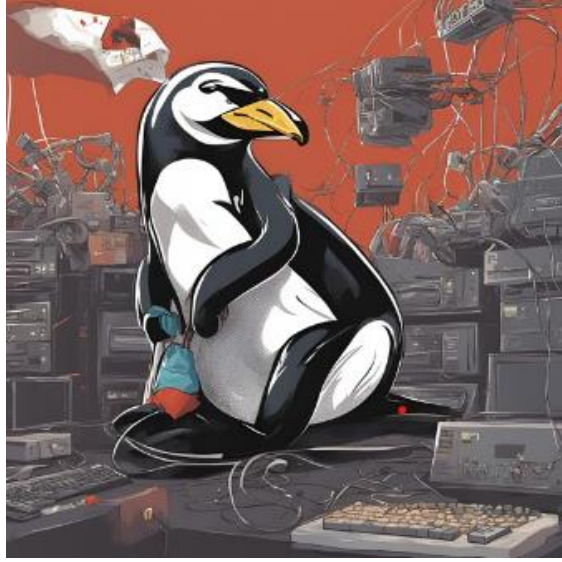


OverTheWire: Bandit

OverTheWire:Bandit, siber güvenlik alanında pratik yapmak ve Linux terminal becerilerini geliřtirmek isteyenler için ideal bir oyundur. Oyuncular, farklı güvenlik zorluklarıyla karşılařtıkları bir dizi seviyeden oluřan bu oyunda, çözümleri bulmak için Linux komut satırını kullanırlar. Temel Linux becerilerini öğrenmek ve güvenlik konusunda deneyim kazanmak isteyenler için mükemmel bir kaynaktır.



Ařağıdaki linkten ulaşabilirsiniz:

https://overthewire.org/wargames/bandit/?source=post_page-----9f9a4283d89c-----

Seviye 0

İlk önce SSH kullanarak giriş yapmamız bekleniyor.

Peki SSH nedir?

SSH (Secure Shell), güvenli bir şekilde bir bilgisayara veya sunucuya uzaktan erişmek için kullanılan bir ağ protokolüdür.

ssh kullanıcı_adi@sunucu_ip_adresi

yazarak sunucuya bağlanabiliriz. Eğer belirli bir porta bağlanmak istiyorsak da -p parametresini de ekleyerek port numarasını yazabiliriz.

Bize kullanıcı adının ve parolanın bandit0 olduğu bilgisi verilmiş.

ssh bandit0@bandit.labs.overthewire.org -p 2220

Komutunu yazdıktan sonra parolayı girerek sunucuya bağlanıyoruz.

Seviye 0 -> 1

Bir sonraki seviyenin şifresinin, ana dizinde bulunan **readme** adlı dosyada saklandığını biliyoruz.

*Dosya içeriğini görüntülemek için **cat** komutunu kullanıyoruz.*

```
bandit0@bandit:~$ cat readme
Congratulations on your first steps into the bandit game!!
Please make sure you have read the rules at https://overthewire.org/rules/
If you are following a course, workshop, walkthrough or other educational activity,
please inform the instructor about the rules as well and encourage them to
contribute to the OverTheWire community so we can keep these games free!

The password you are looking for is: ZjLjTmM6FvvyRnrb2rfNW0Z0Ta6ip5If
```

Seviye 1 -> 2

Bir sonraki seviyeye geçerken de ilk başta yazdığımız komutu kullanıyoruz. Sadece hangi seviyeye geçiyorsak onu kullanıcı adı olarak yazıyoruz. Yani;

ssh bandit1@bandit.labs.overthewire.org -p 2220

Bir sonraki seviyenin şifresinin, ana dizinde bulunan - adlı bir dosyada saklandığını biliyoruz.

“-” adındaki dosyayı okumamız gerekiyor. Ama “cat -” komutu ile okuyamıyoruz. İnternette araştırma yaparak ./- şeklinde okunabildiğini buldum. Komutumuz “cat ./-” olacak.

İlk olarak ./ ifadesi, “şu anki dizin” anlamına gelir. Ardından cat komutu, dosyaların içeriğini ekrana bastırmak için kullanılır. Yani cat ./ ifadesi ile mevcut dizindeki tüm dosyaların içeriğini ekrana yazdırabiliriz.

```
bandit1@bandit:~$ cat ./-
263JGJPfgU6LtdEvgfWU1XP5yac29mFx
```

Seviye 2 -> 3

Bir sonraki seviyenin şifresinin, ana dizinde bulunan **spaces in this filename** adı verilen bir dosyada saklandığını biliyoruz.

Eğer `cat spaces in this filename` yazarsak böyle bir dosyanın veya dizinin olmadığına dair hata alırız. Bunun içinde “ ” kullanmalıyız.

`cat “ ”` ifadesi, boşluk karakterleri içeren dosya adlarına sahip dosyaların içeriğini ekrana bastırmak için kullanılabilir. Normalde, dosya adları arasında boşluk karakterleri varsa terminalde bu dosyaları işlemek zor olabilir çünkü boşluk karakterleri genellikle bir argümanın bitişini belirtir. Ancak, çift tırnak içindeki bir dosya adı, bir bütün olarak kabul edilir ve bu şekilde `cat` komutuna geçirilebilir.

```
bandit2@bandit:~$ cat "spaces in this filename"
```

```
MNk8KNH3Usiio41PRUEoDFPqfxLPIsmx
```

Seviye 3 -> 4

Bir sonraki seviyenin şifresinin, **inhere** dizinindeki gizli bir dosyada saklandığını biliyoruz.

“`cd`” komutu ile “**inhere**” dosyasının içine girdik. `ls` komutu çalıştırınca dosyayı göremedik.

“-a” komutu gizli dosyaları gösterir. Bu yüzden “`ls -la`” komutunu çalıştırdık. Sonra da `cat` ile gizli dosyanın içeriğini okuduk.

```
bandit3@bandit:~$ ls
inhere
bandit3@bandit:~$ cd inhere/
bandit3@bandit:~/inhere$ ls
bandit3@bandit:~/inhere$ ls -la
total 12
drwxr-xr-x 2 root root 4096 Jun 11 04:49 .
drwxr-xr-x 3 root root 4096 Jun 11 04:49 ..
-rw-r----- 1 bandit4 bandit3 33 Jun 11 04:49 ...Hiding-From-You
bandit3@bandit:~/inhere$ cat ...Hiding-From-You
2WmrDFRmJIq3IPxneAaMGhap0pFhF3NJ
```

Seviye 4 -> 5

Bir sonraki seviyenin şifresi, **inhere** dizinindeki insanlar tarafından okunabilen tek dosyada saklandığını biliyoruz. (İpucu: Terminaliniz bozulduysa “reset” komutunu deneyin.)

```
bandit4@bandit:~$ ls
inhere
bandit4@bandit:~$ cd inhere/
bandit4@bandit:~/inhere$ ls
-file00 -file01 -file02 -file03 -file04 -file05 -file06 -file07 -file08 -file09
```

inhere dizininde birden çok dosya olduğunu görüyoruz. Bir dizindeki tüm dosyaların hangi türde olduğunu anlamak için veya belirli bir türde dosyaların bulunup bulunmadığını kontrol etmek için `file ./*` komutu kullanılır.

file komutu dosya türlerini belirlemek için kullanılırken, ./ ifadesi mevcut dizindeki tüm dosyaları seçer. Yani file ./* komutu ile mevcut dizindeki tüm dosyaların türlerini belirleyebiliriz.*

```
bandit4@bandit:~/inhere$ file ./*
./-file00: data
./-file01: data
./-file02: data
./-file03: data
./-file04: data
./-file05: data
./-file06: data
./-file07: ASCII text
./-file08: data
./-file09: data
bandit4@bandit:~/inhere$ cat ./-file07
4oQYVPkxZ00E005pTW81FB8j8lxXGUQw
```

Seviye 5 -> 6

Bir sonraki seviyenin şifresinin, **inhere** dizinin altındaki bir dosyada saklandığını ve aşağıdaki özelliklerin tümüne sahip olduğunu biliyoruz.

- insan tarafından okunabilir
- 1033 bayt boyutunda
- çalıştırılabilir değil

```
bandit5@bandit:~$ ls
inhere
bandit5@bandit:~$ cd inhere/
bandit5@bandit:~/inhere$ ls
maybeh ere00  maybeh ere03  maybeh ere06  maybeh ere09  maybeh ere12  maybeh ere15  maybeh ere18
maybeh ere01  maybeh ere04  maybeh ere07  maybeh ere10  maybeh ere13  maybeh ere16  maybeh ere19
maybeh ere02  maybeh ere05  maybeh ere08  maybeh ere11  maybeh ere14  maybeh ere17
```

find komutu, Linux ve diğer Unix benzeri işletim sistemlerinde dosya ve izin aramak için kullanılan bir araçtır. find komutunu belirli bir dizin hiyerarşisinde dosyaları bulmak için kullanabilir ve çeşitli arama kriterlerine dayalı olarak dosyaları filtreleyebiliriz.

-size seçeneği find komutu ile kullanılarak belirli bir dosya boyutuna göre dosyaları aramak için kullanılır.

c, dosya boyutunu byte cinsinden belirtir. Yani c kullanıldığında belirtilen dosya boyutu byte cinsinden ifade edilir.

Örneğin, =1033c ifadesi tam olarak 1033 byte boyutundaki dosyaları ifade eder.

```
bandit5@bandit:~/inhere$ find -size 1033c
./maybehere07/.file2
bandit5@bandit:~/inhere$ cat ./maybehere07/.file2
HWasnPhtq9AVKe0dmk45nxy20cvUa6EG
```

Seviye 6 -> 7

Bir sonraki seviyenin şifresinin **sunucuda bir yerde** saklandığını ve aşağıdaki özelliklerin tümüne sahip olduğunu biliyoruz.

- bandit7 kullanıcısı tarafından sahip olunan
- grup haydut6'ya ait
- 33 bayt boyutunda

`find / -user bandit7 -group bandit6 -type f -size 33c`

· */: Aramanın başlangıç noktasını belirtir. Tüm sunucuyu arayacak şekilde kök dizini olarak belirtilir.*

· *-user bandit7: Dosyanın "bandit7" kullanıcısına ait olduğunu belirtir.*

· *-group bandit6: Dosyanın "bandit6" grubuna ait olduğunu belirtir.*

· *-type f: "find" komutuna yalnızca düzenli dosyaları (dizinler veya diğer türdeki dosyalar değil) aramasını söyler.*


```
find: '/home/bandit5/inhere': Permission denied
find: '/home/drifter6/data': Permission denied
find: '/home/bandit27-git': Permission denied
find: '/home/drifter8/chroot': Permission denied
find: '/home/ubuntu': Permission denied
find: '/home/bandit29-git': Permission denied
find: '/home/bandit31-git': Permission denied
find: '/var/crash': Permission denied
find: '/var/lib/udisks2': Permission denied
find: '/var/lib/amazon': Permission denied
find: '/var/lib/polkit-1': Permission denied
find: '/var/lib/snapd/void': Permission denied
find: '/var/lib/snapd/cookie': Permission denied
find: '/var/lib/update-notifier/package-data-downloads/partial': Permission denied
find: '/var/lib/chrony': Permission denied
find: '/var/lib/dpkg/info/bandit7.password': Permission denied
find: '/var/lib/private': Permission denied
find: '/var/lib/ubuntu-advantage/apt-esm/var/lib/apt/lists/partial': Permission denied
find: '/var/lib/apt/lists/partial': Permission denied
find: '/var/spool/rsyslog': Permission denied
find: '/var/spool/bandit24': Permission denied
find: '/var/spool/cron/crontabs': Permission denied
find: '/var/log': Permission denied
find: '/var/tmp': Permission denied
find: '/var/cache/pollinate': Permission denied
find: '/var/cache/apparmor/baad73a1.0': Permission denied
find: '/var/cache/apparmor/2425d902.0': Permission denied
find: '/var/cache/private': Permission denied
find: '/var/cache/ldconfig': Permission denied
find: '/var/cache/apt/archives/partial': Permission denied
find: '/tmp': Permission denied
find: '/lost+found': Permission denied
find: '/snap': Permission denied
```

Hata kodlarını görmezden gelmek için Linux komut satırında yaygın olarak kullanılan bir yol, bir komutun çıktısını yönlendirmek ve hata kodlarını görmezden gelmek için `2>/dev/null` ifadesini kullanmaktır.

`2>` ifadesi, standart hata çıktısını yönlendirmek için kullanılır. `/dev/null`, Linux sistemlerinde çıktıyı reddetmek için özel bir dosyadır. Yani, `2>/dev/null` ifadesi, standart hata çıktısını `/dev/null` dosyasına yönlendirir. Bu da hata mesajlarının görmezden gelinmesini sağlar.

```
bandit6@bandit:~$ find / -user bandit7 -group bandit6 -type f -size 33c 2>/dev/null
/var/lib/dpkg/info/bandit7.password
bandit6@bandit:~$ cat /var/lib/dpkg/info/bandit7.password
morbNTDkSW6jIlUc0ymOdMaLn0lFVAaj
```

Seviye 7 -> 8

Bir sonraki seviyenin şifresinin **data.txt** dosyasında **millionth** kelimesinin yanında saklandığını biliyoruz.

Elimizde çok büyük bir “data.txt” dosyası var. Gözümüzle arayıp bulmamız çok zor. Bu yüzden “grep” komutunu kullanacağız.

| sembolü, Linux komut satırında iki komut arasında bağlantı sağlamak için kullanılır.

grep komutu ile | sembolünü birlikte kullanarak, bir komutun çıktısını grep komutunun girdisi olarak sağlayabilir ve belirli bir deseni arayabilirsiniz.

```
Hermite's      ozM4Br1dPtELjIMnwmc33FsvvBm3Izgr
Katmai's      89PqHsfBKSS9TzfokYcM40H8fhMh7RxC
gnat's      LHN3UCoF7dsHM9wLFvjn1QaYKYdOd7VH
Huitzilopitchli 0TGtEC7rXRqFVNFOZu8tPbiTjTMiEE8V
oldest      xClGeDzQZLZ7IH6YecMEjOhOvUSqaLV0
arson's      o9MxROJgU6o8mBOGz2MvjLuy7UL1kLZS
Deloris      4JtCji77K5TDgqLWnQHm3GTN1FCdI7XQ
sensibler     EELAdpHIH2T4VdIZQQETALrMkD1zzQ0g
Satanist      FB8UH9A1hVcdtrPIUL92Cv3Tg7puEWts
bandit7@bandit:~$ cat data.txt | grep millionth
millionth      dfwvzFQi4mU0wfNbFOe9RoWskMLg7eEc
```

Seviye 8 -> 9

Bir sonraki seviyenin şifresinin **data.txt** dosyasında saklandığını ve yalnızca bir kez geçen tek metin satırı olduğunu biliyoruz.

Yalnızca bir kez geçen tek bir metin satırını bulmak için sort, uniq ve grep komutlarını bir arada kullanabiliriz.

sort dosya_adı | uniq -u

Bu komut, dosya_adı dosyasındaki yalnızca bir kez geçen tek satırları ekrana basar. sort komutu, dosyadaki satırları sıralar, uniq -u komutu, yalnızca bir kez geçen satırları alır.

```
bandit8@bandit:~$ sort data.txt | uniq -u
4CKMh1JI91bUIZZPXDqGana14xvAg0JM
```

Seviye 9 -> 10

Bir sonraki seviyenin şifresinin **data.txt** dosyasında, insan tarafından okunabilen birkaç dizeden birinde ve önünde birkaç '=' karakteriyle saklandığını biliyoruz.

İnsan tarafından okunabilir metin dizelerini yani stringsleri bulmak için kullandığımız komut adı üzerinde strings'tir.

strings, bir dosyanın içeriğinden ASCII karakterlerini çıkaran ve bunları ekrana yazdıran bir Linux komutudur. Bu komut özellikle derlenmiş bir program veya diğer ikili dosyalar gibi dosyaların içeriğini incelemek için kullanışlıdır.

```
bandit9@bandit:~$ strings data.txt | grep =
.===== the
6===== passwordF
N===== is
===== FGUW5ilLVJrxX9kMYMmlN4MgbpfMiqey
```

Bu komut, “data.txt” dosyasındaki insan tarafından okunabilir dizeleri bulur ve ardından bu dizeler arasında en az bir “=” karakteri bulunanları görüntüler.

Seviye 10 -> 11

Bir sonraki seviyenin şifresinin, base64 kodlu verileri içeren **data.txt** dosyasında saklandığını biliyoruz.

base64, bir dosyanın veya metnin base64 kodlamasını oluşturmak veya çözmek için kullanılan bir komuttur.

Base64 kodlaması, verileri ASCII karakter seti içinde temsil etmek için kullanılan bir yöntemdir. Bu kodlama, özellikle metin olmayan verilerin (örneğin, resimler veya dosyalar) e-posta gibi metin tabanlı iletişim araçları aracılığıyla iletilmesi gerektiğinde kullanışlıdır, çünkü ASCII karakter seti yoluyla veri aktarımını sağlar.

Base64 kodunu çözmek için:

base64 -d dosya_adı

Eğer bir metin veya dizeden Base64 kodunu çözmek istiyorsanız:

echo "base64_metni" | base64 -d

```
bandit10@bandit:~$ ls
data.txt
bandit10@bandit:~$ cat data.txt
VGhlIHBhc3N3b3JkIGlzIGR0UjE3M2ZaS2IwUlJzREZTR3NnMlJXbnBOVmozcVJyCg==
bandit10@bandit:~$ echo "VGhlIHBhc3N3b3JkIGlzIGR0UjE3M2ZaS2IwUlJzREZTR3NnMlJXbnBOVmozcVJyCg==" | base64 -d
The password is dtR173fZKb0RRsDFSGsg2RWnpNVj3qRr
```

Seviye 11 -> 12

Bir sonraki seviyenin şifresinin, tüm küçük (az) ve büyük (AZ) harflerin 13 konum döndürüldüğü **data.txt** dosyasında saklandığını biliyoruz. Ayrıca Yararlı okuma metninde Rot13'ün bağlantısı verilmiş.

İlk önce Rot13'ün ne olduğuna bakalım.

ROT13 (Rotate by 13 places), bir şifreleme algoritmasıdır ve bir metni alfabedeki her harfi 13 pozisyon kaydırarak şifreler veya çözer. Bu algoritma hem şifreleme hem de çözme işlemlerinde aynıdır, yani metni iki kez ROT13 uyguladığınızda orijinal metne geri dönersiniz.

Örneğin, ROT13 uygulandığında 'a' harfi 'n' harfine, 'b' harfi 'o' harfine ve böylece devam eder. Aynı şekilde, 'n' harfi 'a' harfine, 'o' harfi 'b' harfine ve benzeri şekilde döner.

Aşağıdaki komut, data.txt dosyasındaki metni 13 konum sola kaydırarak şifreyi çözecektir:

```
tr 'a-zA-Z' 'n-za-mN-ZA-M' < data.txt
```

Bu komut, data.txt dosyasındaki metni alır ve tüm küçük ve büyük harfleri alfabe sırasında 13 konum sola kaydırarak şifreyi çözer. tr komutu, karakterleri birbiriyle değiştirmek için kullanılır. 'a-zA-Z' ifadesi, tüm küçük ve büyük harfleri içeren bir karakter sınıfını belirtir. 'n-za-mN-ZA-M' ifadesi, bu harfleri alfabe sırasında 13 konum sola kaydırmak için kullanılır. Bu şekilde, orijinal metni elde edebiliriz.

```
bandit11@bandit:~$ ls
data.txt
bandit11@bandit:~$ cat data.txt
Gur cnffjbeq vf 7k16JArUVv5LxVuJfsSVdbbtaHGlw9D4
bandit11@bandit:~$ tr 'a-zA-Z' 'n-za-mN-ZA-M' < data.txt
The password is 7x16WNeHIi5YkIhWsFIqoognUTyj9Q4
```

Seviye 12 -> 13

Bir sonraki seviyenin parolasının, tekrar tekrar sıkıştırılan bir dosyanın onaltılı dökümü olan **data.txt** dosyasında saklandığını biliyoruz.

Ayrıca “Bu seviye için /tmp altında çalışabileceğiniz bir dizin oluşturmak yararlı olabilir. Tahmin edilmesi zor bir dizin adıyla mkdir kullanın. Veya daha iyisi “mktemp -d” komutunu kullanın. Daha sonra veri dosyasını cp kullanarak kopyalayın ve mv kullanarak yeniden adlandırın (man sayfalarını okuyun!)” bilgilendirmesini görüyoruz.

```
bandit12@bandit:~$ mkdir /tmp/12lvl
bandit12@bandit:~$ cp data.txt /tmp/12lvl
bandit12@bandit:~$ cd /tmp/12lvl
```

Belirtildiği gibi mkdir ile /tmp altına bir dizin oluşturuyorum. Sonra cp komutu ile data.txt'i oluşturduğum dizine kopyalıyorum.

```
bandit12@bandit:/tmp/12lvl$ xxd -r data.txt > 12lvl
bandit12@bandit:/tmp/12lvl$ ls -la
total 40
drwxrwxr-x  2 bandit12 bandit12  4096 Jun 12 13:05 .
drwxrwx-wt 487 root      root      24576 Jun 12 13:05 ..
-rw-rw-r--  1 bandit12 bandit12   610 Jun 12 13:05 12lvl
-rw-r----- 1 bandit12 bandit12  2638 Jun 12 13:04 data.txt
```

Burada data.txt dosyasındaki hexadecimal (onaltılık) veriyi geri dönüştürerek (-r) normal (metin) formata getirip bu dönüştürülmüş veriyi 12lvl adlı oluşturduğumuz dosyaya yönlendiriyoruz.

Özetle xxd -r data.txt komutu, data.txt dosyasındaki onaltılık veriyi normal metin formatına çevirir ve > operatörüyle bu dönüştürülmüş veriyi 12lvl adlı bir dosyaya yazar.

```
bandit12@bandit:/tmp/12lvl$ file 12lvl
12lvl: gzip compressed data, was "data2.bin", last modified: Tue Jun 11 21:29:38 2024
, max compression, from Unix, original size modulo 2^32 577
```

file komutunu kullanarak dosyanın özelliklerine baktığımızda, dosyanın gzip ile sıkıştırılmış olduğunu görüyoruz. Ancak, gzip -d seviye12 komutuyla dosyayı doğrudan açamayız çünkü dosyanın uzantısında ".gzip" olduğu belirtilmemiş. Uzantıyı değiştirerek dosyayı açabiliriz.

```
bandit12@bandit:/tmp/12lvl$ mv 12lvl 12lvl.gz
bandit12@bandit:/tmp/12lvl$ ls -la
total 40
drwxrwxr-x  2 bandit12 bandit12  4096 Jun 12 13:14 .
drwxrwx-wt 489 root      root      24576 Jun 12 13:14 ..
-rw-rw-r--  1 bandit12 bandit12   610 Jun 12 13:05 12lvl.gz
-rw-r----- 1 bandit12 bandit12  2638 Jun 12 13:04 data.txt
bandit12@bandit:/tmp/12lvl$ gzip -d 12lvl.gz
bandit12@bandit:/tmp/12lvl$ ls
12lvl  data.txt
```

mv komutu ile ismini değiştirdik sonra, gzip ile zipten çıkardık.

```
bandit12@bandit:/tmp/12lvl$ file 12lvl
12lvl: bzip2 compressed data, block size = 900k
bandit12@bandit:/tmp/12lvl$ mv 12lvl 12lvl.bz2
```

Dosyanın özelliklerini file 12lvl komutuyla kontrol ettiğimizde, bu sefer dosyanın bzip2 ile sıkıştırıldığını görüyoruz. Dosyanın adını mv 12lvl.bz2 komutuyla değiştirip ardından bzip2 -d seviye12.bz2 komutuyla çıkarabiliriz.

```
bandit12@bandit:/tmp/12lvl$ bzip2 -d 12lvl.bz2
```

```
bandit12@bandit:/tmp/12lvl$ file 12lvl
12lvl: gzip compressed data, was "data4.bin", last modified: Tue Jun 11 21:29:38 2024
, max compression, from Unix, original size modulo 2^32 20480
```

Dosyanın bir kez daha gzip ile sıkıştırıldığını tespit ettik. Önceki yöntemi kullanarak dosyayı çıkarabiliriz.

```
bandit12@bandit:/tmp/12lvl$ mv 12lvl 12lvl.gz
bandit12@bandit:/tmp/12lvl$ gzip -d 12lvl.gz
bandit12@bandit:/tmp/12lvl$ file 12lvl
12lvl: POSIX tar archive (GNU)
```

Dosyanın ardından bir tar dosyası çıkıyor. İsimlendirme için mv 12lvl 12lvl.tar komutunu kullandık ve ardından tar xf 12lvl.tar ile içeriğini çıkardık. Daha sonra ls komutu ile data5.bin adında bir dosya gördük ve aynı şekilde bu dosyayı da tar'dan çıkardık.

```
bandit12@bandit:/tmp/12lvl$ mv 12lvl 12lvl.tar
bandit12@bandit:/tmp/12lvl$ tar xf 12lvl.tar
bandit12@bandit:/tmp/12lvl$ ls -la
total 68
drwxrwxr-x  2 bandit12 bandit12  4096 Jun 12 13:22 .
drwxrwx-wt 499 root      root      24576 Jun 12 13:22 ..
-rw-rw-r--  1 bandit12 bandit12 20480 Jun 12 13:05 12lvl.tar
-rw-r--r--  1 bandit12 bandit12 10240 Jun 11 21:29 data5.bin
-rw-r----- 1 bandit12 bandit12  2638 Jun 12 13:04 data.txt
bandit12@bandit:/tmp/12lvl$ file data5.bin
data5.bin: POSIX tar archive (GNU)
```

Sonrasında ise data6.bin adında bir dosya daha karşımıza çıktı.


```
bandit12@bandit:/tmp/12lvl$ mv data5.bin data5.bin.tar
bandit12@bandit:/tmp/12lvl$ tar xf data5.bin.tar
bandit12@bandit:/tmp/12lvl$ ls -la
total 72
drwxrwxr-x  2 bandit12 bandit12  4096 Jun 12 13:24 .
drwxrwx-wt 505 root      root      24576 Jun 12 13:24 ..
-rw-rw-r--  1 bandit12 bandit12 20480 Jun 12 13:05 12lvl.tar
-rw-r--r--  1 bandit12 bandit12 10240 Jun 11 21:29 data5.bin.tar
-rw-r--r--  1 bandit12 bandit12   221 Jun 11 21:29 data6.bin
-rw-r-----  1 bandit12 bandit12  2638 Jun 12 13:04 data.txt
```

File ile data6.bin'in de bzip2 ile sıkıştırıldığını öğrendik.

```
bandit12@bandit:/tmp/12lvl$ file data6.bin
data6.bin: bzip2 compressed data, block size = 900k
bandit12@bandit:/tmp/12lvl$ mv data6.bin data6.bin.bz2
bandit12@bandit:/tmp/12lvl$ bzip2 -d data6.bin.bz2
bandit12@bandit:/tmp/12lvl$ ls -la
total 80
drwxrwxr-x  2 bandit12 bandit12  4096 Jun 12 13:27 .
drwxrwx-wt 511 root      root      24576 Jun 12 13:27 ..
-rw-rw-r--  1 bandit12 bandit12 20480 Jun 12 13:05 12lvl.tar
-rw-r--r--  1 bandit12 bandit12 10240 Jun 11 21:29 data5.bin.tar
-rw-r--r--  1 bandit12 bandit12 10240 Jun 11 21:29 data6.bin
-rw-r-----  1 bandit12 bandit12  2638 Jun 12 13:04 data.txt
bandit12@bandit:/tmp/12lvl$ file data6.bin
data6.bin: POSIX tar archive (GNU)
```

Aynı yöntemi kullanarak onu da zipten çıkarıyoruz. Ardından, tar ile sıkıştırıldığını görüyoruz ve yine aynı şekilde tar'dan çıkarıyoruz.

```
bandit12@bandit:/tmp/12lvl$ mv data6.bin data6.bin.tar
bandit12@bandit:/tmp/12lvl$ tar xf data6.bin.tar
bandit12@bandit:/tmp/12lvl$ ls -la
total 84
drwxrwxr-x  2 bandit12 bandit12  4096 Jun 12 13:30 .
drwxrwx-wt 512 root      root      24576 Jun 12 13:30 ..
-rw-rw-r--  1 bandit12 bandit12 20480 Jun 12 13:05 12lvl.tar
-rw-r--r--  1 bandit12 bandit12 10240 Jun 11 21:29 data5.bin.tar
-rw-r--r--  1 bandit12 bandit12 10240 Jun 11 21:29 data6.bin.tar
-rw-r--r--  1 bandit12 bandit12    79 Jun 11 21:29 data8.bin
-rw-r-----  1 bandit12 bandit12  2638 Jun 12 13:04 data.txt
```

Daha sonra bir dizindeki tüm dosyaları ve dizinleri ayrıntılı bir şekilde listelemek için `ls -la` komutunu kullanıyoruz ve `data8.bin` ile karşılaşırız.

```
bandit12@bandit:/tmp/12lvl$ file data8.bin
data8.bin: gzip compressed data, was "data9.bin", last modified: Tue Jun 11 21:29:38
2024, max compression, from Unix, original size modulo 2^32 49
bandit12@bandit:/tmp/12lvl$ mv data8.bin data8.bin.gz
bandit12@bandit:/tmp/12lvl$ gzip -d data8.bin.gz
bandit12@bandit:/tmp/12lvl$ ls -la
total 88
drwxrwxr-x  2 bandit12 bandit12  4096 Jun 12 13:32 .
drwxrwx-wt 518 root      root      28672 Jun 12 13:32 ..
-rw-rw-r--  1 bandit12 bandit12 20480 Jun 12 13:05 12lvl.tar
-rw-r--r--  1 bandit12 bandit12 10240 Jun 11 21:29 data5.bin.tar
-rw-r--r--  1 bandit12 bandit12 10240 Jun 11 21:29 data6.bin.tar
-rw-r--r--  1 bandit12 bandit12    49 Jun 11 21:29 data8.bin
-rw-r----- 1 bandit12 bandit12  2638 Jun 12 13:04 data.txt
bandit12@bandit:/tmp/12lvl$ file data8.bin
data8.bin: ASCII text
bandit12@bandit:/tmp/12lvl$ cat data8.bin
The password is F05dwFsc0cbaIiH0h8J2eUks2vdTDwAn
```

`file` komutu ile `data8.bin`'inde `gzip` ile sıkıştırıldığını görüyoruz ve `gzip`ten çıkarıyoruz. Çıkarılmış hali ile tekrar `file` diyoruz ve sonunda `ASCII text`'e ulaşıyoruz.

`cat data8.bin` diyerek bir sonraki seviyenin şifresine ulaşıyoruz.

Seviye 13 -> 14

Bir sonraki seviyenin şifresinin `/etc/bandit_pass/bandit14` dosyasında saklandığını ve yalnızca `bandit14` kullanıcısı tarafından okunabildiğini biliyoruz.

Ayrıca bu seviye için bir sonraki şifreyi almayacağımızı ancak bir sonraki seviyeye giriş yapmak için kullanılabilecek özel bir SSH anahtarı alacağımızı öğreniyoruz. Not olarak da `localhost`'un üzerinde çalıştığımız makineyi ifade eden bir ana bilgisayar adı olduğunu tekrar hatırlıyoruz.

İlk önce `ls -la` ile dosya ve dizinlere bakıyoruz. `sshkey.private` dosyasını görüyoruz.

Daha sonra `ssh` ile direkt `bandit14` kullanıcısına bağlanıyoruz.


```
bandit13@bandit:~$ ls -la
total 24
drwxr-xr-x  2 root    root    4096 Jun 11 21:29 .
drwxr-xr-x 70 root    root    4096 Jun 11 21:30 ..
-rw-r--r--  1 root    root     220 Mar 31 08:41 .bash_logout
-rw-r--r--  1 root    root    3771 Mar 31 08:41 .bashrc
-rw-r--r--  1 root    root     807 Mar 31 08:41 .profile
-rw-r----- 1 bandit14 bandit13 1679 Jun 11 21:29 sshkey.private
bandit13@bandit:~$ ssh bandit14@localhost -i sshkey.private -p 2220
The authenticity of host '[localhost]:2220 ([127.0.0.1]:2220)' can't be established.
ED25519 key fingerprint is SHA256:C2ihUBV7ihnV1wUXRb4RrEcLfXC5CXlhmAAM/urerLY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Could not create directory '/home/bandit13/.ssh' (Permission denied).
Failed to add the host to the list of known hosts (/home/bandit13/.ssh/known_hosts).
```

bandit14@localhost: localhost sunucusundaki bandit14 kullanıcısına bağlanmayı belirler.

-i sshkey.private: Kimlik doğrulaması için sshkey.private dosyasını özel anahtar olarak kullanır. Bu dosya, bağlantı sırasında kimlik doğrulaması için gerekli olan özel anahtarı içerir.

-p 2220: SSH bağlantısı için kullanılacak port numarasını belirtir.

En son da `cat /etc/bandit_pass/bandit14` yazarak şifreye ulaşıyoruz.

```
bandit14@bandit:~$ cat /etc/bandit_pass/bandit14
MU4VWeTyJk8ROof1qqmcBPALh7LDCPvS
```

Seviye 14 ->15

Bir sonraki seviyenin şifresinin, **localhost üzerinde 30000 numaralı porta** mevcut seviyenin şifresi gönderilerek alınabileceğini biliyoruz.

Bunun için ncat veya nckomutunu kullanıyoruz.

nc (netcat), ağ bağlantıları oluşturmak ve yönetmek için kullanılan çok yönlü bir komut satırı aracıdır. Hem TCP hem de UDP protokollerini destekler ve pek çok ağ görevini yerine getirebilir.

ncat komutu, Nmap Security Scanner projesinin bir parçası olarak geliştirilen ve netcat (nc) komutuna benzer işlevler sunan güçlü bir ağ bağlantı aracıdır. ncat; ağ bağlantıları oluşturmak, veri iletimi, dinleme, dosya transferi ve daha birçok ağ ile ilgili görevi gerçekleştirebilir.

nc localhost 30000 yazarak *localhost* üzerindeki 30000 numaralı porta bağlanıyorum.

Komutu yazdıktan sonra enter diyoruz ve bir önceki bulduğumuz şifreyi yazıyoruz.

Sonrasında da bir sonraki seviyenin şifresini görüyoruz.

```
bandit14@bandit:~$ nc localhost 30000
MU4VWeTyJk8R0of1qqmcBPALh7lDCPvS
Correct!
8xCjnmgoKbGLhHFAZlGE5Tmu4M2tKJQo
```

Seviye 15 → 16

“Sonraki seviyenin şifresini, mevcut seviyenin şifresini SSL şifrelemesi kullanarak *localhost* üzerindeki 30001 numaralı porta göndererek alabilirsiniz.”

`openssl s_client -connect localhost:30001`

Komutunu yazıyoruz. Sonra bir önceki seviyenin şifresini giriyoruz ve ente’a basıyoruz.

openssl s_client -connect localhost:30001 komutu, OpenSSL kütüphanesinin *s_client* aracını kullanarak SSL/TLS üzerinden belirtilen hedef sunucuya bağlanmayı sağlar. Bu komutun temel kullanımı ve parametreleri şu şekildedir:

openssl: OpenSSL kriptografik araçlarının genel komutu.

s_client: OpenSSL’in SSL/TLS istemcisi modunu başlatır. Bu mod, sunucu ile SSL/TLS bağlantısı kurarak iletişim kurmayı sağlar.

-connect localhost:30001: Bağlanılacak hedef sunucunun adresi ve portu. Bu örnekte *localhost* üzerindeki 30001 numaralı porta bağlanmayı ifade eder.

```

Start Time: 1718986945
Timeout : 7200 (sec)
Verify return code: 18 (self-signed certificate)
Extended master secret: no
Max Early Data: 0

---
read R BLOCK
---
Post-Handshake New Session Ticket arrived:
SSL-Session:
    Protocol : TLSv1.3
    Cipher : TLS_AES_256_GCM_SHA384
    Session-ID: BC620A0F1845392CB70A628B27453D998A5E690EE1C6188245CC914BF8FBCC9C
    Session-ID-ctx:
    Resumption PSK: 371793A4E32C2B422739BE945DE30C6CE813398BF8041B0CF6E7A820E3EB9E60B00A6C5F1171BF747FF416A59CCB963F
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
0000 - 60 61 e6 2e a8 2b fb 0f-0a aa 39 05 88 21 10 3e 'a ... +....9.. l.>
0010 - a3 13 ff 24 d7 73 e1 b3-64 83 c6 05 6a 50 e2 4e ... $.s..d... jP.N
0020 - c4 4f cf f8 6c e0 00 2c-13 ea cb 71 ae 9c ca 5c .O..l.., ...q... \
0030 - 3e ea e0 00 7d c3 6a 69-f2 06 d9 57 c0 47 fa 26 >... }.j1 ...W.G.6
0040 - 85 ff bf 81 75 70 a7 b3-92 d6 d8 ec 98 aa e2 fa ....up.....
0050 - 11 d0 6a 09 6a c1 e5 d0-7b 35 f5 e4 9e a5 ec bd ..j.j... {5.....
0060 - 2d 9f b6 75 2b fd fb 54-19 63 99 4f 13 a6 e0 eb ~..u+..T.c.O....
0070 - 9d c1 a0 2d 14 ed 0a 6c-c9 f1 2d dc 33 90 57 22 ...-...l...-3.W"
0080 - 4d b6 c1 8c 02 91 49 02-d2 06 18 da 2d 12 98 2e M.....I.....~ ...
0090 - 88 30 fe 85 bc 6a cc eb-1f ae ef c9 4e 43 92 02 .0... j.....NC..
00a0 - 94 87 97 43 80 6b e1 33-7c 3c a8 76 7d 7e e1 53 ...C.k.3|<.v}~.S
00b0 - 32 44 c6 8e a2 74 85 05-a8 66 12 d3 77 f8 44 38 2D... t... f..w.D8
00c0 - bf bb c5 fb aa d5 a2 7e-05 81 11 1f 3f cb 6b 8b .....~.....?.k.
00d0 - 4f 35 7e cf 59 9d a6 7c-2a 89 d1 6a db 9b 80 c3 05~.Y.. |*..j....

Start Time: 1718986945
Timeout : 7200 (sec)
Verify return code: 18 (self-signed certificate)
Extended master secret: no
Max Early Data: 0

---
read R BLOCK
8xCjnmgoKbGLhHFAZlGE5Tmu4M2tKJQo
Correct!
kSkvUpMQ7lBYyCM4G8PvCvT1BfWRy0Dx

```

Seviye 16 -> 17

“Sonraki seviyenin kimlik bilgileri, mevcut seviyenin şifresini 31000 ila 32000 arasındaki localhost portlarından birine göndererek alınabilir. İlk olarak, bu portlardan hangilerinde bir sunucunun dinlediğini bulun. Sonra, bunlardan hangilerinin SSL ile iletişim kurduğunu ve hangilerinin kurmadığını belirleyin. Sadece 1 sunucunun, diğerleri sadece size gönderdiğiniz şeyi geri gönderecek şekilde davrandığını unutmayın.”

`nmap -sV localhost -p 31000-32000`

nmap: Ağdaki hedef sistemler hakkında bilgi toplamak ve ağ keşfi yapmak için kullanılan güçlü bir araçtır.

-sV: Servis sürüm tespiti yapar; yani her port için çalışan servisin sürüm bilgilerini almaya çalışır.

localhost: Bu parametre, nmap’in yerel makineye (bulunduğunuz bilgisayar) tarama yapmasını sağlar. localhost, genellikle 127.0.0.1 IP adresini ifade eder.

-p 31000-32000: Bu seçenek, 31000 ile 32000 arasındaki portları taramak için kullanılır. Yani bu komut, yerel makinenizdeki 31000 ile 32000 arasındaki portlarda çalışan servisleri belirlemeye çalışır.

```
bandit16@bandit:~$ nmap -sV localhost -p 31000-32000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-21 18:55 UTC
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00015s latency).
Not shown: 996 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
31046/tcp  open  echo
31518/tcp  open  ssl/echo
31691/tcp  open  echo
31790/tcp  open  ssl/unknown
31960/tcp  open  echo
```

Nmap, beş bağlantı noktasının açık olduğunu bildiriyor. Bu bağlantı noktalarından sadece ikisi (31518 ve 31790) SSL kullanıyor. Ayrıca, 31518 numaralı bağlantı noktasının yalnızca echo servisini çalıştırdığını belirtiyor. 31790 numaralı bağlantı noktası ise bilinmeyen bir servisi barındırıyor gibi görünüyor.

Portu da bulduktan sonra,

ncat - ssl localhost 31790

komutunu yazıyoruz. Ve bir önceki şifreyi girerek enter'a basıyoruz.

ncat: Ağ bağlantılarını yönetmek, veri iletimi yapmak ve ağ hizmetlerini test etmek için kullanılan bir araçtır.

— **ssl:** SSL/TLS ile güvenli bir bağlantı kurulmasını sağlar. Ncat, veri iletimi sırasında şifreleme ve kimlik doğrulama sağlamak için SSL veya TLS protokollerini kullanır.

31790: Bağlanılacak olan port numarası.

Bu komut çalıştırıldığında, ncat programı localhost üzerindeki 31790 numaralı port'a SSL/TLS ile güvenli bir bağlantı kurmaya çalışır. Eğer 31790 numaralı port bir SSL/TLS servisi tarafından kullanılıyorsa ve bağlantı başarılıysa, ncat ile veri alışverişi yapabilirsiniz.

Karşımıza RSA private key çıkıyor. Kopyalıyoruz.


```
bandit16@bandit:~$ ncat --ssl localhost 31790
kSkvUpMQ7lBYyCM4GBPvCvT1BfWRy0Dx
Correct!
-----BEGIN RSA PRIVATE KEY-----
MIIeogIBAAKCAQEAvm0kuifmMg6HL2YPIOjon6iWfbp7c3jx34YkYWqUH57SudyJ
imZzeyGC0gtZPGujUSxiJSWI/oTqexh+cAMTSMlOJf7+BrJObArnxd9Y7YT2bRPQ
Ja6Lzb558YW3FZl87ORiO+rW4LCDCNd2lUvLE/GL2GWyuKN0K5iCd5TbtJzEkQTu
DSt2mcNn4rHAL+JFr56o4T6z8WWAW18BR6yGrMq7Q/kALHYW3OekePQAZL0VUYbW
JGTi65CxbCnzc/w4+mQYvmzpWtMAzJTzAzQxNbkr2MBGySxDLrjg0LWN6sK7wNX
x0YVztz/zbIkPjfkU1jHS+9EbVNj+D1XFOJuaQIDAQABAoIBABagpxpM1aoLWfvD
KHcj10nqcoBc4oE1laFYQwik7xfW+24pRNUDE6SFth0ar69jp5RlLwD1NhPx3iBl
J9nOM8OJ0VToum43UOS8YxF8WwhXriYGnc1sskbwpX0UDc9uX4+UESzH22P29ovd
d8WErY0gPxun8pbJLmxkAtWNhpMvfe0050vk9TL5wqbu9AlbssgTcCXkMQnPw9nC
YNN6DDP2lbcBrvgT9YCNL6C+ZKufD52yOQ9q0kwFTEQpjtf4uNtJom+asvlpms8A
vLY9r60wYSvmZhNqBURj7lyCtXMIu1kkd4w7F77k+DjHoAXyxcUp1DGL51sOmama
+TOWWgECgYEA8JtPxP0GRJ+IQkX262jM3dEIkza8ky5moIwUqYdsx0NxHgRRhORT
8c8hAuRBb2G82so8vUHK/fur850Efc9TncnCY2crpoqsgghifKLxrLgtT+qDpfZnx
SatLdt8GfQ85yA7hnWWJ2MxF3NaeSDm75Lsm+tBbAiyC9P2jGRNtMSkCgYEAypHd
HCctNi/FwjulhttFx/rHYKhLidZDFYeiE/v45bN4yFm8x7R/b0iE7KaszX+Exdvt
SghaTdcG0Knyw1bpJVusavPzpaJMjdJ6tcFhVAbAjm7enCIvGCSx+X3l5SiWg0A
R57hJglezIiVjv3aGwHwvLZvtszK6zV6oXFAu0ECgYAbjo46T4hyP5tJi93V5Hdi
TtieK7xRVxUL+iU7rWkGAXFpMLFteQEsRr7PJ/lemmEY5eTDAFmLY9FL2m9oQWCg
R8VdwSk8r9FGLS+9aKcV5PI/WEKlwGxinB30hYimtiG2Cg5JCqIZFHxD6MjEG0iu
L8ktHMPvodBwNsSBULpG0QKBgBAPlTfC1HONwiMGOU3KPwYwt006CdTkmJOML8Ni
blh9elyZ9FsGxsgtRBXRsqXuz7wtsQAgLHxbdLq/ZJQ7Yfz0KU4ZxEnabvXnvWkU
Y0djHdS0oKvDQNWu6ucyLRAWFuISeXw9a/9p7ftpxm0TSgyvmfLF2MIAEwyzRqaM
77pBAoGAMmjmIJdjp+Ez8duyn3ieo36yrftF5NSsJLABxFpdlc1gvtGCWW+9Cq0b
dxviW8+TFVEBl104f7HVM6EpTscdDxU+bCXWkfjuRb7Dy9G0tt9JPsx8MBTakzh3
vBgysi/sN3RqRBcGU40f0oZyfAMT8s1m/uYv5206IgeuZ/ujbjY=
-----END RSA PRIVATE KEY-----
```

cd /tmp diyerek tmp dizinine gidiyoruz. touch 16.key adlı bir dosya oluşturuyoruz. Sonra nano 16.key diyerek kopyaladığımız keyi içine yapııştırıyoruz.

```
bandit16@bandit:~$ cd /tmp
bandit16@bandit:/tmp$ touch 16.key
bandit16@bandit:/tmp$ nano 16.key
```

nano, basit ve kullanımı kolay bir metin düzenleyici veya metin editörüdür. Genellikle terminal veya komut satırı üzerinde çalışır. nano'yu kullanarak metin dosyalarını oluşturabilir, düzenleyebilir ve kaydedebilirsiniz.

• Dosyayı kaydetmek için Ctrl + O, ardından Enter tuşuna basarak onaylayın.

· nano editöründen çıkmak için Ctrl + X tuşlarına basın. Eğer dosyada yapılan değişiklikler kaydedilmediyse, size kaydetme seçeneği sunar.)

Sonra,

chmod 700 16.key

komutu ile dosyanın iznini değiştiriyoruz.

(Dosyanın sahibi okuma (4), yazma (2) ve yürütme (1) izinlerine sahiptir. Bu izinlerin toplamı 7'dir. Grubun ve diğerlerinin izni yok (0))

Son olarak da,

ssh -i 16.key bandit17@localhost -p 2220

komutuna yes diyerek 17. Seviyeye geçiyoruz.

Seviye 17 -> 18

“Ana dizinde 2 dosya vardır: **passwords.old** ve **passwords.new**. Bir sonraki seviyenin şifresi **passwords.new** dosyasındadır ve **passwords.old** ile **passwords.new** arasında değiştirilen tek satırdır.”

NOT: Bu seviyeyi çözdüyseniz ve bandit18'e giriş yapmaya çalıştığınızda “Byebye!” görüyorsanız, bu bir sonraki seviye olan bandit19 ile ilgilidir.

Bu bölümü çözmek için diff komutunu kullanabiliriz.

diff komutu, iki dosya veya dizin arasındaki farkları bulmak için kullanılan bir komuttur.

diff passwords.new passwords.old

Bu komut, passwords.new ve passwords.old dosyalarını karşılaştırır ve farklılıkları terminalde gösterir. diff komutunun çıktısı, her farklılık için ayrı satırlar içerir ve bu satırlarda değişikliklerin nerede ve ne şekilde olduğunu gösterir.

· <“**passwords.new**” satırından gelen satırı gösterir

· >“**passwords.old**” daki satırı gösterir.

```
bandit17@bandit:~$ ls -la
total 36
drwxr-xr-x  3 root    root    4096 Jun 20 04:06 .
drwxr-xr-x 70 root    root    4096 Jun 20 04:08 ..
-rw-r----- 1 bandit17 bandit17   33 Jun 20 04:06 .bandit16.password
-rw-r--r--  1 root    root      220 Mar 31 08:41 .bash_logout
-rw-r--r--  1 root    root    3771 Mar 31 08:41 .bashrc
-rw-r----- 1 bandit18 bandit17 3300 Jun 20 04:06 passwords.new
-rw-r----- 1 bandit18 bandit17 3300 Jun 20 04:06 passwords.old
-rw-r--r--  1 root    root      807 Mar 31 08:41 .profile
drwxr-xr-x  2 root    root    4096 Jun 20 04:06 .ssh
bandit17@bandit:~$ diff passwords.new passwords.old
42c42
< x2gLTTjFwMOhQ8oWNbMN362QKxfRqGlO
---
> FtePUTiLiwPzjIFw2T7o57oBS4zUvPpg
```

Enjoy your stay!

Byebye !

Connection to bandit.labs.overthewire.org closed.

Seviye 18 -> 19

“Sonraki seviyenin şifresi, homedizinindeki readme adlı bir dosyada saklanmaktadır. Ne yazık ki birisi **.bashrc** dosyasını SSH ile giriş yaptığınızda sizi oturumdan çıkarmak için değiştirmiş.”

```
ssh bandit18@bandit.labs.overthewire.org -p 2220 -t "/bin/sh"
```

Bu komut, bandit18 kullanıcısı olarak bandit.labs.overthewire.org sunucusuna 2220 portu üzerinden SSH ile bağlanır ve bağlandığında varsayılan shell (bash yerine) /bin/sh shell'ini başlatır.

-t: TTY (teletypewriter) tahsis etmek için kullanılan bir seçenek. Bu, uzaktan komut çalıştırırken interaktif bir terminal oturumu açmanızı sağlar. Örneğin, uzak shell oturumunun interaktif komutları kabul etmesini sağlar.

“/bin/sh”: Uzak sunucuda çalıştırılacak komut veya başlatılacak shell. Bu durumda, /bin/sh shell'i başlatılır. Bu, genellikle daha basit ve hafif bir shell olan Bourne shell'dir.

```
(kali㉿kali)-[~]  
$ ssh bandit18@bandit.labs.overthewire.org -p 2220 -t "/bin/sh"  
  
      _ _ _ _ _  
     / / / / /  
    / / / / /  
   / / / / /  
  / / / / /  
 / / / / /  
/ / / / /  
  
This is an OverTheWire game server.  
More information on http://www.overthewire.org/wargames  
  
bandit18@bandit.labs.overthewire.org's password:  
$ ls  
readme  
$ cat readme  
cGWpMaKXVwDUNgPAVJbWYuGHVn9z13j8  
$
```

[illegible]

“Bir sonraki seviyeye erişim sağlamak için ana dizindeki setuid ikili dosyasını kullanmalısınız. Nasıl kullanılacağını öğrenmek için argümanlar olmadan çalıştırın. Bu seviyenin şifresi, setuid ikilisini kullandıktan sonra her zamanki yerde (/etc/bandit_pass) bulunabilir.”

```
bandit19@bandit:~$ ls -la
total 36
drwxr-xr-x  2 root    root    4096 Jun 20 04:07 .
drwxr-xr-x 70 root    root    4096 Jun 20 04:08 ..
-rwsr-x---  1 bandit20 bandit19 14880 Jun 20 04:07 bandit20-do
-rw-r--r--  1 root    root     220 Mar 31 08:41 .bash_logout
-rw-r--r--  1 root    root    3771 Mar 31 08:41 .bashrc
-rw-r--r--  1 root    root     807 Mar 31 08:41 .profile
bandit19@bandit:~$ ./bandit20-do
Run a command as another user.
Example: ./bandit20-do id
```

Örnek olarak id dediği için o şekilde yazdık.

```
bandit19@bandit:~$ ./bandit20-do id
uid=11019(bandit19) gid=11019(bandit19) euid=11020(bandit20) groups=11019(bandit19)
```

İkili dosyayı kullandığımızda bandit20 için de uid'nin atandığını gözlemliyoruz, bu da sanki bandit20'ymişiz gibi komutları çalıştırabildiğimiz anlamına geliyor.

Artık komutları bandit20 olarak çalıştırabileceğimizi bildiğimize göre, bandit20 kullanıcısının şifresine erişmek için ikili dosyayı kullanalım.

```
bandit19@bandit:~$ ./bandit20-do cat /etc/bandit_pass/bandit20
0qXahG8Zj0VMN9Ghs7iOWsCfZyX0UbY0
```

Seviye 20 -> 21

“Homedirectory’de aşağıdakileri yapan bir setuid ikili dosyası vardır:

komut satırı argümanı olarak belirttiğiniz portta localhost’a bir bağlantı kurar. Daha sonra bağlantıdan bir metin satırı okur ve bunu önceki seviyedeki (bandit20) parolayla karşılaştırır. Parola doğruysa, bir sonraki seviyenin (bandit21) parolasını iletir.”

```
bandit20@bandit:~$ echo -n '0qXahG8Zj0VMN9Ghs7iOWsCfZyX0UbY0' | nc -l -p 1234 &
[5] 3835231
bandit20@bandit:~$ ./suconnect 1234
Read: 0qXahG8Zj0VMN9Ghs7iOWsCfZyX0UbY0
Password matches, sending next password
EeoULMCra2q0dSkYj561DX7s1CpBu0Bt
[5] Done
echo -n '0qXahG8Zj0VMN9Ghs7iOWsCfZyX0UbY0' | nc -l -p 1234
```

1. ‘netcat’ kullanarak, gelen bağlantıları dinleyen sunucu modunda bir bağlantı oluşturabiliriz. Netcat’in şifreyi göndermesi için echo komutunu kullanıp netcat’e yönlendiriyoruz. -n bayrağı girdide yeni satır karakterlerini önlemek içindir. Son olarak, işlemi arka planda & ile çalıştırıyoruz.

2. Setuid ikili dosyasını port 1234 ile çalıştırmak, netcat sunucumuza bağlanmasını sağlar, echo ile gönderilen şifreyi alır ve bir sonraki şifreyi geri gönderir.

NOT: “setuid” (set user ID) bir Unix/Linux özelliğidir ve bir dosyanın çalıştırılabilir dosya olarak işaretlendiğinde belirli bir kullanıcının kimliğini almasını sağlar. Bu özellikle sistem düzeyindeki işlemler için güçlü bir araçtır çünkü normal kullanıcılar için kısıtlı erişim sağlayan bir programın, belirli bir süper kullanıcı (root gibi) kimliğiyle çalışmasına izin verir.

Bir dosya setuid olarak işaretlendiğinde, bu dosya normalde o dosyayı çalıştıran kullanıcının kimliğini alacak şekilde çalışır. Örneğin, bir dosya root kullanıcısı tarafından setuid olarak ayarlandıysa, herhangi bir kullanıcı bu dosyayı çalıştırdığında o dosyanın root haklarına sahip olacaktır.

Bu özellik, sistem yöneticileri tarafından özenle kullanılmalıdır çünkü kötü amaçlı kullanımı, sistem güvenliğini ciddi şekilde tehlikeye atabilir.

Seviye 21 -> 22

*“Zamana dayalı iş planlayıcısı **cron**’dan düzenli aralıklarla otomatik olarak bir program çalışıyor . Yapılandırma için **/etc/cron.d/** dosyasına bakın ve hangi komutun yürütüldüğünü görün.”*

Cron, Unix ve Unix benzeri işletim sistemlerinde zamanlanmış görevlerin otomatik olarak çalıştırılmasını sağlayan bir zaman tabanlı iş planlayıcıdır.

Cron görevleri, kullanıcıların veya sistem yöneticilerinin belirlediği zaman dilimlerinde (günün belirli saatleri, günler, haftalar veya aylar gibi) otomatik olarak çalışacak komutları veya scriptleri tanımlamak için kullanılır. Bu görevler genellikle belirli işlemlerin periyodik olarak veya düzenli aralıklarla gerçekleştirilmesini sağlar.

*Cron, genellikle sistemdeki birden fazla kullanıcı veya sistem hizmetleri için zamanlanmış görevlerin yönetimi için kullanılır. Kullanıcılar, kendi kullanıcı hesapları altında çalışacak görevleri tanımlayabilir veya sistem yöneticileri, sistem düzeyindeki görevleri yönetmek için genellikle **/etc/crontab** veya **/etc/cron.d/** gibi sistem düzeyindeki cron dosyalarını kullanır.*

Öncelikle ‘/etc/cron.d’ klasöründe ne olduğuna bakıyoruz. Özellikle bu seviye için ‘cronjob_bandit22’ cronjob’ına baktım.


```
bandit21@bandit:~$ ls -la /etc/cron.d
total 44
drwxr-xr-x  2 root root  4096 Jun 20 04:08 .
drwxr-xr-x 121 root root 12288 Jun 20 21:05 ..
-rw-r--r--  1 root root   120 Jun 20 04:07 cronjob_bandit22
-rw-r--r--  1 root root   122 Jun 20 04:07 cronjob_bandit23
-rw-r--r--  1 root root   120 Jun 20 04:07 cronjob_bandit24
-rw-r--r--  1 root root   201 Apr  8 14:38 e2scrub_all
-rwx-----  1 root root    52 Jun 20 04:08 otw-tmp-dir
-rw-r--r--  1 root root   102 Mar 31 00:06 .placeholder
-rw-r--r--  1 root root   396 Jan  9 20:31 sysstat
bandit21@bandit:~$ cat /etc/cron.d/cronjob_bandit22
@reboot bandit22 /usr/bin/cronjob_bandit22.sh &> /dev/null
* * * * * bandit22 /usr/bin/cronjob_bandit22.sh &> /dev/null
```

Bandit22 için cron job'a baktığımızda her saniye çalıştırılan bir Shell scriptinin olduğunu görüyoruz.

Bu cronjob, /usr/bin/cronjob_bandit22.sh dosyası bandit22 kullanıcısı olarak çalıştırır. Beş yıldız; her gün, her dakika çalıştırıldığını gösterir.

Tam olarak neyin yürütüldüğünü bilmek için bash dosyasına bir göz atmalıyız.

Script dosyasına baktığımızda, /tmp dizininde t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv adında bir dosya oluşturduğunu ve sonra bir sonraki seviyenin şifresini o dosyaya kaydettiğini görüyoruz.

Komut dosyası tarafından oluşturulan dosyanın içeriğini cat ile görüntüleyelim.

```
bandit21@bandit:~$ cat /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
```

```
bandit21@bandit:~$ cat /usr/bin/cronjob_bandit22.sh
#!/bin/bash
chmod 644 /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
cat /etc/bandit_pass/bandit22 > /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
bandit21@bandit:~$ cat /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
tRae0UfB9v0UzbCdn9cY0gQnds9GF58Q
```

Seviye 22 -> 23

“Bir program , zaman tabanlı iş zamanlayıcısı olan **cron**’dan düzenli aralıklarla otomatik olarak çalışıyor. Yapılandırma için **/etc/cron.d/** dosyasına bakın ve hangi komutun yürütüldüğünü görün.”

“**NOT:** Başkaları tarafından yazılan shell script dosyalarına bakmak çok faydalı bir beceridir. Bu seviyenin senaryosu özellikle okunması kolay hale getirilmiştir. Ne yaptığını anlamakta

sorun yaşıyorsanız, yazdırdığı hata ayıklama bilgilerini görmek için uygulamayı çalıştırmayı deneyin.”

```
bandit22@bandit:~$ ls -la /etc/cron.d/
total 44
drwxr-xr-x  2 root root  4096 Jun 20 04:08 .
drwxr-xr-x 121 root root 12288 Jun 20 21:05 ..
-rw-r--r--  1 root root   120 Jun 20 04:07 cronjob_bandit22
-rw-r--r--  1 root root   122 Jun 20 04:07 cronjob_bandit23
-rw-r--r--  1 root root   120 Jun 20 04:07 cronjob_bandit24
-rw-r--r--  1 root root   201 Apr  8 14:38 e2scrub_all
-rwx-----  1 root root    52 Jun 20 04:08 otw-tmp-dir
-rw-r--r--  1 root root   102 Mar 31 00:06 .placeholder
-rw-r--r--  1 root root   396 Jan  9 20:31 sysstat
bandit22@bandit:~$ cat /etc/cron.d/cronjob_bandit23
@reboot bandit23 /usr/bin/cronjob_bandit23.sh &> /dev/null
* * * * * bandit23 /usr/bin/cronjob_bandit23.sh &> /dev/null
bandit22@bandit:~$ cat /usr/bin/cronjob_bandit23.sh
#!/bin/bash

myname=$(whoami)
mytarget=$(echo I am user $myname | md5sum | cut -d ' ' -f 1)

echo "Copying passwordfile /etc/bandit_pass/$myname to /tmp/$mytarget"
cat /etc/bandit_pass/$myname > /tmp/$mytarget
```

Buraya kadar olan komutlar bir önceki bölümle aynı. Son komutun çıktısında ise;

‘/usr/bin/cronjob_bandit23.sh’ script dosyasına baktığımızda son satır, 22. seviyeye benzer bir yapıdadır. Bu script sadece değişkenler tanımlar. İlk değişken olan ‘myname’, whoami komutunun çıktısını alıp kaydeder. Bu script bandit23 kullanıcısı olarak çalıştırılacak, bu nedenle whoami komutu ‘bandit23’ çıktısını verir. Son satır, bandit23 şifresinin ‘/tmp’ dizininde bir dosyaya yazılacağını belirtir. Dosya adı, echo “I am user \$myname” | md5sum | cut -d ‘ ‘ -f 1 komutu ile oluşturulur. Tek yapmamız gereken \$myname değişkenini bandit23 olarak yerine koyup komutu çalıştırmak ve sonuç dosya adıdır.

```
bandit22@bandit:~$ echo I am user bandit23 | md5sum | cut -d ' ' -f 1
8ca319486bfbbc3663ea0fbe81326349
bandit22@bandit:~$ cat /tmp/8ca319486bfbbc3663ea0fbe81326349
0Zf11ioIjMVN551jX3CmStKLYqjk54Ga
```

Seviye 23 -> 24

“Zamana dayalı iş planlayıcısı **cron’dan** düzenli aralıklarla otomatik olarak bir program çalışıyor. Yapılandırma için **/etc/cron.d/** dosyasına bakın ve hangi komutun yürütüldüğünü görün.”

“NOT: Bu seviye, kendi ilk shell script dosyanızı oluşturmanızı gerektirir. Bu çok büyük bir adım ve bu seviyeyi geçtiğinizde kendinizle gurur duymalısınız!”

“NOT 2: Shell script’in yürütüldükten sonra kaldırıldığını unutmayın, bu nedenle bir kopyasını yanınızda tutmak isteyebilirsiniz...”

```
bandit23@bandit:~$ ls -la /etc/cron.d
total 44
drwxr-xr-x  2 root root  4096 Jun 20 04:08 .
drwxr-xr-x 121 root root 12288 Jun 20 21:05 ..
-rw-r--r--  1 root root   120 Jun 20 04:07 cronjob_bandit22
-rw-r--r--  1 root root   122 Jun 20 04:07 cronjob_bandit23
-rw-r--r--  1 root root   120 Jun 20 04:07 cronjob_bandit24
-rw-r--r--  1 root root   201 Apr  8 14:38 e2scrub_all
-rwx-----  1 root root    52 Jun 20 04:08 otw-tmp-dir
-rw-r--r--  1 root root   102 Mar 31 00:06 .placeholder
-rw-r--r--  1 root root   396 Jan  9 20:31 sysstat
bandit23@bandit:~$ cat /etc/cron.d/cronjob_bandit24
@reboot bandit24 /usr/bin/cronjob_bandit24.sh &> /dev/null
* * * * * bandit24 /usr/bin/cronjob_bandit24.sh &> /dev/null
```

Buralar ve hemen bi sonraki komut önceki bölümlerle aynı.

```
bandit23@bandit:~$ cat /usr/bin/cronjob_bandit24.sh
#!/bin/bash

myname=$(whoami)

cd /var/spool/$myname/foo
echo "Executing and deleting all scripts in /var/spool/$myname/foo:"
for i in * .*;
do
    if [ "$i" != "." -a "$i" != ".." ];
    then
        echo "Handling $i"
        owner="$(stat --format "%U" ./$i)"
        if [ "${owner}" = "bandit23" ]; then
            timeout -s 9 60 ./$i
        fi
        rm -f ./$i
    fi
done

bandit23@bandit:~$ mktemp -d
/tmp/tmp.up7rEQuI8s
bandit23@bandit:~$ cd /tmp/tmp.up7rEQuI8s
bandit23@bandit:/tmp/tmp.up7rEQuI8s$ nano bandit24_pass.sh
```

mktemp -d

İlk olarak geçici bir dizin oluşturmak için mktemp -d komutunu yazdık.

Script dosyalarında veya komut satırı aracılığıyla geçici dosyalar veya dizinler oluşturmanın güvenli bir yöntemi, mktemp -d komutunu kullanmaktır. Bu yöntem, geçici dosya veya dizin oluştururken çakışma riskini minimize eder ve güvenliği artırır.

Sonra cd ile dizine gittik.

cd /tmp/tmp.up7rEQuI8s

Şu anda bandit23 kullanıcısı olarak giriş yaptığımız için, bandit24 şifresini bize verecek bir script oluşturabiliriz. Daha sonra dosyayı '/var/spool/bandit24/foo' klasörüne taşıyacağız.

Nano'nun içine aşağıdaki scripti yazdık.

```
#!/bin/bash
cat /etc/bandit_pass/bandit24 > /tmp/tmp.up7rEQuI8s/password
```

Artık yalnızca klasöre ve dosyaya gerekli izinleri vermeniz gerekiyor.


```
bandit23@bandit:/tmp/tmp.up7rEQuI8s$ chmod +rx bandit24_pass.sh
bandit23@bandit:/tmp/tmp.up7rEQuI8s$ chmod 777 /tmp/tmp.up7rEQuI8s
bandit23@bandit:/tmp/tmp.up7rEQuI8s$ touch password
bandit23@bandit:/tmp/tmp.up7rEQuI8s$ chmod +rwx password
bandit23@bandit:/tmp/tmp.up7rEQuI8s$ ls -la
total 256
drwxrwxrwx  2 bandit23 bandit23  4096 Jun 26 20:44 .
drwxrwx-wt 5980 root      root    249856 Jun 26 20:44 ..
-rwxrwxr-x  1 bandit23 bandit23    73 Jun 26 20:43 bandit24_pass.sh
-rwxrwxr-x  1 bandit23 bandit23     0 Jun 26 20:44 password
```

Son olarak dosyayı doğru klasöre taşıyalım.

```
cp bandit24_pass.sh /var/spool/bandit24/foo
```

Ve cat ile password'u okuyorum.

```
bandit23@bandit:/tmp/tmp.up7rEQuI8s$ cp bandit24_pass.sh /var/spool/bandit24/foo
bandit23@bandit:/tmp/tmp.up7rEQuI8s$ cat password
gb8KRRcSshuZXI0tUuR6ypOFjiZbf3G8
```

Seviye 24 -> 25

“Bir daemon 30002 portunu dinliyor ve bandit24 için şifre ve gizli bir sayısal 4 haneli pin kodu verildiğinde bandit25 için şifreyi size verecek. Brute-force adı verilen 10000 kombinasyonun hepsini denemek dışında pin kodunu almanın bir yolu yok. Her seferinde yeni bağlantılar oluşturmanıza gerek yok.”

Öncelikle nc komutu ile localhost'ta 30002 numaralı porta bir bağlantı kurmayı denedim.

```
bandit24@bandit:/tmp/tmp.BFP8xWV9ok$ nc localhost 30002
I am the pincode checker for user bandit25. Please enter the password for user bandit24 and the secret pincode on a single line, separated by a space.
```

Sonra önceki seviyede de yaptığım gibi geçici bir dizin oluşturdum. Daha sonra nano içine:

```
#!/bin/bash
```

```
for i in {0000..9999}
```

```
do
```

```
    echo gb8KRRcSshuZXI0tUuR6ypOFjiZbf3G8 $i >> ihtimaller.txt
```

```
done
```

```
cat ihtimaller.txt | nc localhost 30002 > sonuc.txt
```


İlk olarak olası tüm pin kodları üzerinde for döngüsü kullanıldı. Her olasılık 'ihtimaller.txt' adlı bir dosyaya eklenir. İkinci bölümde olasılıklar daemon'a gönderilir. Çıktı 'sonuc.txt' adlı bir dosyaya kaydedilir.

Artık script programlandığına göre onu kaydetmemiz gerekiyor. Bir klasör oluşturarak scripti içine kaydediyoruz ve çalıştırma izni vererek çalıştırıyoruz.

Tüm farklı pin kodu olasılıklarının netcat'e gönderilmesinden itibaren tüm yanıtların çıktısını içerecektir. Yani artık yalnızca pin kodunun doğru olduğu yanıt filtrelememiz gerekiyor.

Bunu ilk koşudan öğrendiklerimizle yapabiliriz. Grep kullanıyoruz ve 'Wrong!' ile tüm satırları filtreliyoruz. Sonuç olarak, giriş ve yalnızca doğru pin koduyla üretilen satır döndürülmelidir.

```
bandit24@bandit:~$ mkdir -p /tmp/tmp.BFP8xWV9ok
bandit24@bandit:~$ cd /tmp/tmp.BFP8xWV9ok
bandit24@bandit:/tmp/tmp.BFP8xWV9ok$ nano bf_pin.sh
Unable to create directory /home/bandit24/.local/share/nano/: No such file or directory
It is required for saving/loading search history or cursor positions.

bandit24@bandit:/tmp/tmp.BFP8xWV9ok$ chmod +x bf_pin.sh
bandit24@bandit:/tmp/tmp.BFP8xWV9ok$ ./bf_pin.sh
bandit24@bandit:/tmp/tmp.BFP8xWV9ok$ ls
bf_pin.sh ihtimaller.txt sonuc.txt
bandit24@bandit:/tmp/tmp.BFP8xWV9ok$ sort sonuc.txt | grep -v "Wrong!"

Correct!
I am the pincode checker for user bandit25. Please enter the password for user bandit24 and the secret pincode on a single line, separated by a space.
The password of user bandit25 is iCi86ttT4KSNeIarmKiwbQNmB3YJP3q4
```