlerini sorgulayarak ağ hakkında daha fazla bilgiyi aktif olarak keşfetmeye çalışır. Örnekler arasında html-title (web sitelerinin kök yolunun başlığını alır), smb-enum-shares (Windows paylaşımlarını numaralandırır) ve snmp-sysdescr (SNMP aracılığıyla sistem ayrıntılarını çıkarır) yer almır.

dos ⇒ Bu kategorideki komut dosyaları hizmet reddine neden olabilir. Bazen bu, bir hizmet reddi yöntemine karşı güvenlik açığını test etmek için yapılır, ancak daha yaygın olarak geleneksel bir güvenlik açığını test etmenin istenmeyen ancak gerekli bir yan etkisidir. Bu testler bazen savunmasız hizmetleri çökertebilir.

exploit (İstismar) ⇒ Bu komut dosyaları bazı güvenlik açıklarından aktif olarak yararlanmayı amaçlar. Örnek olarak jdwp-exec ve http-shellshock verilebilir.

external (Dış) ⇒ Bu kategorideki komut dosyaları üçüncü taraf bir veritabanına veya başka bir ağ kaynağına veri gönderebilir. Bunun bir örneği, hedefin adresi hakkında bilgi edinmek için whois sunucularına bağlantı kurulan whois-ip'dir. Üçüncü taraf veritabanı operatörlerinin onlara gönderdiğiniz her şeyi kaydetme olasılığı her zaman vardır, bu da çoğu durumda IP adresinizi ve hedefin adresini içerecektir.Çoğu komut dosyası kesinlikle tarama bilgisayarı ve istemci arasındaki trafiği içerir; bunu yapmayanlar bu kategoride yer alır.

fuzzer ⇒ Bu kategori, sunucu yazılımına her pakette beklenmedik veya rastgele alanlar göndermek için tasarlanmış komut dosyalarını içerir. Bu teknik, yazılımdaki keşfedilmemiş hataları ve güvenlik açıklarını bulmak için yararlı olsa da, hem yavaş bir süreçtir hem de bant genişliği yoğundur. Bu kategorideki bir betik örneği, sunucu çökene veya kullanıcı tarafından belirlenen bir zaman sınırı geçene kadar bir DNS sunucusunu hafif kusurlu alan istekleriyle bombalayan dns-fuzz'dır.

intrusive (Müdaheleci) ⇒ Bunlar, hedef sistemi çökertme, hedef ana bilgisayarda önemli kaynakları kullanma (bant genişliği veya CPU süresi gibi) veya hedefin sistem yöneticileri tarafından kötü niyetli olarak algılanma riskleri çok yüksek olduğu için güvenli kategoride sınıflandırılamayan komut dosyalarıdır. Örnek olarak http-open-proxy (hedef sunucuya HTTP proxy'si olarak kullanmaya çalışır) ve snmp-brute (public, private ve cisco gibi ortak değerler göndererek bir cihazın SNMP topluluk dizesini tahmin etmeye çalışır) verilebilir. Bir komut dosyası özel sürüm kategorisinde olmadığı sürece, güvenli veya müdaheleci olarak kategorize edilmelidir.

malware ⇒ Bu betikler hedef platforma kötü amaçlı yazılım veya arka kapı bulaşıp bulaşmadığını test eder. Örnekler arasında, olağanüstü bağlantı noktası numaralarında çalışan SMTP sunucularını izleyen smtp-strangeport ve bir sorgu almadan önce sahte bir cevap veren identd spoofing daemon'larını tespit eden auth-spoof yer alır. Bu davranışların her ikisi de genellikle kötü amaçlı yazılım bulaşmalarıyla ilişkilidir.

safe ⇒ Hizmetleri çökertmek, büyük miktarda ağ bant genişliği veya diğer kaynakları kullanmak veya güvenlik açıklarından yararlanmak için tasarlanmamış komut dosyaları güvenli olarak kategorize edilir. Bunların uzaktaki yöneticileri rahatsız etme olasılığı daha düşüktür, ancak (diğer tüm Nmap özelliklerinde olduğu gibi) hiçbir zaman olumsuz tepkilere neden olmayacaklarını garanti edemeyiz. Bunların çoğu genel ağ keşfi gerçekleştirir. Örnekler ssh-hostkey (bir SSH ana bilgisayar anahtarını alır) ve html-title (bir web sayfasından başlığı alır). Sürüm kategorisindeki betikler güvenliğe göre kategorize edilmemiştir, ancak güvenli kategorisinde olmayan diğer betikler müdaheleci kategorisine yerleştirilmelidir.

version (Versiyon) ⇒ Bu özel kategorideki komut dosyaları sürüm algılama özelliğinin bir uzantısıdır ve açıkça seçilemezler. Yalnızca sürüm algılama (-sV) istendiğinde çalıştırılmak üzere seçilirler. Çıktıları sürüm algılama çıktısından ayırt edilemez ve hizmet veya ana bilgisayar komut dosyası sonuçları üretmezler. Örnekler skypev2-version, pptp-version ve iax2-version'dır.

vuln ⇒ Bu komut dosyaları bilinen belirli güvenlik açıklarını kontrol eder ve genellikle yalnızca bulunurlarsa sonuçları bildirir. Örnek olarak realvnc-auth-bypass ve afp-path-vuln verilebilir.

Script Types and Phases (Senaryo Türleri ve Aşamaları)

NSE, alındıkları hedeflerin türü ve çalıştırıldıkları tarama aşaması ile ayırt edilen dört tür komut dosyasını destekler. Bireysel komut dosyaları birden fazla işlem türünü destekleyebilir.

Prerule scripts (Kural öncesi komut dosyaları) ⇒ Bu komut dosyaları Nmap'in tarama aşamalarından önce çalışır, bu nedenle Nmap henüz hedefleri hakkında herhangi bir bilgi toplamamıştır. DHCP ve DNS SD sunucularını sorgulamak için ağ yayını istekleri gerçekleştirmek gibi belirli tarama hedeflerine bağlı olmayan görevler için yararlı olabilirler. Bu komut dosyalarından bazıları Nmap'in taraması için yeni hedefler oluşturabilir (yalnızca newtargets NSE bağımsız değişkenini belirtirseniz). Örneğin, dns-zone-transfer bir bölge aktarım isteği kullanarak bir etki alanındaki IP'lerin bir listesini alabilir ve ardından bunları otomatik olarak Nmap'in

tarama hedef listesine ekleyebilir. Ön kural komut dosyaları bir ön kural işlevi içерerek tanımlanabilir ("Kurallar" adlı bölüme bakın).

Host scripts (Ev sahibi komut dosyaları) ⇒ Bu aşamadaki komut dosyaları, Nmap hedef ana bilgisayara karşı ana bilgisayar bulma, bağlantı noktası tarama, sürüm algılama ve işletim sistemi algılama işlemlerini gerçekleştirdikten sonra Nmap'in normal tarama işlemi sırasında çalışır. Bu tür bir komut dosyası, ana bilgisayar kuralı işleviyle eşleşen her hedef ana bilgisayara karşı bir kez çağrılar. Örnekler, bir hedef IP için sahiplik bilgilerini arayan whois-ip ve parçalanma gerektirmeden hedefe ulaşabilecek maksimum IP paketi boyutunu belirlemeye çalışan path-mtu'dur.

Service scripts (Hizmet komut dosyaları) ⇒ Bu betikler hedef ana bilgisayarda dinlenen belirli hizmetlere karşı çalışır. Örneğin, Nmap web sunucularına karşı çalıştırmak için 15'ten fazla http hizmet komut dosyası içerir. Bir ana bilgisayarın birden fazla bağlantı noktasında çalışan web sunucuları varsa, bu komut dosyaları birden çok kez çalışabilir (her bağlantı noktası için bir tane). Bunlar en yaygın Nmap komut dosyası türüdür ve bir komut dosyasının hangi algılanan hizmetlere karşı çalıştırılacağına karar vermek için bir portrule işlevi içermeleri ile ayırt edilirler.

Portrule scripts (Postrule komut dosyaları) ⇒ Bu komut dosyaları Nmap tüm hedeflerini taradıktan sonra çalışır. Nmap çıktısını biçimlendirmek ve sunmak için yararlı olabilirler. Örneğin, ssh-hostkey en çok SSH sunucularına bağlanan, açık anahtarlarını bulan ve bunları yazdırın servis (portrule) betiği ile bilinir. Ancak aynı zamanda taranan tüm ana bilgisayarlar arasında yinelenen anahtarları kontrol eden ve ardından bulunanları yazdırın bir postrule içerir. Postrule komut dosyasının bir başka potansiyel kullanımı da Nmap çıktısının ters indeksini yazdırırmaktır; her bir ana bilgisayardaki hizmetleri listelemek yerine hangi ana bilgisayarların belirli bir hizmeti çalıştığını gösterir. Postrule komut dosyaları bir postrule fonksiyonu içерerek tanımlanır.

Birçok komut dosyası potansiyel olarak hem kural öncesi hem de kural sonrası komut dosyası olarak çalışabilir. Bu gibi durumlarda, tutarlılık için bir kural öncesi kullanmanızı öneririz.

Command-line Arguments (Komut Satırı Bağımsız Değişkenleri)

Bunlar, komut dosyası taramasına özgü beş komut satırı argümanıdır:

`-sC` ⇒ Varsayılan betik kümesini kullanarak bir betik taraması gerçekleştirir. Bu --script=default ile eşdeğerdir. Bu varsayılan kategorideki bazı komut dosyaları müdahaleci olarak kabul edilir ve izinsiz olarak hedef ağıda çalıştırılmamalıdır.

`--script <filename> | <category> | <directory> | <expression> [...]` ⇒ Virgülle ayrılmış dosya adları, komut dosyası kategorileri ve dizinler listesini kullanarak bir komut dosyası taraması çalıştırır. Listedeki her öğe, daha karmaşık bir komut dosyası kümesini tanımlayan bir Boolean ifadesi de olabilir. Her öğe önce bir ifade, sonra bir kategori ve son olarak da bir dosya veya dizin adı olarak yorumlanır. all özel bağımsız değişkeni, Nmap'in komut dosyası veritabanındaki her komut dosyasını çalışmaya uygun hale getirir. NSE istismarlar, kaba kuvvet kimlik doğrulama kırıcıları ve hizmet reddi saldıruları gibi tehlikeli betikler içerebileceğinden all argümanı dikkatli kullanılmalıdır.

Komut dosyası ifade listesindeki her bir öğenin önüne + karakteri eklenderek verilen komut dosyası(ları) portrule veya hostrule işlevlerindeki koşullardan bağımsız olarak çalışmaya zorlanabilir. Bu genellikle yalnızca özel durumlarda ileri düzey kullanıcılar tarafından yapılır. Örneğin, bazıları standart olmayan portlarda çalışan bir grup MS SQL sunucusu üzerinde yapılandırma incelemesi yapmak isteyebilirsiniz. Nmap'in ms-sql hizmetini tanıması için kapsamlı sürüm algılama (-sV --version-all) çalıştırarak Nmap taramasını yavaşlatmak yerine, --script +ms-sql-config komut dosyasını belirterek ms-sql-config komut dosyasını hedeflenen tüm ana bilgisayarlara ve bağlantı noktalarına karşı çalışmaya zorlayabilirsınız.

Dosya ve dizin adları göreli veya mutlak olabilir. Mutlak isimler doğrudan kullanılır. Göreceli yollar, bulunana kadar aşağıdaki yerlerin her birinin komut dosyaları alt dizininde aranır:

```
--datadir
$NMAPDIR
~/ .nmap (not searched on Windows)
<APPDATA>\nmap (only on Windows)
the directory containing the nmap executable
the directory containing the nmap executable, followed by ../share/nmap (not searched on Windows)
NMAPDATADIR (not searched on Windows)
the current directory.
```

ile biten bir dizin adı verildiğinde, Nmap dizindeki adı .nse ile biten her dosyayı yükler. Diğer tüm dosyalar göz ardı edilir ve dizinler özyinelemeli olarak aranmaz. Bir dosya adı verildiğinde, .nse uzantısına sahip olmak zorunda değildir; gerekirse otomatik olarak eklenecektir.

Örnekler ve --script seçeneğinin tam açıklaması için "Komut Dosyası Seçimi" bölümüne bakın.

Nmap komut dosyaları varsayılan olarak Nmap veri dizininin scripts alt dizininde saklanır (bkz. Bölüm 14, Nmap Veri Dosyalarını Anlama ve Özelleştirme). Verimlilik için, betikler scripts/script.db'de depolanan ve her betığın ait olduğu kategori veya kategorileri listeleyen bir veritabanında dizinlenir. all argümanı Nmap komut dosyası veritabanındaki tüm komut dosyalarını çalıştıracaktır, ancak Nmap istismarlar, hizmet reddi saldıruları ve diğer tehlikeli komut dosyalarını içerebileceğiinden dikkatli kullanılmalıdır.

--script-args <args> ⇒ Komut dosyalarına bağımsız değişkenler sağlar. Ayrıntılı açıklama için "Komut Dosyalarına Argümanlar" başlıklı bölüme bakın.

--script-args-file <filename> ⇒ Bu seçenek --script-args ile aynıdır, ancak argümanları komut satırı yerine bir dosyada iletirsiniz. Ayrıntılı açıklama için "Komut Dosyalarının Bağımsız Değişkenleri" başlıklı bölüme bakın.

--script-help <filename> | <category> | <directory> | <expression> [all,...] ⇒ Komut dosyaları hakkında yardım gösterir. Verilen belirtimle eşleşen her betik için Nmap betik adını, kategorilerini ve açıklamasını yazdırır. Belirtimler --script tarafından kabul edilenlerle aynıdır; örneğin, ssl-enum-ciphers betiği hakkında yardım istiyorsanız, nmap --script-help ssl-enum-ciphers komutunu çalıştırırsınız. Betik yardımının bir örneği Örnek 9.2, "Betik Yardımı" bölümünde gösterilmektedir.

Örnek 9.2. Komut dosyası yardımı

```
$ nmap --script-help "afp-* and discovery"
Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-21 14:15 UTC
afp-ls
Categories: discovery safe
https://nmap.org/nsedoc/scripts/afp-ls.html
Attempts to get useful information about files from AFP volumes.
The output is intended to resemble the output of ls.

afp-serverinfo
Categories: default discovery safe
https://nmap.org/nsedoc/scripts/afp-serverinfo.html
Shows AFP server information. This information includes the server's
hostname, IPv4 and IPv6 addresses, and hardware type (for example
Macmini or MacBookPro).

afp-showmount
Categories: discovery safe
https://nmap.org/nsedoc/scripts/afp-showmount.html
Shows AFP shares and ACLs.
```

Eğer -oX seçeneği kullanılırsa, komut dosyası yardımının bir XML gösterimi verilen dosyaya yazılacaktır.

--script-trace ⇒ Bu seçenek --packet-trace seçeneğine benzer, ancak paket paket yerine uygulama düzeyinde çalışır. Bu seçenek belirtilirse, komut dosyaları tarafından gerçekleştirilen tüm gelen ve giden iletişim yazdırılır. Görüntülenen bilgiler iletişim protokolünü, kaynak ve hedef adresleri ve iletilen verileri içerir. İletilen verilerin %5'inden fazlası yazdırılamazsa, bunun yerine hex dökümleri verilir. --packet-trace belirtilmesi komut dosyası izlemeyi de etkinleştirir.

--script-updatedb ⇒ Bu seçenek, Nmap tarafından mevcut varsayılan komut dosyalarını ve kategorileri belirlemek için kullanılan scripts/script.db dosyasında bulunan komut dosyası veritabanını günceller. Veritabanını güncellemek yalnızca varsayılan komut dizinine NSE komut dosyaları eklediğinizde veya kaldırığınızda ya da herhangi bir komut dosyasının kategorilerini değiştirdiğinizde gereklidir. Bu seçenek argüman olmadan tek başına kullanılır: nmap --script-updatedb.

Düzenleme bazı Nmap seçeneklerinin komut dosyası taramaları üzerinde etkileri vardır. Bunlardan en belirgin olanı -sV'dır. Bir sürüm taraması, sürüm kategorisindeki komut dosyalarını otomatik olarak çalıştırır. Bu kategorideki komut dosyaları diğer komut dosyalarından biraz farklıdır, çünkü çıktıları sürüm tarama sonuçlarıyla karışır ve ekrana herhangi bir komut dosyası tarama çıktıları üretmezler. Eğer -oX seçeneği kullanılırsa, tipik komut dosyası çıktı XML çıktı dosyasında mevcut olmaya devam edecektir.

Komut dosyası motorunu etkileyen bir başka seçenek de -A'dır. Agresif Nmap modu -sC seçeneği anlamına gelir.

Script Selection (Senaryo Seçimi)

--script seçeneği virgülle ayrılmış bir kategori, dosya adı ve dizin adı listesi alır. Kullanımına ilişkin bazı basit örnekler:

nmap --script default,safe ⇒ Varsayılan ve güvenli kategorilerdeki tüm komut dosyalarını yükler.

nmap --script smb-os-discovery ⇒ Yalnızca smb-os-discovery betiği yükler. .nse uzantısının isteği bağlı olduğunu unutmuyın.

nmap --script default,banner,/home/user/customscripts ⇒ Varsayılan kategorideki betiği, başlık betliğini ve /home/user/customscripts dizinindeki tüm .nse dosyalarını yükler.

script.db'deki komut dosyalarına isimle atıfta bulunurken, kabuk tarzı bir '*' joker karakteri kullanabilirsiniz.

nmap --script "http-*" ⇒ Adı http- ile başlayan http-auth ve http-open-proxy gibi tüm betikleri yükler. Joker karakteri kabuktan korumak için --script argümanının tırnak içinde olması gerekiyordu.

Boolean ifadeleri oluşturmak için and, or ve not operatörleri kullanılarak daha karmaşık kod seçimi yapılabılır. İşleçler Lua'da olduğu gibi aynı önceliğe sahiptir: not en yüksektir, ardından and ve sonra or gelir. Parantez kullanarak önceliği değiştirebilirsiniz. İfadeler boşluk karakterleri içerdiginden, bunları tırnak içine almak gereklidir.

nmap --script "not intrusive" ⇒ Müdahaleci kategorisindekiler hariç tüm komut dosyalarını yükler.

nmap --script "default or safe" ⇒ Bu işlevsel olarak nmap --script "default,safe" ile eşdeğerdir. Varsayılan kategorideki veya güvenli kategorideki ya da her ikisindeki tüm betikleri yükler.

nmap --script "default and safe" ⇒ Hem varsayılan hem de güvenli kategorilerde bulunan komut dosyalarını yükler.

nmap --script "(default or safe or intrusive) and not http-*" ⇒ Adları http- ile başlayanlar hariç, varsayılan, güvenli veya müdahaleci kategorilerdeki betikleri yükler.

Boole ifadesindeki adlar bir kategori, script.db dosyasından bir dosya adı veya tümü olabilir. Bir ad, operatör olan and, or ve not dizileri hariç olmak üzere, ' ', ',', '(', ')' veya ';' içermeyen herhangi bir karakter dizisidir.

Arguments to Scripts (Komut Dosyalarına Argümanlar)

Bağımsız değişkenler --script-args seçeneği kullanılarak NSE komut dosyalarına aktarılabilir. Argümanlar, anahtar-değer çiftleri ve muhtemelen dizi değerlerinden oluşan bir tabloyu tanımlar. Bağımsız değişkenler komut dosyalarına nmap.registry.args adlı kayıt defterinde bir tablo olarak sağlanır, ancak bunlara normalde stdnse.get_script_args işlevi aracılığıyla erişiliyor.

Kod bağımsız değişkenlerinin sözdizimiLua'nın tablo oluşturucu sözdizimine benzer. Bağımsız değişkenler, virgülle ayrılmış bir name=value çiftleri listesidir. Adlar ve değerler, boşluk veya '{', '}', '=' veya ',' karakterlerini

îçermeyen dizeler olabilir. Bu karakterlerden birini bir dizeye dahil etmek için, dizeyi tek veya çift tırnak içine alın. Tırnak içine alınmış bir dize içinde, '\' bir tırnak işaretinden kaçar. Ters egek çizgi yalnızca bu özel durumda tırnak işaretlerinden kaçmak için kullanılır; diğer tüm durumlarda ters egek çizgi tam anlamıyla yorumlanır.

Değerler, Lua'da olduğu gibi {} içine alınmış tablolar da olabilir. Bir tablo, örneğin proxy ana bilgisayarlarının bir listesi gibi basit dize değerleri veya iç içe tablolar da dahil olmak üzere daha fazla ad-değer çifti içerebilir.

Komut dosyası bağımsız değişkenleri genellikle ilgili komut dosyası adıyla nitelendirilir, böylece bir kullanıcı tek bir genel adla birden fazla komut dosyasını istemeden etkilemez. Örneğin, broadcast-ping.timeout komut dosyasını beklemek istediğiniz süreye ayarlayarak broadcast-ping komut dosyasına (ve yalnızca bu komut dosyasına) verilen yanıtlar için zaman aşımını ayarlayabilirsiniz. Ancak bazen bir komut dosyası argümanının daha geniş çapta uygulanmasını istersiniz. Nitelemeyi kaldırır ve sadece timeout=250ms olarak belirtirseniz, broadcast-ping'e ek olarak bir düzineden fazla betik için değer ayarlamış olursunuz. Nitelikli ve niteliksiz bağımsız değişkenleri bile birleştirebilirsiniz ve en spesifik eşleşme önceliklidir. Örneğin, rlogin-brute.timeout=20s,timeout=250ms şeklinde belirtebilirsiniz. Bu durumda, zaman aşımı rlogin-brute betiği için 20 saniye ve bu değişkeni destekleyen diğer tüm betikler (broadcast-ping, lld-discovery, vb.) için 250 milisaniye olacaktır.

Komut satırında --script-args ile argümanları iletmek yerine, bunları bir dosyada saklayabilir (virgül veya satırsonu ile ayırarak) ve --script-args-file ile sadece dosya adını belirtebilirsiniz. Komut satırında --script-args ile belirtilen seçenekler, bir dosyada verilenlere göre önceliklidir. Dosya adı mutlak bir yol olarak veya Nmap'in olağan arama yoluna (NMAPDIR, vb.) göreli olarak verilebilir.

İşte komut dosyası argümanları ile tipik bir Nmap çağrısı:

```
nmap -sC --script-args 'user=foo,pass=",{}=bar",paths={"/admin,/cgi-bin},xmpp-info.server_name=localhost'
```

Komut dosyası argümanlarının tek tırnak içine alındığını dikkat edin. Bash kabuğu için bu, kabuğun çift tırnak işaretlerini yorumlamasını ve otomatik dize birleştirme yapmasını engeller. Doğal olarak, farklı kabuklar tırnak işaretlerinden kaçmanızı veya farklı tırnak işaretleri kullanmanızı gerektirebilir. İlgili kılavuzuza bakın. Komut buLua tablosuyla sonuçlanır:

```
nmap.registry.args = {  
    user = "foo",  
    pass = ",{}=bar",  
    paths = {  
        "/admin",  
        "/cgi-bin"  
    },  
    xmpp-info.server_name="localhost"  
}
```

Değerlere doğrudan nmap.registry.args'den erişebilseniz de, normalde stdnse.get_script_args işlevini şu şekilde kullanmak daha iyidir:

```
local server_name = stdnse.get_script_args("xmpp-info.server_name")
```

Tüm komut dosyası bağımsız değişkenleri, nmap.registry.args tablosu olan genel bir ad alanını paylaşır. Bu nedenle, user gibi kısa veya belirsiz adlar önerilmez. Bazı betikler argümanlarının önüne smtp-open-relay.domain gibi betik adlarını ekler. Birçok betiği etkileyebilen kütüphaneler tarafından kullanılan bağımsız değişkenlerin adları genellikle kütüphanenin adıyla başlar, smbuser ve creds.snmp gibi.

<https://nmap.org/nsedoc/> adresindeki çevirmiçi NSE Dokümantasyon Portalı, komut dosyasını etkileyebilecek tüm kütüphane argümanları da dahil olmak üzere her komut dosyasının kabul ettiği argümanları listeler.

Complete Examples (Eksiksiz Örnekler)

nmap -sC example.com ⇒ Varsayılan komut dosyası kümesini kullanarak basit bir komut dosyası taraması.

nmap -sn -sC example.com ⇒ Bağlantı noktası taraması olmayan bir komut dosyası taraması; yalnızca ana bilgisayar komut dosyaları çalıştırılabilir.

nmap -Pn -sn -sC example.com ⇒ Ana bilgisayar keşfi veya bağlantı noktası taraması olmayan bir komut dosyası taraması. Tüm ana bilgisayarların açık olduğu varsayılar ve yalnızca ana bilgisayar komut dosyaları çalıştırılabilir.

nmap --script smb-os-discovery --script-trace example.com ⇒ Komut dosyası izleme ile belirli bir komut dosyasını çalıştırın.

nmap --script snmp-sysdescr --script-args creds.snmp=admin example.com ⇒ Bir komut dosyası bağımsız değişkeni alan tek bir komut dosyası çalıştırın.

nmap --script mycustomscripts,safe example.com ⇒ mycustomscripts dizinindeki tüm komut dosyalarının yanı sıra güvenli kategorisindeki tüm komut dosyalarını da çalıştırın.

Script Format (Senaryo Formatı)

NSE komut dosyaları birkaç açıklayıcı alandan, komut dosyasının ne zaman çalıştırılacağını tanımlayan bir kuraldan ve gerçek komut dosyası talimatlarını içeren bir eylem işlevinden oluşur. Açıklayıcı alamlara típki diğer Lua değişkenlerine atadığınız gibi değerler atanabilir. Bu bölümde gösterildiği gibi isimleri küçük harf olmalıdır.

description Field (Açıklama Alanı) ⇒ Açıklama alanı, bir komut dosyasının ne için test edildiğini ve kullanıcının bilmesi gereken önemli notları açıklar. Komut dosyasının karmaşıklığına bağlı olarak, açıklamaların uzunluğu birkaç cümleden birkaç paragrafa kadar değişebilir. İlk paragraf, kullanıcıya tek başına sunulmaya uygun komut dosyası işlevinin kısa bir özeti olmalıdır. Diğer paragraflar çok daha fazla kod detayı sağlayabilir.

description Field (Açıklama Alanı) ⇒ Açıklama alanı, bir komut dosyasının ne için test edildiğini ve kullanıcının bilmesi gereken önemli notları açıklar. Komut dosyasının karmaşıklığına bağlı olarak, açıklamaların uzunluğu birkaç cümleden birkaç paragrafa kadar değişebilir. İlk paragraf, kullanıcıya tek başına sunulmaya uygun komut dosyası işlevinin kısa bir özeti olmalıdır. Diğer paragraflar çok daha fazla kod detayı sağlayabilir.

categories Field (Kategoriler Alanı) ⇒ Kategoriler alanı, bir komut dosyasının ait olduğu bir veya daha fazla kategorisi tanımlar ("Komut Dosyası Kategorileri" başlıklı bölüme bakın). Kategoriler büyük/küçük harfe duyarlı değildir ve herhangi bir sırada belirtilebilir. Bu örnekte olduğu gibi dizi tarzı bir Lua tablosunda listelenirler:

```
categories = {"default", "discovery", "safe"}
```

author Field (Yazar Field) ⇒ Yazar alanı, kod yazarlarının adlarını içerir ve ayrıca iletişim bilgilerini de (ana sayfa URL'leri gibi) içerebilir. Artık e-posta adreslerinin eklenmesini önermiyoruz çünkü spam gönderenler bunları NSEDoc web sitesinden kazıyalabilir. Bu istege bağlı olan NSE tarafından kullanılmaz, ancak komut dosyası yazarlarına hak ettikleri krediyi veya suçlamayı verir.

license Field (Lisans Alanı) ⇒ Nmap bir topluluk projesidir ve NSE betikleri de dahil olmak üzere her türlü kod katkısını memnuniyetle karşılıyoruz. Eğer değerli bir betik yazarsanız, bunu kendinize saklamayın! İstege bağlı lisans alanı, Nmap ile birlikte gelen tüm komut dosyalarını dağıtmak için yasal izne sahip olmamızı sağlamaya yardımcı olur. Bu komut dosyalarının tümü şu anda standart Nmap lisansını kullanmaktadır ("Nmap Telif Hakkı ve Lisanslama" adlı bölümde açıklanmıştır). Aşağıdaki satırı içerirler:

```
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
```

Nmap lisansı GNU GPL'ye benzer. Betik yazarları tercih ederlerse bunun yerine BSD tarzı bir lisans (reklam maddesi yok) kullanabilirler. BSD tarzı bir lisans için lütfen bu satırı ekleyin:

```
license = "Simplified (2-clause) BSD license--See https://nmap.org/svn/docs/licenses/BSD-simplified"
```

dependencies **Field (bağımlılıklar Alanı)** ⇒ Bağımlılıklar alanı, seçildikleri takdirde bu koddan önce çalışması gereken kodların adlarını içeren bir dizidir. Bu, bir betığın diğerinin sonuçlarını kullanabileceğii durumlarda kullanılır. Örneğin, smb-* betiklerinin çoğu smb-brute'a bağımlıdır, çünkü smb-brute tarafından bulunan hesaplar diğer betiklerin daha fazla bilgi almasını sağlayabilir. Bir betığın bağımlılıklar içinde listelenmesi o betığın çalıştırılmasına neden olmaz; yine de --script seçeneği ile veya başka bir şekilde seçilmesi gerekir. bağımlılıklar yalnızca seçilen betikler arasında bir sıralama yapılmasını zorlar. Bu, smb-os-discovery'den bir bağımlılıklar tablosu örneğidir:

```
dependencies = {"smb-brute"}
```

Bağımlılıklar tablosu isteğe bağlıdır. Alan atlanırsa NSE, kodun hiçbir bağımlılığı olmadığını varsayıacaktır.

Bağımlılıklar, her birine "çalışma seviyesi"[12] adı verilen bir sayı atayarak komut dosyalarının dahili bir sıralamasını oluşturur. Komut dosyalarınızı çalıştırırken, NSE'nin çıktısında çalıştırılan her komut dosyası grubunun çalışma düzeyini (toplam çalışma sayısı ile birlikte) göreceksiniz:

```
NSE: Script scanning 127.0.0.1.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 17:38
Completed NSE at 17:38, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 17:38
Completed NSE at 17:38, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 17:38
Completed NSE at 17:38, 0.00s elapsed
NSE: Script Scanning completed.
```

Rules (Kurallar)

Nmap, bir betığın bir hedefe karşı çalıştırılıp çalıştırılmayacağını belirlemek için betik kurallarını kullanır. Bir kural, doğru veya yanlış döndüren bir Lua işlevidir. Kod eylem işlevi yalnızca kural true olarak değerlendirilirse gerçekleştirilir.

Bir komut dosyası, komut dosyasının ne zaman çalıştırılacağını belirleyen aşağıdaki işlevlerden bir veya daha fazlasını içermelidir:

```
prerule()
hostrule(host)
portrule(host, port)
postrule()
```

prerule komut dosyaları, herhangi bir ana bilgisayar taranmadan önce, komut dosyası ön tarama aşamasında bir kez çalışır. hostrule ve portrule komut dosyaları, her ana bilgisayar grubu tarandıktan sonra çalışır. postrule komut dosyaları, tüm ana bilgisayarlar tarandıktan sonra, komut dosyası son tarama aşamasında bir kez çalışır. Bir komut dosyası birden fazla kurala sahipse birden fazla aşamada çalışabilir.

prerule ve postrule argüman kabul etmez. hostrule bir ana bilgisayar tablosu kabul eder ve örneğin hedefin IP adresini veya ana bilgisayar adını test edebilir. portrule açık, açık|filtrelenmiş veya filtrenmemiş bağlantı noktası durumlarındaki herhangi bir bağlantı noktası için hem bir ana bilgisayar tablosu hem de bir bağlantı noktası tablosu kabul eder. Port kuralları genellikle bir porta karşılaştırılıp çalıştırılmayacağına karar verirken port numarası, port durumu veya dinleme hizmeti adı gibi faktörleri test eder. Örnek kurallar "Kural" adlı bölümde gösterilmiştir.

İleri düzey kullanıcılar, --komut dosyası bağımsız değişkeninde komut dosyası adının (veya kategorisinin ya da başka bir ifadenin) önüne + işaretti koyarak bu kural işlevlerinin sonuçlarından bağımsız olarak bir komut dosyasını çalışmaya zorlayabilir.

Bir prerule veya postrule arasında seçim yapmak için geçerli standart şudur: komut dosyası ana bilgisayar keşfi veya başka bir ağ işlemi yapıyorsa, prerule kullanılmalıdır. postrule, tarama sırasında toplanan verilerin ve istatistiklerin raporlanması için ayrılmıştır.

Action (Eylem)

Eylem, bir NSE komut dosyasının kalbidir. Kodun prerule, portrule, hostrule veya postrule tetiklendiğinde yürütülecek tüm talimatları içerir. Kuralla aynı argümanları kabul eden bir Lua fonksiyonudur. Eylem değerinin dönüş değeri bir isim-değer çiftleri tablosu, bir dize veya nil olabilir. Bir NSE eylemi örneği için "Eylem" başlıklı bölüme bakın.

Eylemin çıktısı bir tabloysa, normal (-oN) ve XML (-oX) çıktı biçimlerine dahil edilmek üzere otomatik olarak yapılandırılmış bir şekilde biçimlendirilir. Bir dizeyse, metin normal çıktıda doğrudan görüntülenir ve XML çıktısında bir XML özniteliği olarak yazılır, Kod nil döndürürse hiçbir çıktı üretilmez. Farklı dönüş değerlerinin nasıl işlendiğine ilişkin ayrıntılar için "Yapılandırılmış ve Yapılandırılmamış Çıktı" başlıklı bölüme bakın.

Environment Variables (Ortam Değişkenleri)

Her komut dosyasının kendi ortam değişkenleri kümesi vardır:

`SCRIPT_PATH` ⇒ Komut dosyası yolu.

`SCRIPT_NAME` ⇒ Komut dosyası adı. Bu değişken hata ayıklama çıktısında kullanılabilir.

`SCRIPT_TYPE` ⇒ Bir komut dosyasının birden fazla kural işlevi olabileceğinden, bu ortam değişkeni hangi kuralın komut dosyasını etkinleştirdiğini gösterecektir; bu, komut dosyası farklı Komut Dosyası Tarama aşamaları arasında bazı kodları paylaşmak istiyorsa yararlı olacaktır. Bu dört dize değerinden birini alacaktır: "prerule", "hostrule", "portrule" veya "postrule". Bu değişken sadece kural fonksiyonlarının değerlendirilmesi sırasında ve sonrasında kullanılabilir.

Bu, dns-zone-transfer'den önceki ortam değişkenlerini ve ardından çıktı mesajını kullanan bir hata ayıklama kodu örneğidir:

```
stdnse.print_debug(3, "Skipping '%s' %s, 'dnszonetransfer.server' argument is missing.", SCRIPT_NAME, SCRIPT_TYPE)
```

```
Initiating NSE at 15:31
NSE: Skipping 'dns-zone-transfer' prerule, 'dnszonetransfer.server' argument is missing.
```

Script Language (Senaryo Dili)

Nmap Scripting Engine'in çekirdeği gömülebilir bir Lua yorumlayıcısıdır. Lua genişletilebilirlik için tasarlanmış hafif bir dildir. Nmap gibi diğer yazılımlarla arayüz oluşturmak için güçlü ve iyi belgelenmiş bir API sunar.

Nmap Scripting Engine'in ikinci kısmı, Lua ve Nmap'i birbirine bağlayan NSE Kütüphanesidir. Bu katman, Lua yorumlayıcısının başlatılması, paralel komut dosyası yürütme zamanlaması, komut dosyası alma ve daha fazlası gibi konuları ele alır. Aynı zamanda NSE ağ I/O çerçevesinin ve istisna işleme mekanizmasının kalbidir. Ayrıca, komut dosyalarını daha güçlü ve kullanışlı hale getirmek için yardımcı kütüphaneler de içerir. Yardımcı Kütüphane modülleri ve uzantıları "NSE Kütüphaneleri" adlı bölümde açıklanmaktadır.

Lua Base Language (Lua Temel Dil)

Nmap komut dosyası dili, Nmap ile arayüz oluşturmak için Kütüphanelerle genişletilmiş gömülü bir Lua yorumlayıcısıdır. Nmap API'si nmap Lua isim alanındadır. Bu, Nmap tarafından sağlanan kaynaklara yapılan tüm çağrılarının bir nmap önekine sahip olduğu anlamına gelir. nmap.new_socket(), örneğin, yeni bir soket sarmalayıcı nesnesi döndürür. Nmap kütüphane katmanı ayrıca Lua bağlamının başlatılması, paralel komut dosyalarının zamanlanması ve tamamlanan komut dosyaları tarafından üretilen çıktıların toplanmasıyla da ilgilendir.

Planlama aşamaları sırasında, Nmap komut dosyası için temel olarak birkaç programlama dilini düşündük. Bir başka seçenek de tamamen yeni bir programlama dili uygulamaktı. Kriterlerimiz çok katıldı: NSE'nin kullanımı kolay, boyutu küçük, Nmap lisansı ile uyumlu, ölçülebilir, hızlı ve paralelleştirilebilir olması gerekiyordu. Kendi güvenlik denetim dillerini sıfırdan tasarlamaya yönelik önceki birkaç çaba (diğer projeler tarafından) garip çözümlerle sonuçlandı, bu nedenle bu rotayı izlememeye erken karar verdik. İlk olarak Guile Scheme yorumlayıcısı düşünüldü, ancak daha elverişli lisansı nedeniyle tercih Elk yorumlayıcısına yöneldi. Ancak Elk betiklerini paralelleştirmek zor olacaktı. Buna ek olarak, çoğu Nmap kullanıcısının Scheme gibi fonksiyonel diller yerine prosedürel programlamayı tercih etmesini bekliyoruz. Perl, Python ve Ruby gibi daha büyük yorumlayıcılar iyi bilinir ve sevılır, ancak verimli bir şekilde gömülmesi zordur. Sonunda, Lua tüm kriterlerimizde başarılı oldu. Küçüktür, liberal MIT açık kaynak lisansı altında dağıtılmıştır, verimli paralel komut dosyası yürütme için korutinlere sahiptir, gömülebilirlik göz önünde bulundurularak tasarlanmıştır, mükemmel belgelere sahiptir ve büyük ve kararlı bir topluluk tarafından aktif olarak geliştirilmektedir. Lua artık Wireshark sniffer ve Snort IDS gibi diğer popüler açık kaynak güvenlik araçlarına bile gömülüdür.

NSE Scripts (NSE Komut Dosyaları)

Bu bölüm (kısa özetlerle birlikte NSE komut dosyalarının uzun bir listesi) bu kitabın yalnızca basılı baskısında yer almaktadır çünkü NSE Dokümantasyon Portalı'ndaki bilgilere daha iyi bir çevrimiçi arayüz sağlamaktayız.

NSE Scripts ⇒ <https://nmap.org/nsedoc/scripts/>

NSE Libraries ⇒ <https://nmap.org/nsedoc/lib/>

NSE Libraries (NSE Kütüphaneleri)

Lua'nın önemli yerleşik yeteneklerine ek olarak, komut dosyası yazmayı daha güçlü ve kullanışlı hale getiren birçok uzantı kütüphanesi yazdık veya entegre ettik. Bu kütüphaneler (bazen modül olarak da adlandırılır) gerektiğinde derlenir ve Nmap ile birlikte yüklenir. Kendi dizinleri olan nselib, yapılandırılmış Nmap veri dizinine yüklenir. Komut dosyalarının bunları kullanmak için yalnızca varsayılan kütüphanelere ihtiyaç duyması gereklidir.

List of All Libraries (Tüm Kütüphanelerin Listesi)

Bu liste, hangi kütüphanelerin mevcut olduğu hakkında bir fikir vermek için sadece bir genel bakış niteliğindedir. Geliştiriciler <https://nmap.org/nsedoc/> adresindeki belgelerin tamamına başvurmak isteyeceklerdir.

afp ⇒ Bu kütüphane Patrik Karlsson patrik@cqure.net tarafından Apple AFP Servisi ile iletişimini kolaylaştırmak için yazılmıştır. Tam özellikli değildir ve hala birkaç işlevi eksiktir.

ajp ⇒ Apache mod_proxy_ajp'de bulunan belgelere dayalı temel bir AJP 1.3 uygulaması; http://httpd.apache.org/docs/2.2/mod/mod_proxy_ajp.html

amqp ⇒ AMQP kütüphanesi, bir AMQP sunucusunun özellikleri hakkında bilgi almak için bazı temel işlevler sağlar.

asn1 ⇒ ASN.1 fonksiyonları.

base32 ⇒ Base32 kodlama ve kod çözme. RFC 4648'i takip eder.

base64 ⇒ Base64 kodlama ve kod çözme. RFC 4648'i takip eder.

bin ⇒ İkili verileri paketleyin ve paketten çıkarın.

bit ⇒ Tamsayılar üzerinde bitsel işlemler.

bitcoin ⇒ Bu kütüphane BitCoin protokolünün minimal bir alt kümesini uygular Şu anda sürüm el sıkışmasını ve Addr yanıtlarının işlenmesini desteklemektedir.

bittorrent ⇒ Kullanıcıların bir torrent dosyasından bilgi okumasına, bencoded (bittorrent kodlu) arabellekleri çözmesine, belirli bir torrentle ilişkili eşleri bulmasına ve eş arama sırasında keşfedilen düğümleri almasına olanak tanıyan Bittorrent ve DHT protokol kitabı.

bjnp ⇒ Canon ağ yazıcılarını ve tarayıcı aygıtlarını keşfetmek ve sorgulamak için kullanılan Canon BJNP protokolünün bir uygulaması.

brute ⇒ Brute kütüphanesi, uzak hizmetlere karşı parola tahmini gerçekleştirmek için ortak bir çerçeve oluşturma girişimidir.

cassandra ⇒ Cassandra Thrift iletişimini istemci olarak ele almak için kütüphane yöntemleri

citrixxml ⇒ Bu modül Patrik Karlsson tarafından yazılmıştır ve Citrix XML Hizmeti ile iletişimini kolaylaştırır. Tam özellikli değildir ve birkaç işlev ve parametre eksiktir.

comm ⇒ Banner yakalama ve veri alışverişi gibi ağ keşif görevleri için ortak iletişim işlevleri.

creds ⇒ Kimlik bilgisi sınıfı, bulunan kimlik bilgilerini Nmap kayıt defterinde saklar

cvs ⇒ Şu anda yalnızca kimlik doğrulamayı destekleyen minimal bir CVS (Eşzamanlı Sürümler Sistemi) pserver protokolü uygulaması.

datafiles ⇒ Nmap'in bazı veri dosyalarını okuyun ve ayırtırın: nmap-protocols, nmap-rpc, nmap-services ve nmap-mac-prefixes.

dhcp ⇒ Bir Dinamik Ana Bilgisayar Yapılandırma Protokolü (DHCP) istemcisi uygulayın.

dhcp6 ⇒ Temel DHCP6 isteklerini destekleyen minimalist DHCP6 (IPv6 için Dinamik Ana Bilgisayar Yapılandırma Protokolü) uygulaması Kütüphane aşağıdaki sınıflar etrafında yapılandırılmıştır:

- DHCP6.Option - DHCP6 seçenekleri kodlayıcıları (istekler için) ve kod çözücüleri (yanıtlar için)
- DHCP6.Request - DHCP6 istek kodlayıcı ve kod çözücü
- DHCP6.Response - DHCP6 yanıt kodlayıcısı ve kod çözücü

- Helper - Yardımcı sınıf, birincil komut dosyası arayüzü

`dns` ⇒ Paket oluşturma, kodlama, kod çözme ve sorgulamayı destekleyen basit DNS kütüphanesi.

`dnsbl` ⇒ Çeşitli DNSBL hizmetlerinin sorgulanmasını kolaylaştırmak için uygulanan minimalist bir DNS BlackList kütüphanesi. Mevcut hizmet listesi, aşağıdaki hizmet derlemelerine dayanarak uygulanmıştır:

- http://en.wikipedia.org/wiki/Comparison_of_DNS_blacklists
- <http://www.robtex.com>
- <http://www.sdsc.edu/~jeff/spam/cbc.html>

`dnsd` ⇒ DNS Hizmet Keşfini desteklemek için kütüphane

`drda` ⇒ DRDA Kütüphanesi çok sınırlı bir operasyon alt kümesini desteklemektedir.

`eap` ⇒ Sınırlı bir özellik alt kümesini destekleyen EAP (Genişletilebilir Kimlik Doğrulama Protokolü) kütüphanesi.

`eigrp` ⇒ Cisco'nun EIGRP paketlerinin sınırlı bir alt kümesini ayırtmayı ve oluşturmayı destekleyen bir kütüphane.

`ftp` ⇒ FTP işlevleri.

`giop` ⇒ Çok sınırlı bir operasyon alt kümesini destekleyen GIOP Kütüphanesi

`gps` ⇒ Küçük bir gps ayrıştırma modülü. Şu anda GPRMC NMEA kod çözme yapıyor

`http` ⇒ HTTP istemci protokolünü Nmap komut dosyalarının yararlanabileceği standart bir biçimde uygular.

`httpspider` ⇒ Temel örümcekleme yetenekleri sağlayan küçük bir httpspider kütüphanesi Aşağıdaki sınıflardan oluşur:

`iax2` ⇒ Minimalist bir Asterisk IAX2 (Inter-Asterisk eXchange v2) VoIP protokolü uygulaması. Kütüphane, kaba kuvvet şifre tahminini gerçekleştirmek için gereken minimum uygulamayı gerçekleştirir.

`imap` ⇒ IMAP protokolünün küçük bir alt kümesini, şu anda CAPABILITY, LOGIN ve AUTHENTICATE işlevlerini uygulayan bir kütüphane. Kütüphane ilk olarak Brandon Enright tarafından yazıldı ve daha sonra Patrik Karlsson tarafından genişletildi ve OO formuna dönüştürüldü patrik@cqure.net

`informix` ⇒ Informix işlemelerinin çok sınırlı bir alt kümesini destekleyen Informix Kütüphanesi

`ipOps` ⇒ IP adreslerini işlemek ve karşılaştırmak için yardımcı fonksiyonlar.

`ipp` ⇒ Küçük bir CUPS ipp (İnternet Yazdırma Protokolü) kütüphane uygulaması

`iscsi` ⇒ Patrik Karlsson tarafından yazılan bir iSCSI kütüphanesi uygulaması patrik@cqure.net Kütüphane şu anda hedef bulma ve oturum açmayı desteklemektedir.

`isns` ⇒ Minimal bir İnternet Depolama Adı Hizmeti (iSNS) uygulaması

`jdwp` ⇒ JDWP (Java Debug Wire Protocol) kütüphanesi, uzaktan hata ayıklama portunu kullanmak ve java bytecode enjekte etmek için gereken bir dizi komutu uygular.

`json` ⇒ JSON verilerini işlemek için kütüphane yöntemleri. RFC 4627'ye göre JSON kodlama ve kod çözme işlemlerini gerçekleştirir.

`ldap` ⇒ LDAP'yı işlemek için kütüphane yöntemleri.

`lfs` ⇒ Verilen yolun içeriğini listeleyen bir dizin yineleyici döndürür Yineleyici dir_obj ile her çağrıda, bir dizin girdisinin adını bir dize olarak döndürür veya daha fazla girdi yoksa nil döndürür.

`listop` ⇒ Fonksiyonel tarzda liste işlemleri.

`match` ⇒ Tamponlanmış ağ I/O yardımcı fonksiyonları.

`membase` ⇒ Couchbase Membase TAP protokolünün küçük bir uygulaması Couchbase Wiki'deki az sayıda belgeye dayanmaktadır: x <http://www.couchbase.org/wiki/display/membase/SASL+Authentication+Example>

`mobileme` ⇒ "iPhone'umu bul" işlevini kullanarak Apple aygıtlarını keşfetmeyi sağlayan bir MobileMe web hizmeti istemcisi.

`mongodb` ⇒ MongoDB'yi işlemek, paketleri oluşturmak ve ayırtmak için kütüphane yöntemleri.

`msrpc` ⇒ Bu kütüphane smb kütüphanesini yoğun bir şekilde kullanarak çeşitli MSRPC fonksiyonlarını çağıracaktır. Burada kullanılan fonksiyonlara 445 ve 139 numaralı TCP portları üzerinden, kurulmuş bir oturumla erişilebilir. NULL oturum (varsayılan) bazı işlevler ve işletim sistemleri (ya da yapılandırmalar) için işe yararken diğerleri için işe yaramayacaktır.

`mspcperformance` ⇒ Bu modül, kayıt defterinde HKEY_PERFORMANCE_DATA altında depolanan PERF_DATA_BLOCK yapısını ayırtmak için tasarlanmıştır. Bu yapıyı sorgulayarak, neler olup bittiği hakkında pek çok bilgi edinebilirsiniz.

`msrpctypes` ⇒ Bu modül, Microsoft RPC (MSRPC) çağrıları için parametreleri toplamak üzere yazılmıştır. İçeri ve dışarı aktarılan değerler, protokol tarafından tanımlanan ve Samba geliştiricileri tarafından belgelenen yapılara dayanmaktadır. Türlerin ayrıntılı dökümleri için Samba 4.0'ın .idl dosyalarına bir göz atın.

`mssql` ⇒ MSSQL Kütüphanesi çok sınırlı bir işlem alt kümesini destekler.

`mysql` ⇒ Çok sınırlı bir işlem alt kümesini destekleyen basit MySQL Kütüphanesi.

`natpmp` ⇒ Bu kütüphane, NAT Bağlantı Noktası Eşleme Protokolü (NAT-PMP) taslağında açıklandığı gibi NAT-PMP'nin temellerini uygular: o <http://tools.ietf.org/html/draft-cheshire-nat-pmp-03>

`ncp` ⇒ Netware Çekirdek Protokolü'nün (NCP) küçük bir uygulaması. NCP başlangıçta sadece Netware protokolü iken, şimdi Novell eDirectory çalıştırın hem Linux hem de Windows platformlarında mevcuttur.

`ndmp` ⇒ Minimalist bir NDMP (Ağ Veri Yönetimi Protokolü) kütüphanesi

`netbios` ⇒ NetBIOS trafiğini oluşturur ve ayırtır. Bunun birincil kullanımı NetBIOS ad istekleri göndermektir.

`nmap` ⇒ Nmap dahili arayüzü.

`nrc` ⇒ Domino RPC'yi desteklemek için minimalist bir kütüphane

`nsedebug` ⇒ Nmap komut dosyaları için hata ayıklama işlevleri.

`omp2` ⇒ Bu kütüphane, OMP (OpenVAS Yönetim Protokolü) sürüm 2 kullanan OpenVAS Manager sunucuları ile etkileşimi kolaylaştmak için yazılmıştır.

`openssl` ⇒ OpenSSL bağıları.

`ospf` ⇒ Şu anda IPv4 ve aşağıdaki OSPF mesaj türlerini destekleyen minimalist bir OSPF (Open Shortest Path First yönlendirme protokolü) Kütüphanesi: MERHABA

`packet` ⇒ Ham paketleri işlemek için olanaklar.

`pcre` ⇒ Perl Uyumlu Düzenli İfadeler.

`pgsql` ⇒ Protokolün hem sürüm 2 hem de sürüm 3'ünü destekleyen PostgreSQL kütüphanesi. Kütüphane şu anda kimlik doğrulamasını gerçekleştirmek için gereken asgari bilgileri içermektedir. Kimlik doğrulama, SSL etkin olsun veya olmasın ve düz metin veya MD5 kimlik doğrulama mekanizmaları kullanılarak desteklenir.

`pop3` ⇒ POP3 işlevleri.

`pppoe` ⇒ PPPoE Keşif ve Yapılandırma istekleri için temel desteği uygulayan minimalist bir PPPoE (Ethernet üzerinden noktadan noktaya protokol) kütüphanesi. PPPoE protokolü ethernet tabanlıdır ve bu nedenle herhangi bir IP veya port numarası kullanmaz.

- `proxy` ⇒ Proxy testi için işlevler.
- `rdp` ⇒ Minimal bir RDP (Uzak Masaüstü Protokolü) kütüphanesi. Şu anda şifreleme ve şifre desteğini belirleme işlevine sahiptir.
- `redis` ⇒ Minimalist bir Redis (bellek içi anahtar-değer veri deposu) kütüphanesi.
- `rmi` ⇒ RMI üzerinden iletişim için kütüphane yöntemi (JRMP + java serileştirme)
- `rpc` ⇒ Çok sınırlı bir işlem alt kümesini destekleyen RPC Kütüphanesi.
- `rpcap` ⇒ Bu kütüphane WinPcap Remote Capture Daemon ile iletişim kurmak için gereken temelleri uygular. Şu anda NULL- veya Parola tabanlı kimlik doğrulama kullanarak hizmete kimlik doğrulamayı desteklemektedir. Ek olarak, kılçılık için kullanılabilen arayüzleri listeleyen yeteneklerine sahiptir.
- `rsync` ⇒ Minimalist bir RSYNC (uzaktan dosya senkronizasyonu) kütüphanesi
- `rtsp` ⇒ Bu Gerçek Zamanlı Akış Protokolü (RTSP) kütüphanesi, mevcut komut dosyaları tarafından ihtiyaç duyulan protokolün yalnızca minimum bir alt kümesini uygular.
- `sasl` ⇒ Basit Kimlik Doğrulama ve Güvenlik Katmanı (SASL).
- `shortport` ⇒ Kısa portreler oluşturmak için işlevler.
- `sip` ⇒ SIP komutlarının ve yöntemlerinin sınırlı bir alt kümesini destekleyen bir SIP kütüphanesi
- `smb` ⇒ Bir Windows protokolü olan Sunucu İleti Bloğu (SMB, CIFS'in bir uzantısı) trafigiyle ilgili işlevleri uygular.
- `smbauth` ⇒ Bu modül SMB'de kullanılan kimlik doğrulama ile ilgilenir (LM, NTLM, LMv2, NTLMv2).
- `smtp` ⇒ Basit Posta Aktarım Protokolü (SMTP) işlemleri.
- `snmp` ⇒ SNMP işlevleri.
- `socks` ⇒ Küçük bir SOCKS sürüm 5 proxy protokolü uygulaması
- `srvcloc` ⇒ Hizmet Konum Protokolü'nün nispeten küçük bir uygulaması. Başlangıçta Novell NCP sunucularını keşfetme isteklerini desteklemek için tasarlanmıştır, ancak başka herhangi bir hizmet için de çalışmalıdır.
- `ssh1` ⇒ SSH-1 protokolü için işlevler. Bu modül ayrıca anahtar parmak izlerini biçimlendirmek için işlevler içerir.
- `ssh2` ⇒ SSH-2 protokolü için işlevler.
- `sslcert` ⇒ SSL sertifikalarını toplamak ve bunları ana bilgisayar tabanlı kayıt defterinde saklamak için işlevler sağlayan bir kütüphane.
- `stdnse` ⇒ Standart Nmap Scripting Engine fonksiyonları. Bu modül, kendi modüllerini haklı çıkarmak için çok küçük olan çeşitli karışıklı işlevler içerir.
- `strbuf` ⇒ Dize tampon olanakları.
- `strict` ⇒ Katı bildirimli global kütüphane. Çalışma zamanı yürütmesi sırasında bildirilmemiş global değişkenleri kontrol eder.
- `stun` ⇒ RFC3489 ve RFC5389 uyarınca STUN protokolünün (Session Traversal Utilities for NAT) temellerini uygulayan bir kütüphane. Protokole genel bir bakış <http://en.wikipedia.org/wiki/STUN> adresinde mevcuttur.
- `tab` ⇒ Çıktıyi tablolar halinde düzenleyin.
- `target` ⇒ Nmap tarama kuyruğuna yeni keşfedilen hedefler eklemek için yardımcı fonksiyonlar.
- `tftp` ⇒ Minimal bir TFTP sunucusu uygulayan kütüphane
- `tns` ⇒ Oracle işlemlerinin çok sınırlı bir alt kümesini destekleyen TNS Kütüphanesi
- `unpwdbs` ⇒ Kullanıcı adı/şifre veritabanı kütüphanesi.

upnp ⇒ Başlangıçta Thomas Buchanan tarafından yazılan upnp-info koduna dayalı bir UPNP kütüphanesi. Kod, upnp-info'dan çıkarıldı ve çok noktaya yayın isteklerini desteklemek için Patrik Karlsson patrik@cqure.net tarafından kısmen yeniden yazıldı.

url ⇒ URI ayrıştırma, kompozisyon ve göreli URL çözümlemesi.

versant ⇒ Versant nesne veritabanı yazılımindan bazı temel bilgilerin numaralandırılmasına izin veren küçük bir kütüphane (bkz. http://en.wikipedia.org/wiki/Versant_Corporation). Kod tamamen Versant Management Center yönetim uygulaması kullanılırken yakalanan paket dökümlerine dayanmaktadır.

vnc ⇒ VNC kütüphanesi, VNC sunucularıyla ve Tight- veya Ultra- VNC gibi türevleriyle iletişim kurmak için gereken bazı temel işlevleri sağlar.

vulns ⇒ Güvenlik açığı yönetimi için işlevler.

vuzedht ⇒ Aşağıdaki belgelere dayalı bir Vuze DHT protokolü uygulaması: o

http://wiki.vuze.com/w/Distributed_hash_table

wsdd ⇒ Komut dosyalarının Web Hizmeti Dinamik Keşif problemleri göndermesini ve yanıtların bazı çok temel kod çözme işlemlerini gerçekleştirmesini sağlayan bir kütüphane. Kütüphane hiçbir şekilde tam bir WSDD uygulaması değildir, daha ziyade bazı paket yakalamalarının ve bazı yaratıcı kodlamaların sonucudur.

xmcp ⇒ XDMCP'nin (X Display Manager Control Protocol) uygulanması: x
<http://www.xfree86.org/current/xmcp.pdf>

xmpp ⇒ Bir XMPP (Jabber) kütüphanesi, kimlik doğrulama brute-force yapmak için yeterli protokolün minimal bir alt kümesini uygular.

Hacking NSE Libraries (NSE Kütüphanelerini Hacklemek)

Kütüphaneleri düzenlerken yapılan yaygın bir hata, yerel bir değişken yerine yanlışlıkla global bir değişken kullanmaktadır. Aynı global değişkeni kullanan farklı kütüphaneler gizemli hatalara neden olabilir. Lua'nın kapsam ataması varsayılan olarak globaldir, bu nedenle bu hatayı yapmak kolaydır.

Bu sorunu düzeltmeye yardımcı olmak için NSE, strict.lua adlı standart Lua dağıtımından uyarlanmış bir kütüphane kullanır. Kütüphane, dosya kapsamında bildirilmemiş bir global değişkene herhangi bir erişim veya değişiklik yapıldığında çalışma zamanı hatası verecektir. Kütüphane dosya kapsamında global isme (nil bile olsa) bir atama yaparsa, global değişken bildirilmiş kabul edilir.

Adding C Modules to Nselib (Nselib'e C Modülleri Ekleme)

Nselib'de bulunan modüllerden birkaçı Lua yerine C veya C++ ile yazılmıştır. İki örnek bit ve pcre'dir. Mümkünse modüllerin Lua dilinde yazılmasını öneriyoruz, ancak performans kritikse veya (pcre ve openssl modüllerinde olduğu gibi) mevcut bir C kütüphanesine bağlantı veriyorsanız C ve C++ daha uygun olabilir. Bu bölümde, nselib'e kendi derlenmiş uzantılarınızı nasıl yazacağınız açıklanmaktadır.

Nselib'de bulunan modüllerden birkaçı Lua yerine C veya C++ ile yazılmıştır. İki örnek bit ve pcre'dir. Mümkünse modüllerin Lua dilinde yazılmasını öneriyoruz, ancak performans kritikse veya (pcre ve openssl modüllerinde olduğu gibi) mevcut bir C kütüphanesine bağlantı veriyorsanız C ve C++ daha uygun olabilir. Bu bölümde, nselib'e kendi derlenmiş uzantılarınızı nasıl yazacağınız açıklanmaktadır.

NSE ile birlikte gelen en basit derlenmiş modül openssl'dir. Bu modül, yeni başlayan bir modül yazarı için iyi bir örnek teşkil eder. openssl kaynağı için kaynak kodu nse_openssl.cc ve nse_openssl.h içinde bulunmaktadır. Derlenen diğer modüllerin çoğu bu nse_<modül adı>.cc adlandırma kuralını takip eder.

openssl modülü incelendiğinde, nse_openssl.cc'deki fonksiyonlardan birinin MD5 özetini hesaplayan l_md5 olduğu görülür. Fonksiyon prototipi şöyledir:

```
static int l_md5(lua_State *L);
```

Prototip, I_md5'in lua_CFunction türüyle eşleştiğini gösterir. Fonksiyon statiktir çünkü diğer derlenmiş kodlar tarafından görülebilir olması gerekmektedir. Sadece Lua'ya kaydetmek için bir adres gereklidir. Dosyanın ilerleyen kısımlarında, I_md5, luaL_Reg türünde bir diziye girilir ve md5 adıyla ilişkilendirilir:

```
static const struct luaL_Reg openssllib[] = {
{ "md5", I_md5 },
{ NULL, NULL }
};
```

Bu fonksiyon artık NSE'de md5 olarak bilinecektir. Daha sonra kütüphane, aşağıda gösterildiği gibi, luaopen_openssl başlatma işlevi içinde luaL_newlib'e yapılan bir çağrı ile kaydedilir. OpenSSL BIGNUM tiplerinin kaydı ile ilgili bazı satırlar atlanmıştır:

```
LUALIB_API int luaopen_openssl(lua_State *L) {
    luaL_newlib(L, openssllib);
    return 1;
}
```

luaopen_openssl işlevi, dosyada nse_openssl.h içinde açık olan tek işlevdir. OPENSSLNAME basitçe "openssl" dizesidir.

Derlenmiş bir modül yazıldıktan sonra, nse_main.cc'deki standart kütüphaneler listesine dahil edilerek NSE'ye eklenmelidir. Daha sonra modülün kaynak dosya isimleri Makefile.in dosyasında uygun yerlere eklenmelidir. Bu iki görev için de diğer C modülleri örneğini takip edebilirsiniz. Windows derlemesi için, yeni kaynak dosyaları MS Visual Studio kullanılarak mswin32/nmap.vcproj proje dosyasına eklenmelidir ("Kaynak Koddan Derleme" adlı bölüme bakın).

Nmap API

NSE komut dosyaları, esnek ve zarif komut dosyaları yazmak için çeşitli Nmap olanaklarına erişebilir. API, bağlantı noktası durumları ve sürüm algılama sonuçları gibi hedef ana bilgisayar ayrıntılarını sağlar. Ayrıca verimli ağ I/O için Nsock Kütüphanesine bir arayüz sunar.

Information Passed to a Script (Komut Dosyasına Aktarılan Bilgiler)

Etkili bir Nmap komut dosyası motoru, bir Lua yorumlayıcısından daha fazlasını gerektirir. Kullanıcıların Nmap'in hedef ana bilgisayarlar hakkında öğrendiği bilgilere kolay erişmesi gerekmektedir. Bu veriler NSE komut dosyasının eylem yöntemine bağımsız değişkenler olarak aktarılır. Bağımsız değişkenler olan ana bilgisayar ve bağlantı noktası, betiğin yürütüldüğü hedef hakkında bilgi içerenLua tablolarıdır. Bir betik bir ana bilgisayar kurallıyla eşleşirse, yalnızca ana bilgisayar tablosunu alır ve bir port kurallıyla eşleşirse hem ana bilgisayarı hem de portu alır. Aşağıdaki listede bu iki tablodaki her değişken açıklanmaktadır.

host ⇒ Bu tablo kural ve eylem fonksiyonlarına parametre olarak aktarılır. Ana bilgisayar tarafından çalıştırılan işletim sistemi (-O anahtarı sağlanmışsa), IP adresi ve taranan hedefin ana bilgisayar adı hakkında bilgi içerir.

host.os ⇒ İşletim sistemi eşleşme tabloları dizisi. Bir işletim sistemi eşleşmesi, insan tarafından okunabilir bir ad ve bir dizi işletim sistemi sınıfından oluşur. Her işletim sistemi sınıfı bir satıcı, işletim sistemi ailesi, işletim sistemi nesli, cihaz türü ve sınıf için bir dizi CPE girişinden oluşur. (İşletim sistemi eşleşme alanlarının açıklaması için "Referans Parmak İzi Formatının Kodunu Çözme" bölümüne bakın). Tanımlanmamışlara alanlar nil olabilir. host.os tablosu bu genel yapıya sahiptir:

```
host.os = {
{
    name =<string>,
    classes = {
    {
        vendor =<string>,
        osfamily =<string>,
        osgen =<string>,
        type =<string>,
        cpe = {
            "cpe:/<...>",
            [More CPE]
        }
    },
    [More classes]
},
},
[More OS matches]
}
```

Örneğin, bu nmap-os-db girişinde bir işletim sistemi eşleşmesi:

```
Fingerprint Linux 2.6.32 - 3.2
Class Linux | Linux | 2.6.X | general purpose
CPE cpe:/o:linux:linux_kernel:2.6
Class Linux | Linux | 3.X | general purpose
CPE cpe:/o:linux:linux_kernel:3
```

bu host.os tablosuyla sonuçlanacaktır:

```
host.os = {
{
    name = "Linux 2.6.32 - 3.2",
    classes = {
    {
        vendor = "Linux",
        osfamily = "Linux",
        osgen = "2.6.X",
        type = "general purpose",
        cpe = { "cpe:/o:linux:linux_kernel:2.6" }
},
{
        vendor = "Linux",
        osfamily = "Linux",
        osgen = "3.X",
        type = "general purpose",
        cpe = { "cpe:/o:linux:linux_kernel:3" }
    }
}
```

```
        }
    },
}
```

Sadece mükemmel işletim sistemi eşleşmelerine karşılık gelen girdiler host.os tablosuna yerleştirilir. Nmap -O seçeneği olmadan çalıştırıldığında, host.os nil olur.

`host.ip` ⇒ Hedef ana bilgisayarın IP adresinin bir dize gösterimini içerir. Tarama bir ana bilgisayar adına karşı çalıştırıldığında ve DNS araması birden fazla IP adresi döndürdüğünde, tarama için seçilen IP adresinin aynısı kullanılır.

`host.name` ⇒ Taranan hedef ana bilgisayarın dize olarak gösterilen ters DNS girişini içerir. Ana bilgisayarın ters DNS giriş yoksa, alanın değeri boş bir dizedir.

`host.targetname` ⇒ Komut satırında belirtlen ana bilgisayarın adını içerir. Komut satırında verilen hedef bir netmask içeriyorsa veya bir IP adresi ise alanın değeri nil olur.

`host.reason` ⇒ Hedef ana bilgisayarın geçerli durumunda olmasının nedeninin bir dize gösterimini içerir. Sebep, durumu belirleyen paketin türü tarafından verilir. Örneğin, canlı bir ana bilgisayardan gelen bir yanıt yanıtı.

`host.reason_ttl` ⇒ Hedef ana bilgisayarın durumunu belirlemek için kullanılan yanıt paketinin geldiği zamanki TTL değerini içerir. Bu yanıt paketi, host.reason öğesini ayarlamak için de kullanılan pakettir.

`host.directly_connected` ⇒ Hedef ana bilgisayarın Nmap çalıştırılan ana bilgisayara doğrudan bağlı olup olmadığını (yani aynı ağ segmentinde olup olmadığını) gösteren bir Boolean değeri.

`host.mac_addr` ⇒ Varsa hedef ana bilgisayarın MAC adresi (altı bayt uzunlığında ikili dize), aksi takdirde nil. MAC adresi genellikle yalnızca bir LAN'a doğrudan bağlı ana bilgisayarlar için ve yalnızca Nmap SYN taraması gibi bir ham paket taraması yapıyorsa kullanılabilir.

`host.mac_addr_next_hop` ⇒ Ana bilgisayara giden rotadaki ilk atlamanın MAC adresi veya mevcut değilse nil.

`host.mac_addr_src` ⇒ Ana bilgisayara bağlanmak için kullanılan kendi MAC adresimiz (ya ağ kartımızın ya da (--spoof-mac ile) sahte adres).

`host.interface` ⇒ Ana bilgisayara paketlerin gönderildiği arabirim adını (dnet tarzı) içeren bir dize.

`host.interface_mtu` ⇒ Host.interface için MTU (maksimum iletim birimi) veya bilinmiyorsa 0.

`host.bin_ip` ⇒ Hedef ana bilgisayarın IP adresi 4 baytlık (IPv4) veya 16 baytlık (IPv6) bir dize olarak.

`host.bin_ip_src` ⇒ Ana bilgisayamızın (Nmap çalıştırılan) kaynak IP adresi 4 baytlık (IPv4) veya 16 baytlık (IPv6) bir dize olarak.

`host.times` ⇒ Bu tablo Nmap'in ana bilgisayar için zamanlama verilerini içerir ("Gidiş Dönüş Süresi Tahmini" adlı bölümde bakın). Anahtarları srtt (düzeltilmiş gidiş-dönüş süresi), rttvar (gidiş-dönüş süresi varyansı) ve timeout (prob zaman aşımı) olup, hepsi kayan noktalı saniye cinsinden verilmiştir.

`host.traceroute` ⇒ Bu, --traceroute seçeneği kullanıldığından mevcut olan traceroute atlamalarının bir dizisidir. Her girdi, ad, ip ve srtt (gidiş-dönüş süresi) alanlarına sahip bir konak tablosudur. Bir giriş için TTL, tablodaki konumu göz önüne alındığında örtüktür. Boş bir tablo zaman aşımına uğramış bir atlamayı temsil eder.

`host.os_fp` ⇒ İşletim sistemi algılaması yapıldıysa, bu, ana bilgisayar için işletim sistemi parmak izini içeren bir dizedir. Biçim "Nmap Parmak İzini Anlamak" başlıklı bölümde açıklanmıştır.

`port` ⇒ Bağlantı noktası tablosu, ana bilgisayar tablosuyla aynı şekilde bir NSE hizmet betiğine (yani yalnızca ana bilgisayar yerine bağlantı noktası kuralına sahip olanlara) aktarılır. Komut dosyasının çalıştığı bağlantı

noktası hakkında bilgi içerir. Bu tablo ana bilgisayar betiklerine aktarılmazken, hedef üzerindeki bağlantı noktası durumları nmap.get_port_state() ve nmap.get_ports() çağrıları kullanılarak Nmap'ten talep edilebilir.

`port.number` ⇒ Hedef portun port numarasını içerir.

`port.protocol` ⇒ Hedef portun protokolünü tanımlar. Geçerli değerler "tcp" ve "udp" dir.

`port.service` ⇒ Nmap hizmet tespiti tarafından tespit edildiği şekliyle port.number üzerinde çalışan hizmetin bir dize gösterimini içerir. Eğer port.version.service_dtype alanı "table" ise, Nmap port numarasına göre servisi tahmin etmiştir. Aksi takdirde, sürüm algılama dinleme hizmetini belirleyebildi ve bu alan port.version.name'e eşittir.

`port.reason` ⇒ Hedef bağlantı noktasının mevcut durumunda (port.state tarafından verilen) olmasının nedeninin bir dize gösterimini içerir. Sebep, durumu belirleyen paketin türüne göre verilir. Örneğin, kapalı bir bağlantı noktasından gelen bir RST paketi veya açık bir bağlantı noktasından gelen SYN-ACK.

`port.reason_ttl` ⇒ Hedef portun durumunu belirlemek için kullanılan yanıt paketinin geldiğinde TTL değerini içerir. Bu yanıt paketi, port.reason'ı ayarlamak için de kullanılan pakettir.

`port.version` ⇒ Bu girdi, Nmap sürüm tarama motoru tarafından alınan bilgileri içeren bir tablodur. Bazı değerler (hizmet adı, hizmet türü güveni ve RPC ile ilgili değerler gibi) bir sürüm taraması yapılmamış olsa bile Nmap tarafından alınabilir. Belirlenmemiş olan değerler varsayılan olarak nil'dir. Her bir değerin anlamı aşağıdaki tabloda verilmiştir:

Tablo 9.1. port.version değerleri

Name (İsim)	Description (Açıklama)
<code>name</code>	Nmap'in bağlantı noktası için karar verdiği hizmet adını içerir.
<code>name_confidence</code>	Nmap'in adın doğruluğu konusunda ne kadar emin olduğunu 1 (en az emin) ile 10 arasında değerlendirir. Eğer port.version.service_dtype "table" ise, bu 3'tür.
<code>product</code> , <code>version</code> , <code>extrainfo</code> , <code>hostname</code> , <code>ostype</code> , <code>devicetype</code>	Bu beş değişken <versioninfo> altında "match Directive" bölümünde açıklananlarla aynıdır.
<code>service_tunnel</code>	Nmap'in hizmeti tespit etmek için SSL tünelleme kullanıp kullanmadığına bağlı olarak "none" veya "ssl" dizesini içerir.
<code>service_fp</code>	Varsa, hizmet parmak izi bu değerde sağlanır. Bu, "Topluluk Katkıları" adlı bölümde açıklanmaktadır.
<code>service_dtype</code>	Nmap'in nmap-services dosyasından veya bir servis prob eşleşmesinden port.version.name'i çıkarıp çıkarmadığına bağlı olarak "table" veya "probed" dizesini içerir.
<code>cpe</code>	Tespit edilen hizmet için CPE kodlarının listesi. Resmi CPE spesifikasyonunda açıklanıldığı gibi, bu dizelerin tümü cpe:/ önekiyle başlar.

`port.state` ⇒ Portun durumu hakkında bilgi içerir. Hizmet komut dosyaları yalnızca açık veya açık|filtrelenmiş durumda bağlantı noktalarına karşı çalıştırılır, bu nedenle port.state genellikle bu değerlerden birini içerir. Port tablosu get_port_state veya get_ports fonksiyonlarının bir sonucuya başka değerler de görünebilir. Port durumunu nmap.set_port_state() çağrısını kullanarak ayarlayabilirsiniz. Bu normalde açık|filtrelenmiş bir portun açık olduğu belirlendiğinde yapılır.

Network I/O API (Ağ I/O API'si)

Verimli ve paralelleştirilebilir ağ G/C'sine izin vermek için NSE, Nmap soket kütüphanesi olan Nsock'a bir arayüz sağlar. Nsock'un kullandığı akıllı geri arama mekanizması NSE komut dosyalarına tamamen şeffaftır. NSE'nin soketlerinin temel faydası, G/C işlemlerinde asla bloke olmamaları ve birçok komut dosyasının paralel

olarak çalıştırılmasına izin vermeleridir. G/Ç paralelliği NSE komut dosyası yazarları için tamamen şeffaftır. NSE'de ya bloklama yapmayan tek bir soket kullanılmış gibi ya da bağlantınız bloklama yapıyormuş gibi programlayabilirsiniz. Engelleyen G/Ç çağrıları bile belirli bir zaman aşımı aşıldığında geri döner. İki tür Ağ G/Ç'si desteklenir: bağlantı tarzı ve ham paket.

Connect-style network I/O (Connect tarzı ağ I/O)

Ağ API'sinin bu kısmı çoğu klasik ağ kullanımı için uygun olmalıdır: Kullanıcılar bir soket oluşturur, uzak bir adrese bağlar, veri gönderir ve alır ve son olarak soketi kapatır. Taşıma katmanına (TCP, UDP veya SSL) kadar olan her şey kütüphane tarafından gerçekleştirilir.

Bir NSE soketi, bir soket nesnesi döndüren nmap.new_socket çağrıları yapılarak oluşturulur. Soket nesnesi olağan bağlanma, gönderme, alma ve kapatma yöntemlerini destekler. Ayrıca receive_bytes, receive_lines ve receive_buf fonksiyonları veri alımı üzerinde daha fazla kontrol sağlar. Örnek 9.3, connect tarzı ağ işlemlerinin kullanımını göstermektedir. try fonksiyonu, "İstisna İşleme" bölümünde açıklandığı gibi hata işleme için kullanılır.

Örnek 9.3. Bağlantı tarzı G/Ç

```
require("nmap")

local socket = nmap.new_socket()
socket:set_timeout(1000)
try = nmap.new_try(function() socket:close() end)
try(socket:connect(host.ip, port.number))
try(socket:send("login"))
response = try(socket:receive())
socket:close()
```

Raw packet network I/O (Ham paket ağ G/Ç)

Bağlantı odaklı yaklaşımın çok üst düzey olduğu durumlar için NSE, kod geliştiricilere ham paket ağ G/Ç seçeneği sunar.

Ham paket alımı, Nsock kütüphanesi içindeki bir Libpcap sarmalayıcı aracılığıyla gerçekleştiriliyor. Adımlar, bir yakalama aygıtı açmak, dinleyicileri aygıtına kaydetmek ve ardından paketleri alındıkça işlemektir.

pcap_open yöntemi, sıradan bir soket nesnesinden ham soket okumaları için bir tanıtıcı oluşturur. Bu yöntem, bir paketten (başlıklar dahil) bir paket karması hesaplayan bir geri arama işlevi alır. Bu karma, daha sonra pcap_register işleviyle kaydedilen dizelerle karşılaştırılan herhangi bir ikili dizeyi döndürebilir. Paket hash geri çağrıları normalde paketin kaynak adresi gibi bir kısmını çıkaracaktır.

Pcap okuyucusuna pcap_register fonksiyonu kullanılarak belirli paketleri dinlemesi söylenir. Fonksiyon, alınan her paketin hash değeriyle karşılaştırılan ikili bir dizgi alır. Hash'leri kayıtlı dizelerle eşleşen paketler pcap_receive yöntemi tarafından döndürülecektir. Tüm paketleri almak için boş dizeyi kaydedin.

Bir kod, pcap_receive yöntemini çağırarak bir dinleyicinin kaydedildiği tüm paketleri alır. Yöntem, bir paket alınana veya bir zaman aşımı gerçekleşene kadar bloklanır.

Paket hash hesaplama işlevi ne kadar genel tutulursa, o kadar çok betik paketi alabilir ve yürütmeye devam edebilir. Kodunuzun içinde paket yakalamayı işlemek için önce nmap.new_socket ile bir soket oluşturmanız ve daha sonra soketi socket_object:close ile kapatmanız gereklidir - tipki bağlantı tabanlı ağ I/O'sunda olduğu gibi.

Paketleri almak önemli olsa da, onları göndermek de kesinlikle önemli bir özelliktir. Bunu başarmak için NSE, IP ve Ethernet katmanlarında göndermeye erişim sağlar. Ham paket yazma işlemleri ham paket okuma işlemleri ile

aynı soket nesnesini kullanmaz, bu nedenle nmap.new_dnet işlevi gönderme için gerekli nesneyi oluşturmak üzere çağrılar. Bundan sonra, bir ham soket veya Ethernet arayüz tanıtıcısı kullanım için açılabilir.

dnet nesnesi oluşturulduktan sonra, IP gönderimi için nesneyi başlatmak üzere ip_open işlevi çağrılabılır. ip_send, IP başlığıyla başlaması gereken gerçek ham paketi gönderir. dnet nesnesi, hangi IP ana bilgisayarlarına gönderilebileceği konusunda herhangi bir kısıtlama getirmez, bu nedenle aynı nesne açıkken birçok farklı ana bilgisayara göndermek için kullanılabilir. Ham soketi kapatmak için ip_close çağrı yapın.

IP'den daha düşük seviyede gönderme için, NSE Ethernet çerçeveleri yazmak için fonksiyonlar sağlar. ethernet_open bir Ethernet arayüzü açarak gönderme için dnet nesnesini başlatır. Ham çerçeve ethernet_send ile gönderilir. Tanıtıcıyı kapatmak için ethernet_close çağrı yapılır.

IP'den daha düşük seviyede gönderme için, NSE Ethernet çerçeveleri yazmak için fonksiyonlar sağlar. ethernet_open bir Ethernet arayüzü açarak gönderme için dnet nesnesini başlatır. Ham çerçeve ethernet_send ile gönderilir. Tanıtıcıyı kapatmak için ethernet_close çağrı yapılır.

Bazen karmaşık API'leri anlamadan en kolay yolu örnek vermektedir. Nmap ile birlikte gelen ipidseq betiği, ana bilgisayarı Nmap'in Idle Scan (-sl) özelliğine uygunluk açısından test etmek için ham IP paketlerini kullanır. Yine Nmap ile birlikte gelen sniffer-detect betiği, ağdaki karışık modlu makineleri (sniffer çalışanları) tespit etmek için ham Ethernet çerçevelerini kullanır.

Structured and Unstructured Output (Yapilandırılmış ve Yapılandırılmamış Çıktı)

NSE komut dosyaları genellikle çıktılarını temsil eden, güzel bir şekilde organize edilmiş ve özenle seçilmiş anahtarlarla sahip bir tablo döndürmelidir. Böyle bir tablo ekran çıktısı için otomatik olarak biçimlendirilecek ve XML çıktısında iç içe geçmiş öğeler olarak saklanacaktır. XML çıktısının mantıksal olarak anahtarlarla ve değerlere ayrılması, diğer araçların kod çıktısını kullanmasını kolaylaştırır. Bir betığın yalnızca bir dize döndürmesi mümkün değildir, ancak bunu yapmak kullanımından kaldırılmıştır. Geçmişte, betikler yalnızca bir dize döndürebiliyor ve çıktıları basitçe bir metin bloğu olarak XML'e kopyalayıyordu - bu artık "yapilandırılmamış çıktı" olarak bilinmektedir.

user-list adlı bir kodun bu kod örneğinde gösterildiği gibi bir tablo döndürdüğü varsayılmı. Aşağıdaki paragraflarda normal ve XML çıktısında nasıl göründüğü gösterilmektedir.

```
local output = stdnse.output_table()
output.hostname = "slimer"
output.users = {}
output.users[#output.users + 1] = "root"
output.users[#output.users + 1] = "foo"
output.users[#output.users + 1] = "bar"
return output
```

Bir Lua tablosu normal çıktı için bir dizeye dönüştürülür. Bunun çalışma şekli şöyledir: her iç içe tablo yeni bir girinti seviyesi alır. Dize anahtarlı tablo girişlerinden önce anahtar ve iki nokta üst üste gelir; tamsayı anahtarlı girişler basitçe sırayla görünür. Sırasız olan normalLua tablolarının aksine, stdnse.output_table'dan gelen bir tablo, anahtarlarını eklendikleri sırada tutacaktır. Örnek 9.4, "NSE yapılandırılmış çıktısının otomatik biçimlendirilmesi", örnek tablonun normal çıktıda nasıl göründüğünü gösterir.

Örnek 9.4. NSE yapılandırılmış çıktısının otomatik biçimlendirilmesi

```

PORT      STATE SERVICE
1123/tcp open  unknown
| user-list:
|   hostname: slimer
|   users:
|     root
|     foo
|     bar

```

Bir Lua tablosunun XML gösterimi aşağıdaki gibi oluşturulur. İç içe tablolar tablo elemanları olur. Kendileri tablo olmayan tabloların girişleri elem elemanı olur. Dize anahtarlı girdiler (ister tablo ister eleman olsun) bir anahtar niteliği alır (örneğin <elem key="username">foo</elem>); tamsayı anahtarlı girdilerin anahtar ögesi yoktur ve anahtarları gördündükleri sırada örtütür.

Yukarıdakilere ek olarak, kodun ürettiği normal çıktı (otomatik olarak oluşturulmuş olsa bile) kod ögesinin çıktı niteliğine kopyalanır. Yeni satırlar ve diğer özel karakterler, örneğin gibi XML karakter varlıklarını olarak kodlanacaktır. Örnek 9.5, "XML'de NSE yapılandırılmış çıktısı" örnek tablonun XML'de nasıl görüneceğini gösterir.

Örnek 9.5. XML'de NSE yapılandırılmış çıktısı

```

<script id="t" output="&#xa;hostname: slimer&#xa;users: &#xa;  root&#xa;  foo&#xa;  bar">
  <elem key="hostname">slimer</elem>
  <table key="users">
    <elem>root</elem>
    <elem>foo</elem>
    <elem>bar</elem>
  </table>
</script>

```

Bazı komut dosyaları normal çıktılarından daha fazla kontrole ihtiyaç duyar. Örneğin, karmaşık tabloları görüntülemesi gereken komut dosyaları için durum böyledir. Çıktı üzerinde tam kontrol için, bu komut dosyaları aşağıdakilerden birini yapabilir:

ikinci dönüş değeri olarak bir dize döndürür veya

döndürülen tablo üzerinde __tostring metodunu ayarlar.

Elde edilen dize normal çıktıda kullanılacak ve tablo her zamanki gibi XML'de kullanılacaktır. Biçimlendirilmiş dize, birden fazla satır olarak görünmesi için satırsonu karakterleri içerebilir.

Yukarıdaki kod örneği biçimlendirilmiş bir dize döndürmek için bu şekilde değiştirilirse,

```

local output = stdnse.output_table()
output.hostname = "slimer"
output.users = {}
output.users[#output.users + 1] = "root"
output.users[#output.users + 1] = "foo"
output.users[#output.users + 1] = "bar"
local output_str = string.format("hostname: %s\n", output.hostname)
output_str = output_str .. "\n" .. stringaux.strjoin(", ", output.users)
return output, output_str

```

o zaman normal çıktı aşağıdaki gibi görünecektir:

```
PORT      STATE SERVICE
1123/tcp  open  unknown
| user-list:
|   hostname: slimer
|_  users: root, foo, bar
```

Yapılardırılmış çıktıda belirli veri türlerinin biçimlendirilmesine ilişkin kurallar vardır. NSE çıktıları kullanıcıları, tarih ve saat gibi bazı veri türlerinin farklı komut dosyalarında bile aynı şekilde biçimlendirildiğini varsayıarak faydalı sağlar.

Ağ adresleri, örneğin IPv4, IPv6 ve MAC, dizeler olarak gösterilir.

Açık anahtar parmak izleri gibi uzun onaltılık dizeler küçük harfli alfabetik karakterler kullanılarak ve iki nokta üst üste gibi ayırcılar olmadan yazılmalıdır.

Tarihler ve saatler RFC 3339'a göre biçimlendirilir. Saat dilimi ofseti biliniyorsa, bu örneklerdeki gibi görünmelidirler:

```
2012-09-07T23:37:42+00:00
2012-09-07T23:37:42+02:00
```

Saat dilimi ofseti bilinmiyorsa (belirtilmemiş bir yerel saatı temsil ediyorsa), ofset kısmını atlayın:

```
2012-09-07T23:37:42
```

Kütüphane işlevi `datetime.format_timestamp` kodu, yapılandırılmış çıktı için zamanları biçimlendirmek üzere mevcuttur. Saniye cinsinden isteğe bağlı bir zaman dilimi ofseti alır ve tarihi otomatik olarak bu ofset içinde doğru olacak şekilde kaydırır.

```
datetime.format_timestamp(os.time(), 0) --> "2012-09-07T23:37:42+00:00"
```

Exception Handling (İstisna İşleme)

NSE, temel Lua dilinde bulunmayan bir istisna işleme mekanizması sağlar. Özellikle ağ I/O işlemleri için uyarlanmıştır ve nesne yönelimli bir paradigmadan ziyade işlevsel bir programlama paradigmmasını takip eder. Bir istisna işleyicisi oluşturmak için `nmap.new_try` API yöntemi kullanılır. Bu yöntem, başka bir işlevin dönüş değerleri olduğu varsayılan değişken sayıda bağımsız değişken alan bir işlev döndürür. Dönüş değerlerinde bir istisna tespit edilirse (ilk dönüş değeri `false` ise), kod yürütmesi iptal edilir ve hiçbir çıktı üretilemez. İsteğe bağlı olarak, `new_try` öğesine bir istisna yakalandığında çağrılabilecek bir işlev aktarabilirsiniz. Bu fonksiyon genellikle gerekli temizleme işlemlerini gerçekleştirir.

Örnek 9.6'da temizleme istisnalarının işlenmesi gösterilmektedir. Bir hata durumunda yeni oluşturulan soketi kapatmak için `catch` adında yeni bir fonksiyon tanımlanmıştır. Daha sonra bu soket üzerindeki bağlantı ve iletişim girişimlerini korumak için kullanılır. Herhangi bir `catch` fonksiyonu belirtilmezse, betiğin yürütülmesi daha fazla uzatılmadan iptal edilir-açık soketlerLua'nın çöp toplayıcısının bir sonraki çalışmasına kadar açık kalacaktır. Eğer `verbosity` seviyesi en az bir ise veya tarama hata ayıklama modunda yapılıyorsa, yakalanmamış hata durumunun bir açıklaması standart çıktıya yazdırılır. Şu anda birkaç deyimi tek bir `try` bloğunda graplamanın kolaylıkla mümkün olmadığını unutmayın.

Örnek 9.6. İstisna işleme örneği

```
local result, socket, try, catch
```

```

result = ""
socket = nmap.new_socket()
catch = function()
socket:close()
end
try = nmap.new_try(catch)

try(socket:connect(host.ip, port.number))
result = try(socket:receive_lines(1))
try(socket:send(result))

```

try/catch mekanizması tarafından düzgün bir şekilde ele alınan bir fonksiyon yazmak basittir. Fonksiyon birden fazla değer döndürmeliidir. İlk değer, fonksiyon başarılı bir şekilde tamamlandığında true, aksi takdirde false (veya nil) olan bir Boolean olmalıdır. İşlev başarıyla tamamlandırsa, try yapısı gösterge değerini tüketir ve kalan değerleri döndürür. İşlev başarısız olduysa, döndürülen ikinci değer hata durumunu açıklayan bir dize olmalıdır. Değer nil veya false değilse true olarak değerlendirilir, bu nedenle normal durumda değerinizi döndürebilir ve bir hata oluşursa nil, <hata açıklaması> döndürebilirsiniz.

The Registry (Kayıt Defteri)

Komut dosyaları, değerleri tüm komut dosyaları tarafından erişilebilen özel bir tablo olan bir kayıt defterinde depolayarak bilgi paylaşabilir. Tüm komut dosyaları tarafından paylaşılan nmap.registry adında global bir kayıt defteri vardır. Her ana bilgisayarın ayrıca host.registry adında kendi kayıt defteri vardır; burada host, bir betiğe aktarılan ana bilgisayar tablosudur. Kayıt defterlerindeki bilgiler Nmap yürütümleri arasında saklanmaz.

Genel kayıt defteri tüm tarama oturumu boyunca devam eder. Komut dosyaları, örneğin daha sonra bir kural sonrası komut dosyası tarafından görüntülenecek değerleri depolamak için bunu kullanabilir. Öte yandan, ana bilgisayar başına kayıtlar yalnızca bir ana bilgisayar taranırken var olur. Bir komut dosyasından aynı ana bilgisayarda çalışan başka bir komut dosyasına bilgi göndermek için kullanılabilirler. Mümkün olduğunda, ana bilgisayar başına kayıt defterini kullanın; bu sizi yalnızca anahtar adlarını ana bilgisayarlar arasında benzersiz yapmak zorunda kalmaktan kurtarmakla kalmaz, aynı zamanda kayıt defteri tarafından kullanılan belleğin artık ihtiyaç duyulmadığında geri alınmasına da olanak tanır.

İşte her iki kayıt defterinin kullanımına ilişkin örnekler:

ssh-hostkey betiğinin portrule'ü SSH anahtar parmak izlerini toplar ve daha sonra postrule tarafından yazdırılabilmeleri için bunları global nmap.registry'de saklar.

ssl-cert betiği SSL sertifikalarını toplar ve bunları ana bilgisayar başına kayıt defterinde saklar, böylece ssl-google-cert-catalog betiği sunucuya başka bir bağlantı yapmak zorunda kalmadan bunları kullanabilir.

Her komut dosyası genel kayıt defteri tablosuna yazabildiğinden, diğer komut dosyalarının (veya paralel olarak çalışan aynı komut dosyasının) anahtarlarının üzerine yazılmasını önlemek için kullandığınız anahtarları benzersiz yapmak önemlidir.

Başka bir komut dosyasının sonuçlarını kullanan komut dosyaları, önceki komut dosyasının önce çalıştığından emin olmak için dependencies değişkenini kullanarak bunu bildirmelidir.

Script Writing Tutorial (Senaryo Yazma Eğitimi)

Diyelim ki NSE'nin gücüne ikna oldunuz. Kendi komut dosyanızı nasıl yazarsınız? Diyelim ki bir TCP portunu dinleyen sürecin sahibini belirlemek için bir identd sunucusundan bilgi almak istiyorsunuz. Bu aslında identd'nin

amaçları değildir (giden bağlantıların sahibini sorgulamak için, dinleyen daemonları değil), ancak birçok identd sunucusu yine de bunu izin verir. Nmap eskiden bu işlevi sahipti (ident scan olarak adlandırılır), ancak yeni bir tarama motoru mimarisine geçerken kaldırıldı. identd'nin kullandığı protokol oldukça basittir, ancak yine de Nmap'in sürüm algılama dili ile başa çıkmak için çok karmaşıktır. İlk olarak, tanımlama sunucusuna bağlanırsınız ve <port-on-server>, <port-on-client> şeklinde bir sorgu gönderirsiniz ve bir satırsonu karakteri ile sonlandırırsınız. Sunucu daha sonra sunucu bağlantı noktası, istemci bağlantı noktası, yanıt türü ve adres bilgilerini içeren bir dizeyle yanıt vermelidir. Bir hata varsa adres bilgisi atlanır. RFC 1413'te daha fazla ayrıntı mevcuttur, ancak bu açıklama bizim amaçlarımız için yeterlidir. Protokol, iki nedenden dolayı Nmap'in sürüm tespit dilinde modellenemez. Birincisi, bir bağlantının hem yerel hem de uzak bağlantı noktasını bilmeniz gereklidir. Sürüm tespiti bu verileri sağlamaz. İkinci ve daha ciddi engel ise hedefe iki açık bağlantıya ihtiyacınız olmasıdır; biri tanımlama sunucusuna diğerinin de sorgulamak istediğiniz dinleme portuna. Her iki engel de NSE ile kolayca aşılabilir.

Bir komut dosyasının anatomisi "Komut Dosyası Formatı" adlı bölümde açıklanmıştır. Bu bölümde, açıklanan yapının nasıl kullanıldığını göstereceğiz.

The Head (Kafa)

Komut dosyasının başı esasen meta bilgisidir. Bu, şu alanları içerir: açıklama, kategoriler, bağımlılıklar, yazar ve lisansın yanı sıra kullanım, args ve çıktı etiketleri gibi ilk NSEDoc bilgileri ("Kod Dokümantasyonu Yazma (NSEDoc)" başlıklı bölüme bakın).

Açıklama alanı, kodun ne yaptığını açıklayan bir veya daha fazla paragraf içermelidir. Kod sonuçlarıyla ilgili herhangi bir şey kullanıcıların kafasını karıştırabilir veya onları yanlış yönlendirebilirse ve bu sorunu kodu veya sonuç metnini iyileştirmek ortadan kaldırılamazsanız, bu durum açıklamada belgelenmelidir. Birden fazla paragraf varsa, ilk paragraf gerektiğinde kısa bir özeti olarak kullanılır. İlk paragrafin tek başına bir özeti olarak kullanılabilcecinden emin olun. Bu açıklama kısa çünkü çok basit bir senaryo:

```
description = [[  
Attempts to find the owner of an open TCP port by querying an auth  
(identd - port 113) daemon which must also be open on the target system.  
]]
```

Sırada NSEDoc bilgileri var. Bu kod çok basit olduğu için yaygın @usage ve @args etiketlerini içermez, ancak bir NSEDoc @output etiketine sahiptir:

```
---  
--@output  
-- 21/tcp open  ftp    ProFTPD 1.3.1  
-- |_auth-owners: nobody  
-- 22/tcp open  ssh    OpenSSH 4.3p2 Debian 9etch2 (protocol 2.0)  
-- |_auth-owners: root  
-- 25/tcp open  smtp   Postfix smtspd  
-- |_auth-owners: postfix  
-- 80/tcp open  http   Apache httpd 2.0.61 ((Unix) PHP/4.4.7 ...)  
-- |_auth-owners: dhttpd  
-- 113/tcp open  auth?  
-- |_auth-owners: nobody  
-- 587/tcp open  submission Postfix smtspd  
-- |_auth-owners: postfix
```

```
-- 5666/tcp open  unknown  
-- |_ auth-owners: root
```

Ardından yazar, lisans ve kategoriler etiketleri geliyor. Bu betik kasaya aittir çünkü hizmeti amaçlanmadığı bir şey için kullanmıyoruz. Bu betik varsayılan olarak çalışması gereken bir betik olduğu için aynı zamanda varsayılan kategorisindedir. İşte bağlam içindeki değişkenler:

```
author = "Diman Todorov"  
  
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"  
  
categories = {"default", "safe"}
```

The Rule (Kural)

Kural bölümü, betığın eyleminin atlanması veya yürütülmesine karar veren bir Lua yöntemidir. Bu karar genellikle kuralın türüne ve kendisine iletilen ana bilgisayar ve bağlantı noktası bilgilerine dayanır. Bir prerule veya postrule her zaman true olarak değerlendirilecektir. Tanımlama betiği söz konusu olduğunda durum bundan biraz daha karmaşıktır. Tanımlama betığının belirli bir bağlantı noktasına karşı çalıştırılıp çalıştırılmayacağına karar vermek için hedef makinede çalışan bir auth sunucusu olup olmadığını bilmemiz gereklidir. Başka bir deyişle, betik yalnızca o anda taranan TCP bağlantı noktasını açıksa ve TCP bağlantı noktası 113 de açıksa çalıştırılmalıdır. Simdilik kimlik sunucularının TCP port 113'ü dinlediği gerçeğine güveneceğiz. Ne yazık ki NSE bize sadece o anda taranan port hakkında bilgi veriyor.

113 numaralı portun açık olup olmadığını öğrenmek için nmap.get_port_state fonksiyonunu kullanırız. Eğer auth portu taranmamışsa get_port_state fonksiyonu nil döndürür. Bu yüzden tablonun nil olmadığını kontrol ediyoruz. Ayrıca her iki portun da açık durumda olup olmadığını kontrol ediyoruz. Eğer durum böyleyse eylem gerçekleştiriliyor, aksi takdirde eylemi atlarız.

```
portrule = function(host, port)  
    local auth_port = { number=113, protocol="tcp" }  
    local identd = nmap.get_port_state(host, auth_port)  
  
    return identd ~= nil  
        and identd.state == "open"  
        and port.protocol == "tcp"  
        and port.state == "open"  
end
```

The Action (Eylem)

Sonunda gerçek işlevselliği uyguluyoruz! Betik önce tanımlama sunucusunu bulmayı beklediğimiz porta bağlanır, daha sonra hakkında bilgi istediğimiz porta bağlanır. Bunu yapmak için önce nmap.new_socket'i çağırarak iki soket seçeneği oluşturuyoruz. Daha sonra, hata tespit edildiğinde bu soketleri kapatın bir hata yakalama fonksiyonu tanımlarız. Bu noktada ağ soketi üzerinde işlem yapmak için open, close, send ve receive gibi nesne yöntemlerini güvenle kullanabiliriz. Bu durumda bağlantıları yapmak için connect'i çağırırız. Aşırı hata işleme kodundan kaçınmak için NSE'nin istisna işleme mekanizması kullanılır. Ağ çağrılarını basitçe bir try çağrısına sarıyoruz ve bu çağrı da herhangi bir şey yanlış giderse catch fonksiyonumuzu çağırıyor.

İki bağlantı başarılı olursa, bir soru dizesi oluşturur ve yanıt ayırtırız. Tatmin edici bir yanıt alırsak, alınan bilgileri döndürürüz.

```

action = function(host, port)
    local owner = ""

    local client_ident = nmap.new_socket()
    local client_service = nmap.new_socket()

    local catch = function()
        client_ident:close()
        client_service:close()
    end

    local try = nmap.new_try(catch)

    try(client_ident:connect(host.ip, 113))
    try(client_service:connect(host.ip, port.number))

    local localip, localport, remoteip, remoteport =
        try(client_service:get_info())

    local request = port.number .. ", " .. localport .. "\r\n"

    try(client_ident:send(request))

    owner = try(client_ident:receive_lines(1))

    if string.match(owner, "ERROR") then
        owner = nil
    else
        owner = string.match(owner,
            "%d+%s*,%s*%d+%s*:;%s*USERID%s*:;%s*.+%s*:;%s*(.+)\r?\n")
    end

    try(client_ident:close())
    try(client_service:close())

    return owner
end

```

Uzak bağlantı noktasının port.number'da saklandığını bildiğimiz için, client_service:get_info() işlevinin son iki dönüş değerini bu şekilde yok sayabileceğimizi unutmayın:

```
local localip, localport = try(client_service:get_info())
```

Bu örnekte, hizmet bir hata ile yanıt verirse sessizce çıkıyoruz. Bu, döndürülecek olan sahip değişkenine nil atanarak yapılır. NSE komut dosyaları genellikle yalnızca başarılı olduklarında mesaj döndürür, böylece kullanıcıyı anlamsız uyarılarla doldurmazlar.

Writing Script Documentation (NSEDoc) (Kod Dokümantasyonu Yazma (NSEDoc))

Komut dosyaları yazarlarından daha fazlası tarafından kullanılır, bu nedenle iyi belgelendirilmeleri gereklidir. NSE modülleri, geliştiricilerin bunları komut dosyalarında kullanabilmeleri için belgelendirmeye ihtiyaç duyar. NSE'nin bu bölümde açıklanan dokümantasyon sistemi bu iki ihtiyacı da karşılamayı amaçlamaktadır. Bu bölüm okurken, bu sistem kullanılarak oluşturulan NSE'nin çevrimiçi belgelerine göz atmak isteyebilirsiniz. <https://nmap.org/nsedoc/> adresinde bulunmaktadır.

NSE, NSEDoc adı verilen LuaDoc dokümantasyon sisteminin özelleştirilmiş bir versiyonunu kullanır. Kodlar ve modüller için dokümantasyon, özel bir forma sahip yorumlar olarak kaynak kodlarında bulunur. Örnek 9.7, stdnse.print_debug() fonksiyonundan alınmış bir NSEDoc yorumudur.

Örnek 9.7. Bir fonksiyon için NSEDoc yorumu

```
---  
-- Prints a formatted debug message if the current verbosity level is greater  
-- than or equal to a given level.  
--  
-- This is a convenience wrapper around  
-- <code>nmap.log_write</code>. The first optional numeric  
-- argument, <code>level</code>, is used as the debugging level necessary  
-- to print the message (it defaults to 1 if omitted). All remaining arguments  
-- are processed with Lua's <code>string.format</code> function.  
-- @param level Optional debugging level.  
-- @param fmt Format string.  
-- @param ... Arguments to format.
```

Dokümantasyon yorumları üç çizgi ile başlar: ---. Yorumun gövdesi aşağıdaki kodun açıklamasıdır. Açıklamanın ilk paragrafi kısa bir özeti olmalıdır, sonraki paragraflar daha fazla ayrıntı sağlamağıdır. İle başlayan özel etiketler belgenin diğer bölümlerini işaretler. Yukarıdaki örnekte, bir fonksiyonun her bir parametresini tanımlamak için kullanılan @param etiketini görüyorsunuz. Dokümantasyon etiketlerinin tam listesi "NSE Dokümantasyon Etiketleri" adlı bölümde bulunabilir.

HTML benzeri `<>` ve `</>` etiketleri içine alınmış metinler monospace yazı tipinde işlenecektir. Bu, değişken ve işlev adlarının yanı sıra çok satırlı kod örnekleri için kullanılmalıdır. Bir dizi satır "*" karakterleriyle başladığında, madde işaretli bir liste olarak işlenecektir. Her liste öğesi tamamen tek bir fiziksel satırda olmalıdır.

Bir kod veya modüldeki her genel işlevi ve tabloyu belgelemek iyi bir uygulamadır. Ayrıca her kod ve modül kendi dosya düzeyinde belgelendirmeye sahip olmalıdır. Bir dosyanın başında yer alan bir belgeleme yorumu (ardından bir fonksiyon veya tablo tanımı gelmeyen) tüm dosya için geçerlidir. Dosya düzeyindeki belgeler, bir modülü kullanan bir geliştirici veya bir komut dosyasını çalıştırın bir kullanıcı için yararlı olan tüm üst düzey bilgileri içeren birkaç paragraf uzunluğunda olabilir ve olmalıdır. Örnek 9.8, comm modülünün belgelerini göstermektedir (yerden tasarruf etmek için birkaç paragraf çıkarılmıştır).

Örnek 9.8. Bir modül için NSEDoc yorumu

```
---  
-- Common communication functions for network discovery tasks like  
-- banner grabbing and data exchange.  
--  
-- These functions may be passed a table of options, but it's not required. The
```

```
-- keys for the options table are <code>"bytes"</code>, <code>"lines"</code>,
-- <code>"proto"</code>, and <code>"timeout"</code>. <code>"bytes"</code> sets
-- a minimum number of bytes to read. <code>"lines"</code> does the same for
-- lines. <code>"proto"</code> sets the protocol to communicate with,
-- defaulting to <code>"tcp"</code> if not provided. <code>"timeout"</code>
-- sets the socket timeout (see the socket function <code>set_timeout</code>
-- for details).
--
-- @author Kris Katterjohn 04/2008
-- @copyright Same as Nmap--See https://nmap.org/book/man-legal.html
```

Fonksiyonlar ve modüller yerine betiklerin belgelenmesi için bazı özel hususlar vardır. Özellikle, betiklerin aksi takdirde @ etiketi yorumlarına ait olacak bazı bilgiler için özel değişkenleri vardır (betik değişkenleri "Betik Biçimi" bölümünde açıklanmıştır). Özellikle, bir komut dosyasının açıklaması bir dokümantasyon yorumu yerine description değişkenine aittir ve @author ve @copyright içinde yer alacak bilgiler bunun yerine author ve license değişkenlerine aittir. NSEDoc bu değişkenleri bilir ve yorumlardaki alanları tercih ederek bunları kullanır. Kodlarda ayrıca örnek çıktıları gösteren @output ve @xmloutput etiketlerinin yanı sıra uygun olan yerlerde @args ve @usage etiketleri de bulunmalıdır. Örnek 9.9, belgeleme yorumları ve NSE değişkenlerinin bir kombinasyonunu kullanarak kod düzeyinde belgeleme için uygun biçimini göstermektedir.

Örnek 9.9. Bir kod için NSEDoc yorumu

```
description = [[
Maps IP addresses to autonomous system (AS) numbers.

The script works by sending DNS TXT queries to a DNS server which in
turn queries a third-party service provided by Team Cymru
(team-cymru.org) using an in-addr.arpa style zone set up especially for
use by Nmap. The responses to these queries contain both Origin and Peer
ASNs and their descriptions, displayed along with the BGP Prefix and
Country Code. The script caches results to reduce the number of queries
and should perform a single query for all scanned targets in a BGP
Prefix present in Team Cymru's database.
```

```
Be aware that any targets against which this script is run will be sent
to and potentially recorded by one or more DNS servers and Team Cymru.
In addition your IP address will be sent along with the ASN to a DNS
server (your default DNS server, or whichever one you specified with the
<code>dns</code> script argument).
```

```
]]
```

```
---
-- @usage
-- nmap --script asn-query [--script-args dns=<DNS server>] <target>
-- @args dns The address of a recursive nameserver to use (optional).
-- @output
-- Host script results:
-- | asn-query:
-- | BGP: 64.13.128.0/21 | Country: US
```

```

-- | Origin AS: 10565 SVOLO-AS - Silicon Valley Colocation, Inc.
-- | Peer AS: 3561 6461
-- | BGP: 64.13.128.0/18 | Country: US
-- | Origin AS: 10565 SVOLO-AS - Silicon Valley Colocation, Inc.
-- |_ Peer AS: 174 2914 6461

author = "jah, Michael"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"discovery", "external", "safe"}

```

Derlenmiş NSE modülleri de Lua kaynak kodu olmamasına rağmen NSEDoc ile belgelenir. Her derlenmiş modülün, Lua modülleriley birlikte nselib dizininde tutulan bir <modulename>.luadoc dosyası vardır. Bu dosya, derlenen modüldeki fonksiyonları ve tabloları Lua'da yazılmış gibi listeler ve belgeler. Her fonksiyonun sadece adı gereklidir, tanımı değil (sonu bile değil). Belgeleme ayırtırıcısına tanımlamada yardımcı olmak için bir tabloyu belgelendirirken @name ve @class etiketlerini kullanmalısınız. Nmap kaynak dağıtımında (nmap.luadoc, Ifs.luadoc ve pcre.luadoc dahil) bu belgeleme yönteminin birkaç örneği vardır.

NSE Documentation Tags (NSE Dokümantasyon Etiketleri)

Aşağıdaki etiketler NSEDoc tarafından anlaşılır:

@param ⇒ Bir fonksiyon parametresini tanımlar. Param'dan sonra gelen ilk sözcük açıklanan parametrenin adıdır. Etiket, bir fonksiyonun her parametresi için bir kez görünebilir.

@see ⇒ Başka bir fonksiyona veya tabloya çapraz referans ekler.

@return ⇒ Bir fonksiyonun geri dönüş değerini tanımlar. Birden fazla geri dönüş değeri için @return birden fazla kez kullanılabilir.

@usage ⇒ Bir fonksiyon, komut dosyası veya modülün kullanım örneğini sağlar. Bir fonksiyon söz konusu olduğunda, örnek Lua kodudur; bir komut dosyası için bu bir Nmap komut satırıdır; ve bir modül için genellikle bir kod örneğidir. use birden fazla kez verilebilir. Bir kodda atlanırsa, NSEDoc varsayılan standartlaştırılmış bir kullanım örneği oluşturur.

@name ⇒ Belgelenen fonksiyon veya tablo için bir isim tanımlar. Bu etiket normalde gereklidir çünkü NSEDoc isimleri kod analizi yoluyla çıkarır.

@class ⇒ Belgelenen nesnenin "sınıfını" tanımlar: fonksiyon, tablo veya modül. @name gibi, bu da normalde otomatik olarak çıkarılır.

@field ⇒ Bir tablonun dokümantasyonunda @field, adlandırılmış bir alanın değerini tanımlar.

@args ⇒ --script-args seçeneğiyile kullanıldığı gibi bir kod argümanını açıklar ("Komut Dosyalarının Argümanları" bölümünde bakın). Args'den sonraki ilk sözcük bağımsız değişkenin adıdır ve bunu izleyen her şey açıklamadır. Bu etiket kod düzeyindeki yorumlara özeldir.

@output ⇒ Kod düzeyinde yorumlara özel olan bu etiket, bir koddan örnek çıktı gösterir.

@xmloutput ⇒ Kodun "Yapilandırılmış ve Yapılandırmamış Çıktı" adlı bölümünün XML'e yazıldığında nasıl göründüğünü gösterir. XML örneği, <script> ve </script> etiketlerini içermemeli ve hiyerarşiyi göstermek için girintili olmalıdır.

@author ⇒ Birden fazla kez verilebilen bu etiket, bir NSE modülünün yazarlarını listeler. Komut dosyaları için bunun yerine yazar değişkenini kullanın.

@copyright ⇒ Bu etiket bir modülün telif hakkı durumunu açıklar. Kodlar için bunun yerine lisans değişkenini kullanın.

Script Parallelism in NSE (NSE'de Komut Dosyası Paralelliği)

"Network I/O API" adlı bölümde, NSE'nin ağ işlemlerini otomatik olarak paralelleştirdiğinden bahsedilmiştir. Genellikle bu işlem bir komut dosyası yazarı için şeffaftır, ancak nasıl çalıştığını dair bilgi gerektiren bazı gelişmiş teknikler vardır. Bu bölümde ele alınan teknikler, birden fazla komut dosyasının bir kütüphanede nasıl etkileşime gireceğini kontrol etmek, paralel olarak birden fazla iş parçacığı kullanmak ve özel durumlar için paralelliği devre dışı bırakmaktır.

Paralel yürütme için standart mekanizma bir iş parçacığıdır. Bir iş parçacığı, bir betiğin yürütme akışını ve verilerini kapsüller. Lua iş parçacığı, başka bir komut dosyası üzerinde çalışmaya devam etmek için rastgele konumlarda verilebilir. Tipik olarak, bu verim konumları nmap kütüphanesindeki soket işlemleri engeller. Kodun geri dönüşü de soket işleminin bir yan etkisi olarak şeffaftır.

Bazı genel terminolojinin üzerinden geçelim. Bir betik, ikili bir yürütülebilir dosyaya benzer; bir betiği yürütmek için gerekli bilgileri tutar. Bir iş parçacığı (bir Lua coroutine) bir sürece benzer; bir ana bilgisayara ve muhtemelen bağlantı noktasına karşı bir komut dosyası çalıştırır. Bazen terminolojiyi kötüye kullanız ve çalışan bir iş parçacığını çalışan bir "betik" olarak adlandırırız, ancak bunun gerçek anlamı, bir işlemin bir yürütülebilir dosyanın örneklenmesi olduğu gibi, bir betiğin örneklenmesidir.

NSE, kod başına bir iş parçacığından oluşan temel paralellik modelini genişletmek için gereken temel unsurları sağlar: yeni bağımsız iş parçacıkları, muteksler ve koşul değişkenleri.

Worker Threads (İşçi Konuları)

Bir komut dosyasının, genel bir komut dosyası ile varsayılan olarak sunulanın ötesinde paralel yürütme ile ilgili olarak daha ince kontrole ihtiyaç duyduğu çeşitli durumlar vardır. Yaygın bir ihtiyaç, aynı anda birden fazla soketten okuma yapmaktr. Örneğin, bir HTTP spidering betiği, web sunucusu kaynaklarını paralel olarak sorgulayan birden fazlaLua iş parçacığına sahip olmak isteyebilir. Bu ihtiyaça cevap vermek için NSE, işçi iş parçacıkları oluşturmak için stdnse.new_thread işlevini sunar. Bu işçi iş parçacıkları bağımsız komut dosyalarının tüm gücüne sahiptir, tek kısıtlama komut dosyası çiktısını rapor edememeleridir.

Bir kod tarafından başlatılan her bir iş parçacığına bir ana işlev ve NSE tarafından ana işlevde aktarılacak değişken sayıda bağımsız değişken verilir:

```
worker_thread, status_function = stdnse.new_thread(main, ...)
```

stdnse.new_thread iki değer döndürür: çalışan iş parçacığınızı benzersiz bir şekilde tanımlayan Lua iş parçacığı (coroutine) ve yeni çalışanınızın durumunu sorgulayan bir durum sorgu işlevi. Durum sorgulama fonksiyonu iki değer döndürür:

```
status, error_object = status_function()
```

İlk geri dönüş değeri basitçe işçi iş parçacığı coroutine'i üzerinde çalıştırılan coroutine.status'un geri dönüş değeridir. (Daha doğrusu, temel coroutine. "Temel iş parçacığı" başlıklı bölümde temel coroutine hakkında daha fazla bilgi edinin). İkinci dönüş değeri, çalışan iş parçacığının sonlandırılmasına neden olan bir hata nesnesi veya herhangi bir hata atılmadıysa nil içerir. Bu nesne, çoğu Lua hatası gibi tipik olarak bir dizedir. Ancak, herhangi bir Lua türü, nil bile olsa bir hata nesnesi olabilir. Bu nedenle, ikinci dönüş değeri olan hata nesnesini yalnızca çalışanın durumu "ölü" ise inceleyin.

NSE, çalışan iş parçacığı yürütmemeyi bitirdiğinde ana işlevden dönen tüm değerleri atar. Çalışan iş parçacığınızla ana işlev parametreleri, üst değerler veya işlev ortamları kullanarak iletişim kurmalısınız. Bir örnek için Örnek 9.10'a bakın.

Son olarak, işçi iş parçacıkları kullanırken bunları koordine etmek için her zaman koşul değişkenleri veya muteksler kullanmalısınız. Nmap tek iş parçacıklıdır, bu nedenle endişelenecek bellek senkronizasyonu sorunları yoktur; ancak kaynaklar için çekişme vardır. Bu kaynaklar genellikle ağ bant genişliği ve soketleri içerir. Koşul değişkenleri, tek bir iş parçacığı için yapılan iş dinamikse de kullanılmalıdır. Örneğin, işçi havuzuna sahip bir web sunucusu örümcek komut dosyası başlangıçta tek bir kök HTML belgesine sahip olacaktır. Kök belgenin alınmasını takiben, her yeni belge getirilecek yeni URL'ler eklediğinden, alınacak kaynak kümesi (işçinin işi) çok büyük hale gelebilir.

Örnek 9.10. İşçi iş parçacıkları

```
local requests = {"/", "/index.html", --[[ long list of objects ]]}

function thread_main (host, port, responses, ...)
    local condvar = nmap.condvar(responses);
    local what = {n = select("#", ...), ...};
    local allReqs = nil;
    for i = 1, what.n do
        allReqs = http.pGet(host, port, what[i], nil, nil, allReqs);
    end
    local p = assert(http.pipeline(host, port, allReqs));
    for i, response in ipairs(p) do responses[#responses+1] = response end
    condvar "signal";
end

function many_requests (host, port)
    local threads = {};
    local responses = {};
    local condvar = nmap.condvar(responses);
    local i = 1;
    repeat
        local j = math.min(i+10, #requests);
        local co = stdnse.new_thread(thread_main, host, port, responses,
            unpack(requests, i, j));
        threads[co] = true;
        i = j+1;
    until i > #requests;
    repeat
        for thread in pairs(threads) do
            if coroutine.status(thread) == "dead" then threads[thread] = nil end
        end
        if ( next(threads) ) then
            condvar "wait"
        end
    until next(threads) == nil;
    return responses;
end
```

Kısa olması için, bu örnekte geleneksel bir web örümceğinin tipik davranışlarına yer verilmemiştir. İstekler tablosunun, işçi iş parçacıklarının kullanılmasını gerektirecek kadar nesne içeriği varsayılmaktadır. Bu

örnekteki kod, 11 kadar göreli URL ile yeni bir iş parçacığı gönderir. Çalışan iş parçacıkları ucuzdur, bu nedenle bunlardan çok sayıda oluşturmaktan korkmayın. Tüm bu iş parçacıklarını gönderdikten sonra kod, her iş parçacığının işi bitene kadar bir koşul değişkeninde bekler ve son olarak yanıt tablosunu döndürür.

stdnse.new_thread tarafından döndürülen durum fonksiyonunu kullanmadığımızı fark etmiş olabilirsiniz. Bunu genellikle hata ayıklama için ya da programınızın çalışan iş parçacıklarından biri tarafından atılan hataya bağlı olarak durması gerekiyorsa kullanırsınız. Basit örneğimiz bunu gerektirmemi ancak daha hataya dayanıklı bir kütüphane bunu gerektirebilir.

Mutexes

Bu bölümün başından itibaren, her bir komut dosyası yürütme iş parçacığının (örneğin, bir hedef ana bilgisayardaki bir FTP sunucusuna karşı çalışan ftp-anon) ağ nesneleri üzerinde bir çağrı yaptığıında (veri gönderme veya alma) diğer komut dosyalarına teslim olduğunu hatırlayın. Bazı betikler iş parçacığı yürütme üzerinde daha hassas eşzamanlılık kontrolü gerektirir. Her bir hedef IP adresi için whois sunucularını sorgulayan whois betiği buna bir örnektir. Çok sayıda eşzamanlı sorgu IP'nizin kötüye kullanım nedeniyle yasaklanması neden olabileceğinden ve tek bir sorgu betiği başka bir örneğinin talep etmek üzere olduğu aynı bilgiyi döndürebileceğinden, bir iş parçacığı bir sorgu gerçekleştirirken diğer iş parçacıklarının duraklatılması yararlıdır.

Bu sorunu çözmek için NSE, komut dosyaları tarafından kullanılabilen bir mutex (karşılıklı dışlama nesnesi) sağlayan bir mutex işlevi içerir. Mutex, bir nesne üzerinde aynı anda yalnızca bir iş parçacığının çalışmasına izin verir. Bu nesne üzerinde çalışmak için bekleyen rakip iş parçacıkları, mutex üzerinde bir "kilit" elde edene kadar bekleme kuyruğuna alır. Yukarıdaki whois sorunu için bir çözüm, her iş parçacığının ortak bir dize kullanarak bir mutex üzerinde bloke olmasını sağlamak ve böylece bir seferde yalnızca bir iş parçacığının bir sunucusu sorgulamasını sağlamaktır. Uzak sunucuları sorgulama işlemi bittiğinde, iş parçacığı sonuçları NSE kayıt defterinde saklayabilir ve mutex'in kilidini açabilir. Uzak sunucusunu sorgulamak için bekleyen diğer komut dosyaları daha sonra bir kilit alabilir, önceki bir sorgudan kullanılabilir bir sonuç için önbelleği kontrol edebilir, kendi sorgularını yapabilir ve mutex'in kilidini açabilir. Bu, uzak bir kaynağa erişimi serileştirmenin iyi bir örneğidir.

Bir mutex kullanmanın ilk adımı nmap.mutex çağrıları ile bir mutex oluşturmaktır.

```
mutexfn = nmap.mutex(object)
```

Döndürülen mutexfn, aktarılan nesne için mutex olarak çalışan bir fonksiyondur. Bu nesne nil, Boolean ve sayı dışında herhangi bir Lua veri tipi olabilir. Dönen fonksiyon mutex'i kilitlemenizi, kilitlemeye çalışmanızı ve serbest bırakmanızı sağlar. Tek argümanı aşağıdakilerden biri olmalıdır:

"lock" ⇒ Mutex üzerinde bloklama kilidi oluşturur. Mutex meşgulse (başka bir iş parçacığının üzerinde kilit varsa), iş parçacığı teslim olur ve bekler. Fonksiyon mutex kilitli olarak geri döner.

"trylock" ⇒ Mutex üzerinde bloklama olmayan bir kilitleme yapar. Mutex meşgulse, hemen false dönüşü degeriyle geri döner. Aksi takdirde, mutex kilitler ve true değerini döndürür.

"done" ⇒ Mutex serbest bırakır ve başka bir iş parçacığının onu kilitmesine izin verir. Eğer iş parçacığının mutex üzerinde bir kilidi yoksa, bir hata oluşacaktır.

"running" ⇒ Mutex üzerinde kilitli olan iş parçacığını veya mutex kilitli değilse nil değerini döndürür. Bu, bitmiş iş parçacıklarının çöp toplamasına müdahale ettiğin için yalnızca hata ayıklama için kullanılmalıdır.

NSE mutexe zayıf bir referans tutar, böylece aynı nesneye nmap.mutex'e yapılan diğer çağrılar aynı mutex işlevini döndürür. Ancak, mutex referansınızı atarsanız, o zaman toplanabilir ve nesne ile nmap.mutex'e yapılan sonraki çağrılar farklı bir işlev döndürecektil. Bu nedenle mutexınızı ihtiyaç duyduğunuz sürece kalıcı olan bir (yerel) değişkene kaydedin.

API'nin kullanımına ilişkin basit bir örnek Örnek 9.11'de verilmiştir. Gerçek hayattan örnekler için Nmap dağıtımındaki asn-query ve whois betiklerini okuyun.

Örnek 9.11. Mutex manipülasyonu

```
local mutex = nmap.mutex("My Script's Unique ID");
function action(host, port)
    mutex "lock";
    -- Do critical section work - only one thread at a time executes this.
    mutex "done";
    return script_output;
end
```

Condition Variables (Durum Değişkenleri)

Koşul değişkenleri, stdnse.new_thread işlevi tarafından oluşturulan işçi iş parçacıklarıyla koordinasyon sağlama ihtiyacından ortaya çıkmıştır. Bir koşul değişkeni, birçok iş parçacığının bir nesne üzerinde beklemesine ve bazı koşullar karşılandığında bunlardan birinin veya hepsinin uyandırılmasına olanak tanır. Başka bir deyişle, birden fazla iş parçacığı koşul değişkeni üzerinde bekleyerek koşulsuz olarak bloke olabilir. Diğer iş parçacıkları bekleyen iş parçacıklarını uyandırmak için koşul değişkenini kullanabilir.

Örneğin, daha önceki Örnek 9.10, "İşçi iş parçacıkları"nı düşünün. Tüm işçiler bitirene kadar denetleyici iş parçacığı uyumak zorundadır. Geleneksel bir işletim sistemi iş parçacığında olduğu gibi sonuçları sorgulayamayacağımızı unutmayın çünkü NSE, Lua iş parçacıklarının önüne geçmez. Bunun yerine, denetleyici iş parçacığının bir işçi tarafından uyandırılana kadar beklediği bir koşul değişkeni kullanınız. Denetleyici, tüm işçiler sonlandırıldıkça kadar sürekli olarak bekleyecektir.

Koşul değişkeni kullanmanın ilk adımı nmap.condvar çağrıları ile bir koşul değişkeni oluşturmaktır.

`condvarfn = nmap.condvar(object)` ⇒ Koşul değişkenlerinin semantiği mutexlerinkine benzer. Döndürülen condvarfn, geçirilen nesne için bir koşul değişkeni olarak çalışan bir fonksiyondur. Bu nesne nil, Boolean ve sayı dışında herhangi bir Lua veri tipi olabilir. Döndürülen işlev, koşul değişkeni üzerinde beklemenize, sinyal vermenize ve yayın yapmanıza olanak tanır. Tek argümanı aşağıdakilerden biri olmalıdır:

`"wait"` ⇒ Koşul değişkenini bekleyin. Bu, geçerli iş parçacığını koşul değişkeni için bekleme kuyruğuna ekler. Başka bir iş parçacığı koşul değişkeniyle ilgili sinyal verdiğiinde veya yayın yaptığında yürütmeye devam eder.

`"signal"` ⇒ Koşul değişkenine sinyal gönderin. Koşul değişkeninin bekleme kuyruğundaki iş parçacıklarından biri devam ettirilecektir.

`"broadcast"` ⇒ Koşul değişkeninin bekleme kuyruğundaki tüm iş parçacıklarını devam ettirir.

Mutexlerde olduğu gibi, NSE koşul değişkenine zayıf bir referans tutar, böylece aynı nesneye nmap.condvar'a yapılan diğer çağrılar aynı işlevi döndürür. Ancak, koşul değişkenine olan referansınızı atarsanız, o zaman toplanabilir ve nesneye nmap.condvar'a yapılan sonraki çağrılar farklı bir işlev döndürür. Bu nedenle, koşul değişkeninizi ihtiyaç duyduğunuz sürece kalıcı olan bir (yerel) değişkene kaydedin.

Koşul değişkenlerini kullanırken, beklemeden önce ve sonra yüklemi kontrol etmek önemlidir. Yüklem, bir alt iş parçacığı veya denetleyici iş parçacığı içinde iş yapmaya devam edilip edilmeyeceğine ilişkin bir testtir. Çalışan iş parçacıkları için bu, en azından denetleyicinin hala hayatı olup olmadığını görmek için bir test içerecektir. Sonuçlarınızı kullanacak bir iş parçacığı yokken iş yapmaya devam etmek istemezsiniz. Beklemeden önce tipik bir test şöyle olabilir: Kontrolörün hala çalışıp çalışmadığını kontrol edin; çalışmayıorsa çıkış. Yapılacak iş olup olmadığını kontrol edin; yoksa bekleyin.

Bir koşul değişkenini bekleyen bir iş parçacığı, başka herhangi bir iş parçacığı koşul değişkeni üzerinde "signal" veya "broadcast" çağrıları yapmadan devam ettirilebilir (sahte uyandırma). Bunun olağan, ancak tek nedeni koşul değişkenini kullanan iş parçacıklarından birinin sonlandırılması değildir. Bu, NSE'nin sağladığı önemli bir garantidir ve bir çalışan ya da denetleyicinin koşul değişkenine sinyal göndermeden sonlanan bir iş parçacığının kendisini uyandırmamasını beklediği durumlarda kilitlenmeyi önlemenizi sağlar.

Collaborative Multithreading (İşbirlikçi Multithreading)

Lua'nın en az bilinen özelliklerinden biri korutinler aracılığıyla işbirlikçi çoklu iş parçacığıdır. Bir coroutine verilebilir ve devam ettirilebilir bağımsız bir yürütme yiğini sağlar. Standart coroutine tablosu, coroutine'lerin oluşturulması ve manipülasyonuna erişim sağlar. Lua'nın çevrimiçi ilk baskısı Programming in Lua, korutinlere mükemmel bir giriş içerir. Aşağıda, bütünlük için burada korutinlerin kullanımına genel bir bakış yer almaktadır, ancak bu kesin referansın yerini tutmaz.

Bu bölüm boyunca coroutine'lerden thread olarak bahsettiğimizde,Lua'daki bir coroutine'in türüdür ("thread"). Bunlar programcılar beklediği gibi önceliğe sahip iş parçacıkları değildir. Lua iş parçacıkları paralel kodlama için temel sağlar, ancak aynı anda yalnızca bir iş parçacığı çalışır.

BirLua işlevi birLua iş parçacığının üstünde çalışır. İş parçacığı, etkin işlevlerin, yerel değişkenlerin ve geçerli komut işaretçisinin bir yiğinini tutar. Çalışan iş parçacığını açıkça teslim ederek korutinler arasında geçiş yapabiliriz. Verilen iş parçacığını devam ettiren coroutine çalışmaya devam eder. Örnek 9.12, sayıları yazdırma için korutinlerin kısa bir kullanımını göstermektedir.

Örnek 9.12. Temel Korutin Kullanımı

```
local function main ()  
    coroutine.yield(1)  
    coroutine.yield(2)  
    coroutine.yield(3)  
end  
local co = coroutine.create(main)  
for i = 1, 3 do  
    print(coroutine.resume(co))  
end  
→ true 1  
→ true 2  
→ true 3
```

Korutinler, NSE'nin komut dosyalarını paralel olarak çalıştırmasını sağlayan bir olanaktır. Tüm komut dosyaları, engelleme bir soket işlevi çağrılarında verim veren korutinler olarak çalıştırılır. Bu, NSE'nin diğer komut dosyalarını çalıştırmasını ve daha sonra G/C işlemi tamamlandığında engellenen komut dosyasını devam ettirmesini sağlar.

Bazen korutinler tek bir kod içindeki bir iş için en iyi araçtır. Soket programlamada yaygın kullanım alanlarından biri verileri filtrelemektir. Bir HTML belgesindeki tüm bağlantıları üreten bir fonksiyon yazabilirsiniz. string.gmatch kullanan bir yineleyici yalnızca tek bir kalıbı yakalayabilir. Bazı karmaşık eşleşmeler birçok farklıLua kalıbı alabileceğinden, bir coroutine kullanmak daha uygundur. Örnek 9.13 bunun nasıl yapılacağını göstermektedir.

Örnek 9.13. Bağlantı Oluşturucu

```
function links (html_document)  
    local function generate ()  
        for m in string.gmatch(html_document, "url%((.-)%)") do  
            coroutine.yield(m) -- css url  
        end  
        for m in string.gmatch(html_document, "href%s*=%s*\"(.+)\\"") do  
            coroutine.yield(m) -- anchor link  
        end  
    end  
    return generate  
end
```

```

    end
    for m in string.gmatch(html_document, "src%s*=%s*\"(.-)\"") do
        coroutine.yield(m) -- img source
    end
end
return coroutine.wrap(generate)
end

function action (host, port)
    -- ... get HTML document and store in html_document local
    for link in links(html_document) do
        links[#links+1] = link; -- store it
    end
    -- ...
end

```

The base thread (Temel iplik)

Komut dosyaları kendi çoklu iş parçacıkları için korutinleri kullanabileceğinden, bir kaynağın sahibini tanımlayabilmek veya komut dosyasının hala canlı olup olmadığını belirlemek önemlidir. NSE bu amaç için stdnse.base fonksiyonunu sağlar.

Özellikle bir önbellek veya soketin sahipliğini bir betiğe atayan bir kütüphane yazarken, betiğin hala çalışıp çalışmadığını belirlemek için temel iş parçacığını kullanabilirsiniz. coroutine.status temel iş parçacığı üzerinde betiğin mevcut durumunu verecektir. Betiğin "ölü" olduğu durumlarda, kaynağı serbest bırakmak isteyeceksiniz. Bu iş parçacıklarına referansları tutarken dikkatli olun; NSE, yürütülmesi bitmemiş olsa bile bir komut dosyasını atabilir. İş parçacığı yine de "askıya alındı" durumunu bildirecektir. Bu gibi durumlarda iş parçacığının toplanabilmesi için zayıf bir referans tutmalısınız.

Version Detection Using NSE (NSE Kullanarak Sürüm Algılama)

Nmap'te yerleşik sürüm algılama sistemi, basit bir prob ve desen eşleştirme sözdizimi ile protokollerin büyük çoğunluğunu verimli bir şekilde tanımak için tasarlanmıştır. Bazı protokoller sürüm tespitinin üstesinden gelebileceğinden daha karmaşık iletişim gerektirir. NSE tarafından sağlanan genelleştirilmiş bir komut dosyası dili bu zorlu durumlar için mükemmeldir.

NSE'nin sürüm kategorisi, standart sürüm algılamayı geliştiren betikler içerir. Bu kategorideki betikler -sV ile sürüm tespiti talep ettiğinizde çalıştırılır; bunları çalıştmak için -sC kullanmanız gerekmek. Bu diğer yolu da keser: -sC kullanırsanız, -sV'yi de kullanmadığınız sürece sürüm betiklerini alamazsınız.

Normal sürüm tespiti ile tespit edemediğimiz bir protokol Skype sürüm 2'dir. Bu protokol muhtemelen telekom bağlantılı Internet servis sağlayıcılarının Skype'ı rakip olarak görüp trafiğe müdahale edebileceği korkusuyla tespit edilmesini engellemek için tasarlanmıştır. Yine de bunu tespit etmenin bir yolunu bulduk. Skype bir HTTP GET isteği alırsa, bir web sunucusu gibi davranışıyor ve 404 hatası döndürüyor. Ancak diğer talepler için rastgele görünen bir veri yiğinini geri gönderir. Doğru tanımlama, iki prob göndermeyi ve iki yanıtı karşılaştırmayı gerektirir - NSE için ideal bir görev. Bunu gerçekleştiren basit NSE komut dosyası Örnek 9.14'te gösterilmektedir.

Örnek 9.14. Tipik bir sürüm algılama komut dosyası (Skype sürüm 2 algılama)

```

description = [[
    Detects the Skype version 2 service.
]]
---  

-- @output  

-- PORT STATE SERVICE VERSION  

-- 80/tcp open skype2 Skype  

author = "Brandon Enright"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"version"}  

require "comm"
require "shortport"  

portrule = function(host, port)
    return (port.number == 80 or port.number == 443 or
            port.service == nil or port.service == "" or
            port.service == "unknown")
        and port.protocol == "tcp" and port.state == "open"
        and port.service ~= "http" and port.service ~= "ssl/http"
        and not(shortport.port_is_excluded(port.number, port.protocol)))
end  

action = function(host, port)
    local status, result = comm.exchange(host, port,
                                         "GET / HTTP/1.0\r\n\r\n", {bytes=26, proto=port.protocol})
    if (not status) then
        return
    end
    if (result ~= "HTTP/1.0 404 Not Found\r\n\r\n") then
        return
    end
    -- So far so good, now see if we get random data for another request
    status, result = comm.exchange(host, port,
                                   "random data\r\n\r\n", {bytes=15, proto=port.protocol})  

    if (not status) then
        return
    end
    if string.match(result, "[^%s!-~].*[^%s!-~].*[^%s!-~]") then
        -- Detected
        port.version.name = "skype2"
        port.version.product = "Skype"
        nmap.set_port_version(host, port)
        return

```

```
end  
return  
end
```

Betik Skype'ı tespit ederse, port tablosunu artık bilinen isim ve ürün alanlarıyla genişletir. Daha sonra nmap.set_port_version çağrıları yaparak bu yeni bilgiyi Nmap'e gönderir. Biliniyorsa ayarlanabilecek başka sürüm alanları da vardır, ancak bu durumda elimizde yalnızca ad ve ürün vardır. Sürüm alanlarının tam listesi için nmap.set_port_version belgesine bakın.

Bu betiğin protokolü algılamadığı sürece hiçbir şey yapmadığını dikkat edin. Bir betik, hiçbir şey öğrenmediğini söylemek için çıktı (hata ayıklama çıktısı dışında) üretmemelidir.

Example Script: finger (Örnek Kod: parmak)

Parmak senaryosu, kısa ve basit bir NSE senaryosuna mükemmel bir örnektir.

Önce bilgi alanları atanır. Kodun gerçekte ne yaptığına ilişkin ayrıntılı bir açıklama açıklama alanına gider.

```
description = [[  
    Attempts to get a list of usernames via the finger service.  
]]  
  
author = "Eddie Bell"  
  
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
```

Kategoriler alanı, komut dosyasının ait olduğu tüm kategorileri içeren bir tablodur. Bunlar --script seçeneği ile komut dosyası seçimi için kullanılır:

```
categories = {"default", "discovery", "safe"}
```

Her iyi kod, NSEDoc yorumunda çıktısının bir örneğiyle birlikte gelir.

```
---  
-- @output  
-- PORT STATE SERVICE  
-- 79/tcp open  finger  
-- | finger:  
-- | Welcome to Linux version 2.6.31.12-0.2-default at linux-pb94.site !  
-- | 01:14am up 18:54, 4 users, load average: 0.14, 0.08, 0.01  
-- |  
-- | Login  Name          Tty  Idle Login Time Where  
-- | Gutek  Ange Gutek    *:0   -   Wed 06:19 console  
-- | Gutek  Ange Gutek    pts/1  18:54   Wed 06:20  
-- | Gutek  Ange Gutek    *pts/0  -   Thu 00:41  
-- | _Gutek Ange Gutek    *pts/4  3   Thu 01:06
```

require ile nselib ("NSE Kütüphaneleri" adlı bölüm) tarafından sağlanan olanakları kullanabilirsiniz. Burada ortak iletişim fonksiyonlarını ve daha kısa port kurallarını kullanmak istiyoruz:

```
require "comm"  
require "shortport"
```

Komut dosyasını finger hizmetine karşı çalıştmak istiyoruz. Bu nedenle, iyi bilinen finger portunu (79/tcp) kullanıp kullanmadığını veya sürüm algılama sonuçlarına veya port numarasının nmap-services'teki listesine göre hizmetin "finger" olarak adlandırılıp adlandırılmadığını test ediyoruz:

```
portrule = shortport.port_or_service(79, "finger")
```

İlk olarak, komut dosyası bir hata durumunda komut dosyasından çıkacak bir istisna işleyicisi oluşturmak için nmap.new_try kullanır. Ardından, kontrolü ağ işlemini gerçekleştiren comm.exchange'e aktarır. Burada, en az 100 satır alana, en az 5 saniye bekleyene veya uzak taraf bağlantıyi kapatana kadar iletişim değişiminde beklememizi istedik. Herhangi bir hata try istisna işleyicisi tarafından ele alınır. Komut dosyası, comm.exchange() çağrısının başarılı olması durumunda bir dize döndürür.

```
action = function(host, port)  
local try = nmap.new_try()  
return try(comm.exchange(host, port, "\r\n",  
{ilines=100, proto=port.protocol, timeout=5000}))  
end
```

Implementation Details (Uygulama Detayları)

Şimdi sıra NSE uygulama detaylarını derinlemesine incelemeye geldi. NSE'nin nasıl çalıştığını anlamak, verimli komut dosyaları ve kütüphaneler tasarlamak için yararlıdır. NSE uygulaması için temel referans kaynak kodudur, ancak bu bölüm temel ayrıntılara genel bir bakış sağlar. NSE kaynak kodunu anlamaya ve genişletmeye çalışan kişilerin yanı sıra komut dosyalarının nasıl yürütüldüğünü daha iyi anlamak isteyen komut dosyası yazarları için de değerli olacaktır.

Initialization Phase (Başlatma Aşaması)

NSE, Nmap ilk başladığında open_nse fonksiyonu tarafından herhangi bir taramadan önce başlatılır. open_nse, programdan çıkışa kadar ana bilgisayar grupları arasında devam edecek yeni bir Lua durumu oluşturur. Daha sonra standart Lua kütüphanelerini ve derlenmiş NSE Kütüphanelerini yükler. Standart Lua KütüphaneleriLua Referans Kılavuzunda belgelenmiştir. NSE için kullanılabilen standart Lua Kütüphaneleri debug, io, math, os, package, string ve table'dır. Derlenmiş NSE Kütüphaneleri, bir Lua dosyası yerine bir C++ dosyasında tanımlanan kütüphanelerdir. Bunlar arasında nmap, pcre, bin, bit ve openssl (varsayılan) bulunur.

Temel kütüphaneler yüklenikten sonra open_nse nse_main.lua dosyasını yükler. NSE'nin çekirdeği bu dosyadır-Lua kodu komut dosyalarını yönetir ve uygun ortamı kurar. Bu durumda Lua gerçekten bir tutkal dili olarak parlar. C++ ağ çerçevesini ve düşük seviyeli kütüphaneleri sağlamak için kullanılır. Lua, verileri yapılandırmak, hangi komut dosyalarının yükleneceğini belirlemek ve komut dosyalarını zamanlamak ve yürütmek için kullanılır.

nse_main.lua,Lua ortamını daha sonra betik taramasına hazır olacak şekilde ayarlar. Kullanıcının seçtiği tüm betikleri yükler ve open_nse'ye gerçek betik taramasını yapan bir fonksiyon döndürür.

Kodların standart NSE kütüphanesine erişimini sağlamak için nselib dizini Lua yoluna eklenir. NSE, standart coroutine işlevleri için değiştirmeler yükler, böylece NSE tarafından başlatılan verimler yakalanır ve NSE zamanlayıcısına geri yayılır.

nse_main.lua daha sonra kurulum sırasında kullanılacak sınıfları ve işlevleri tanımlar. Betik argümanları (--script-args) nmap.registry.args dosyasına yüklenir. Halihazırda mevcut değilse veya --script-updatedb ile talep

edilmişse bir komut dosyası veritabanı oluşturular.

Son olarak, komut satırında listelenen komut dosyaları yüklenir. `get_chosen_scripts` işlevi, kategorileri, dosya adlarını ve dizin adlarını karşılaştırarak seçilen komut dosyalarını bulmak için çalışır. Komut dosyaları daha sonra kullanılmak üzere belleğe yüklenir. `get_chosen_scripts`, --script argümanını bir Lua kodu bloğuna dönüştürerek ve ardından çalıştırarak çalışır. (and, or ve not operatörleri bu şekilde desteklenir.) Doğrudan bir kategoriyle veya `script.db`'deki bir dosya adıyla eşleşmeyen tüm belirtimler dosya ve dizin adlarına karşı kontrol edilir. Eğer belirtim normal bir dosya ise, yüklenir. Bir dizin ise, içindeki tüm *.nse dosyaları yüklenir. Aksi takdirde, motor bir hata verir.

`get_chosen_scripts` seçilen komut dosyalarını bağımlılıklarına göre düzenleyerek bitirir ("bağımlılıklar Alanı" adlı bölüme bakın). Hiçbir bağımlılığı olmayan betikler çalışma düzeyi 1'dedir. Bunlara doğrudan bağımlı olan komut dosyaları çalışma düzeyi 2'dedir ve bu böyle devam eder. Bir komut dosyası taraması çalıştırıldığında, her çalışma seviyesi ayrı ayrı ve sırayla çalıştırılır.

`nse_main.lua` iki sınıf tanımlar: Script ve Thread. Bu sınıflar, NSE komut dosyalarını ve bunların komut dosyası iş parçacıklarını temsil eden nesnelerdir. Bir script yüklendiğinde, `Script.new` yeni bir Script nesnesi oluşturur. Komut dosyası Lua'ya yüklenir ve daha sonra kullanılmak üzere kaydedilir. Bu sınıflar ve yöntemleri, her bir komut dosyası ve iş parçacığı için gereken verileri kapsüllemek üzere tasarlanmıştır. `Script.new` ayrıca komut dosyasının eylem işlevi gibi gerekli alanlara sahip olduğundan emin olmak için sağlık kontrolleri içerir.

Script Scanning (Komut Dosyası Tarama)

NSE bir komut dosyası taraması çalıştırıldığında, `script_scan nse_main.cc` içinde çağrılır. Üç kod tarama aşaması olduğundan, `script_scan` iki argüman kabul eder, bu değerlerden biri olabilen bir kod tarama türü:

`SCRIPT_PRE_SCAN` (Kod Ön tarama aşaması) veya `SCRIPT_SCAN` (Kod tarama aşaması) veya `SCRIPT_POST_SCAN` (Kod Son tarama aşaması) ve kod tarama aşaması `SCRIPT_SCAN` ise taranacak hedeflerin bir listesi olan ikinci bir argüman. Bu hedefler tarama için `nse_main.lua` ana fonksiyonuna aktarılacaktır.

Bir kod taraması için ana işlev, kural işlevinin true döndürüp döndürmediğine bağlı olarak bir dizi kod iş parçacığı oluşturur. Oluşturulan iş parçacıkları bir çalışma seviyesi listesinde saklanır. Her çalışma seviyesi listesi ayrı ayrı run fonksiyonuna aktarılır. Run fonksiyonu, NSE için tüm sihrin gerçekleştiği ana işçi fonksiyonudur.

Run fonksiyonunun amacı, bir çalışma seviyesindeki tüm iş parçacıklarını, hepsi bitene kadar çalıştmaktır. Ancak bunu yapmadan önce run, C kodunun çalışmasına yardımcı olan bazıLua kayıt defteri değerlerini yeniden tanımlar. Bu fonksiyonlardan biri olan `_R[WAITING_TO_RUNNING]`, C ile yazılmış ağ Kütüphanesi bağlayıcısının bir iş parçacığını bekleme kuyruğundan çalışma kuyruğuna taşımamasına izin verir. Komut dosyaları, çalışan ve bekleyen kuyrukların her ikisi de boş olana kadar çalıştırılır. Sonuç veren iş parçacıkları bekleme kuyruğuna taşınır; devam etmeye hazır olan iş parçacıkları tekrar çalışma kuyruğuna taşınır. Döngü, iş parçacığı çıkana veya hata ile sonlanana kadar devam eder. Bekleme ve çalışma kuyruklarının yanı sıra bir de bekleyen kuyruk vardır. Bu kuyruk, çalışan kuyruğun yeni bir yinelemesi başlamadan önce bekleme kuyruğundan çalışan kuyruğa geçen iş parçacıkları için geçici bir konum görevi görür.

Chapter 10. Detecting and Subverting Firewalls and Intrusion Detection Systems (Bölüm 10. Güvenlik Duvarlarını ve Saldırı Tespit Sistemlerini Tespit Etme ve Yıkma)

- Introduction (Giriş)

- Why Would Ethical Professionals (White-hats) Ever Do This? (Etik Profesyonelleri (Beyaz Şapkalılar) Bunu Neden Yapsın?)
- Determining Firewall Rules (Güvenlik Duvarı Kurallarını Belirleme)
 - Standard SYN Scan (Standart SYN Taraması)
 - Sneaky firewalls that return RST (RST döndüren sinsi güvenlik duvarları)
 - ACK Scan (ACK Scan)
 - IP ID Tricks (IP Kimliği Hileleri)
 - UDP Version Scanning (UDP Sürüm Taraması)
- Bypassing Firewall Rules (Güvenlik Duvarı Kurallarını Atlama)
 - Exotic Scan Flags (Egzotik Tarama Bayrakları)
 - Source Port Manipulation (Kaynak Bağlantı Noktası Manipülasyonu)
 - IPv6 Attacks (IPv6 Saldırıları)
 - IP ID Idle Scanning (IP Kimliği Boşta Tarama)
 - Multiple Ping Probes (Çoklu Ping Problemleri)
 - Fragmentation (Parçalanma)
 - Proxies (Proxyler)
 - MAC Address Spoofing (MAC Adresi Sahtekarlığı)
 - Source Routing (Kaynak Yönlendirme)
 - FTP Bounce Scan (FTP Sıçrama Taraması)
 - Take an Alternative Path (Alternatif Bir Yol İzleyin)
 - A Practical Real-life Example of Firewall Subversion (Güvenlik Duvarı Subversion'ının Gerçek Hayattan Pratik Bir Örneği)
- Subverting Intrusion Detection Systems (İzinsiz Giriş Tespit Sistemlerini Yıkmak)
 - Intrusion Detection System Detection (İzinsiz Giriş Tespit Sistemi Tespiti)
 - Reverse probes (Ters probalar)
 - Sudden firewall changes and suspicious packets (Ani güvenlik duvarı değişiklikleri ve şüpheli paketler)
 - Naming conventions (Adlandırma kuralları)
 - Unexplained TTL jumps (Açıklanamayan TTL atlamları)
 - Avoiding Intrusion Detection Systems (İzinsiz Giriş Tespit Sistemlerinden Kaçınma)
 - Slow down (Yavaşla)
 - Scatter probes across networks rather than scanning hosts consecutively (Ana bilgisayarları art arda taramak yerine probayı ağırlara dağıtın)
 - Fragment packets (Paketleri parçalama)
 - Evade specific rules (Belirli kurallardan kaçınmak)
 - Avoid easily detected Nmap features (Kolayca tespit edilen Nmap özelliklerinden kaçının)

- Misleading Intrusion Detection Systems (Yanıltıcı Saldırı Tespit Sistemleri)
 - Decoys ()
 - Port scan spoofing (Port tarama sahtekarlığı)
 - Idle scan (Boşta tarama)
 - DNS proxying (DNS proxyleme)
- DoS Attacks Against Reactive Systems (Reaktif Sistemlere Karşı DoS Saldırıları)
- Exploiting Intrusion Detection Systems (İzinsiz Giriş Tespit Sistemlerinden Yararlanma)
- Ignoring Intrusion Detection Systems (İzinsiz Giriş Tespit Sistemlerini Görmezden Gelmek)
- Detecting Packet Forgery by Firewall and Intrusion Detection Systems (Güvenlik Duvarı ve Saldırı Tespit Sistemleri ile Paket Sahteciliğinin Tespiti)
 - Look for TTL Consistency (TTL Tutarlılığına Bakın)
 - Look for IP ID and Sequence Number Consistency (IP Kimliği ve Sıra Numarası Tutarlılığına Bakın)
 - The Bogus TCP Checksum Trick (Sahte TCP Checksum Hilesi)
 - Round Trip Times (Gidiş Dönüş Süreleri)
 - Close Analysis of Packet Headers and Contents (Paket Başlıklarının ve İçeriklerinin Yakın Analizi)
 - Unusual Network Uniformity (Olağandışı Ağ Tekdüzeliği)

Introduction (Giriş)

Birçok İnternet öncüsü, herhangi iki düğüm arasında sanal bağlantılarla izin veren evrensel bir IP adres alanına sahip küresel bir açık ağ öngörmüştür. Bu sayede ana bilgisayarlar gerçek eşler olarak hareket edebilecek, birbirlerine bilgi sunabilecek ve birbirlerinden bilgi alabileceklerdi. İnsanlar iş yerlerinden tüm ev sistemlerine erişebilir, klima kontrol ayarlarını değiştirebilir veya erken gelen misafirler için kapıların kilidini açabilir. Bu evrensel bağlantı vizyonu, adres alanı sıkıntısı ve güvenlik endişeleri nedeniyle engellendi. 1990'ların başında kuruluşlar, bağlanabilirliği azaltmak amacıyla güvenlik duvarları kullanmaya başladı. Büyük ağlar, uygulama proxy'leri, ağ adresi çeviri cihazları ve paket filtreleri ilefiltrelenmemiş Internet'ten kordon altına alındı. Sınırsız bilgi akışı yerini onaylı iletişim kanallarının ve bu kanallar üzerinden geçen içeriğin sıkı bir şekilde düzenlenmesine bıraktı.

Güvenlik duvarları gibi ağ engelleri bir ağın harmasını çıkarmayı son derece zorlaştırabilir. Sıradan keşifleri bastırmak genellikle cihazların uygulanmasının temel bir amacı olduğundan, daha da kolaylaşmayacaktır. Bununla birlikte, Nmap bu karmaşık ağları anlamaya yardımcı olmak ve filtrelerin amaçlandığı gibi çalıştığını doğrulamak için birçok özellik sunar. Hatta kötü uygulanan savunmaları atlatmak için mekanizmaları bile destekler. Ağ güvenlik duruşunu anlamanın en iyi yöntemlerinden biri onu yenmeye çalışmaktır. Kendinizi bir saldırganın yerine koyun ve bu bölümdeki teknikleri ağlarınıza karşı kullanın. Bir FTP sıçrama taraması, boşta tarama, parçalanma saldırısı başlatın veya kendi proxy'lerinizden biri üzerinden tünel açmayı deneyin.

Ağ faaliyetlerini kısıtlamanın yanı sıra, şirketler saldırı tespit sistemleri (IDS) ile trafiği giderek daha fazla izlemektedir. Tüm büyük IDS'ler Nmap taramalarını tespit etmek için tasarlanmış kurallarla birlikte gönderilir, çünkü taramalar bazen saldırının öncüsüdür. Bu ürünlerin çoğu, kötü niyetli olduğu düşünülen trafiği aktif olarak engelleyen saldırı önleme sistemlerine (IPS) dönüşmüştür. Ne yazık ki ağ yöneticileri ve IDS satıcıları için, paket verilerini analiz ederek kötü niyetleri güvenilir bir şekilde tespit etmek zor bir sorundur. Sabır, beceri ve

belirli Nmap seçeneklerinin yardımıyla saldırganlar genellikle IDS'leri tespit edilmeden geçebilirler. Bu arada, yöneticiler masum faaliyetlerin yanlış teşhis edildiği ve uyarıldığı veya engellendiği çok sayıda yanlış pozitif sonuçla başa çıkmak zorundadır.

Why Would Ethical Professionals (White-hats) Ever Do This? (Etik Profesyonelleri (Beyaz Şapkaları) Bunu Neden Yapsın?)

Bazı beyaz şapkalı okuyucular bu bölümü atlamak isteyebilir. Kendi ağlarınıza karşı yetkili kullanım için, neden kendi güvenlik sistemlerinizden kaçmak isteyesiniz ki? Çünkü gerçek saldırganların tehlikesini anlamana yardımcı olur. Nmap doğrudan RPC taraması kullanarak engellenmiş bir portmapper bağlantı noktasının etrafından dolaşabiliyorsanız, kötü adamlar da yapabilir. Karmaşık güvenlik duvarlarını ve diğer cihazları yapılandıırken hata yapmak kolaydır. Hatta birçoğu, bilinçli kullanıcıların bulup kapatması gereken göze çarpan güvenlik açıklarıyla birlikte gelir. Düzenli ağ taraması, saldırganlardan önce tehlikeli örtük kuralların (örneğin Checkpoint Firewall-1 veya Windows IPsec filtrelerinizde) bulunmasına yardımcı olabilir.

IDS'lerden kaçmak için iyi nedenler de vardır. Ürün değerlendirme en yaygın olanlardan biridir. Eğer saldırganlar sadece bir ya da iki Nmap bayrağı ekleyerek radarın altından kayabiliyorsa, sistem çok fazla koruma sunmuyor demektir. Yine de script kiddie'leri ve solucanları yakalayabilir, ancak bunlar genellikle zaten çok açıktır.

Zaman zaman insanlar Nmap'in güvenlik duvari kurallarından kaçmak veya IDS'leri gizlice geçmek için özellikler sunmaması gerektiğini öne sürüyorlar. Bu özelliklerin saldırganlar tarafından kötüye kullanılma olasılığının, yöneticiler tarafından güvenliği artırmak için kullanılma olasılığı kadar yüksek olduğunu savunurlar. Bu mantıkla ilgili sorun, bu yöntemlerin saldırganlar tarafından kullanılmaya devam edeceği ve saldırganların başka araçlar bulacağı ya da işlevselliği Nmap'e yamalayacağıdır. Bu arada, yöneticilerin işlerini yapmaları çok daha zor olacaktır. Yalnızca modern, yamalı FTP sunucuları dağıtmak, FTP sıçrama saldırısını uygulayan araçların dağıtımını engellemeye çalışmaktan çok daha güçlü bir savunmadır.

Determining Firewall Rules (Güvenlik Duvarı Kurallarını Belirleme)

Güvenlik duvarı kurallarını atlamanın ilk adımı onları anlamaktır. Nmap, mümkün olan yerlerde, erişilebilir ancak kapalı olan ve aktif olarakfiltrelenen bağlantı noktaları arasında ayrımlı yapar. Etkili bir teknik, normal bir SYN bağlantı noktası taramasıyla başlamak, ardından ağı daha iyi anlamak için ACK taraması ve IP Kimliği sıralaması gibi daha egzotik tekniklere geçmektir.

Standard SYN Scan (Standart SYN Taraması)

TCP protokolünün yararlı bir özelliği, sistemlerin RFC 793 tarafından beklenmedik bağlantı isteklerine TCP RST (sıfırlama) paketi şeklinde olumsuz bir yanıt göndermesi gerekliliğidir. RST paketi, kapalı bağlantı noktalarının Nmap tarafından tanınmasını kolaylaştırır. Öte yandan, güvenlik duvarları gibi filtreleme cihazları, izin verilmeyen bağlantı noktalarına yönelik paketleri düşürme eğilimindedir. Bazı durumlarda bunun yerine ICMP hata mesajları (genellikle porta ulaşılamıyor) gönderilirler. Düşen paketler ve ICMP hataları RST paketlerinden kolayca ayırt edilebildiğinden, Nmapfiltrelenmiş TCP portlarını açık veya kapalı olanlardan güvenilir bir şekilde tespit edebilir ve bunu otomatik olarak yapar. Bu durum Örnek 10.1'de gösterilmektedir.

Örnek 10.1. Kapalı vefiltrelenmiş TCP portlarının tespiti

```
# nmap -sS -T4 scanme.nmap.org
Starting Nmap ( https://nmap.org )
Nmap scan report for scanme.nmap.org (64.13.134.52)
Not shown: 994 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    closed smtp
53/tcp    open  domain
70/tcp    closed gopher
80/tcp    open  http
113/tcp   closed auth

Nmap done: 1 IP address (1 host up) scanned in 5.40 seconds
```

Örnek 10.1'deki en önemli satırlardan biri "Gösterilmemiştir: 994 filtrelenmiş bağlantı noktası". Başka bir deyişle, bu ana bilgisayarın uygun bir varsayılan güvenlik duvari politikası vardır. Yalnızca yöneticinin açıkça izin verdiği bağlantı noktalarına erişilebilirken, varsayılan eylem bunları reddetmektedir (filtrelemektedir). Numaralandırılmış bağlantı noktalarından üçü açık durumdadır (22, 53 ve 80) ve diğer üçü kapalıdır (25, 70 ve 113). Test edilen 994 portun geri kalanına bu standart tarama ile ulaşılamaz (filtrelenmiştir).

Sneaky firewalls that return RST (RST döndüren sinsi güvenlik duvarları)

Kapalı TCP portları (bir RST paketi döndüren) ve filtrelenmiş portlar (hiçbir şey döndürmeyen veya bir ICMP hatası döndüren) arasındaki Nmap ayrimı genellikle doğru olsa da, birçok güvenlik duvari cihazı artık RST paketlerini hedef ana bilgisayardan geliyormuş gibi taklit edebiliyor ve portun kapalı olduğunu iddia edebiliyor. Bu yeteneğin bir örneği, istenmeyen paketleri reddetmek için birçok yöntem sunan Linux iptables sistemidir. iptables man sayfası bu özelliği aşağıdaki gibi belgelemektedir:

--reject-with type ⇒ Verilen tür icmp-net-unreachable, icmp-host-unreachable, icmp-port-unreachable, icmp-proto-unreachable, icmp-net-prohibited veya icmp-host-prohibited olabilir, bunlar uygun ICMP hata mesajını döndürür (port-unreachable varsayılandır). tcp-reset seçeneği yalnızca TCP protokolüyle eşleşen kurallarda kullanılabilir: bu, bir TCP RST paketinin geri gönderilmesine neden olur. Bu, özellikle bozuk posta ana bilgisayarlarına posta gönderirken sıkılıkla ortaya çıkan (aksi takdirde postanızı kabul etmeyecek olan) ident (113/tcp) problemini engellemek için kullanışlıdır.

Güvenlik duvarları ve IDS/IPS tarafından RST paketlerinin taklit edilmesi, yasal ağ operatörleri için kafa karıştırıcı olabileceğinden ve ayrıca tarayıcıların düşen paketlerin neden olduğu zaman aşımını beklemeden hemen bir sonraki bağlantı noktasına geçmesine izin verdiginden, 113 numaralı bağlantı noktası dışında özellikle yaygın değildir. Bununla birlikte, bu durum gerçekleşmektedir. Böyle bir sahtecilik genellikle RST paketinin makine tarafından gönderilen diğer paketlerle karşılaştırılarak dikkatli bir şekilde analiz edilmesiyle tespit edilebilir. "Güvenlik Duvarı ve Saldırı Tespit Sistemleri Tarafından Paket Sahteciliğinin Tespit Edilmesi" başlıklı bölümde bunun için etkili teknikler açıklanmaktadır.

ACK Scan (ACK Taraması)

"TCP ACK Taraması (-sA)" adlı bölümde ayrıntılı olarak açıklandıgı gibi, ACK taraması TCP paketlerini yalnızca ACK biti ayarlanmış olarak gönderir. Portlar açık ya da kapalı olsun, RFC 793'e göre hedefin bir RST paketiyle yanıt vermesi gereklidir. Öte yandan, probu engelleyen güvenlik duvarları genellikle yanıt vermez veya bir ICMP hedefine ulaşamıyor hatası gönderir. Bu ayrim, Nmap'in ACK paketlerinin filtrelenip filtrelenmediğini rapor etmesini sağlar. Bir Nmap ACK taraması tarafından bildirilen filtrelenmiş bağlantı noktaları kümesi genellikle aynı makineye karşı bir SYN taramasından daha küçütür, çünkü ACK taramalarının filtrelenmesi daha zordur. Birçok ağ neredeyse sınırsız giden bağlantılarla izin verir, ancak internet ana bilgisayarlarının kendilerine geri bağlantı başlatmasını engellemek ister. Gelen SYN paketlerini (ACK biti ayarlanmadan) engellemek bunu yapmanın kolay bir yoludur, ancak yine de ACK paketlerinin geçmesine izin verir. Bu ACK paketlerini engellemek daha zordur,

çünkü bağlantıyi hangi tarafın başlattığını söylemezler. Meşru bağlantılarla ait ACK paketlerine izin verirken istenmeyen ACK paketlerini (Nmap ACK taraması tarafından gönderildiği gibi) engellemek için, güvenlik duvarları belirli bir ACK'nın uygun olup olmadığını belirlemek için kurulan her bağlantı durumsal olarak izlemelidir. Bu durumsal güvenlik duvarları genellikle daha güvenlidir çünkü daha kısıtlayıcı olabilirler. ACK taramalarını engellemek mevcut ekstra kısıtlamalardan biridir. Dezavantajları, çalışmak için daha fazla kaynak gerektirmeleri ve durum bilgisine sahip bir güvenlik duvarının yeniden başlatılmasının bir cihazın durumunu kaybetmesine ve içinden geçen tüm kurulu bağlantıların sonlandırılmasına neden olabilmesidir.

Durum bilgisine sahip güvenlik duvarları yaygın ve popüleritesi artıyor olsa da, durum bilgisine sahip olmayan yaklaşım hala oldukça yaygındır. Örneğin, Linux Netfilter/iptables sistemi yukarıda açıklanan durumsuz yaklaşımın uygulanmasını kolaylaştırmak için --syn kolaylık seçeneğini desteklemektedir.

Önceki bölümde, bir SYN taraması scanme.nmap.org üzerindeki 1.000 ortak bağlantı noktasından altı tanesi hariç hepsinin filtrelenmiş durumda olduğunu göstermiştir. Örnek 10.2, durum bilgisi içeren bir güvenlik duvari kullanıp kullanmadığını belirlemek için aynı ana bilgisayara karşı bir ACK taramasını göstermektedir.

Örnek 10.2. Scanme'ye karşı ACK taraması

```
# nmap -sA -T4 scanme.nmap.org
Starting Nmap ( https://nmap.org )
Nmap scan report for scanme.nmap.org (64.13.134.52)
Not shown: 994 filtered ports
PORT      STATE      SERVICE
22/tcp    unfiltered ssh
25/tcp    unfiltered smtp
53/tcp    unfiltered domain
70/tcp    unfiltered gopher
80/tcp    unfiltered http
113/tcp   unfiltered auth

Nmap done: 1 IP address (1 host up) scanned in 5.96 seconds
```

SYN taramasında görüntülenen aynı altı bağlantı noktası burada gösterilir. Diğer 994 port hala filtrelenmektedir. Bunun nedeni Scanme'nin şu durum tabanlı iptables yönergesi tarafından korunuyor olmasıdır: iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT. Bu sadece kurulmuş bir bağlantının parçası olan veya onunla ilgili olan paketleri kabul eder. Nmap tarafından gönderilen istenmeyen ACK paketleri, gösterilen altı özel bağlantı noktasında düşürülür. Özel kurallar 22, 25, 53, 70 ve 80 numaralı bağlantı noktalarına gelen tüm paketlere ve 113 numaralı bağlantı noktası probleme yanıt olarak bir RST paketi gönderilmesine izin verir. ACK taraması bunları açık (22, 53 ve 80) veya kapalı (25, 70, 113) olarak ayıramadığından, gösterilen altı bağlantı noktasının filtrelenmemiş durumda olduğunu unutmayın.

Şimdi başka bir örneğe bakalım. Yerel ağındaki Para adlı bir Linux ana bilgisayarı aşağıdaki (yer kazanmak için basitleştirilmiş) güvenlik duvari komut dosyasını kullanıyor:

```

#!/bin/sh
#
# A simple, stateless, host-based firewall script.

# First of all, flush & delete any existing tables
iptables -F
iptables -X

# Deny by default (input/forward)
iptables --policy INPUT DROP
iptables --policy OUTPUT ACCEPT
iptables --policy FORWARD DROP

# I want to make ssh and www accessible from outside
iptables -A INPUT -m multiport -p tcp --destination-port 22,80 -j ACCEPT

# Allow responses to outgoing TCP requests
iptables -A INPUT --proto tcp ! --syn -j ACCEPT

```

Bu güvenlik duvarı --state seçeneği ya da -m state modül isteğinden hiçbir iz olmadığı için durumsuzdur. Örnek 10.3 bu ana bilgisayara karşı SYN ve ACK taramalarını göstermektedir.

Örnek 10.3. Para'ya karşı SYN ve ACK taramalarının karşılaştırılması

```

# nmap -sS -p1-100 -T4 para
Starting Nmap ( https://nmap.org )
Nmap scan report for para (192.168.10.191)
Not shown: 98 filtered ports
PORT      STATE    SERVICE
22/tcp     open     ssh
80/tcp     closed   http
MAC Address: 00:60:1D:38:32:90 (Lucent Technologies)

Nmap done: 1 IP address (1 host up) scanned in 3.81 seconds

# nmap -sA -p1-100 -T4 para
Starting Nmap ( https://nmap.org )
All 100 scanned ports on para (192.168.10.191) are: unfiltered
MAC Address: 00:60:1D:38:32:90 (Lucent Technologies)

Nmap done: 1 IP address (1 host up) scanned in 0.70 seconds

```

SYN taramasında 100 bağlantı noktasından 98'i filtrelenmiştir. Ancak ACK taraması, taranan her bağlantı noktasınınfiltrelenmediğini gösterir. Başka bir deyişle, tüm ACK paketleri engellenmeden gizlice geçmeyece ve RST yanıtlarını ortaya çıkarmaktadır. Bu yanıtlar ayrıca zaman aşımalarını beklemek zorunda kalmadığı için taramayı beş kattan daha hızlı hale getirir.

Artık durumlu ve durumsuz güvenlik duvarları arasında nasıl ayırm yapacağımızı biliyoruz, ama bu ne işe yarar? Para'nın ACK taraması, bazı paketlerin muhtemelen hedef ana bilgisayara ulaşlığını gösteriyor. Muhtemelen diyorum çünkü güvenlik duvari sahteciliği her zaman mümkündür. Bu bağlantı noktalarına TCP bağlantıları kuramasınız da, hangi IP adreslerinin kullanıldığını belirlemek, işletim sistemi algılama testleri, belirli IP kimliği saçmalıkları ve bu makinelerde yüklü rootkit'lere komutları tünellemek için bir kanal olarak yararlı olabilirler. FIN taraması gibi diğer tarama türleri, hangi portların açık olduğunu belirleyebilir ve böylece ana bilgisayarların amacını çıkarabilir. Bu tür ana bilgisayarlar IP ID boşta taraması için zombi olarak faydalı olabilir.

Bu tarama çifti aynı zamanda liman durumu dediğimiz şeyin yalnızca limanın kendisine ait bir özellik olmadığını da göstermektedir. Burada, aynı port numarası bir tarama türü tarafından filtrelenmiş ve bir diğeri tarafından filtrelenmemiş olarak kabul edilir. Hangi IP adresinden tarama yaptığınız, yol üzerindeki filtreleme cihazlarının kuralları ve hedef makinenin hangi arayüzüne eriştiğiniz Nmap'in portları nasıl gördüğünü etkileyebilir. Port tablosu yalnızca Nmap'in belirli bir makineden, belirli bir zamanda, tanımlanmış bir dizi seçenekle çalışırken gördüklerini yansıtır.

IP ID Tricks (IP Kimliği Hileleri)

IP başlıklarındaki mütevazı tanımlama alanı şartsızca miktarda bilgiyi ifşa edebilir. Bu bölümün ilerleyen kısımlarında bağlantı noktası taraması (boş tarama teknigi) ve güvenlik duvarı ve saldırı tespit sistemlerinin RST paketlerini korumalı ana bilgisayarlardan geliyormuş gibi gösterdiğini tespit etmek için kullanılacaktır. Bir başka güzel numara da hangi kaynak adreslerinin güvenlik duvarından geçtiğini anlamaktır. Eğer yol üzerindeki bir güvenlik duvarı bu tür paketlerin tümünü düşürüyorsa, 192.168.0.1 "adresinden" kör bir sahtekarlık saldırısı için saatler harcamanın bir anlamlı yoktur.

Bu durumu genellikle Nmap ile birlikte gelen ücretsiz ağ tarama aracı Nping ile test ediyorum. Bu oldukça karmaşık bir tekniktir, ancak bazen değerli olabilir. İşte benim attığım adımlar:

1. Dahili ağdaki bir makinenin en az bir erişilebilir (açık veya kapalı) portunu bulun. Yönlendiriciler, yazıcılar ve Windows kutuları genellikle iyi çalışır. Linux, Solaris ve OpenBSD'nin son sürümleri tahmin edilebilir IP ID sıra numaraları sorununu büyük ölçüde çözmüştür ve çalışmayaçaktır. Kafa karıştırıcı sonuçlardan kaçınmak için seçilen makine çok az ağ trafiğine sahip olmalıdır.
2. Makinenin tahmin edilebilir IP ID dizilerine sahip olduğunu doğrulayın. Aşağıdaki komut Playground adlı bir Windows XP makinesini test eder. Nping seçenekleri, 80 numaralı bağlantı noktasına birer saniye arayla beş SYN paketi gönderilmesini ister.

```
# nping -c 5 --delay 1 -p 80 --tcp playground
Starting Nping ( https://nmap.org/nping )
SENT (0.0210s) TCP 192.168.0.21:42091 > 192.168.0.40:80 S ttl=64 id=48089 iplen=40 seq=136013019 win=1480
RCVD (0.0210s) TCP 192.168.0.40:80 > 192.168.0.21:42091 RA ttl=128 id=4900 iplen=40 seq=0 win=0
SENT (1.0220s) TCP 192.168.0.21:42091 > 192.168.0.40:80 S ttl=64 id=41250 iplen=40 seq=136013019 win=1480
RCVD (1.0220s) TCP 192.168.0.40:80 > 192.168.0.21:42091 RA ttl=128 id=4901 iplen=40 seq=0 win=0
SENT (2.0240s) TCP 192.168.0.21:42091 > 192.168.0.40:80 S ttl=64 id=10588 iplen=40 seq=136013019 win=1480
RCVD (2.0250s) TCP 192.168.0.40:80 > 192.168.0.21:42091 RA ttl=128 id=4902 iplen=40 seq=0 win=0
SENT (3.0270s) TCP 192.168.0.21:42091 > 192.168.0.40:80 S ttl=64 id=55928 iplen=40 seq=136013019 win=1480
RCVD (3.0280s) TCP 192.168.0.40:80 > 192.168.0.21:42091 RA ttl=128 id=4903 iplen=40 seq=0 win=0
SENT (4.0300s) TCP 192.168.0.21:42091 > 192.168.0.40:80 S ttl=64 id=3309 iplen=40 seq=136013019 win=1480
RCVD (4.0300s) TCP 192.168.0.40:80 > 192.168.0.21:42091 RA ttl=128 id=4904 iplen=40 seq=0 win=0

Max rtt: 0.329ms | Min rtt: 0.288ms | Avg rtt: 0.300ms
Raw packets sent: 5 (200B) | Rcvd: 5 (230B) | Lost: 0 (0.00%)
Tx time: 4.00962s | Tx bytes/s: 49.88 | Tx pkts/s: 1.25
Rx time: 5.01215s | Rx bytes/s: 45.89 | Rx pkts/s: 1.00
Nping done: 1 IP address pinged in 5.03 seconds
```

IP ID alanları mükemmel bir şekilde sıralı olduğundan, bir sonraki teste geçebiliriz. Rastgele ya da birbirinden çok uzak olsalardır, yeni bir erişilebilir ana bilgisayar bulmamız gereklidir.

3. Kendi ana bilgisayarınıza yakın bir ana bilgisayardan (hemen hemen her ana bilgisayarınızı görür) hedefe sonda göndermeye başlayın. Örnek bir komut nping -S scanme.nmap.org --rate 10 -p 80 -c 10000 --tcp playground şeklindedir. scanme.nmap.org yerine istediğiniz başka bir ana bilgisayarı ve playground yerine de hedef ana bilgisayarınızı yazın. Yanıt almak gereklidir, çünkü amaç sadece IP ID dizilerini artırmaktır. Nping'i çalıştırığınız makinenin gerçek adresini kullanmayın. Kendi ISP'nizin paketleri engellemesi olasılığını azaltmak için ağ üzerinde yakındakı bir makineyi kullanmanız önerilir.

Bu işlem devam ederken, bir önceki adımdaki testi hedef makinenizde yeniden yapın.

```
# nping -c 5 --delay 1 -p 80 --tcp playground

Starting Nping ( https://nmap.org/nping )
SENT (0.0210s) TCP 192.168.0.21:1781 > 192.168.0.40:80 S ttl=64 id=61263iplen=40 seq=292367194 win=1480
RCVD (0.0220s) TCP 192.168.0.40:80 > 192.168.0.21:1781 RA ttl=128 id=5755iplen=40 seq=0 win=0
SENT (1.0220s) TCP 192.168.0.21:1781 > 192.168.0.40:80 S ttl=64 id=30096iplen=40 seq=292367194 win=1480
RCVD (1.0220s) TCP 192.168.0.40:80 > 192.168.0.21:1781 RA ttl=128 id=5766iplen=40 seq=0 win=0
SENT (2.0240s) TCP 192.168.0.21:1781 > 192.168.0.40:80 S ttl=64 id=26815iplen=40 seq=292367194 win=1480
RCVD (2.0240s) TCP 192.168.0.40:80 > 192.168.0.21:1781 RA ttl=128 id=5777iplen=40 seq=0 win=0
SENT (3.0260s) TCP 192.168.0.21:1781 > 192.168.0.40:80 S ttl=64 id=49116iplen=40 seq=292367194 win=1480
RCVD (3.0270s) TCP 192.168.0.40:80 > 192.168.0.21:1781 RA ttl=128 id=5788iplen=40 seq=0 win=0
SENT (4.0290s) TCP 192.168.0.21:1781 > 192.168.0.40:80 S ttl=64 id=29161iplen=40 seq=292367194 win=1480
RCVD (4.0300s) TCP 192.168.0.40:80 > 192.168.0.21:1781 RA ttl=128 id=57991iplen=40 seq=0 win=0

Max rtt: 0.342ms | Min rtt: 0.242ms | Avg rtt: 0.272ms
Raw packets sent: 5 (200B) | Rcvd: 5 (230B) | Lost: 0 (0.00%)
Tx time: 4.00853s | Tx bytes/s: 49.89 | Tx pkts/s: 1.25
Rx time: 5.01106s | Rx bytes/s: 45.90 | Rx pkts/s: 1.00
Nping done: 1 IP address pinged in 5.03 seconds
```

Bu kez IP ID'leri saniyede bir yerine kabaca 11 artırıyor. Hedef saniyede 10 sahte paketimizi alıyor ve her birine yanıt veriyor. Her yanıt IP kimliğini artırır. Bazı ana bilgisayarlar iletişim kurdukları her IP adresi için benzersiz bir IP ID dizisi kullanır. Durum böyle olsaydı, IP kimliğinin bu şekilde sıçradığını görmezdi ve ağ üzerinde farklı bir hedef ana bilgisayar aramamız gerekiirdi.

4. Güvenlik duvarından geçmesine izin verildiğinden veya güvenildiğinden şüphelendiğiniz sahte adresleri kullanarak 3. adımı tekrarlayın. Güvenlik duvarının arkasındaki adreslerin yanı sıra 10.0.0.0/8, 192.168.0.0/16 ve 172.16.0.0/12 gibi RFC 1918 özel ağlarını deneyin. Ayrıca localhost (127.0.0.1) ve 127.0.0.1'in sabit kodlandığı durumları tespit etmek için 127.0.0.0/8'den başka bir adres deneyin. Kötü şöhretli Land hizmet reddi saldırısı da dahil olmak üzere sahte localhost paketleriyle ilgili birçok güvenlik açığı olmuştur. Yanlış yapılandırılmış sistemler bazen geri döngü arayüzünden gelip gelmediğini kontrol etmeden bu adreslere güvenirler. Bir kaynak adresi son ana bilgisayara ulaşırsa, IP Kimliği 3. adımda görüldüğü gibi atlayacaktır. Adım 2'de olduğu gibi yavaşça artmaya devam ederse, paketler muhtemelen bir güvenlik duvarı veya yönlendirici tarafından düşürülmüştür.

Bu teknigin nihai sonucu, güvenlik duvarından geçmesine izin verilen ve engellenen kaynak adres netbloklarının bir listesidir. Bu bilgi birkaç nedenden dolayı değerlidir. Bir şirketin engellemeyi veya izin vermeyi seçtiği IP adresleri, hangi adreslerin dahili olarak kullanıldığı veya güvenildiği konusunda ipuçları verebilir. Örneğin, bir şirketin üretim ağındaki makineler şirket ağındaki IP adreslerine güvenebilir veya bir sistem yöneticisinin kişisel makinesine güvenebilir. Aynı üretim ağındaki makineler bazen birbirlerine ya da localhost'a da güvenebilir. Yaygın IP tabanlı güven ilişkileri NFS dışa aktarmalarında, ana bilgisayar güvenlik duvari kurallarında, TCP sarmalayıcılarında, özel uygulamalarda, rlogin'de vb. görülür. Bir başka örnek de SNMP'dir; Cisco yönlendiricisine yapılan sahte bir istek, yönlendiricinin yapılandırma verilerini saldırgana geri aktarmasına (TFTP) neden olabilir. Bu sorunları bulmak ve istismar etmek için önemli bir zaman harcamadan önce, sahte paketlerin ulaşıp ulaşmadığını belirlemek için burada açıklanan testi kullanın.

Bu güvenilir kaynak adresi sorununa somut bir örnek olarak, bir keresinde bir şirketin özel UDP hizmetinin, yapılandırma dosyasına girilen özel netbloklardan gelen kullanıcıların kimlik doğrulamasını atlamasına izin verdiği bulmuştum. Bu ağ blokları farklı kurumsal konumlara karşılık geliyordu ve bu özellik yönetim ve hata ayıklamayı kolaylaştırmayı amaçlıyordu. İnternete bakan güvenlik duvari akıllıca bir şekilde bu adresleri engellemeye çalıştı, çünkü gerçek çalışanlar bunun yerine özel bir bağlantından üretme erişebiliyor. Ancak bu bölümde açıklanan teknikleri kullanarak güvenlik duvarının yapılandırma dosyasıyla tam olarak senkronize olmadığını gördüm. UDP kontrol mesajlarını başarıyla taklit edebildiğim ve uygulamalarını devralabildiğim birkaç adres vardı.

Bu güvenlik duvarı kurallarını haritalama teknigi Nmap kullanmaz, ancak sonuçlar gelecekteki çalışmalar için değerlidir. Örneğin, bu test belirli tuzakların (-D) kullanılıp kullanılmayacağını gösterebilir. En iyi tuzaklar hedef sisteme kadar ulaşacaktır. Buna ek olarak, IP ID boşta taramasının (daha sonra ele alınacaktır) çalışması için sahte paketlerin geçmesi gereklidir. Potansiyel kaynak IP'leri bu teknikle test etmek genellikle bir ağdaki her

potansiyel boşta proxy makinesini bulup test etmekten daha kolaydır. Potansiyel boştaki proxy'lerin yalnızca yukarıdaki iki numaralı adımı geçmeleri halinde test edilmeleri gereklidir.

UDP Version Scanning (UDP Sürüm Taraması)

Önceki bölümlerin hepsi yaygın TCP protokolüne odaklanmıştır. UDP ile çalışmak genellikle daha zordur çünkü protokol TCP'nin yaptığı gibi açık portların onaylanması sağlamaz. Birçok UDP uygulaması beklenmedik paketleri görmezden gelerek Nmap'i portun açık mı yoksafiltrelenmiş mi olduğundan emin olamayacaktır. Bu yüzden Nmap bu belirsiz portları Örnek 10.4'te gösterdiği gibi açık|filtrelenmiş durumuna yerleştirir.

Örnek 10.4. Güvenlik duvarlı ana bilgisayara karşı UDP taraması

```
# nmap -sU -p50-59 scanme.nmap.org

Starting Nmap ( https://nmap.org )
Nmap scan report for scanme.nmap.org (64.13.134.52)
PORT      STATE      SERVICE
50/udp    open|filtered re-mail-ck
51/udp    open|filtered la-maint
52/udp    open|filtered xns-time
53/udp    open|filtered domain
54/udp    open|filtered xns-ch
55/udp    open|filtered isi-gl
56/udp    open|filtered xns-auth
57/udp    open|filtered priv-term
58/udp    open|filtered xns-mail
59/udp    open|filtered priv-file

Nmap done: 1 IP address (1 host up) scanned in 1.38 seconds
```

Bu 10 portlu tarama pek yardımcı olmadı. Hiçbir port prob paketlerine yanıt vermedi ve bu yüzden hepsi açık veyafiltrelenmiş olarak listelendi. Hangi portların gerçekten açık olduğunu daha iyi anlamanın bir yolu, açık portlardan bir yanıt alma umuduyla düzinelere farklı bilinen UDP hizmeti için bir sürü UDP probu göndermektir. Nmap sürüm tespiti (Bölüm 7, Hizmet ve Uygulama Sürüm Tespiti) tam olarak bunu yapar. Örnek 10.5, aynı taramayı sürüm algılama (-sV) eklenmiş olarak göstermektedir.

Örnek 10.5. Güvenlik duvarlı ana bilgisayara karşı UDP sürüm taraması

```
# nmap -sV -sU -p50-59 scanme.nmap.org

Starting Nmap ( https://nmap.org )
Nmap scan report for scanme.nmap.org (64.13.134.52)
PORT      STATE      SERVICE      VERSION
50/udp    open|filtered re-mail-ck
51/udp    open|filtered la-maint
52/udp    open|filtered xns-time
53/udp    open      domain      ISC BIND 9.3.4
54/udp    open|filtered xns-ch
55/udp    open|filtered isi-gl
56/udp    open|filtered xns-auth
57/udp    open|filtered priv-term
58/udp    open|filtered xns-mail
59/udp    open|filtered priv-file

Nmap done: 1 IP address (1 host up) scanned in 56.59 seconds
```

Sürüm tespiti, 53 numaralı bağlantı noktasının (etki alanı) açık olduğunu ve hatta ne çalıştığını şüpheye yer bırakmayacak şekilde gösterir. Diğer portlar hala açık|filtrelenmiş çünkü problemlerin hiçbirine yanıt vermediler.

Muhtemelen filtrelenmişlerdir, ancak bu garanti değildir. SNMP gibi yalnızca doğru topluluk dizesine sahip paketlere yanıt veren bir hizmet çalıştırıyor olabilirler. Ya da Nmap sürüm tespit probunun bulunmadığı belirsiz veya özel bir UDP hizmeti çalıştırıyor olabilirler. Ayrıca bu taramanın bir önceki taramanın 40 katından daha uzun süրdüğünü dikkat edin. Tüm bu problemleri her porta göndermek nispeten yavaş bir işlemdir. --version-intensity 0 seçeneğinin eklenmesi, yalnızca belirli bir bağlantı noktası numarasındaki hizmetlerden yanıt alma olasılığı en yüksek olan problemleri göndererek tarama süresini önemli ölçüde azaltacaktır.

Bypassing Firewall Rules (Güvenlik Duvarı Kurallarını Atlama)

Güvenlik duvarı kurallarını haritalamak değerli olsa da, kuralları atlama genellikle birincil hedeftir. Nmap bunu yapmak için birçok teknik uygular, ancak çoğu yalnızca kötü yapılandırılmış ağlara karşı etkilidir. Ne yazık ki, bunlar yaygındır. Her bir tekniğin başarı olasılığı düşüktür, bu nedenle mümkün olduğunda çok sayıda farklı yöntem deneyin. Saldırmanın başarılı olmak için yalnızca bir yanlış yapılandırma bulması gereklidir, ağ savunucuları her deliği kapatmalıdır.

Exotic Scan Flags (Egzotik Tarama Bayrakları)

Önceki bölümde hangi hedef ağ bağlantı noktalarınınfiltrelendiğini belirlemek için ACK taramasının kullanılması ele alınmıştır. Ancak, erişilebilir bağlantı noktalarından hangilerinin açık veya kapalı olduğunu belirleyemedi. Nmap, istenen port durumu bilgisini sağlamaya devam ederken güvenlik duvarlarını gizlice geçmede iyi olan birkaç tarama yöntemi sunar. FIN taraması böyle bir tekniktir. "ACK Taraması" adlı bölümde, Para adlı bir makineye karşı SYN ve ACK taramaları çalıştırılmıştır. SYN taraması, muhtemelen güvenlik duvarı kısıtlamaları nedeniyle yalnızca iki açık bağlantı noktası gösterdi. Bu arada, ACK taraması açık portları kapalı olanlardan ayırt edememektedir. Örnek 10.6, Para'ya karşı bu kez FIN taraması kullanılarak yapılan bir başka tarama girişimini göstermektedir. Çıplak bir FIN paketi ayarlandığı için, bu paket SYN paketlerini engelleyen kuralları aşıyor. SYN taraması 100'ün altında yalnızca bir açık port bulurken, FIN taraması her ikisini de bulur.

Örnek 10.6. Durum bilgisi olmayan güvenlik duvarına karşı FIN taraması

```
# nmap -sF -p1-100 -T4 para
Starting Nmap ( https://nmap.org )
Nmap scan report for para (192.168.10.191)
Not shown: 98 filtered ports
PORT      STATE      SERVICE
22/tcp    open|filtered ssh
53/tcp    open|filtered domain
MAC Address: 00:60:1D:38:32:90 (Lucent Technologies)

Nmap done: 1 IP address (1 host up) scanned in 1.61 seconds
```

Hedef güvenlik duvarı kuralları ve hedef ana bilgisayar türü hangi tekniklerin işe yarayacağını belirlediğinden, diğer birçok tarama türü denmeye değerdir. Özellikle değerli bazı tarama türleri FIN, Maimon, Window, SYN/FIN ve NULL taramalarıdır. Bunların hepsi Bölüm 5, Port Tarama Teknikleri ve Algoritmaları'nda açıklanmıştır.

Source Port Manipulation (Kaynak Bağlantı Noktası Manipülasyonu)

Şaşkıncı derecede yaygın bir yanlış yapılandırma, trafiğe yalnızca kaynak bağlantı noktası numarasına göre güvenmektir. Bunun nasıl ortaya çıktığını anlamak kolaydır. Bir yönetici pırıl pırıl yeni bir güvenlik duvarı kuracak, ancak uygulamaları çalışmayı durdurmak nankör kullanıcıların şikayetleriyle dolup taşacaktır. Özellikle DNS bozulmuş olabilir çünkü harici sunuculardan gelen UDP DNS yanıtları artık ağa giremez. FTP başka bir yaygın

örnektir. Aktif FTP aktarımlarında, uzak sunucu istenen dosyayı aktarmak için istemciyle tekrar bağlantı kurmaya çalışır.

Bu sorunlara genellikle uygulama düzeyinde proxy'ler veya protokol ayırtıran güvenlik duvarı modülleri şeklinde güvenli çözümler mevcuttur. Ne yazık ki daha kolay, güvensiz çözümler de vardır. DNS yanıtlarının 53 numaralı bağlantı noktasından ve aktif FTP'nin 20 numaralı bağlantı noktasından geldiğine dikkat çeken birçok yönetici, bu bağlantı noktalarından gelen trafiğe izin verme tuzağına düşmüştür. Genellikle hiçbir saldırganın bu tür güvenlik duvarı açıklarını fark etmeyeceğini ve istismar etmeyeceğini varsayarlar. Diğer durumlarda, yöneticiler bunu daha güvenli bir çözüm uygulayana kadar kısa vadeli bir geçici önlem olarak görürler. Sonra da güvenlik yükseltmesini unuturlar.

Bu tuzağa düşenler sadece fazla çalışan ağ yöneticileri değildir. Çok sayıda ürün bu güvensiz kurallarla birlikte gönderilmiştir. Microsoft bile suçludur. Windows 2000 ve Windows XP ile birlikte gelen IPsec filtreleri, 88 numaralı bağlantı noktasından (Kerberos) gelen tüm TCP veya UDP trafiğine izin veren örtük bir kural içermektedir. Apple hayranları bu konuda fazla sevinmemeli çünkü Mac OS X Tiger ile birlikte gelen güvenlik duvarı da aynı derecede kötü. Jay Beale, güvenlik duvarı GUI'sindeki "UDP Trafiğini Engelle" kutucuğunu etkinleştirseniz bile, 67 (DHCP) ve 5,353 (Zeroconf) numaralı bağlantı noktalarından gelen paketlerin doğrudan geçtiğini keşfetti. Bu yapılandırmanın bir başka acıklı örneği de Zone Alarm kişisel güvenlik duvarının (2.1.25'e kadar olan sürümler) 53 (DNS) veya 67 (DHCP) kaynak bağlantı noktasına sahip tüm gelen UDP paketlerine izin vermesidir.

Nmap bu zayıflıklardan yararlanmak için -g ve --source-port seçeneklerini (eşdeğerdirler) sunar. Basitçe bir port numarası verin ve Nmap mümkün olduğunda bu porttan paketler gönderecektir. Belirli işletim sistemi tespit testlerinin düzgün çalışması için Nmap'in farklı port numaraları kullanması gereklidir. SYN taraması da dahil olmak üzere çoğu TCP taraması, UDP taraması gibi bu seçeneği tamamen desteklemektedir. Mayıs 2004'te JJ Gray, Bugtraq'ta Windows IPsec kaynak portu 88 hatasının kendi istemicilerinden birine karşı kullanıldığını gösteren örnek Nmap taramaları yayımlamıştır. Normal bir tarama ve ardından -g 88 taraması Örnek 10.7'de gösterilmektedir. Kısalık ve anlaşılırlık için bazı çıktılar çıkarılmıştır.

Örnek 10.7. Kaynak bağlantı noktası 88 kullanarak Windows IPsec filtresini atlama

```
# nmap -sS -v -v -Pn 172.25.0.14
Starting Nmap ( https://nmap.org )
Nmap scan report for 172.25.0.14
Not shown: 1658 filtered ports
PORT      STATE SERVICE
88/tcp    closed  kerberos-sec

Nmap done: 1 IP address (1 host up) scanned in 7.02 seconds

# nmap -sS -v -v -Pn -g 88 172.25.0.14
Starting Nmap ( https://nmap.org )
Nmap scan report for 172.25.0.14
Not shown: 1653 filtered ports
PORT      STATE SERVICE
135/tcp   open   msrpc
139/tcp   open   netbios-ssn
445/tcp   open   microsoft-ds
1025/tcp  open   NFS-or-IIS
1027/tcp  open   IIS
1433/tcp  open   ms-sql-s

Nmap done: 1 IP address (1 host up) scanned in 0.37 seconds
```

Kapalı olan 88 numaralı bağlantı noktasının, JJ'nin bu bağlantı noktasını kaynak bağlantı noktası olarak kullanmayı denemesine yol açan ipucu olduğunu unutmayın. Bu güvenlik açığı hakkında daha fazla bilgi için Microsoft Bilgi Bankası Makale 811832'ye bakın.

IPv6 Attacks (IPv6 Saldırıları)

IPv6 dünyayı tam olarak kasıp kavurmamış olsa da, Japonya ve diğer bazı bölgelerde oldukça popülerdir. Kuruluşlar bu protokolü benimsediklerinde, içgüdüşel olarak IPv4 ile yapmayı öğrendikleri gibi kilitlemeyi genellikle unuturlar. Ya da bunu yapmayı deneyebilirler ancak donanımlarının IPv6 filtreleme kurallarını desteklemediğini fark edebilirler. IPv6'yi filtrelemek bazen IPv4'ten daha kritik olabilir, çünkü genişletilmiş adres alanı genellikle normalde RFC 1918 tarafından belirtilen özel IPv4 adreslerini kullanmak zorunda olan ana bilgisayarlara global olarak adreslenebilir IPv6 adreslerinin tahsis edilmesine izin verir.

IPv4 varsayılanı yerine IPv6 taraması yapmak genellikle komut satırına -6 eklemek kadar kolaydır. İşletim sistemi algılama ve UDP taraması gibi bazı özellikler henüz bu protokol için desteklenmemektedir, ancak en popüler özellikler çalışmaktadır. Örnek 10.8, tanınmış bir IPv6 geliştirme ve savunma kuruluşunun uzun zaman önce gerçekleştirdiği IPv4 ve IPv6 taramalarını göstermektedir.

Örnek 10.8. IPv4 ve IPv6 taramalarını karşılaştırma

```
> nmap www.kame.net

Starting Nmap ( https://nmap.org )
Nmap scan report for kame220.Kame.net (203.178.141.220)
Not shown: 984 closed ports
Port      State    Service
19/tcp    filtered chargen
21/tcp    open     ftp
22/tcp    open     ssh
53/tcp    open     domain
80/tcp    open     http
111/tcp   filtered sunrpc
137/tcp   filtered netbios-ns
138/tcp   filtered netbios-dgm
139/tcp   filtered netbios-ssn
513/tcp   filtered login
514/tcp   filtered shell
2049/tcp  filtered nfs
2401/tcp  open     cvspserver
5999/tcp  open     ncd-conf
7597/tcp  filtered qaz
31337/tcp filtered Elite

Nmap done: 1 IP address (1 host up) scanned in 34.47 seconds

> nmap -6 www.kame.net

Starting Nmap ( https://nmap.org )
Nmap scan report for 3ffe:501:4819:2000:210:f3ff:fe03:4d0
Not shown: 994 closed ports
Port      State    Service
21/tcp    open     ftp
22/tcp    open     ssh
53/tcp    open     domain
80/tcp    open     http
111/tcp   open     sunrpc
2401/tcp  open     cvspserver

Nmap done: 1 IP address (1 host up) scanned in 19.01 seconds
```

İlk taramada SunRPC, Windows NetBIOS ve NFS gibi sıkılıkla istismar edilebilen hizmetler de dahil olmak üzere çok sayıda filtrelenmiş bağlantı noktası gösteriliyor. Ancak aynı ana bilgisayar IPv6 ile tarandığında filtrelenmiş hiçbir bağlantı noktası gösterilmiyor! Birdenbire SunRPC (111 numaralı bağlantı noktası) kullanılabilir hale geldi ve IPv6 özellikli bir rpcinfo veya IPv6'yi destekleyen Nmap sürüm tespiti tarafından sorgulanmayı bekliyor. Sorunu kendilerine bildirdikten kısa bir süre sonra düzelttiler.

İlk taramada SunRPC, Windows NetBIOS ve NFS gibi sıkılıkla istismar edilebilen hizmetler de dahil olmak üzere çok sayıda filtrelenmiş bağlantı noktası gösteriliyor. Ancak aynı ana bilgisayar IPv6 ile tarandığında filtrelenmiş hiçbir bağlantı noktası gösterilmiyor! Birdenbire SunRPC (111 numaralı bağlantı noktası) kullanılabilir hale geldi ve IPv6 özellikli bir rpcinfo veya IPv6'yi destekleyen Nmap sürüm tespiti tarafından sorgulanmayı bekliyor. Sorunu kendilerine bildirdikten kısa bir süre sonra düzelttiler.

IP ID Idle Scanning (IP Kimliği Boşta Tarama)

IP ID boşta taraması, gerçek adresinizden hedefe hiçbir paket gönderilmediği için en gizli tarama türlerinden biri olarak ün yapmıştır. Açık portlar, seçilen bir zombi makinenin IP ID dizilerinden çıkarılır. Boşta taramanın daha az bilinen bir özelliği, elde edilen sonuçların aslında zombinin hedef ana bilgisayarı doğrudan taraması durumunda elde edeceğiniz sonuçlar olmasıdır. g seçenekinin güvenilir kaynak bağlantı noktalarından yararlanmaya izin vermesine benzer bir şekilde, boşta tarama bazen güvenilir kaynak IP adreslerinden yararlanabilir. İlk olarak güvenlik araştırmacısı Antirez tarafından tasarlanan bu dahiyane tarama türü "TCP Idle Scan (-sl)" adlı bölümde tam olarak açıklanmıştır.

Multiple Ping Probes (Çoklu Ping Problemleri)

Güvenlik duvarlı ağlar üzerinden tarama yapmaya çalışırken sık karşılaşılan bir sorun, ping problemlerinin düşmesi sonucu ana bilgisayarların gözden kaçmasıdır. Bu sorunu azaltmak için Nmap çok çeşitli problemlerin paralel olarak gönderilmesine izin verir. Umarım en azından bir tanesi başarılı olur. Bölüm 3, Ana Bilgisayar Bulma ("Ping Tarama"), en iyi güvenlik duvarı kırma tekniklerine ilişkin deneyel veriler de dahil olmak üzere bu teknikleri derinlemesine tartısmaktadır.

Fragmentation (Parçalanma)

Bazı paket filtreleri IP paket parçalarıyla başa çıkmakta zorlanır. Paketleri kendileri yeniden birleştirebilirler, ancak bu ekstra kaynak gerektir. Ayrıca parçaların farklı yollar izleyerek yeniden birleştirilmeyi engelleme olasılığı da vardır. Bu karmaşalık nedeniyle, bazı filtreler tüm parçaları yok sayarken, diğerleri ilk parça hariç hepsini otomatik olarak geçirir. Eğer ilk parça TCP başlığının tamamını içerecek kadar uzun değilse ya da ikinci paket kısmen onun üzerine yazıyorsa ilginç şeyler olabilir. Bu sorunlara karşı savunmasız olan filtreleme cihazlarının sayısı azalıyor, ancak denemekten asla zarar gelmez.

Bir Nmap taraması -f belirtirse küçük IP parçaları kullanacaktır. Varsayılan olarak Nmap her parçaya sekiz bayta kadar veri ekler, böylece tipik bir 20 veya 24 baytlık (seçeneklere bağlı olarak) TCP paketi üç küçük parça halinde gönderilir. Her -f örneği maksimum parça veri boyutuna sekiz ekler. Böylece -f -f her bir parça içinde en fazla 16 veri baytına izin verir. Alternatif olarak, --mtu seçeneğini belirtebilir ve maksimum veri baytını bir argüman olarak verebilirsınız. --mtu argümanı sekizin katı olmalıdır ve -f seçeneğiyle birleştirilemez.

Bazı kaynak sistemler giden paketleri çekirdekte birleştirir. İptables bağlantı izleme modülüne sahip Linux buna bir örnektir. Gönderilen paketlerin parçalandığından emin olmak için Wireshark gibi bir sniffer çalışırken bir tarama yapın. Ana bilgisayar işletim sisteminiz sorunlara neden oluyorsa, IP katmanını atlamak ve ham ethernet çerçeveleri göndermek için --send-eth seçeneğini deneyin.

Parçalama yalnızca Nmap'in TCP ve UDP bağlantı noktası taramalarını (bağlantı taraması ve FTP sıçrama taraması hariç) ve işletim sistemi algılamasını içeren ham paket özellikleri için desteklenir. Sürüm algılama ve Nmap Scripting Engine gibi özellikler genellikle parçalanmayı desteklemez çünkü hedef hizmetlerle iletişim kurmak için ana bilgisayarınızı TCP yiğinına güvenirler.

Sıra dışı ve kısmen çakışan IP parçaları Ağ araştırması ve istismarı için yararlı olabilir, ancak bu Nmap'ten daha düşük seviyeli bir ağ aracı gerektirir. Nmap, parçaları herhangi bir çakışma olmadan sırayla gönderir.

Parçalanmış bir port taraması geçerse, Fragroute gibi bir araç, ana bilgisayara saldırmak için kullanılan diğer araçları ve açıkları parçalamak için kullanılabilir.

Proxies (Vekiller)

Özellikle Web için uygulama düzeyinde proxy'ler, algılanan güvenlik ve ağ verimliliği (önbellege alma yoluyla) avantajları nedeniyle popüler hale gelmiştir. Güvenlik duvarları ve IDS'ler gibi, yanlış yapılandırılmış proxy'ler de çözüdüklerinden çok daha fazla güvenlik sorununa neden olabilir. En sık karşılaşılan sorun, uygun erişim kontrollerinin ayarlanmamasıdır. İnternette yüz binlerce açık proxy bulunmakta ve herkesin bunları diğer internet sitelerine anonim atlama noktaları olarak kullanmasına izin vermektedir. Düzinelerce kuruluş bu açık proxy'leri bulmak ve IP adreslerini dağıtmak için otomatik tarayıcılar kullanıyor. Proxy'ler zaman zaman Çin hükümetinin ülke sakinlerine uyguladığı acımasız sansürden kaçmak gibi tartışmalı olumlu amaçlar için de kullanılıyor. Bu "Çin'in büyük güvenlik duvarının" New York Times web sitesinin yanı sıra hükümetin aynı fikirde olmadığı diğer haber, siyasi ve ruhani siteleri engellediği bilinmektedir. Ne yazık ki, açık proxy'ler, anonim olarak sitelere girmek, kredi kartı dolandırıcılığı yapmak ya da interneti spam ile doldurmak isteyen daha kötü niyetli kişiler tarafından daha sık kötüye kullanılıyor.

Internet kaynaklarına açık bir proxy barındırmak çok sayıda soruna neden olabilirken, daha ciddi bir durum açık proxy'lerin korumalı ağa geri bağlantılarla izin vermesidir. Dahili ana bilgisayarların İnternet kaynaklarına erişmek için bir proxy kullanması gerektiğine karar veren yöneticiler genellikle yanlışlıkla ters yöndeği trafiğe de izin verirler. Hacker Adrian Lamo, genellikle bu ters proxy teknğini kullanarak Microsoft, Excite, Yahoo, WorldCom, New York Times ve diğer büyük ağlara girmesiyle ünlüdür.

"ÇÖZÜM: Açık Proxy Tespiti gibi Özel İhtiyaçlara Uygun Sürüm Tespiti" başlıklı bölümde Nmap sürüm tespiti kullanarak açık proxy'leri bulmanın bir yolu anlatılmaktadır. Buna ek olarak, Packet Storm gibi Internet sitelerinde çok sayıda özel ücretsiz proxy tarayıcısı mevcuttur. Binlerce açık proxy'den oluşan listeler de yaygındır.

MAC Address Spoofing (MAC Adresi Sahtekarlığı)

Ethernet cihazları (Wi-Fi dahil) altı baytlık benzersiz bir ortam erişim kontrolü (MAC) adresi ile tanımlanır. İlk üç bayt, kurumsal olarak benzersiz bir tanımlayıcı (OUI) oluşturur. Bu önek IEEE tarafından bir satıcıya atanır. Satıcı daha sonra kalan üç baytı sattığı adaptörlere ve cihazlara benzersiz bir şekilde atamaktan sorumludur. Nmap, OUI'leri atandıkları satıcı adlarıyla eşleştirilen bir veritabanı içerir. Bu, bir ağı tararken cihazların tanımlanmasına yardımcı olur, ancak bu bölümde neden tamamen güvenilemeyeceği açıklanmaktadır. OUI veritabanı dosyası, nmap-mac-prefixes, "MAC Adresi Satıcı Önekleri: nmap-mac-prefixes" adlı bölümde açıklanmıştır.

MAC adresleri ethernet cihazlarına önceden atanmış olsa da, mevcut donanımların çoğunda bir sürücü ile değiştirilebilirler. Ancak çok az kişi MAC adresini değiştirdiğinden (hatta bir MAC adresi olduğunu bildiğinden), birçok ağ bu adresi tanımlama ve yetkilendirme amacıyla kullanır. Örneğin, çoğu kablosuz erişim noktası, erişimi belirli bir MAC adresi kümesiyle sınırlamak için bir yapılandırma seçeneği sunar. Benzer şekilde, bazı ücretli veya özel ağlar, bir web formu kullanarak bağlandıktan sonra sizin kimlik doğrulamaya veya ödeme yapmaya zorlayacaktır. Daha sonra MAC adresinize göre ağın geri kalanına erişmenize izin vereceklerdir. MAC adreslerini koklamadan (gönderilen ve alınan her çerçevede gönderilmeleri gereklidir) ve ardından ağa yetkisiz erişim elde etmek için bu MAC'i taklit etmenin genellikle kolay olduğu göz önüne alındığında, bu erişim kontrolü biçimi oldukça zayıftır. Ayrıca, bir yönlendiriciden geçerken bir son ana bilgisayarın MAC adresi değiştirildiğinden, yalnızca bir ağın kenarlarında etkilidir.

Erişim kontrolüne ek olarak, MAC adresleri bazen hesap verebilirlik için de kullanılır. Ağ yöneticileri, bir DHCP kiralaması aldıklarında veya ağıda yeni bir makine iletişim kurduğunda MAC adreslerini kaydedeler. Daha sonra ağ istismarı veya korsanlık şikayetleri gelirse, IP adresi ve olay zamanına göre MAC adresini bulurlar. Daha

sonra sorumlu makineyi ve sahibini bulmak için MAC'i kullanırlar. MAC adresi sahteciliğinin kolaylığı bu yaklaşımı bir dereceye kadar zayıflatır. Kullanıcılar suçu olsalar bile, sorumluluktan kaçmak için MAC adresi sahteciliği iddiasını ortaya atabilirler.

Nmap, --spoof-mac seçeneği ile MAC adresi sahteciliğini destekler. Verilen argüman çeşitli şekillerde olabilir. Eğer sadece 0 sayısı ise, Nmap oturum için tamamen rastgele bir MAC adresi seçer. Verilen dize çift sayıda onaltılık basamaktan oluşuyorsa (çiftler isteğe bağlı olarak iki nokta üst üste ile ayrılır), Nmap MAC olarak bunları kullanır. Eğer 12'den az hex hanesi verilmişse, Nmap altı baytin geri kalanını rastgele değerlerle doldurur. Eğer argüman sıfır veya onaltılık bir dize değilse, Nmap verilen dizeyi içeren bir satıcı adı bulmak için nmap-mac-prefixes'e bakar (büyük/küçük harfe duyarlı değildir). Bir eşleşme bulunursa, Nmap satıcının OUI'sini kullanır ve kalan üç bayti rastgele doldurur. Geçerli --spoof-mac bağımsız değişken örnekleri Apple, 0, 01:02:03:04:05:06, deadbeefcafe, 0020F2 ve Cisco'dur. Bu seçenek, Nmap'in gerçekten ethernet düzeyinde paketler gönderdiğinde emin olmak için --send-eth'i ima eder. Bu seçenek sadece SYN taraması veya OS tespiti gibi ham paket taramalarını etkiler, sürüm tespiti veya Nmap Scripting Engine gibi bağlantı odaklı özelliklerini etkilemez.

MAC adresi sahteciliği ağ erişimi için gerekli olmadığına bile aldatma amacıyla kullanılabilir. Eğer bir konferanstaysam ve Thinkpad'ımden --spoof-mac Apple ile bir tarama başlatırsam, şüpheli gözler odadaki MacBook kullanıcılarına söylebilir.

Source Routing (Kaynak Yönlendirme)

Bu eski usul teknik bazı durumlarda hala etkilidir. Yol üzerindeki belirli bir yönlendirici size sorun çıkarıyorrsa, onun etrafında bir rota bulmaya çalışın. Bu tekniğin etkinliği sınırlıdır çünkü paket filtreleme sorunları genellikle hedef ağ üzerinde veya yakınında meydana gelir. Bu makinelerin ya tüm kaynak yönlendirmeli paketleri düşürmesi ya da ağa giden tek yol olması muhtemeldir. Nmap --ip-options seçeneğini kullanarak hem gevşek hem de katı kaynak yönlendirmeyi destekler. Örneğin, --ip-options "L 192.168.0.7 192.168.30.9" belirtilerek paketin bu iki IP yol noktası üzerinden gevşek kaynak yönlendirmesi yapılması istenir. Sıkı kaynak yönlendirme için L yerine S belirtin. Sıkı kaynak yönlendirmeyi seçerseniz, yol boyunca her bir atlama belirtmeniz gerekeceğini unutmayın.

Modern bir ağda filtreleme politikalarından kaçmak için kullanılan kaynak yönlendirmenin gerçek hayattan bir örneği için "Güvenlik Duvarı Yıkımının Gerçek Hayattan Pratik Bir Örneği" adlı bölümme bakın. IPv4 kaynak yönlendirmesi çok yaygın olarak engellenirken, IPv6 kaynak yönlendirme biçimi çok daha yaygındır. Bu sorunla ilgili ilginç bir makaleye <http://lwn.net/Articles/232781/> adresinden ulaşabilirsiniz.

Nmap ile bir hedef makineye giden bir kaynak yönlendirilmiş yol keşfedilirse, istismar edilebilirlik port taramasıyla sınırlı değildir. Ncat, kaynak yönlendirmeli yollar üzerinden TCP ve UDP iletişimini etkinleştirilebilir (-g seçeneğini kullanın).

FTP Bounce Scan (FTP Sıçrama Taraması)

FTP sunucularının sadece küçük bir yüzdesi hala savunmasız olsa da, tüm müşterilerinizin sistemlerini bu sorun için kontrol etmeye değer. En azından, dışarıdan saldırganların diğer tarafları taramak için savunmasız sistemleri kullanmasına izin verir. Daha kötü yapılandırmalar saldırganların kuruluşun güvenlik duvarlarını aşmasına bile izin verir. Bu tekniğin ayrıntıları ve örnekleri "TCP FTP Bounce Scan (-b)" adlı bölümde verilmiştir. Örnek 10.9 bir HP yazıcısının bağlantı noktası taramasını aktarmak için kullanıldığını göstermektedir. Bu yazıcı kuruluşun güvenlik duvarının arkasındaysa, normalde erişilememeyen (salırgan için) dahili adresleri de taramak için kullanılabilir.

Örnek 10.9. FTP sıçrama taraması ile bir yazıcıdan yararlanma

```

felix~> nmap -p 22,25,135 -Pn -v -b XXX.YY.111.2 scanme.nmap.org

Starting Nmap ( https://nmap.org )
Attempting connection to ftp://anonymous:-wwwuser@XXX.YY.111.2:21
Connected:220 JD FTP Server Ready
Login credentials accepted by ftp server!
Initiating TCP ftp bounce scan against scanme.nmap.org (64.13.134.52)
Adding open port 22/tcp
Adding open port 25/tcp
Scanned 3 ports in 12 seconds via the Bounce scan.
Nmap scan report for scanme.nmap.org (64.13.134.52)
PORT      STATE    SERVICE
22/tcp    open     ssh
25/tcp    open     smtp
135/tcp   filtered msrpc

Nmap done: 1 IP address (1 host up) scanned in 21.79 seconds

```

Take an Alternative Path (Alternatif Bir Yol İzleyin)

"Kutunun dışında düşün" klijesini aşırı kullanmaktan nefret ediyorum, ancak iyi korunan bir ağın ön kapısına sürekli vurmak her zaman en iyi yaklaşım değildir. İçeri girmenin başka yollarını arayın. Telefon hatlarını dinleyin, özel ağ erişimi olan yan kuruluşlara saldırın ya da Wi-Fi koklama ekipmanıyla ofislerine gidin, hatta gizlice girip uygun bir ethernet girişine takın. Nmap tüm bu bağlantıları üzerinden iyi çalışır. Müşteriniz sizi ninja kıyafeti içinde veri merkezlerinin çatısına çıkarken yakalamadan önce sizme testi sözleşmenizin bu yöntemleri kapsadığından emin olun.

A Practical Real-life Example of Firewall Subversion (Güvenlik Duvarı Subversion'ının Gerçek Hayattan Pratik Bir Örneği)

Güvenlik duvarı kurallarını atlamak için birçok bireysel teknik ele alındığına göre, şimdi bunları gerçek hayatıki bir sizme testi senaryosunda bir araya getirmenin zamanı geldi. Her şey güvenlik uzmanı Michael Cain'in SecurityFocus sizme testi listesine gönderdiği bir gönderi ile başladı. O ve iş arkadaşı Demetris Papapetrou büyük bir şirketin iç ağında sizme testi yapıyordular ve bir VLAN'ın diğerine erişimini engelleyen güvenlik duvarı kurallarını atlatmışlardır. Bu başarıyı Nmap kullanarak gerçekleştirdiklerini okumaktan memnuniyet duydum ve hikayenin tamamı için onlara yazdım. Bu hikaye hem öğretici hem de ilham verici, çünkü en yaygın istismarlar başarısız olsa bile azmin ve bildiğiniz her tekniği denemenin değerini gösteriyor. Güvenlik duvarının sizi yenmesine izin vermeyin!

Hikaye, Michael ve Demetris'in yoğun bir şekilde filtrelenmiş bir ağda sıkışık kaldıklarını gösteren bir Nmap taraması yapmalarıyla başlıyor. Bazı kurumsal sunuculara ulaşabiliyorlar, ancak ağ üzerinde bir yerlerde bulunması gereken (potansiyel olarak savunmasız) masaüstü istemci makinelerin hiçbirine ulaşamıyorlar. Belki de kısıtlı bir konferans odası ya da lobi alanında ya da kurumsal misafirler için kurulmuş bir kablosuz erişim noktasındadırlar. Keşfedilen bazı ana bilgisayarlar ve ağlar Örnek 10.10'da gösterilmektedir. Bu hikayedeki birkaç ayrıntı (IP adresleri gibi) gizlilik nedeniyle değiştirilmiştir. Hedef şirkete Megacorp adını vereceğim.

Örnek 10.10. Megacorp'taki bazı ilginç ana bilgisayarlar ve ağlar

```

10.10.5.1 - A router/firewall which will give us grief later
10.10.5.42 - Our protagonists are scanning from this machine
10.10.6.30 - files2.megacorp.com; Nmap shows this is a Windows machine
              with port 445 open.
10.10.6.60 - mail.megacorp.com; Nmap OS detection shows that it is
              Solaris 8. Port 25 is open and accessible.
10.10.10.0/24 - Nothing shows up here, but many of the IPs have

```

reverse-DNS names, so Demetris suspects that a firewall may be blocking his probes. The goal is to reach any available hosts on this subnet.

Demetris, 10.10.10.0/24 ağında herhangi bir ana bilgisayarın saklanıp saklanmadığını belirleme hedefi gözü önüne alındığında, ICMP eko istek sorgularını (-PE) kullanarak basit bir ping taramasıyla başlar. Sonuçlar Örnek 10.11'de gösterilmektedir.

Örnek 10.11. Hedef ağa karşı ping taraması

```
# nmap -n -sn -PE -T4 10.10.10.0/24
Starting Nmap ( https://nmap.org )
Nmap done: 256 IP addresses (0 hosts up) scanned in 26.167 seconds
```

Ping taraması yanıt veren herhangi bir ana bilgisayar bulamıyor. Demetris anlaşılabilir bir şekilde hayal kırıklığına uğrar, ancak en azından bu bölümü daha ilginç ve öğretici hale getirir. Belki de ağ gerçekten boştur, ancak Demetris'in erişmesinin engellendiği savunmasız makinelerle de dolu olabilir. Daha derine inmesi gerekmektedir. Örnek 10.12'de, Demetris bu ağ üzerinde bir IP secer ve bir ping taraması gerçekleştirir. Paket düzeyinde neler olup bittiğini belirlemek için paket izleme (--packet-trace) ve ekstra ayrıntı (-vv) seçeneklerini belirtir. Sadece bir IP seçmenin nedeni, yüzlerce paketten oluşan kafa karıştırıcı bir selden kaçınılmaktır.

Örnek 10.12. Tek bir IP'ye karşı paket izleme

```
# nmap -vv -n -sn -PE -T4 --packet-trace 10.10.10.7
Starting Nmap ( https://nmap.org )
SENT (0.3130s) ICMP 10.10.5.42 > 10.10.10.7 echo request (type=8/code=0)
ttl=41 id=7193 ipLen=28
RCVD (0.3130s) ICMP 10.10.5.1 > 10.10.5.42 host 10.10.10.7 unreachable
(type=3/code=1) ttl=255 id=25980 ipLen=56
Nmap done: 1 IP address (0 hosts up) scanned in 0.313 seconds
```

Görünüşe göre Demetris bu IP'leri (ya da en azından bunu) taramaya çalışırken ICMP host unreachable mesajları alıyor. Yönlendiriciler genellikle bir ana bilgisayar kullanılamadığında ve bu nedenle bir MAC adresi belirleyemediklerinde bunu yapar. Bazen de filtrelemeden kaynaklanır. Demetris ağdaki diğer ana bilgisayarıları tarar ve aynı şekilde davrandıklarını doğrular. Yalnızca ICMP paketlerinin filtrelenmiş olması mümkündür, bu nedenle Demetris bir TCP SYN taraması denemeye karar verir. nmap -vv -n -sS -T4 -Pn --reason 10.10.10.0/24 komutunu çalıştırır. Tüm bağlantı noktaları filtrelenmiş olarak gösterilir ve --reason sonuçları bazı ana bilgisayar ulaşılamaz mesajlarını ve bazı yanıt vermeyen bağlantı noktalarını suçlar. Yanıt vermeyen bağlantı noktaları, yönlendirici tarafından gönderilen ana bilgisayar ulaşılamaz mesajlarının hız sınırlamasından kaynaklanıyor olabilir. Birçok yönlendirici her birkaç saniyede bir bunlardan yalnızca birini gönderir. Demetris, taramayı tekrar çalıştırarak ve ana bilgisayara ulaşılamıyor mesajlarının tam olarak aynı bağlantı noktası kümesi için gelip gelmediğine bakarak hız sınırlamasının neden olup olmadığını doğrulayabilir. Eğer portlar aynıysa, sorun belirli bir port tabanlı filtre olabilir. Nmap her seferinde farklı portlar için host-unreachable mesajları alıysa, bunun nedeni muhtemelen hız sınırlamasıdır.

Soruna bir filtre neden oluyorsa, yönlendiriciler ve anahtarlarla yaygın olarak bulunan basit bir durumsuz güvenlik duvarı olabilir. Önceki bölümlerde tartışıldığı gibi, bunlar bazen TCP ACK paketlerinin ihlal edilmeden geçmesine izin verir. Demetris taramayı tekrarlar, ancak ACK taraması için -sS yerine -sA belirtir. Taramada bulunanfiltrelenmemiş bağlantı noktaları ACK paketlerinin hedef ana bilgisayardan TCP RST yanıtı aldığı gösterir. Ne yazık ki, SYN taramasında olduğu gibi bu durumda da tüm sonuçlar filtrelenmiştir.

Demetris daha gelişmiş bir şey denemeye karar verir. İlk Nmap taramasından 445 numaralı portun 10.10.6.30 (files2.megacorp.com) adresindeki Windows makinesinde açık olduğunu zaten bilmektedir. Demetris 10.10.10.0/24 ağına doğrudan ulaşamamış olsa da, belki de files2 (önemli bir şirket dosya sunucusu olarak) bu IP aralığına erişebilmektedir. Demetris, IPID Idle taramasını kullanarak taramalarını files2'den sektirmeyi denemeye karar verir. İlk olarak, files2'nin bir zombi olarak çalıştığından emin olmak için onu 10.10.6.60'a karşı test etmek ister - 25 numaralı bağlantı noktası açık olan ve yanıt verdiği bilinen bir makine. Bu testin sonuçları Örnek 10.13'te gösterilmektedir.

Örnek 10.13. Boşta taramayı test etme

```
# nmap -vv -n -Pn -sI 10.10.6.30:445 -p 25 10.10.6.60
Starting Nmap ( https://nmap.org )

Initiating idle scan against 10.10.6.60 at 13:10
Idle scan using zombie 10.10.6.30 (10.10.6.30:445); Class: Incremental
Even though your Zombie (10.10.6.30) appears to be vulnerable to IP ID
sequence prediction (class: Incremental), our attempts have failed. This
generally means that either the Zombie uses a separate IP ID base for each
host (like Solaris), or because you cannot spoof IP packets (perhaps your ISP
has enabled egress filtering to prevent IP spoofing), or maybe the target
network recognizes the packet source as bogus and drops them
QUITTING!
```

10.10.6.30'u Boşta Çalışan Zombi olarak kullanmak pek işe yaramadı. Eğer sorun yoğun trafikten kaynaklanıysa, gece yarısı tekrar deneyebilirdi. "TCP Idle Scan (-sI)" adlı bölümün iyice okunmasıyla birlikte -packet-trace seçeneği, 10.10.6.30'un neden zombi olarak çalışmadığını belirtmeye yardımcı olabilir. Demetris ağ üzerinde bulduğu diğer birkaç ana bilgisayarı dener ve hiçbir zombi olarak çalışmaz.

Demetris 10.10.10.0/24 ağına girip giremeyeceği konusunda endişelenmeye başlar. Neyse ki bu konuda tecrübelidir ve elinde başka bir numara vardır: IP kaynak yönlendirmesi. İnternetin ilk günlerinde (ve hatta bugün IPv6 ile), kaynak yönlendirme önemli ve yaygın olarak kullanılan bir ağ tanılama özelliği idi. Normal yönlendirme kurallarına güvenmek yerine bir paketin hedefine gitmesini istediğiniz atlamları belirtmenize olanak tanır. Katı kaynak yönlendirme ile her atlamayı belirtmeniz gereklidir. Gevşek kaynak yönlendirme, önemli IP yol noktalarını doldurmanıza izin verirken, normal İnternet yönlendirmesi bu yol noktaları arasındaki atlama ayrıntılarını doldurur.

Uzun zaman önce ağ topluluğu, kaynak yönlendirmenin değerinden daha fazla sorun (özellikle güvenlik açısından) olduğu konusunda fikir birliğine vardı. Birçok yönlendirici (çoğu olmasa da) kaynak yönlendirmeli IPv4 paketlerini düşürecek şekilde yapılandırılmıştır, bu nedenle bazı kişiler 90'ların başından beri sorunun çözüldüğünü düşünmektedir. Yine de kaynak yönlendirme, SYN flooding ve Telnet şifre koklama gibi, nadir ama güçlü bir risk olarak devam etmektedir. Demetris, 10.10.6.60 posta sunucusu üzerinden gevşek kaynak yönlendirmeli paketler kullanarak files2'yi (10.10.6.30) pin taramasıyla bu saldırıyı test eder. Sonuçlar Örnek 10.14'te gösterilmektedir.

Örnek 10.14. Kaynak yönlendirmeyi test etme

```
# nmap -n -sn -PE --ip-options "L 10.10.6.60" --reason 10.10.6.30
Starting Nmap ( https://nmap.org )
Host 10.10.6.30 appears to be up, received echo-reply.
Nmap done: 1 IP address (1 host up) scanned in .313 seconds
```

Demetris testin işe yaramasına hem şaşırır hem de çok sevinir. Hemen dikkatini gerçek hedef ağına yöneltir ve ilk ping taramasını ek bir seçenekle tekrarlar: --ip-options "L 10.10.6.60". Bu kez Nmap 10.10.10.7 adresindeki

makinelerin yanıt verdiği bildirir. Demetris, 10.10.10.0/24 ve 10.10.5.0/24 alt ağlarının birbirleriyle iletişim kurmalarını engellemek için yapılandırılmış farklı yönlendirici VLAN'larında olması nedeniyle daha önce ulaşılamadığını öğrenir. Demetris'in kaynak yönlendirme tekniği bu politikada büyük bir boşluk açtı! Demetris, Örnek 10.15'te gösterildiği gibi 10.10.10.7 makinesini SYN taramasıyla takip eder.

Örnek 10.15. Sonunda başarı

```
# nmap -vv -n -sS -Pn --ip-options "L 10.10.6.60" --reason 10.10.10.7
Starting Nmap ( https://nmap.org )
Nmap scan report for 10.10.10.7
Not shown: 988 closed ports
Reason: 988 resets
PORT      STATE    SERVICE          REASON
21/tcp     filtered  ftp              no-response
23/tcp     filtered  telnet          no-response
25/tcp     open     smtp             syn-ack
80/tcp     open     http             syn-ack
135/tcp    open     msrpc            syn-ack
139/tcp    open     netbios-ssn     syn-ack
443/tcp    open     https            syn-ack
445/tcp    open     microsoft-ds   syn-ack
515/tcp    open     printer          syn-ack
1032/tcp   open     iad3             syn-ack
1050/tcp   open     java-or-OTGfileshare syn-ack
3372/tcp   open     msdtc            syn-ack
Nmap done: 1 IP address (1 host up) scanned in 21.203 seconds
```

Demetris bu ilk taramada işletim sistemi ve sürüm tespitini atmamıştır, ancak açık port profilinden bu bir Windows makinesi gibi görünmektedir. Demetris artık kaynak yönlendirme seçenekleri sunan Ncat gibi araçlar kullandığı sürece bu bağlantı noktalarına bağlanabilir ve erişebilir. Hikayede bundan sonra ne olacağını bilmiyorum ama tahminimce Demetris'in ağa tamamen girmesi ve ardından şirketin ağı daha güvenli bir şekilde yeniden tasarlamasına yardımcı olması gerekiyor.

Subverting Intrusion Detection Systems (İzinsiz Giriş Tespit Sistemlerini Yıkmak)

Modern saldırganların karşılaştığı tek engel güvenlik duvarları değildir. İzinsiz giriş tespit ve önleme sistemleri de sorun yaratabilir. Ağ yönetimi personeli, IDS'den gelen gece 02:00 saldırısı uyarı sayfalarını her zaman iyi karşılamaz. Düşünceli bilgisayar korsanları, eylemlerinin ilk etapta tüm bu uyarılara neden olmasını önlemek için özen gösterirler. İlk adım, bir IDS'in mevcut olup olmadığını tespit etmektir - birçok küçük şirket bunları kullanmaz. Eğer bir IDS'den şüphelenilir ya da tespit edilirse, onu alt etmek için birçok etkili teknik vardır. Bunlar, izinsiz girişe göre değişen üç kategoriye ayrılır: saldırgan orada değilmiş gibi IDS'den kaçınmak, yaniltıcı verilerle IDS'nin kafasını karıştırmak ve daha fazla ağ ayrıcalığı elde etmek veya sadece onu kapatmak için IDS'yi istismar etmek. Alternatif olarak, gizlilikle ilgilenmeyen saldırganlar, hedef ağa saldırırken IDS'yi tamamen görmezden gelebilirler.

Intrusion Detection System Detection (İzinsiz Giriş Tespit Sistemi Tespit)

Ağ yöneticileri ve kötü niyetli bilgisayar korsanları arasındaki bitmek bilmeyen savaşın başlarında, yöneticiler sistemleri sertleştirerek ve hatta bir çevre bariyeri olarak hareket etmek için güvenlik duvarlarını kurarak bölgelerini savundular. Bilgisayar korsanları ise güvenlik duvarlarını aşmak ya da etrafından dolaşmak ve savunmasız ana bilgisayarları istismar etmek için yeni araçlar geliştirdiler. Yöneticilerin sürekli olarak sinsi faaliyetleri izleyen saldırısı tespit sistemlerini devreye sokmasıyla silahlanma yarışı başladı. Saldırganlar da elbette

IDS'leri tespit etmek ve aldatmak için sistemler geliştirecek karşılık verdi. Saldırı tespit sistemlerinin pasif cihazlar olması amaçlansa da, birçoğu saldırganlar tarafından ağ üzerinden tespit edilebilmektedir.

En az göze çarpan IDS, hiç iletim yapmadan ağ trafiğini pasif olarak dinleyen bir IDS'dir. Saldırganlar tarafından tehlkiye atılsa bile IDS'nin iletim yapamamasını sağlamak için özel ağ musluk donanım cihazları mevcuttur. Böyle bir kurulumun güvenlik avantajlarına rağmen, pratik hususlar nedeniyle yaygın olarak kullanılmamaktadır. Modern IDS'ler merkezi yönetim konsollarına ve benzerlerine uyarı gönderebilmeyi bekler. Eğer IDS'in ürettiği tek şey bu olsaydı, risk minimum olurdu. Ancak uyarı hakkında daha kapsamlı veri sağlamak için, genellikle saldırganlar tarafından görülebilecek problemler başlatırlar.

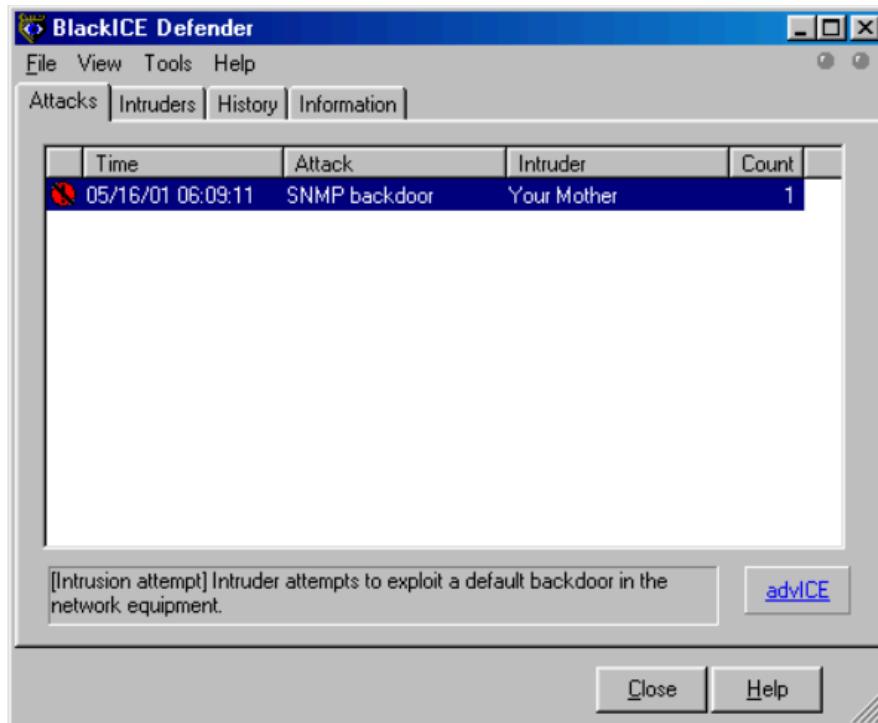
Reverse probes (Ters probalar)

IDS'ler tarafından yaygın olarak başlatılan bir sonda, saldırganın IP adresinin ters DNS sorusudur. Sonuçta bir uyarındaki alan adı IP adresinden daha değerlidir. Ne yazık ki, kendi rDNS'lerini kontrol eden saldırganlar (oldukça yaygındır) günlükleri gerçek zamanlı olarak izleyebilir ve tespit edildiklerini öğrenebilirler. Bu, saldırganların talep eden IDS'ye sahte isimler ve önbellek girdileri gibi yanlış bilgiler vermesi için iyi bir zamandır.

Bazı IDS'ler çok daha ileri gider ve görünürdeki saldırganlara daha müdahaleci problemler gönderir. Bir saldırgan hedefinin kendisini geri taradığını gördüğünde, alarmları harekete geçirdiğine şüphe yoktur. Bazı IDS'ler Windows NetBIOS bilgi isteklerini saldırgana geri gönderir. ISS BlackICE Defender bunu varsayılan olarak yapan (ya da en azından yapan) bir satıcıdır. Dinleyen BlackICE örneklerinden bir uyarı oluşturan basit bir paket gönderen icepick adında küçük bir araç yazdım. Daha sonra NetBIOS sorgularını izler ve bulunan BlackICE kurulumlarını rapor eder. Birisi bu IDS'yi arayan büyük ağları kolayca tarayabilir ve daha sonra bu bölümde daha sonra tartışılan açıkları kullanarak onları istismar etmeye çalışabilir.

Sadece BlackICE kurulumlarını bulmakla ya da sizme testleri sırasında tespit etmekle yetinmeyerek, sondaya yanlış bilgilerle yanıt veren windentd adlı basit bir Unix programı yazdım. Şekil 10.1, windentd ve icepick sayesinde Davetsiz Misafirin "Anneniz" olarak listelendiği bir BlackICE konsolunu göstermektedir. Bu basit araçlar desteklenmese de <https://nmap.org/presentations/CanSecWest01/> adresinden edinilebilir.

Şekil 10.1. BlackICE alışılmadık bir davetsiz misafir keşfeder



Sudden firewall changes and suspicious packets (Anı güvenlik duvarı değişiklikleri ve şüpheli paketler)

Birçok saldırı tespit sistemi, pazarlama departmanlarının saldırısı önleme sistemleri olarak adlandırdığı sistemlere dönüştürülmüştür. Bazıları sadece normal bir IDS gibi ağı koklayabilir ve tetiklenmiş paket yanıtlarını gönderebilir. En iyi IPS sistemleri, şüpheli bir faaliyet tespit edildiğinde paket akışını kısıtlayabilmek için ağ üzerinde sıralıdır. Örneğin, bir IPS, port taraması yaptığına inandığı veya bir arabellek taşıması istismarı girişiminde bulunan bir IP adresinden gelen trafiği engelleyebilir. Saldırganlar, bir sistemi port taramasından geçirdikten sonra bildirilen açık portlara bağlanamazlarsa bunu fark edebilirler. Saldırganlar başka bir IP adresinden bağlanmayı deneyerek engellendiklerini doğrulayabilirler.

Şüpheli yanıt paketleri, bir saldırının eylemlerinin bir IDS tarafından işaretlendiğine dair bir ipucu da olabilir. Özellikle, ağ üzerinde inline olmayan birçok IDS, bağlantıları koparmak için RST paketlerini taklit edeceklerdir. Bu paketlerin sahte olduğunu belirlemenin yolları "Güvenlik Duvarı ve Saldırı Tespit Sistemleri Tarafından Paket Sahteciliğini Tespit Etme" bölümünde ele alınmaktadır.

Naming conventions (Adlandırma kuralları)

Adlandırma kuralları IDS'nin varlığına dair başka bir ipucu olabilir. Bir Nmap liste taraması realsecure, ids-monitor veya dragon-ids gibi ana bilgisayar adları döndürürse, bir saldırı tespit sistemi bulmuş olabilirsiniz. Yöneticiler bu bilgiyi istemeden vermiş olabilirler ya da bunu ev ve araba camlarındaki alarm etiketleri gibi düşünüyor olabilirler. Belki de script çocukların IDS ile ilgili isimlerden korkup kaçacaklarını düşünüyorlardır. Yanlış bilgilendirme de olabilir. DNS adlarına asla tam olarak güvenemezsiniz. Örneğin, bugzilla.securityfocus.com'un popüler Bugzilla web tabanlı hata izleme yazılımını çalıştırın bir web sunucusu olduğunu varsayıbilirsiniz. Ama öyle değildir. Örnek 10.16'daki Nmap taraması bunun yerine muhtemelen bir Symantec Raptor güvenlik duvarı olduğunu göstermektedir. Hiçbir web sunucusuna erişilememektedir, ancak Raptor'un arkasında gizlenmiş bir web sunucusu olabilir.

Örnek 10.16. Ana bilgisayar adları aldatıcı olabilir

```
# nmap -sS -sV -T4 -p1-24 bugzilla.securityfocus.com

Starting Nmap ( https://nmap.org )
Nmap scan report for 205.206.231.82
Not shown: 21 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp-proxy Symantec Enterprise Firewall FTP proxy
22/tcp    open  ssh?
23/tcp    open  telnet   Symantec Raptor firewall secure gateway telnetd

Nmap done: 1 IP address (1 host up) scanned in 0.94 seconds
```

Unexplained TTL jumps (Açıklanamayan TTL atlamaları)

Belirli IDS'leri tespit etmenin bir başka yolu da traceroute'lardaki açıklanamayan boşlukları (veya şüpheli makineleri) izlemektir. Çoğu işletim sistemi bir traceroute komutu içerirken (Windows'ta tracert olarak kısaltılır), Nmap --traceroute seçeneği ile daha hızlı ve daha etkili bir alternatif sunar. Standart traceroute'un aksine, Nmap problemlerini paralel olarak gönderir ve tarama sonuçlarına göre ne tür bir probun en etkili olacağını belirleyebilir. Basitlik için uydurulmuş olan Örnek 10.17'de traceroute beşinci atlama hıçbir şey bulamamaktadır. Bu, hedef şirketi koruyan bir hat içi IDS ya da güvenlik duvarı olabilir. Elbette bu, rotanın bir parçası olmadan ağı pasif olarak koklayanların aksine yalnızca hat içi IDS'leri tespit edebilir. Bazı inline cihazlar bile TTL'yi azaltmadıkları veya ICMP ttl-exceeded mesajlarını korunan ağdan geri iletmemeyi reddettikleri için tespit edilemeyebilir.

Örnek 10.17. Traceroute ile TTL boşluklarını not etme

```
# nmap --traceroute www.target.com
Interesting ports on orestes.red.target.com (10.0.0.6)
Not shown: 996 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
80/tcp    open  http
113/tcp   closed auth

TRACEROUTE (using port 22/tcp)
HOP RTT    ADDRESS
1  1.10  gw (205.217.153.49)
2  10.40  metrol-ge-152.pa.meer.net (205.217.152.1)
3  12.02  208.185.168.171 (208.185.168.171)
4  14.74  p4-2-0-0.r06.us.bb.verio.net (129.250.9.129)
5  ...
6  15.07  orestes.red.target.com (10.0.0.6)

Nmap done: 1 IP address (1 host up) scanned in 4.35 seconds
```

Traceroute bu bilgiyi elde etmek için en iyi bilinen yöntem olsa da, tek yöntem değildir. IPv4, bu bilgileri toplamak için record route adı verilen belirsiz bir seçenek sunar. Maksimum IP başlık boyutu nedeniyle, en fazla dokuz atlama kaydedilebilir. Ayrıca, bazı ana bilgisayarlar ve yönlendiriciler bu seçenek ayarlandığında paketleri düşürür. Yine de geleneksel traceroute'un başarısız olduğu zamanlar için kullanışlı bir numaradır. Bu seçenek, seçeneği ayarlamak için --ip-options R ve yanittan okumak için --packet-trace kullanılarak Nmap ile belirtilebilir. Genellikle bir ICMP ping taraması (-sn -PE) ile birlikte kullanılır. Çoğu işletim sistemi, bu amaç için Nmap'ten daha kolay kullanılabilen ping komutuna bir -R seçeneği sunar. Bu tekninin bir örneği Örnek 10.18'de verilmiştir.

Örnek 10.18. IP kaydı rota seçeneğini kullanma

```

> ping -R 151.164.184.68
PING 151.164.184.68 (151.164.184.68) 56(124) bytes of data.
64 bytes from 151.164.184.68: icmp_seq=1 ttl=126 time=11.7 ms
NOP
RR:    192.168.0.100
       69.232.194.10
       192.168.0.6
       192.168.0.100

--- 151.164.184.68 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 11.765/11.765/11.765/0.000 ms

```

Avoiding Intrusion Detection Systems (İzinsiz Giriş Tespit Sistemlerinden Kaçınma)

Saldırı tespit sistemlerini alt etmenin en ince yolu, onların dikkatli bakışlarından tamamen kaçınılmaktır. Gerçek şu ki, IDS'leri yöneten kurallar oldukça kırılgandır, çünkü genellikle saldırıyı biraz manipüle ederek yenilebilirler. Saldırganlar, istismarlarının IDS tarafından tespit edilmesinden kaçmak için URL kodlamadan polimorfik kabuk kodu üreticilerine kadar düzinelere teknike sahiptir. Bu bölüm, güvenlik açıklarını gizlice kullanmaktan daha kolay olan gizli port taramasına odaklanmaktadır.

Slow down (Yavaşla)

IDS uyarılarından kaçınmak söz konusu olduğunda, sabır bir erdemdir. Port tarama tespiti genellikle eşik tabanlıdır. Sistem belirli bir zaman diliminde belirli sayıda probu izler. Bu, masum kullanıcılarından gelen yanlış pozitifleri önlemeye yardımcı olur. Aynı zamanda kaynakları korumak için de gereklidir - bağlantı problemlerini sonsuza kadar kaydetmek belleği tüketir ve gerçek zamanlı liste aramasını çok yavaşlatır. Bu eşik yaklaşımının dezavantajı, saldırıların tarama oranlarını eşliğin hemen altında tutarak bundan kaçınabilecekleridir. Nmap bunu başarmak için -T seçeneği ile seçilebilecek çeşitli hazır zamanlama modları sunar. Örneğin, -T paranoid seçeneği Nmap'in her seferinde sadece bir prob göndermesine ve aralarında beş dakika beklemesine neden olur. Büyük bir tarama haftalar sürebilir, ancak en azından muhtemelen tespit edilmeyecektir. T sneaky seçeneği de benzerdir, ancak probalar arasında yalnızca 15 saniye bekler.

Sinsi gibi hazır zamanlama modlarını belirtmek yerine, zamanlama değişkenleri --max-parallelism, --min-rtt-timeout ve --scan-delay gibi seçeneklerle tam olarak özelleştirilebilir. Bölüm 6, Nmap Performansını Optimize Etme, bunları derinlemesine açıklamaktadır.

A practical example: bypassing default Snort 2.2.0 rules (Pratik bir örnek: varsayılan Snort 2.2.0 kurallarını atlama)

Kullanışlı açık kaynak Snort IDS'i incelemek, radarın altına gizlice girme konusunda bir ders sağlar. Snort birkaç nesil port tarama dedektörüne sahiptir. Flow-Portscan modülü oldukça zorludur. Bunu atlatan bir tarama muhtemelen diğer birçok IDS tarafından da tespit edilemeyecektir.

Flow-portscan, bağlantı noktası tarayıcılarını tespit etmek için birlikte çalışabilen (veya ayrı ayrı etkinleştirilebilen) iki algılama sisteminde oluşur. Sistem ve düzinelere yapılandırma değişkeni Snort dağıtımındaki docs/README.flow-portscan dosyasında belgelenmiştir, ancak ben hızlı bir özet sunacağım.

Flow-portscan'deki daha basit algılama yöntemi sabit zaman ölçüği olarak bilinir. Bu basitçe tarayıcı-sabit-pencere saniyeleri içinde tarayıcı-sabit-eşik prob paketlerini izler. Snort.conf dosyasında ayarlanan bu iki değişkenin her biri varsayılan olarak 15'tir. Sayacın tek bir makineden korunan ağdaki herhangi bir ana bilgisayara gönderilen tüm probları içerdigini unutmayın. Dolayısıyla, korunan 15 makinenin her birinde tek bir bağlantı noktasını hızlı bir şekilde taramak, tek bir makinede 15 bağlantı noktasını taramak kadar kesin bir uyarı oluşturacaktır.

Eğer tek tespit yöntemi bu olsaydı, çözüm oldukça kolay olurdu. Nmap'in prob gönderme arasında 1.075 saniye beklemesini sağlamak için --scan-delay 1075ms seçeneğini geçin. Sezgisel seçim, 15 saniyede 15 paketten

kaçınmak için paketler arasında bir saniye beklemek olabilir, ancak bu yeterli değildir. İlk paketin gönderilmesi ile on beşinci paketin gönderilmesi arasında sadece 14 bekleme süresi vardır, bu nedenle bekleme süresi en az 15/14 veya 1,07143 saniye olmalıdır. Scan-delay 1000ms seçeneğini seçen zavallı bir aptal, alarmı tetiklemeye devam ederken taramayı önemli ölçüde yavaşlatabaktır. Ağ üzerinde birden fazla ana bilgisayar taranıyorsa, alarmın tetiklenmesini önlemek için bunların ayrı ayrı taraması gereklidir. max-hostgroup 1 seçeneği her seferinde yalnızca bir ana bilgisayarın taramasını sağlar, ancak bir ana bilgisayara gönderilen son prob ile diğerine gönderilen ilk prob arasındaki --scan-delay değerini uygulamayacağı için tamamen güvenli değildir.

Ana bilgisayar başına en az 15 bağlantı noktası tarandığı sürece, --scan-delay değerini en az 1155 ms yaparak telafi edebilir veya bir kabuk betiğinden tek hedefli Nmap örneklerini başlatarak aralarında 1075 ms bekleyebilirsiniz. Örnek 10.19, bir ağ üzerindeki birkaç makinenin bu şekilde gizlice taramasını göstermektedir. Çoklu Nmap örnekleri Bash kabuk sözdizimi kullanılarak işlenir. Burada IP'ler manuel olarak belirtilir. Eğer çok sayıda hedef isteniyorsa, bunlar -sL (list scan) seçeneği ile bir dosyada numaralandırılabilir, ardından Nmap normal bir kabuk döngüsü kullanarak her birine karşı başlatılabilir. Bu taramaların port başına 1.075 saniyeden fazla süremesinin nedeni,filtrelenen portların ağ tıkanıklığı nedeniyle düşmediğinden emin olmak için yeniden iletişimlerin gerekli olmasıdır.

Örnek 10.19. Varsayılan Snort 2.2.0 Flow-portscan sabit zamanlı tarama algılama yöntemini atlamak için yavaş tarama

```
felix~# for target in 205.217.153.53 205.217.153.54 205.217.153.62; \
do nmap --scan-delay 1075ms -p21,22,23,25,53 $target; \
usleep 1075000; \
done

Starting Nmap ( https://nmap.org )
Nmap scan report for insecure.org (205.217.153.53)
PORT      STATE    SERVICE
21/tcp    filtered  ftp
22/tcp    open     ssh
23/tcp    filtered  telnet
25/tcp    open     smtp
53/tcp    open     domain

Nmap done: 1 IP address (1 host up) scanned in 10.75 seconds

Starting Nmap ( https://nmap.org )
Nmap scan report for lists.insecure.org (205.217.153.54)
PORT      STATE    SERVICE
21/tcp    filtered  ftp
22/tcp    open     ssh
23/tcp    filtered  telnet
25/tcp    open     smtp
53/tcp    open     domain

Nmap done: 1 IP address (1 host up) scanned in 10.78 seconds

Starting Nmap ( https://nmap.org )
Nmap scan report for scanme.nmap.org (205.217.153.62)
PORT      STATE    SERVICE
21/tcp    filtered  ftp
22/tcp    open     ssh
23/tcp    filtered  telnet
25/tcp    open     smtp
53/tcp    open     domain

Nmap done: 1 IP address (1 host up) scanned in 10.80 seconds
```

Ne yazık ki port tarama meraklıları için Snort'u yenmek o kadar kolay değildir. Kayan zaman ölçüği olarak bilinen başka bir algılama yöntemine sahiptir. Bu yöntem, bir ana bilgisayardan yeni bir prob algılandığında pencereyi artırması dışında, az önce tartışılan sabit pencere yöntemine benzer. Pencere sırasında tarayıcı kayma eşiği problemleri algılanırsa bir alarm verilir. Pencere scanner-sliding-window saniyesinde başlar ve

algılanan her prob için pencerede o ana kadar geçen süre ile scanner-sliding-scale-factor çarpımı kadar artar. Bu üç değişken snort.conf dosyasında varsayılan olarak 40 prob, 20 saniye ve 0,5 faktör olarak ayarlanmıştır.

Kayan ölçek, yeni paketler geldikçe sürekli olarak büyümesi bakımından oldukça sinsidir. En basit (yavaş olsa da) çözüm, her 20,1 saniyede bir sonda göndermek olacaktır. Bu, hem varsayılan sabit hem de kayan ölçeklerden kaçınacaktır. Bu tıpkı Örnek 10.19'daki gibi yapılabilir, ancak daha yüksek bir değer kullanılabilir. Çok hızlı bir şekilde 14 paket göndererek, pencerenin dolması için 20 saniye bekleyerek ve ardından 14 sonda daha göndererek bunu büyük ölçüde hızlandırırsınız. Bunu Nmap'i kontrol eden bir kabuk betiği ile yapabilirsiniz, ancak bu özel iş için kendi basit SYN tarama programınızı yazmanız tercih edilebilir.

Scatter probes across networks rather than scanning hosts consecutively (Ana bilgisayarları art arda taramak yerine problemleri ağlara dağıtan)

Önceki bölümde tartışıldığı gibi, IDS'ler genellikle yalnızca bir şüpheli etkinlik eşigine ulaşıldıkten sonra alarm verecek şekilde programlanır. Bu eşik genellikle globaldir, sadece tek bir ana bilgisayar yerine IDS tarafından korunan tüm ağa uygulanır. Bazen belirli bir kaynak adresten ardışık ana bilgisayarlara giden trafiği özellikle izlerler. Eğer bir ana bilgisayar 10.0.0.1 numaralı ana bilgisayarın 139 numaralı portuna bir SYN paketi gönderirse, bu kendi başına çok şüpheli değildir. Ancak bu probu 10.0.0.2, .3, .4 ve .5'e benzer paketler izlerse, bir port taraması açıkça belirtilir.

Bu alarmları tetiklemekten kaçınmanın bir yolu, problemleri art arda taramak yerine çok sayıda ana bilgisayar arasında dağıtmaktır. Bazen aynı ağ üzerindeki çok sayıda ana bilgisayarı taramaktan kaçınabilirsiniz. Yalnızca bir araştırma anketi yürütüyorsanız, büyük bir ağı taramak yerine -iR ile problemleri tüm Internet'e dağıtmayı düşünün. Sonuçların her halükarda daha temsili olması muhtemeldir.

Çoğu durumda, belirli bir ağı taramak istersiniz ve Internet başında örnekleme yeterli değildir. Ardışık ana bilgisayar prob alarmlarından kaçınmak kolaydır. Nmap, hedef ağları 16384 IP'lik bloklara ayıran ve ardından her bloktaki ana bilgisayarı rastgele hale getiren --randomize-hosts seçeneğini sunar. Eğer B sınıfı ya da daha büyük bir ağı tariyorsanız, daha büyük blokları rastgele hale getirerek daha iyi (daha gizli) sonuçlar elde edebilirsiniz. Bunu nmap.h dosyasında PING_GROUP_SZ değerini artırarak ve ardından yeniden derleyerek elde edebilirsiniz. Bir --randomize-hosts taramasında kullanılan blok boyutu PING_GROUP_SZ değerinin dört katıdır. Daha yüksek PING_GROUP_SZ değerlerinin daha fazla ana bilgisayar belleği tükettiğini unutmayın. Alternatif bir çözüm, hedef IP listesini bir liste taraması (-sL -n -oN <dosya adı>) ile oluşturmak, bir Perl betiği ile rastgele hale getirmek ve ardından tüm listeyi -iL ile Nmap'e sağlamaktır. Eğer 10.0.0.0/8 gibi büyük bir ağı tariyorsanız ve 16 milyon IP adresinin tamamının rastgele olmasını istiyorsanız muhtemelen bu yaklaşımı kullanmanız gerekecektir.

Fragment packets (Paketleri parçalama)

IP parçaları saldırısı tespit sistemleri için büyük bir sorun teşkil edebilir, çünkü özellikle üst üste binen parçalar ve parçalanma montajı zaman aşımı gibi tuhaflıkların ele alınması belirsizdir ve platformlar arasında önemli ölçüde farklılık gösterir. Bu nedenle, IDS genellikle uzak sistemin bir paketi nasıl yorumlayacağını tahmin etmek zorundadır. Parça birleştirme de yoğun kaynak gerektirebilir. Bu nedenlerden dolayı, birçok saldırısı tespit sistemi hala parçalanmayı çok iyi desteklememektedir. Bir port taramasının küçük (8 veri baytı veya daha az) IP parçaları kullanmasını belirtmek için -f belirtin. Daha önemli ayrıntılar için "Parçalama" adlı bölüme bakın.

Evade specific rules (Belirli kurallardan kaçınmak)

Çoğu IDS satıcısı kaç uyarıları destekledikleri konusunda övünür, ancak çoğu (çoğu değilse de) atlamat kolaydır. Nmap kullanıcıları arasında en popüler IDS açık kaynaklı Snort'tur. Örnek 10.20, Snort 2.0.0'da Nmap'e referans veren tüm varsayılan kuralları göstermektedir.

Örnek 10.20. Nmap'i referans alan varsayılan Snort kuralları

```

felix~/src/snort-2.0.0/rules> grep -i nmap *
icmp.rules:alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING NMAP";
  dsize:0;itype: 8;reference:arachnids,162;
  classtype:attempted-recon;sid:469;rev:1;)
scan.rules:alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap XMAS";
  flags:FPU;reference:arachnids,30;classtype:attempted-recon; sid:1228;rev:1;)
scan.rules:alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap TCP";
  flags:A;ack:0;reference:arachnids,28;classtype:attempted-recon;sid:628;rev:1;)
scan.rules:alert tcp $EXTERNAL_NET any ->
  $HOME_NET any (msg:"SCAN nmap fingerprint attempt";
  flags:SFPU;reference:arachnids,05;classtype:attempted-recon; sid:629; rev:1;)
web-attacks.rules:alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
  (msg:"WEB-ATTACKS nmap command attempt";
  flow:to_server,established;content:"nmap%20";
  nocase;sid:1361;classtype:web-application-attack; rev:4;)

```

Şimdi bu kurallara bir saldırganın gözünden bakalım. İlk kural herhangi bir yükü olmayan bir ICMP ping paketi arar (dsize:0). Sıfır olmayan bir --data-length seçeneği belirtmek bu kuralı geçersiz kılacaktır. Ya da kullanıcı TCP SYN ping gibi tamamen farklı bir ping tarama türü belirtebilir.

Bir sonraki kural FIN, PSH ve URG bayrakları ayarlanmış TCP paketlerini arar (flags:FPU) ve bir Nmap Xmas tarama uyarısı verir. Xmas tarama seçeneklerine --scanflags FINPSH seçeneğinin eklenmesi URG bayrağını kaldıracaktır. Tarama bekleniği gibi çalışmaya devam edecek, ancak kural tetiklenmeyecektir.

Listedeki üçüncü kural ACK biti ayarlanmış ancak onay numarası sıfır olan TCP paketlerini arar (flags:A;ack:0). Nmap'in eski sürümlerinde bu davranış vardı, ancak 1999'da Snort kuralına yanıt olarak düzeltildi.

Dört numaralı kural SYN, FIN, PSH ve URG bayrakları ayarlanmış TCP paketlerini arar (flags:SFPU). Daha sonra bir Nmap OS parmak izi girişimi bildirir. Bir saldırgan -O bayrağını atlayarak bunu işaretlemekten kaçınabilir. Eğer gerçekten işletim sistemi tespiti yapmak istiyorsa, bu tek test osscan2.cc dosyasında yorumlanmaya bilir. İşletim sistemi tespiti hala oldukça doğru olacaktır, ancak IDS uyarısı işaretlenmeyecektir.

Son kural, web sunucularına "nmap" dizesini gönderen kişileri arar. Web sunucusu üzerinden komut çalışma girişimlerini arıyorlar. Bir saldırgan Nmap'i yeniden adlandırarak, boşluk yerine sekme karakteri kullanarak veya varsa SSL şifrelemesiyle bağlanarak bunu aşabilir.

Elbette adında Nmap olmayan ancak yine de izinsiz port taramaları tarafından işaretlenebilecek başka ilgili kurallar da vardır. Gelişmiş saldırganlar, ilgilendikleri IDS'leri kendi ağlarına kurar, ardından alarmları tetiklemediklerinden emin olmak için taramaları önceden değiştirir ve test ederler.

Snort bu örnek için seçilmiştir çünkü kural veritabanı herkese açıktır ve açık kaynaklı bir ağ güvenlik aracıdır. Ticari IDS'ler de benzer sorunlardan muzdariptir.

Avoid easily detected Nmap features (Kolayca tespit edilen Nmap özelliklerinden kaçının)

Nmap'in bazı özellikleri diğerlerine göre daha dikkat çekicidir. Özellikle, sürüm tespiti birçok farklı hizmete bağlanır, bu da genellikle bu makinelerde günlükler bırakır ve saldırısı tespit sistemlerinde alarmları tetikler. İşletim sistemi tespitinin saldırısı tespit sistemleri tarafından tespit edilmesi de kolaydır, çünkü testlerden birkaçı oldukça sıra dışı paketler ve paket dizileri kullanır. Örnek 10.20, "Nmap'i referans alan varsayılan Snort kuralları" bölümünde gösterilen Snort kuralları tipik bir Nmap OS algılama imzasını göstermektedir.

Gizli kalmak isteyen pen-test uzmanları için bir çözüm, bu göze çarpan problemleri tamamen atlamaktır. Hizmet ve işletim sistemi tespiti değerlidir, ancak başarılı bir saldırısı için gerekli değildir. Ayrıca, tüm hedef ağı bunlarla araştırmak yerine, ilginç görünen makinelere veya bağlantı noktalarına karşı vaka bazında da kullanılabilirler.

Misleading Intrusion Detection Systems (Yanlıltıcı Saldırı Tespit Sistemleri)

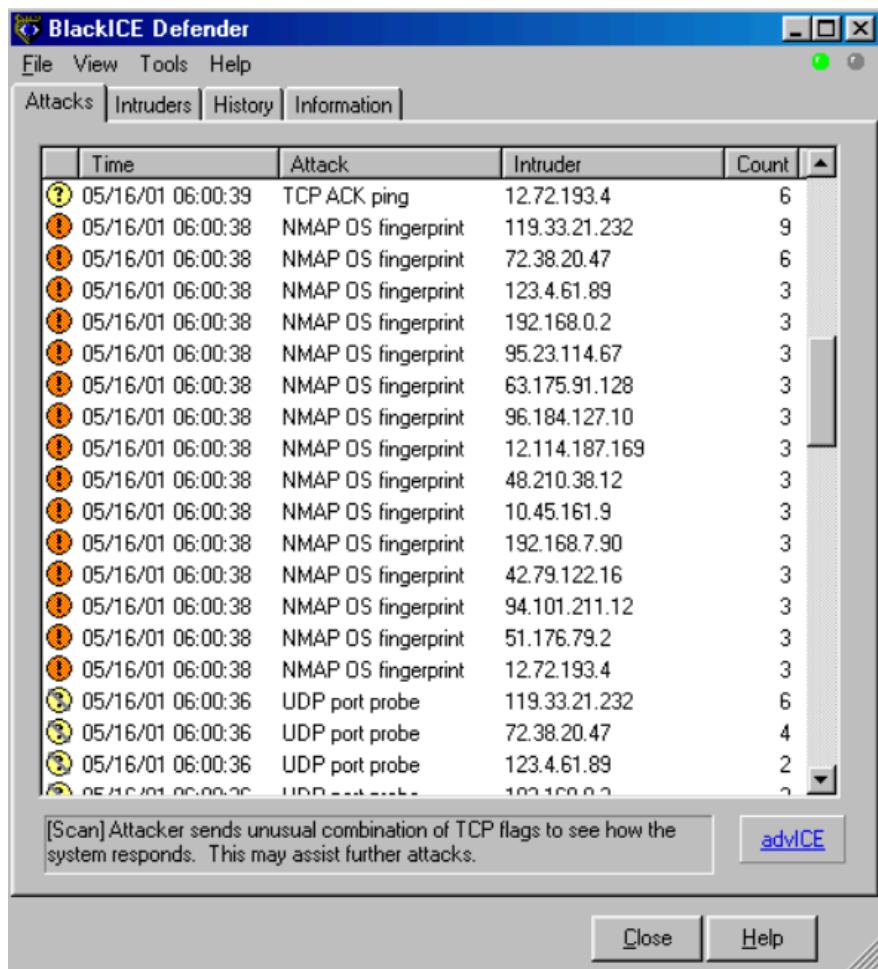
Bir önceki bölümde saldırısı tespit sistemlerinin dikkatli gözlerinden kaçınmak için kurnazlık kullanmaktan bahsedilmişti. Alternatif bir yaklaşım ise paket sahteciliği ile IDS'yi aktif olarak yanıltmak ya da kafasını

karışmaktadır. Nmap bunu gerçekleştirmek için çok sayıda seçenek sunar.

Decoys ()

Sokak suçluları, bir suç işledikten sonra yetkililerden kaçmanın etkili yollarından birinin yakındaki kalabalığın arasına karışmak olduğunu bilirler. Polis, kapkaççıyı yoldan geçen masum insanlardan ayırt edemeyebilir. Ağ alanında, Nmap dünyanın dört bir yanındaki düzinelerce ana bilgisayardan geliyormuş gibi görünen bir tarama oluşturabilir. Hedef, hangi ana bilgisayarın saldırganları temsil ettiğini ve hangilerinin masum yemler olduğunu belirlemekte zorlanacaktır. Bu durum yönlendirici yolu izleme, yanıt bırakma ve diğer aktif mekanizmalarla aşılabile de, genellikle tarama kaynağını gizlemek için etkili bir tekniktir. Şekil 10.2'de tuzaklarla dolu bir BlackICE rapor ekranı gösterilmektedir. Yönetici listedeki her İSS için sağlayıcılara şikayette bulunamaz. Bu uzun zaman alacaktır ve ana bilgisayarların biri hariç hepsi masumdu.

Şekil 10.2. Düzinelerce tuzak tarafından maskelenmiş bir saldırın



Dikkat ⇒ Birçok perakende (çevirmeli, kablolu modem, DSL, vb.) İSS sahte paketlerin çoğunu filtreler, ancak sizinkiyle aynı ağ aralığından gelen sahte paketler geçebilir. Önce Internet üzerinden kontrol ettiğiniz bir makineye karşı bazı testler yapın ya da "IP ID Hileleri" bölümünde anlatılanlara benzer IP ID hileleri kullanarak bunu 3. taraf sunuculara karşı bile test edebilirsiniz.

Tuzaklar -D seçeneği ile eklenir. Bağımsız değişken, virgülle ayrılmış bir ana bilgisayar listesidir. ME dizesi, gerçek kaynak ana bilgisayarın tarama sıralamasında nerede görünmesi gerektiğini temsil etmek için tuzaklardan biri olarak kullanılabilir. Aksi takdirde rastgele bir konum olacaktır. ME'nin listede 6. pozisyonuna veya

daha ileriye dahil edilmesi, bazı yaygın port tarama dedektörlerinin etkinliği rapor etmesini engeller. Örneğin, Solar Designer'ın mükemmel Scanlogd'u, günlüklerinin tuzaklarla dolmasını önlemek için yalnızca ilk beş tarama kaynağını rapor eder.

Rastgele, rezerve edilmemiş bir IP adresi istemek için RND'yi veya <sayı> rastgele adresler oluşturmak için RND:<sayı>'yı da kullanabilirsiniz.

Yem olarak kullanılan ana bilgisayarların çalışır durumda olması gerektiğini unutmayın. AĞDA sadece bir ana bilgisayar çalışıyorsa hangi ana bilgisayarın tarama yaptığı belirlemek oldukça kolay olacaktır. Çok fazla aşağı tuzak kullanmak, SYN seli olarak bilinen bir durum nedeniyle hedef bağlantı noktalarının geçici olarak yanıt vermemesine de neden olabilir. Tuzak ağlarının isim sunucusu günlüklerinde görünmekten kaçınmak için isimler yerine IP adreslerinin kullanılması tavsiye edilir. Hedeflerin kendileri de ideal olarak IP adresleriyle ifade edilmelidir.

Tuzaklar hem ilk ping taramasında (ICMP, SYN, ACK veya her neyse kullanılarak) hem de gerçek port tarama aşamasında kullanılır. Tuzaklar ayrıca uzak işletim sistemi tespiti sırasında da kullanılır. DNS sorguları veya hizmet/sürüm tespiti için kullanılmazlar, bu nedenle -sV veya -A gibi seçenekleri kullanırsanız kendinizi ele verirsınız. Çok fazla tuzak kullanmak bir taramayı önemli ölçüde yavaşlatabilir ve hatta bazen daha az doğrumasına neden olabilir.

Port scan spoofing (Port tarama sahtekarlığı)

Büyük bir tuzak grubu, bir port taramasının gerçek kaynağını gizlemede oldukça etkili olsa da, IDS uyarıları birisinin tuzak kullandığını açıkça ortaya koyacaktır. Daha incelikli ancak sınırlı bir yaklaşım ise tek bir adresten sahte port taraması yapmaktır. S ve ardından bir kaynak IP belirtin ve Nmap istenen port taramasını verilen kaynaktan başlatacaktır. Hedef sahte IP'ye yanıt vereceğinden ve Nmap bu yanıtları görmeyeceğinden hiçbir yararlı Nmap sonucu elde edilemeyecektir. Hedefteki IDS alarmları tarama için sahte kaynağı suçlayacaktır. Nmap'in sahte paketleri göndereceği uygun arayüz adını (eth0, ppp0, vb.) seçmek için -e <arayüzadı> belirtmeniz gerekebilir. Bu, masum tarafları tuzağa düşürmek, IDS'nin doğruluğu konusunda yöneticinin zihninde şüphe uyandırmak ve "Reaktif Sistemlere Karşı DoS Saldırıları" adlı bölümde ele alınacak olan hizmet reddi saldırıları için yararlı olabilir.

Idle scan (Boşta tarama)

Boşta tarama, önceki bölümde tartışıldığı gibi kaynak IP adresinin yanıltılmasına izin verirken, yine de doğru TCP bağlantı noktası tarama sonuçları elde etmeyi sağlayan akıllıca bir tekniktir. Bu, birçok sistem tarafından uygulandığı gibi IP tanımlama alanının özelliklerini kötüye kullanarak yapılır. "TCP Idle Scan (-sI)" adlı bölümde çok daha ayrıntılı olarak açıklanmıştır.

DNS proxying (DNS proxyleme)

En dikkatle hazırlanmış planlar bile gözden kaçan küçük bir ayrıntıyla bozulabilir. Eğer plan ultra-gizli port taraması içeriyorsa, bu küçük ayrıntı DNS olabilir. "DNS Çözünürlüğü" adlı bölümde tartışıldığı gibi, Nmap her yanıt veren ana bilgisayara karşı varsayılan olarak ters DNS çözünürlüğünü gerçekleştirir. Eğer hedef ağ yöneticileri her şeyi günlüğe kaydeden paranoid tiplerse ya da son derece hassas bir IDS'leri varsa, bu DNS arama problemleri tespit edilebilir. Liste taraması (-sL) gibi müdahaleci olmayan bir şey bile bu şekilde tespit edilebilir. Problemler Nmap çalıştırılan makine için yapılandırılmış DNS sunucusundan gelecektir. Bu genellikle İSS'niz veya kuruluşunuz tarafından tutulan ayrı bir makinedir, ancak bazen kendi sisteminiz de olabilir.

Bu riski ortadan kaldırmanın en etkili yolu, tüm ters DNS çözümlemesini devre dışı bırakmak için -n belirtmektir. Bu yaklaşımıla ilgili sorun, DNS tarafından sağlanan değerli bilgileri kaybetmenizdir. Neyse ki Nmap, kaynağı gizlerken bu bilgileri toplamanın bir yolunu sunuyor. İnternet üzerindeki DNS sunucularının önemli bir yüzdesi herhangi birinden gelen özyinelemeli sorgulara açıktır. Nmap'in --dns-servers seçeneğine bu ad sunucularından bir veya daha fazlasını belirtin ve tüm rDNS sorguları bunlar aracılığıyla proxy'llenecektir. Örnek 10.21 bu teknigi, bazı SecurityFocus IP'lerinin liste taramasını yaparken, herhangi bir izi örtmek için genel

özyinelemeli DNS sunucuları 4.2.2.1 ve 4.2.2.2'yi kullanarak göstermektedir. İleri DNS'nin hala ana bilgisayarınızın yapılandırılmış DNS sunucusunu kullandığını unutmayın, bu nedenle bu küçük potansiyel bilgi sızıntısını bile önlemek için alan adları yerine hedef IP adreslerini belirtin. Bu nedenle, Örnek 10.21 ilk olarak Nmap komut satırında ana bilgisayar adını belirtmek yerine www.securityfocus.com adresini aramak için kullanılan Linux host komutunu göstermektedir. Tek bir DNS sunucusundan gelen istek sayısına dayalı IDS eşiklerinden kaçınmak için --dns-servers komutuna virgülle ayrılmış düzinelere DNS sunucusu belirtebilirsiniz ve Nmap isteklerini bunlar arasında yuvarlayacaktır.

Örnek 10.21. SecurityFocus'un Gizli Liste Taraması için DNS Proxy'lerini (Özyinelemeli DNS) Kullanma

```
# host www.securityfocus.com 4.2.2.1
Using domain server:
Address: 4.2.2.1#53

www.securityfocus.com has address 205.206.231.12
www.securityfocus.com has address 205.206.231.15
www.securityfocus.com has address 205.206.231.13

# nmap --dns-servers 4.2.2.1,4.2.2.2 -sL 205.206.231.12/28

Starting Nmap ( https://nmap.org )
Host 205.206.231.0 not scanned
Host mail2.securityfocus.com (205.206.231.1) not scanned
Host ns1.securityfocus.com (205.206.231.2) not scanned
Host sgs1.securityfocus.com (205.206.231.3) not scanned
Host sgs2.securityfocus.com (205.206.231.4) not scanned
Host 205.206.231.5 not scanned
Host adserver.securityfocus.com (205.206.231.6) not scanned
Host datafeeds.securityfocus.com (205.206.231.7) not scanned
Host sfcn.securityfocus.com (205.206.231.8) not scanned
Host mail.securityfocus.com (205.206.231.9) not scanned
Host www.securityfocus.com (205.206.231.10) not scanned
Host www1.securityfocus.com (205.206.231.11) not scanned
Host www2.securityfocus.com (205.206.231.12) not scanned
Host www3.securityfocus.com (205.206.231.13) not scanned
Host media.securityfocus.com (205.206.231.14) not scanned
Host www5.securityfocus.com (205.206.231.15) not scanned
Nmap done: 16 IP addresses (0 hosts up) scanned in 0.27 seconds
```

DoS Attacks Against Reactive Systems (Reaktif Sistemlere Karşı DoS Saldırıları)

Birçok saticı, saldırı önleme sistemleri olarak adlandırdıkları sistemleri öne çıkarıyor. Bunlar temel olarak trafiği aktif olarak engelleylebilen ve kötü niyetli olduğu düşünülen bağlantıları sıfırlayabilen IDS'lerdir. Bunlar genellikle ağ üzerinde satır içi veya ağ etkinliği üzerinde daha fazla kontrol için ana bilgisayar tabanlıdır. Diğer (satır içi olmayan) sistemler gelişigüzel dinleme yapar ve TCP RST paketlerini taklit ederek şüpheli bağlantılarla başa çıkmaya çalışır. Çok çeşitli şüpheli etkinlikleri engellemeye çalışan geleneksel IPS saticılarına ek olarak, Port Sentry gibi birçok popüler küçük program özellikle port tarayıcılarını engellemek için tasarlanmıştır.

Port tarayıcılarını engellemek ilk başta iyi bir fikir gibi görünse de, bu yaklaşımla ilgili birçok sorun vardır. En bariz olanı, önceki bölümlerde gösterildiği gibi, port taramalarının taklit edilmesinin genellikle oldukça kolay olmasıdır. Saldırganların bu tür bir tarama engelleme yazılımının mevcut olduğunu anlamaları da genellikle kolaydır, çünkü bir port taraması yaptıktan sonra sözde açık portlara bağlanamayacaklardır. Başka bir sistemden tekrar deneyecekler ve başarılı bir şekilde bağlanarak orijinal IP'nin engellendiğini doğrulayacaklardır. Saldırganlar daha sonra, hedef ana bilgisayarın saldırının istediği herhangi bir sistemi engellemesine neden olmak için daha önce tartışılan ana bilgisayar sahtekarlığı tekniklerini (-S seçeneği) kullanabilir. Bunlar arasında önemli DNS sunucuları, büyük web siteleri, yazılım güncelleme arşivleri, posta

sunucuları ve benzerleri yer alabilir. Reaktif engellemeyi devre dışı bırakmak için meşru yöneticiyi yeterince rahatsız etmek muhtemelen uzun sürmeyecektir. Bu tür ürünlerin çoğu belirli önemli ana bilgisayarların engellenmesini önlemek için bir beyaz liste seçeneği sunsa da, bunların hepsini listelemek olağanüstü zordur. Saldırganlar genellikle engellemek için yaygın olarak kullanılan yeni bir ana bilgisayar bulabilir ve yönetici sorunu belirleyip beyaz listeyi buna göre ayarlayana kadar kullanıcıları rahatsız edebilir.

Exploiting Intrusion Detection Systems (İzinsiz Giriş Tespit Sistemlerinden Yararlanma)

Saldırı tespit sistemlerini alt etmenin en cüretkar yolu onları hacklemektir. Birçok ticari ve açık kaynak tedarikçisinin ürün istismar edilebilirliği konusunda acıncak güvenlik kayıtları vardır. Internet Security System'in amiral gemisi RealSecure ve BlackICE IDS'leri, Witty solucanının on binden fazla kurulumu tehlikeye atmasına izin veren bir güvenlik açığına sahipti, ardından dosya sistemlerini bozarak IDS'leri devre dışı bıraktı. Cisco, Checkpoint, Netgear ve Symantec gibi diğer IDS ve güvenlik duvarı satıcıları da uzaktan istismar edilebilen ciddi güvenlik açıklarına maruz kalmıştır. Snort, Wireshark, tcpdump, FakeBO ve diğerlerinde bulunan istismar edilebilir hatalarla açık kaynak kodlu koklayıcıların durumu da pek iyi değildir. Protokollerin güvenli ve verimli bir şekilde ayrıştırılması son derece zordur ve uygulamaların çoğunun yüzlerce protokolü ayrıştırması gereklidir. IDS'yi çökerten (genellikle tek bir paketle) hizmet reddi saldıruları, bu ayrıcalık yükseltme güvenlik açıklarından bile daha yaygındır. Çökmüş bir IDS herhangi bir Nmap taraması tespit etmeyecektir.

Ignoring Intrusion Detection Systems (İzinsiz Giriş Tespit Sistemlerini Görmezden Gelmek)

İleri düzey saldırganlar genellikle bu bölümde açıklanan IDS yıkma tekniklerini kullanırlarken, çok daha yaygın olan acemi saldırganlar (script kiddies) IDS'lerle nadiren ilgilenirler. Birçok şirket IDS bile kullanmaz ve kullananlar da genellikle yanlış yapılandırır ya da uyarılara çok az dikkat eder. İnternete dönük bir IDS, script kiddie ve solucanlardan o kadar çok saldırı görecektir ki, savunmasız bir hizmeti bulmak için yapılan birkaç Nmap taramasının herhangi bir bayrak kaldırması pek olası değildir.

Böyle bir saldırgan ağı ele geçirse, izlenen bir IDS tarafından tespit edilse ve ardından sistemlerden atılsa bile, bu küçük bir kayıptır. Bilgisayar korsanlığı onlar için genellikle bir sayı oyunudur, bu nedenle binlerce ağ içinde ele geçirilen bir ağı kaybetmek öbensizdir. Böylesine iyi korunan bir ağı, muhtemelen kullanımlarını (hizmet reddi saldıruları, toplu tarama veya spam gönderimi gibi) hızlı bir şekilde fark eder ve onları yine de kapatır. Bilgisayar korsanları, suç faaliyetleri için uzun ömürlü düğüm noktaları sağlayacak, ihmalkar bir şekilde yönetilen ve kötü izlenen ağları tehlikeye atmak ister.

Takip edilmek ve kovuşturtmaya uğramak, IDS'i yok sayanların nadiren ilgilendiği bir konudur. Saldırılarını genellikle gerçek konumlarından birkaç dünya atlaması uzakta olan diğer tehlikeye atılmış ağlardan başlatırlar. Ya da bazı internet kafeler, okul bilgisayar laboratuarları, kütüphaneler veya yaygın açık kablosuz erişim noktaları tarafından sağlanan anonim bağlantıyı kullanabilirler. Kullan at çevirmeli hesaplar da yaygın olarak kullanılmaktadır. Hesaptan atılsalar bile, başka (ya da aynı) bir sağlayıcıya yeniden kaydolmak sadece dakikalar alır. Saldırganların çoğu Romanya, Çin, Güney Kore ve kovuşturmanın pek olası olmadığı diğer ülkelerden geliyor.

Internet solucanları, IDS'den kaçınma konusunda nadiren sıkıntı yaşayan bir başka saldırı sınıfıdır. Milyonlarca IP adresinin utanmazca taranması, gizliliği vurgulayan dikkatli ve hedefli bir yaklaşma kıyasla saat başına daha fazla tehlikeye yol açtığı için hem solucanlar hem de script kiddie'ler tarafından tercih edilmektedir.

Çoğu saldırı gizlilik için çaba sarf etmese de, çoğu saldırısı tespit sisteminin bu kadar kolay alt edilebilmesi büyük bir endişe kaynağıdır. Yetenekli saldırganlar küçük bir azınlıktır, ancak genellikle en büyük tehdittir. IDS'lerden gelen çok sayıda uyarı sizin uwagęte sürüklemeşin. Her şeyi tespit edemezler ve genellikle en önemli olanı gözden kaçırırlar.

Yetenekli bilgisayar korsanları bile bazen ilk keşif için IDS endişelerini görmezden gelirler. Diğer tüm saldırganların arasına karışmayı ve İnternet'teki trafiği incelemeyi umarak izlenmemeyen bir IP adresinden tarama yaparlar. Sonuçları analiz ettikten sonra, diğer sistemlerden daha dikkatli, gizli saldırular başlatabilirler.

Detecting Packet Forgery by Firewall and Intrusion Detection Systems (Güvenlik Duvarı ve Saldırı Tespit Sistemleri ile Paket Sahteciliğinin Tespiti)

Önceki bölümlerde, bazı güvenlik duvari ve saldırısı tespit sistemlerinin, paketleri cihazın arkasındaki korunan sistemlerden birinden geliyormuş gibi gösterecek şekilde yapılandırılabileceğinden bahsedilmişti. TCP RST paketleri sık rastlanan bir örnektir. Yük dengeleyiciler, SSL hızlandırıcıları, ağ adresi çeviri cihazları ve bazı honeyneler de kafa karıştırıcı veya tutarsız sonuçlara yol açabilir. Nmap'in yanıtları nasıl yorumladığını anlamak, karmaşık uzak ağ topolojilerini bir araya getirmede büyük ölçüde yardımcı olur. Nmap olağanüstü veya beklenmedik sonuçlar bildirdiğinde, Nmap'in sonuçlarını dayandırdığı ham paketleri görmek için --packet-trace seçeneğini ekleyebilirsiniz. Şaşırtıcı durumlarda, daha da ileri gitmeniz ve özel problemler başlatmanız ve Nping ve Wireshark gibi diğer araçlarla paketleri analiz etmeniz gerekebilir. Amaç genellikle gerçek ağ kurulumunu anlamanıza yardımcı olacak tutarsızlıkları bulmaktır. Aşağıdaki bölümlerde bunu yapmak için birkaç yararlı teknik açıklanmaktadır. Bu testlerin çoğu doğrudan Nmap'i içermese de, beklenmedik Nmap sonuçlarını yorumlamak için yararlı olabilirler.

Look for TTL Consistency (TTL Tutarlılığına Bakın)

Güvenlik duvarları, yük dengeleyiciler, NAT ağ geçitleri ve benzeri cihazlar genellikle korudukları makinelerin bir veya daha fazla atlama mesafesinde bulunur. Bu durumda, paketler ağ cihazına ulaşacak ancak son ana bilgisayara ulaşmayacak şekilde bir TTL ile oluşturulabilir. Böyle bir probdan bir RST alınırsa, cihaz tarafından gönderilmiş olmalıdır.

Gayri resmi bir değerlendirme sırasında, büyük bir dergi yayıcısının ağını Internet üzerinden taradım ("ÇÖZÜM: Belirli Bir Açık TCP Bağlantı Noktası için Büyük Bir Ağ Tarayıñ" başlıklı bölümde hatırlayabilirsiniz). Neredeyse her IP adresinde 113 numaralı bağlantı noktası kapalıydı. Bir güvenlik duvarı tarafından RST sahteciliğinden şüphelenerek biraz daha derine indim. Açık, kapalı ve filtrelenmiş portlar içerdiginden, özellikle bu ana bilgisayara odaklanmaya karar verdim:

```
# nmap -sS -Pn -T4 mx.chi.playboy.com
Starting Nmap ( https://nmap.org )
Nmap scan report for mx.chi.playboy.com (216.163.143.4)
Not shown: 998 filtered ports
PORT      STATE    SERVICE
25/tcp    open     smtp
113/tcp   closed   auth

Nmap done: 1 IP address (1 host up) scanned in 53.20 seconds
```

113 numaralı bağlantı noktası gerçekten kapalı mı, yoksa güvenlik duvari RST paketlerini mi yanıltıyor? Örnek 10.22'de gösterildiği gibi, ücretsiz hping2 yardımcı programının özel traceroute modunu kullanarak 25 ve 113 numaralı bağlantı noktalarına olan mesafeyi (ağ atlamaları cinsinden) saydım. Bunu yapmak için daha hızlı olan Nmap --traceroute seçeneğini kullanabilirdim, ancak o sırada bu seçenek mevcut değildi.

Örnek 10.22. Kapalı ve filtrelenmiş TCP bağlantı noktalarının algılanması

```

# hping2 -t 5 --traceroute -p 25 -S mx.chi.playboy.com
[combined with results from hping2 -i 1 --ttl \* -p 25 -S mx.chi.playboy.com]
5->TTL 0 during transit from 64.159.2.97 (ae0-54.mp2.SanJosel.Level3.net)
6->TTL 0 during transit from 64.159.1.34 (so-3-0-0.mp2.Chicago1.Level3.net)
7->TTL 0 during transit from 200.247.10.170 (pos9-0.core1.Chicago1.level3.net)
8->TTL 0 during transit from 200.244.8.42 (gige6-0.ipcol01.Chicago1.Level3.net)
9->TTL 0 during transit from 166.90.73.205 (ge1-0.brl.ord.playboy.net)
10->TTL 0 during transit from 216.163.228.247 (f0-0.b1.chi.playboy.com)
11->No response
12->TTL 0 during transit from 216.163.143.130 (fw.chi.playboy.com)
13->46 bytes from 216.163.143.4: flags=SA seq=0 ttl=52 id=48957 rtt=75.8 ms

# hping2 -t 5 --traceroute -p 113 -S mx.chi.playboy.com
[ results augmented again ]
5->TTL 0 during transit from 64.159.2.97 (ae0-54.mp2.SanJosel.Level3.net)
6->TTL 0 during transit from 64.159.1.34 (so-3-0-0.mp2.Chicago1.Level3.net)
7->TTL 0 during transit from 200.247.10.170 (pos9-0.core1.Chicago1.level3.net)
8->TTL 0 during transit from 200.244.8.42 (gige6-0.ipcol01.Chicago1.Level3.net)
9->TTL 0 during transit from 166.90.73.205 (ge1-0.brl.ord.playboy.net)
10->TTL 0 during transit from 216.163.228.247 (f0-0.b1.chi.playboy.com)
11->Nothing
12->46 bytes from 216.163.143.4: flags=RA seq=0 ttl=48 id=53414 rtt=75.0 ms

```

Bu özel traceroute, 25 numaralı açık bağlantı noktasına ulaşmanın 13 atlama gerektirdiğini gösteriyor. 12 atlama uzakta Chicago'da fw.chi.playboy.com olarak adlandırılan bir güvenlik duvarı var. Aynı makinedeki farklı bağlantı noktalarının aynı hop mesafesi uzaklığında olması beklenir. Ancak 113 numaralı bağlantı noktası yalnızca 12 atlamanın ardından bir RST ile yanıt veriyor. Bu RST fw.chi.playboy.com tarafından taklit ediliyor. Güvenlik duvarının 113 numaralı bağlantı noktası yanıtlarını taklit ettiği biliindiğinden, bu paketler belirli bir IP adresinde bir ana bilgisayarın mevcut olduğunu göstergesi olarak alınmamalıdır. ICMP eko istekleri (-PE) ve 22 ve 80 numaralı portlara SYN paketleri (-PS22,80) gibi yaygın prob türlerini kullanarak, ancak TCP port 113'ü içeren herhangi bir ping probunu atlayarak ağı tekrar tarayarak mevcut ana bilgisayarları bulduk.

Look for IP ID and Sequence Number Consistency (IP Kimliği ve Sıra Numarası Tutarlılığına Bakın)

Her IP paketi, birleştirme için kullanılan 16 bitlik bir tanımlama alanı içerir. Ayrıca uzak ana bilgisayarlar hakkında şüpçü miktarda bilgi edinmek için de kullanılabilir. Bu, Nmap boşta tarama tekniğini kullanarak port taraması, trafik tahmini, ana bilgisayar takma ad tespiti ve çok daha fazlasını içerir. Ayrıca yük dengeleyiciler gibi birçok ağ cihazının tespit edilmesine de yardımcı olabilir. Bir keresinde beta.search.microsoft.com adresini tararken garip işletim sistemi algılama sonuçları fark ettim. Bu yüzden neler olduğunu öğrenmek için TCP port 80'e karşı hping2 SYN problemini başlattım. Örnek 10.23 sonuçları göstermektedir.

Örnek 10.23. IP Kimliği sıra numarası tutarlığını test etme

```

# hping2 -c 10 -i 1 -p 80 -S beta.search.microsoft.com
HPING beta.search.microsoft.com. (eth0 207.46.197.115): S set, 40 headers
46 bytes from 207.46.197.115: flags=SA seq=0 ttl=56 id=57645 win=16616
46 bytes from 207.46.197.115: flags=SA seq=1 ttl=56 id=57650 win=16616
46 bytes from 207.46.197.115: flags=RA seq=2 ttl=56 id=18574 win=0
46 bytes from 207.46.197.115: flags=RA seq=3 ttl=56 id=18587 win=0
46 bytes from 207.46.197.115: flags=RA seq=4 ttl=56 id=18588 win=0
46 bytes from 207.46.197.115: flags=SA seq=5 ttl=56 id=57741 win=16616
46 bytes from 207.46.197.115: flags=RA seq=6 ttl=56 id=18589 win=0
46 bytes from 207.46.197.115: flags=SA seq=7 ttl=56 id=57742 win=16616
46 bytes from 207.46.197.115: flags=SA seq=8 ttl=56 id=57743 win=16616
46 bytes from 207.46.197.115: flags=SA seq=9 ttl=56 id=57744 win=16616

```

IP ID numaralarının sırasına bakıldığından (kalın harflerle), bu IP adresini bir tür yük dengeleyici aracılığıyla paylaşan gerçekten iki makine olduğu açıktır. Bir 57K aralığında IP ID dizilerine sahipken, diğer 18K kullanıyor. Bu bilgiler göz önüne alındığında, Nmap'in tek bir işletim sistemi tahmininde bulunmakta zorlanması şüpçidir. Çok farklı sistemler üzerinde çalışıyor olabilirler.

Benzer testler TCP zaman damgası seçeneği veya açık portlar tarafından döndürülen ilk sıra numarası gibi diğer sayısal alanlar üzerinde de gerçekleştirilebilir. Bu özel durumda, TCP pencere boyutu ve TCP bayraklarının da ana bilgisayarları ele verdiği görebilirsiniz.

The Bogus TCP Checksum Trick (Sahte TCP Checksum Hilesi)

Bir IDS veya güvenlik duvarının yanıt paketlerinde sahtecilik yapıp yapmadığını belirlemek için bir başka kullanışlı numara da sahte TCP sağlama toplamı içeren problemler göndermektir. Esasen tüm uç ana bilgisayarlar daha fazla işlem yapmadan önce sağlama toplamını kontrol eder ve bu bozuk paketlere yanıt vermez. Öte yandan, güvenlik duvarları genellikle performans nedenleriyle bu kontrolü atlar. Bu davranışın Örnek 10.24'te gösterildiği gibi --badsum seçeneği ile tespit edebiliriz.

Örnek 10.24. Kötü TCP sağlama toplamlarına sahip bir güvenlik duvari bulma

```
# nmap -sS -p 113 -Pn --badsum google.com
Starting Nmap ( https://nmap.org )
Warning: Hostname google.com resolves to 3 IPs. Using 64.233.187.99.
Nmap scan report for jc-in-f99.google.com (64.233.187.99)
PORT      STATE    SERVICE
113/tcp    closed   auth

Nmap done: 1 IP address (1 host up) scanned in 0.44 seconds
```

Örnek 10.24'ten, TCP sağlama toplamlarını doğrulamadan 113 numaralı bağlantı noktasından google.com adresine giden paketleri işleyen bir tür ağ cihazı, belki de bir güvenlik duvarı olduğu sonucunu çıkarabiliriz. Normalde, bir uç ana bilgisayar kötü TCP sağlama toplamlarına sahip paketleri sessizce bırakır ve biz de kapalı bir bağlantı noktası yerine filtrelenmiş bir bağlantı noktası görürüz. --badsum ayrıca IP'nin yanı sıra UDP, ICMP ve IGMP gibi diğer protokoller için de kötü sağlama toplamları kullanacaktır.

Bu teknik, kasıtlı olarak hatalı biçimlendirilmiş sağlama toplamlarına sahip paketler göndermenin diğer nedenleriyle birlikte, Ed3f tarafından Phrack 60, makale 12'de daha ayrıntılı olarak açıklanmıştır. Bu bazen faydalı bir teknik olsa da, dikkate alınması gereken birkaç uyarı vardır:

1. Birçok modern güvenlik duvari artık bu bilginin sizmasını önlemek için TCP sağlama toplamlarını (en azından bir pakete yanıt verilip verilmeyeceğini belirlerken) doğrulamaktadır. Dolayısıyla bu teknik, filtrelenmiş bir --badsum probunun bir uç ana bilgisayar tarafından düşürüldüğünü kanıtlamaktan ziyade, bir --badsum prob yanıtının bir güvenlik duvarı (veya eksik TCP yiğinına sahip başka bir cihaz) tarafından gönderildiğini kanıtlamak için daha kullanışlıdır.
2. --badsum kullanmak, paketlerin tüm platformlarda kötü sağlama toplamlarıyla gönderileceğini garanti etmez. Bazı sistemlerde çekirdek veya ağ kartı sağlama toplamı hesaplamasını yapar ve istenen kötü değerin üzerine doğru değeri ekler. Bunun sizin başına gelmediğinden emin olmanın bir yolu, gönderdiğiniz paketleri koklamak için uzak bir makine kullanmaktır. Örneğin, tcpdump ile koklama yaparken, kötü TCP sağlama toplamlarına sahip paketler [bad tcp cksum aa79 (→ab79)!] gibi gösterilecektir. Başka bir yaklaşım da ana bilgisayarlarınızdan birine karşı normal bir SYN taraması yapmaktır (en az bir açık port ile). Sonra aynı taramayı --badsum ile yapın. Aynı bağlantı noktaları hala açık olarak gösteriliyorsa, --badsum muhtemelen sizin için çalışmıyor demektir. Lütfen sorunu "Hatalar" bölümünde açıkladığı gibi bildirin.

Round Trip Times (Gidiş Dönüş Süreleri)

Bir güvenlik duvarı bir prob yanıtını taklit ettiğinde, bu yanıt genellikle gerçek hedef ana bilgisayardan gelen bir yanıtta biraz daha erken döner. Sonuçta, güvenlik duvarı genellikle en az bir atlama daha yakındır. Ayrıca paketleri hızlı bir şekilde ayırtırmak ve işlemek için optimize edilmiştir ve başka pek bir şey yapmaz. Öte yandan, hedef ana bilgisayar uygulamaları çalıştırırmakla o kadar meşgul olabilir ki bir proba yanıt vermesi birkaç

milisaniye daha uzun sürer. Bu nedenle, gidiş dönüş sürelerinin yakından karşılaştırılması genellikle güvenlik duvarı maskaralıklarını ele verebilir.

Bu teknikle ilgili bir zorluk, güvenlik duvari yanıtı ile gerçek hedef yanıt arasındaki zaman farkının milisaniyenin bir kesri kadar olabilmesidir. Normal gidiş dönüş süresi farklılıklarını bundan daha büyük olabilir, bu nedenle sadece iki prob göndermek (biri hedef ana bilgisayardan geldiği bilinen bir yanıt isteyen ve diğer güvenlik duvarından gelebilecek şüpheli bir yanıt) nadiren yeterli olur. Her bir prob türünden bin tane göndermek RTT farklılığının çoğunu ortadan kaldırır, böylece temel farklılıklar ayırt edilebilir. Bunun o kadar uzun süremesi gerekmez -c 1000 --rate 20 seçenekleriyle bin probu bir dakikadan kısa bir sürede gönderir. Bu sonuçlardan, size verdiği ortalamayı kullanmak yerine medyanı hesaplayın. Bu, çok büyük sürelerin (iki saniye sonra yeniden iletilen kayıp bir yanıt gibi) verileri çarpıtmasını önler. Sonuçların ne kadar tutarlı olduğunu belirlemek için bin probu bir veya iki kez daha yapın. Ardından aynı işlemi şüpheli sonda ile deneyin ve ikisini karşılaştırın. Süreler son anlamlı basamağa kadar tamamen aynıysa, muhtemelen her iki yanıt da aynı ana bilgisayar gönderiyordur. Sürekli olarak bir prob türünün diğerinden daha hızlı yanıt verdiği görüyorsanız, bundan paket sahteciliği sorumlu olabilir.

Bu yöntem mükemmel değildir. Bir zaman tutarsızlığı, güvenlik duvarı dışında herhangi bir sayıda başka faktörden kaynaklanabilir. Paket sahteciliği gibi ağ anormalliklerini tespit etmek bir mahkeme davasını kanıtlamak gibi olduğu hala değerli bir tekniktir. Her küçük kanıt parçası bir sonuca ulaşmaya yardımcı olur. Tutarsızlık, güvenlik duvarı sahteciliğinden daha ilginç keşiflere bile yol açabilir. Belki de hedef üzerindeki belirli portlar, saldıruları daha iyi incelemek için bir bal ağına yönlendiriliyor.

Close Analysis of Packet Headers and Contents (Paket Başlıklarının ve İçeriklerinin Yakın Analizi)

Küçük bir TCP başlığında bile ne kadar çok ögenin farklı olabileceği şartsızdır. Farklı bir işletim sisteminin göstergesi olabilecek düzinelere ince ayrıntı için Bölüm 8, Uzak İşletim Sistemi Algılama'ya bakın. Örneğin, farklı sistemler farklı TCP seçenekleri, RST paket metni, hizmet türü değerleri vb. ile yanıt verir. Bir yük dengeleyicinin arkasında birkaç sistem varsa veya paketler güvenlik duvarı veya izinsiz giriş tespit sistemleri tarafından gönderiliyorsa, paketler nadiren tam olarak eşleşecektir.

Paket başlıklarını incelemek için mükemmel bir araç Wireshark'tır çünkü başlığı tek tek alanlara ayıralabilir ve paketin içili içeriğinin metinsel açıklamalarını sağlayabilir. Paketleri karşılaşmanın püf noktası, bir güvenlik duvarından gelebileceğini düşündüğünüz bir paketi ve hedef ana bilgisayardan veya hedef işletim sisteminden gelen aynı türden başka bir paketi toplamaktır. Toplayabileceğiniz iki paket türü TCP sıfırlama paketleri ve ICMP hata paketleridir. Nping veya --scanflags Nmap seçeneğini kullanarak farklı IP, TCP veya ICMP başlıklarına sahip yanıtları ortaya çıkarmak mümkün olmalıdır.

Unusual Network Uniformity (Olağandışı Ağ Tekdüzeligi)

Yanıt paketleri bir güvenlik duvarı tarafından gönderildiğinde, genellikle tek tek makinelerden oluşan kümelerden beklenenden daha tekdüze olurlar. Önceki TTL kontrol bölümünde tartışılan dergi şirketini tararken, yüzlerce sıralı IP makinesinin 113 numaralı bağlantı noktasına bir RST ile yanıt verdiği gördüm. Gerçek bir makine kümesinde, belirli bir zamanda en az birkaç tanesinin çevrimdışı olmasını beklersiniz. Ayrıca, bu adreslerin çoğundan başka türde bir yanıt almadım. Bu şüpheli sonuç beni TTL testleri yapmaya yöneltti ve bu da fw.chi ana bilgisayarının aslında RST paketlerini taklit ettiğini gösterdi.

Bir güvenlik duvarının kendini ele vermesi için paketleri taklit etmesi bile gerekmez. Bir başka yaygın güvenlik duvari yapılandırması da paketleri belirli portlara düşürmektedir. Birçok İSS, solucanların yayılmasını azaltmak için Windows 135, 139 ve 445 numaralı bağlantı noktalarını filtreler. Çok sayıda bitişik canlı ana bilgisayarda aynıfiltrelenmiş bağlantı noktasının kümesi görünyorsa, olası suçlu bir ağ güvenlik duvarıdır. Bir güvenlik duvarı tarafından hangi bağlantı noktalarının filtrelendiğini belirledikten sonra,filtrelenen bağlantı noktaları için birçok ağ bloğunu tarayarak bu güvenlik duvarı kuralları tarafından kaç ana bilgisayarın korunduğunu genellikle haritalayabilirsiniz. Bu, kazara oluşan deliklerin veya kuruluşun DMZ'sinin (askerden arındırılmış bölge)

keşfedilmesine yol açabilir; bu bölge genellikle kamu hizmetlerini barındırır ve çok daha gevşek güvenlik duvarı kurallarına sahiptir.

next next next