



OverTheWire — Bandit (Writeup)

BEU CYBER TEAM
DUYGU KAÇAR | SUDENUR MURATOĞULLARI

<https://www.linkedin.com/in/sudenur-murato%C4%9Fullar%C4%B1-7b1406223/>

<https://www.linkedin.com/in/duygu-ka%C3%A7ar-b82888254>

<https://www.linkedin.com/company/beu-cyber/>

İçindekiler

OVERTHEWIRE — BANDİT (WRİTEUP)	4
LİNUX DOSYA SİSTEMİ DİZİNLERİ	5
NAVİGASYON KOMUTLARI	6
pwd komutu	6
cd komutu	6
ls komutu	6
file komutu	7
mv komutu	7
cp komutu	7
find komutu	8
YAZI MANİPÜLASYON	9
vi komutu	9
cat komutu	9
tr komutu	9
strings komutu	10
“-“isimli dosya okuma işlemi	10
grep komutu	11
sort komutu	11
uniq komutu	11
xxd komutu	12
base64 komutu	12
rot-13 ve bash yoluyla kullanımı	13
BİLGİ EDİNME KOMUTLARI	13
id komutu	13
more komutu	13
man komutu	14
du komutu	14
mkdir komutu	14
Unix 'job control' (bg, fg, jobs, &, CTRL-Z, ...)	14
GİT	15
DİĞER KOMUT VE SERVİSLERİ	25
SSH NEDİR VE NASIL KULLANILIR	25
Telnet ve SSH FARKI nedir?	25
Suid bit	26

chmod	27
openssl	29
s_client	30
TELNET	30
netcat	31
nmap	32
tmux	32
cron	33
bash	34
screen	35
tar komutu	36
bzip2 komutu	36
gzip komutu	37
BANDİT SORU ÇÖZÜMLERİ	37
Bandit Seviye 0	37
Bandit Seviye 0 → Seviye 1	38
Bandit Seviye 1 → Seviye 2	39
Bandit Seviye 2 → Seviye 3	39
Bandit Seviye 3 → Seviye 4	40
Bandit Seviye 4 → Seviye 5	40
Bandit Seviye 5 → Seviye 6	41
Bandit Seviye 6 → Seviye 7	42
Bandit Seviye 7 → Seviye 8	44
Bandit Seviye 8 → Seviye 9	44
Bandit Seviye 9 → Seviye 10	45
Bandit Seviye 10 → Seviye 11	45
Bandit Seviye 11 → Seviye 12	46
Bandit Seviye 12 → Seviye 13	47
Bandit Seviye 13 → Seviye 14	50
Bandit Seviye 14 → Seviye 15	52
Bandit Seviye 15 → Seviye 16	53
Bandit Seviye 16 → Seviye 17	54
Bandit Seviye 17 → Seviye 18	56
Bandit Seviye 18 → Seviye 19	57

Bandit Seviye 19 → Seviye 20.....	58
Bandit Seviye 20 → Seviye 21.....	59
Bandit Seviye 21 → Seviye 22	60
Bandit Seviye 22 → Seviye 23.....	61
Bandit Seviye 23 → Seviye 24	63
Bandit Seviye 24 → Seviye 25	65
Bandit Seviye 25 → Seviye 26	66
Bandit Seviye 26 → Seviye 27	70
Bandit Seviye 27 → Seviye 28	70
Bandit Seviye 28 → Seviye 29	71
Bandit Seviye 29 → Seviye 30	74
Bandit Seviye 30 → Seviye 31.....	77
Bandit Seviye 31 → Seviye 32	78
Bandit Seviye 32 → Seviye 33.....	80

OVERTHEWIRE — BANDIT (WRİTEUP)

OVER_THE_WIRE platformunda, siber güvenlik konusunda eğitim amacıyla düzenlenen bir platformdur. Bu platformda yer alan **BANDIT** oyunu, Linux komut satırı hakkında temel bilgi sahibi olmak isteyenler için tasarlanmış bir oyundur.

BANDIT oyununda, oyuncular birçok farklı seviyeden oluşan bir labirentte ilerleyerek, her seviyede belirli bir amacı tamamlamaya çalışırlar. Seviyeler, sırasıyla daha zor hale gelir ve her seviye, oyuncuların Linux komut satırı hakkında yeni bilgiler edinmelerine yardımcı olur.

Her seviye, kullanıcılarından gizlenen bir şifre içerir ve oyuncuların bu şifreyi bulmaları ve sonraki seviyeye geçmeleri gerekmektedir. Ayrıca, her seviyede, kullanıcılar yeni Linux komutlarını ve özelliklerini öğrenmeleri için farklı senaryolar sunulur.

BANDIT oyunu hem yeni başlayanlar hem de deneyimli Linux kullanıcıları için faydalı bir kaynak olabilir. Yeni başlayanlar, oyunda öğrendikleri bilgileri gerçek hayatı kullanarak Linux işletim sistemine daha iyi hâkim olabilirler. Deneyimli kullanıcılar ise, oyundaki daha ileri seviyelerde karşılaştıkları zorluklar sayesinde Linux komut satırı hakkında daha fazla bilgi edinebilirler.

Bu nedenle, **BANDIT** oyunu, siber güvenlik ve Linux komut satırı hakkında bilgi edinmek isteyen herkes için faydalı bir kaynak olabilir. Ayrıca, **OVER_THE_WIRE** platformunda yer alan diğer oyunlar da siber güvenlik konusunda kendini geliştirmek isteyenler için öğretici ve eğlenceli bir yol sunar.

...

İçeriğindeki komutları da ayrıntılı bir şekilde anlattık, keyifli ve verimli okumalar dileriz.

LINUX DOSYA SİSTEMİ DİZİNLERİ

/: Kök dizindir.

/bin: Yürütilebilir dosyaların bulunduğu yerdir. Tüm kullanıcılar tarafından kullanılabilir.

/boot: Önyükleyici ve önyükleme dosyalarının bulunduğu yerdir. Sistemi başlatmak için gerekli dosyalar burada bulunur.

/dev: Tüm fiziksel aygıtların bağlandığı yerdir, USB ve CD gibi aygıtlar burada bulunur.

/etc: Sistemde kurulu olan yazılımların konfigürasyon dosyalarının bulunduğu yerdir.

/home: Kullanıcıların kişisel dosyalarının bulunduğu dizindir, her kullanıcı için ayrı bir dizin bulunur.

/lib: Paylaşılan kütüphaneler ve bazen çekirdekle ilgili dosyaların bulunduğu dizindir, uygulamalar tarafından kullanılan kodları içeren dosyalar burada bulunur.

/media: Harici aygıtların ve dosyalarının bulunduğu dizindir, CD ve USB bellekler gibi aygıtlara buradan erişebilirsiniz.

/mnt: Manuel olarak bağlanan depolama aygıtlarının veya bölümlerinin bulunduğu dizindir, günümüzde çok fazla kullanılmaz. CD-ROM ve disket sürücüsü gibi aygıtlara buradan erişebilirsiniz.

/opt: İsteğe bağlı olarak yüklenebilen yazılımların bulunduğu dizindir, paket yöneticisi tarafından yönetilir.

/proc: Sanal bir dizindir, bilgisayar hakkında ve Linux sistemi hakkında bilgiler içerir. Dosyalar ve dizinler sistem çalışırken veya başlamadan önce oluşturulur.

/root: Kök (yönetici) kullanıcısının ana dizinidir.

/run: Geçici verilerin depolanması için kullanılır.

/sbin: Genellikle sistem yönetimi için yürütülebilir dosyalar içerir. Sadece kök kullanıcı için erişilebilir yükleme, silme ve biçimlendirme yapabilen araçları içerir.

/tmp: Sistem önyüklemleri arasında kullanılan geçici dosyaların bulunduğu yerdir. Dosyalar ve dizinler, uygulamanın şu anda ihtiyaç duymadığı ama daha sonra ihtiyaç duyabileceği verileri içerebilir.

/usr: Paylaşılan yardımcı programlar ve dosyaların bulunduğu Linux yoludur. /usr uygulamalar, kitaplıklar, belgeler, duvar kağıtları, simgeler ve uygulamalar ve hizmetler tarafından paylaşılan verileri içeren dizinler dizisini içerir.

/var: Sistem günlükleri ve değişken verilerin bulunduğu yerdir. Genellikle değişken uzunlukta dosyalar (örneğin, günlük dosyaları) ve diğer verilerin bulunduğu dosya türleri içerir.

/srv: Sunucularla ilgili verilerin bulunduğu yerdir. Örneğin, bir web sunucusu çalıştırılırsa, HTML dosyaları /srv/http (veya /srv/www) yerine yerleşir. FTP sunucusu çalıştırılırsa, dosyalar /srv/ftp yerine yerleşir.

/sys: Başka bir sanal dizindir (örneğin, /proc ve /dev gibi) ve bilgisayara bağlı cihazlardan gelen bilgileri içerir.

/kernel: Çekirdek dosyalarının bulunduğu yerdir.

NAVİGASYON KOMUTLARI

pwd komutu

Bu komut, kullanıcının bulunduğu dizinin tam yolunu ekrana yazdırır.

cd komutu

"cd" komutu kullanıcının bulunduğu dizini değiştirmesine olanak tanır.

Örnek kullanım:

```
$ cd /home/user/documents
```

Bu komut, "/home/user/documents" dizinine geçiş yapar ve bulunduğuuz dizin "/home/user/documents" dizinine değiştir.

Ayrıca "cd .." komutu ile bulunduğuuz dizinin bir alt dizinine geçiş yapabilirsiniz.

Örnek kullanım:

```
$ cd ..
```

ls komutu

"ls" komutu, kullanıcının bulunduğu dizindeki dosyalar ve klasörlerin listesini verir ve bu dosyaların adı, dosya tipi, boyutu, oluşturulma tarihi gibi bilgilerini görüntüler.

"ls" komutu, farklı seçenekler ile kullanılabildiği gibi, basit bir şekilde de kullanılabilir.

Örneğin:

```
ls
```

Bu komut, bulunduğuuz dizindeki dosyaların ve klasörlerin listesini verir.

```
ls -l
```

Bu komut, bulunduğunuz dizindeki dosyaların ve klasörlerin detaylı bir listesini verir ve dosya tipi, boyutu, tarihi, sahibi gibi bilgileri içerir.

```
$ ls -la
```

"ls -l" komutunun bir alternatifi olarak, "ls -la" komutu, dizindeki tüm dosyaların ve klasörlerin detaylı bir listesini verir ve gizli dosyaları da dahil eder.

file komutu

"file" komutu dosyaların türünü belirlemek için kullanılır. Komut, dosya içeriğine bakmaksızın, dosyanın ne olduğunu tahmin eder ve dosya türü hakkında bilgi verir (örneğin, metin dosyası, resim dosyası, ses dosyası, vb.).

Örnek kullanım:

```
$ file picture.jpg
picture.jpg: JPEG image data, JFIF standard 1.01, resolution (DPI), density
```

Bu komut, "picture.jpg" isimli dosyanın bir JPEG resim dosyası olduğunu belirler.

mv komutu

mv komutu dosya ve dizinlerin adlarının değiştirilmesini veya başka bir dizine taşınmasını sağlar. Kullanım şekli şu şekildedir:

```
mv [kaynak dosya/dizin] [hedef dosya/dizin]
```

Örnek kullanım:

```
mv file1.txt ~/documents/file1_backup.txt
```

Bu komut "file1.txt" dosyasının adını "file1_backup.txt" olarak "~/documents/" dizinine taşır.

cp komutu

"cp" komutu dosya ve dizinlerin kopyalanmasını sağlar. Kullanım şekli şu şekildedir:

```
cp [kaynak dosya/dizin] [hedef dosya/dizin]
```

[kaynak dosya/dizin]: Kopyalanmak istenen dosya veya dizinin yolu.

[hedef dosya/dizin]: Kopyalanan dosyanın veya dizinin yeni yeri.

```
cp file1.txt ~/documents/
```

Bu komut "file1.txt" dosyasını "~/documents/" dizinine kopyalar.

find komutu

"find" komutu belirtilen bir dizin veya dosya sistemi altındaki dosyaları ve dizinleri aramaya yarar. Komut, dosya adı, dosya tarihi, dosya tipi, dosya boyutu gibi çeşitli ölçütlerde göre dosya ve dizinleri arayabilir.

Örnek kullanım:

```
$ find /path/to/directory -name "*.txt"  
/path/to/directory/file1.txt  
/path/to/directory/subdirectory/file2.txt
```

Bu komut, "/path/to/directory" dizininde ve alt dizinlerinde ".txt" uzantısına sahip dosyaları bulur.

"find" komutunun genel söz dizimi şu şekildedir:

```
find [başlangıç dizini] [seçenekler] [aramak istenen dosya adı/özelliği]
```

Örneğin, "~/Documents" dizininde tüm dosyaları listelemek için şu kod kullanılabilir:

```
find ~/Documents
```

"find" komutunun birçok seçeneği bulunmaktadır, bazı yaygın olarak kullanılanlar şunlardır:

-name: Dosya adına göre arama yapar

-type: Dosya türüne göre arama yapar (örneğin, dizin, dosya, bağlantı, gibi)

-mtime: Dosyanın değiştirilme tarihine göre arama yapar

-size: Dosya boyutuna göre arama yapar

-iname: Dosya adına göre arama yapar, ancak büyük/küçük harf duyarlığını atlar

Örnek olarak, "~/Documents" dizininde ".txt" uzantılı dosyaları bulmak için şu kod kullanılabilir:

```
find ~/Documents -name "*.txt"
```

YAZI MANİPÜLASYON

vi komutu

“vi” Unix/Linux işletim sistemlerinde kullanılan bir metin editörüdür. Kullanımı, komut satırından yapılır; dosyaları açmak, değiştirmek ve kaydetmek için kullanılabilir.

Aşağıdaki örnek, vi kullanarak bir dosya oluşturmayı ve değiştirmeyi gösterir:

1. Terminalden "vi test.txt" komutunu çalıştırın.
2. Girdi modunu açın ve "Hello World!" metnini girin.
3. "Esc" tuşuna basın ve ":wq" komutunu girin.
4. Terminalden "cat test.txt" komutunu çalıştırarak dosyanın içeriğini görüntüleyin.

cat komutu

“cat” komutu dosyaların içeriğini ekrana yazdırma olanak tanır.

Örnek kullanım:

```
$ cat file.txt
```

Bu komut, "file.txt" isimli dosyanın içeriğini ekrana yazdırır. Ayrıca, birden fazla dosya ismi verildiğinde, belirtilen dosyaların içeriği sırayla ekrana yazdırılabilir.

Örnek kullanım:

```
$ cat file1.txt file2.txt > output.txt
```

Bu komut, "file1.txt" ve "file2.txt" isimli dosyaların içeriğini sırayla ekrana yazdırır ve sonuçları "output.txt" isimli bir dosyaya yazdırır.

“cat” komutunu kullanırken dosya adındaki boşluklar için tırnak işaretleri kullanabilirsiniz:

```
cat "bu bir dosya"
```

tr komutu

"tr" komutu, girdi olarak verilen metnin belirli karakterlerini diğer karakterlere ya da karakter gruplarına çevirmeye yarar. Bu komut, metnin sadeleştirilmesi, dönüştürülmesi veya karakter düzenlemeleri gibi amaçlar için kullanılabilir.

```
echo "Merhaba Dünya" | tr [:lower:] [:upper:]
```

Bu örnekte, girdi olarak verilen metin, küçük harfleri büyük harflere çevrilir ve ekrana yazdırılır.

```
echo "Hello World" | tr [:space:] \\n
```

Bu örnekte, girdi olarak verilen metindeki boşluklar "\n" (yeni satır) karakterine çevrilir ve ekrana yazdırılır.

"tr" komutu, aşağıdaki gibi seçeneklere de sahiptir:

- d: Karakterleri siler (örneğin, "Hello World" metnindeki boşlukları silmek için).

- s: Tekrar eden karakterleri tek bir karakter olarak çevirir (örneğin, "aaabbccc" metnindeki tekrar eden karakterleri "abc" olarak çevirmek için).

strings komutu

"strings" komutu, belirli bir dosya içerisindeki metin dizelerini bulmaya ve ekrana yazdırılmaya yarar. Bu komut genellikle binary dosyalar (örneğin, .exe, .so, .dll vb.) içindeki metin dizelerini bulmak için kullanılır.

```
strings file.bin
```

Bu örnekte, "file.bin" dosyasındaki metin dizeleri bulunur ve ekrana yazdırılır.

"strings" komutu, aşağıdaki gibi seçeneklere de sahiptir:

- n: Aranacak en kısa metin dizisi uzunluğunu belirtir.

- t: Metin dizelerinin nasıl ekrana yazdırılacağını belirtir (örneğin, 10 ile 16 arasındaki baytların hexadecimal gösterimi).

“-“ isimli dosya okuma işlemi

"-" isimli bir dosya Unix ve Unix benzeri işletim sistemlerinde geçersiz bir dosya adıdır ve kullanıcı tarafından oluşturulduğunda yanlış sonuçlar doğurabilir. Bu nedenle, bu dosya adı kullanılmamalıdır.

Eğer "-" isimli bir dosya okumak istiyorsanız, "cat" komutunun yerine "less" veya "more" gibi diğer dosya görüntüleme komutlarını kullanabilirsiniz. Örnek olarak:

```
$ less -
```

Bu komut, kullanıcı tarafından girdi olarak verilen içeriği görüntüler.

"cat < -" Bu komut, standart girdi (STDIN) kaynağından okunan verileri dosya olarak görüntüler. "-" simbolü, standart girdi kaynağını belirtir ve kullanıcı tarafından girilen verileri okumak için kullanılabilir. Örnek olarak:

```
$ echo "Hello World" | cat < -
Hello World
```

Bu komut, "Hello World" metnini standart girdi kaynağı olarak verir ve "cat" komutu bu veriyi görüntüler.

grep komutu

"grep" komutu, dosyalarda veya girdilerde belirli bir metin dizesini aramaya yarar. Aşağıdaki gibi kullanabilir:

```
grep "pattern" file1.txt file2.txt
```

"pattern" parametresi aranacak metin dizesini, "file1.txt file2.txt" parametreleri ise arama yapılacak dosyaları belirtir. Eğer bir dosya yerine bir girdi verilirse, o girdide "pattern" metnin aranır.

```
echo "hello world" | grep "hello"
```

Bu örnekte, "hello world" girdisi verilmiş ve "hello" metni içerisinde aranmıştır. Grep komutu, aradığı metnin bulunması durumunda, bulunan satırları veya bulunan satırların tamamını ekrana yazdırabilir.

sort komutu

"sort" komutu, verilen girdileri veya dosyaları sıralamaya yarar. Sıralama, varsayılan olarak alfabetik olarak yapılır ancak belirli bir sıralama kriteri belirtilerek farklı sıralamalar da yapılabilir.

Örnek kullanım:

```
sort file.txt
```

Bu örnekte, "file.txt" dosyasının içeriği sıralanır ve sıralı sonuç ekrana yazdırılır.

```
echo "apple\norange\nbanana" | sort
```

Bu örnekte, girdi olarak verilen metin dizesi sıralanır ve sıralı sonuç ekrana yazdırılır.

"sort" komutu, çok sayıda seçenek sunar ve sıralama kriterini belirlemek için kullanılabilir. Örneğin, tarihe göre sıralama yapmak için "-t" ve ":" seçeneği kullanılabilir.

uniq komutu

"uniq" komutu, verilen girdileri veya dosyalardaki tekrar eden satırları silerek tekrar etmeyen satırları göstermeye yarar.

Örnek kullanım:

```
uniq file.txt
```

Bu örnekte, "file.txt" dosyasındaki tekrar eden satırlar silinir ve tekrar etmeyen satırlar ekrana yazdırılır.

```
echo "apple\norange\nbanana\norange" | uniq
```

Bu örnekte, girdi olarak verilen metin dizesindeki tekrar eden satırlar silinir ve tekrar etmeyen satırlar ekrana yazdırılır.

"uniq" komutunun seçenekleri şunlardır:

- c: Tekrar eden satırların sayısını gösterir.
- d: Sadece tekrar eden satırları gösterir.
- u: Sadece tekrar etmeyen satırları gösterir.

Örnek kullanım:

```
uniq -c file.txt
```

Bu örnekte, "file.txt" dosyasındaki tekrar eden satırların sayısı ekrana yazdırılır.

xxd komutu

"xxd" komutu, dosyaların hexadecimal (onaltılık) ve ASCII formatında görüntülenmesini sağlar. Bu komut, dosya içeriğinin incelenmesi ve düzenlenmesi gibi amaçlar için kullanılabilir.

```
xxd file.txt
```

Bu örnekte, "file.txt" dosyası hexadecimal ve ASCII formatında ekranda görüntülenir.

```
xxd -r file.hex
```

Bu örnekte, "file.hex" adlı hexadecimal formatlı dosya, orijinal dosya formatına dönüştürülür ve "file" adıyla kaydedilir. "-r" seçeneği dosyanın hexadecimal formatından orijinal dosya formatına dönüştürülmesini belirtir.

base64 komutu

"base64" komutu, bir dosya veya girdinin içeriğini Base64 kodlamasına çevirmeye veya Base64 kodlamasından orijinal içeriğine çevirmeye yarar. Base64 kodlaması, herhangi bir binary veriyi ASCII karakterleri kullanarak kodlamaya yarar ve veri aktarımı veya depolama sırasında veri bozulmalarını önler.

"base64" komutu, aşağıdaki gibi seçeneklere de sahiptir:

- w: Çıktıyı kaç baytlık bloklara böleceğini belirtir.
- d / --decode: Base64 kodlamasından orijinal içeriğe çevirir.

-e / --encode: Dosya veya girdinin içeriğini Base64 kodlamasına çevirir.

Örnek kullanım:

```
(kali㉿kali)-[~/Desktop]
$ cat dosyaa.txt
c2VsYWFhbW1tIDop

(kali㉿kali)-[~/Desktop]
$ cat dosyaa.txt | base64 -d
selaaammm :)
```

rot-13 ve bash yoluyla kullanımı

"rot-13" (rotate-13), bir şifreleme yöntemidir. Bu yöntem, her bir harfin alfabetik sırasını 13 harf ileri atarak şifrelenmesini sağlar. İşlenen metin, harf sırasına göre simetrik bir şekilde şifrelendiğinden, şifrelenmiş metnin tekrar açılması için aynı işlem ters olarak yapılır.

```
echo "Hello World" | tr '[A-Za-z]' '[N-ZA-Mn-za-m]'
```

Bu örnekte, "Hello World" metni rot-13 şifrelemesi uygulanır ve "Uryyb Jbeyq" olarak görüntülenir.

```
echo "Uryyb Jbeyq" | tr '[N-ZA-Mn-za-m]' '[A-Za-z]'
```

Bu örnekte, "Uryyb Jbeyq" metni rot-13 şifrelemesi çözülür ve "Hello World" olarak görüntülenir.

BİLGİ EDİNME KOMUTLARI

id komutu

"id" komutu, bir kullanıcının kimliği ve grup üyelikleri hakkında bilgi verir. Bu komut, kullanıcının UID (kullanıcı tanımlayıcı) ve GID (grup tanımlayıcı) numaralarını, aynı zamanda bu numaralara atanmış kullanıcı adı ve grup adlarını görüntüler.

Kullanımı: "id [kullanıcı adı]" şeklindedir. Eğer bir kullanıcı adı belirtilmmezse, komut, o anda oturum açmış olan kullanıcının bilgilerini verir.

Örnek:

```
$ id
```

```
uid=1000(user) gid=1000(user)
groups=1000(user),4(adm),24(cdrom),27(sudo),46(plugdev),108(lpadmin),124(sambashare)
```

more komutu

"more" komutu, Linux ve Unix benzeri işletim sistemlerinde bir metin dosyasının içeriğini ekran boyutu kadar sayfa sayfa görüntülemeyi sağlar. More komutunun üstündeki satır sayısı

belirtilebilir ve kullanıcı metin dosyasını bir sonraki sayfaya geçmek için "Enter" tuşuna basabilir veya "q" tuşuna basarak çıkabilir.

man komutu

Bu komut, kullanıcıların sistemde bulunan programlar, fonksiyonlar veya komutlar hakkında bilgi almasına olanak tanır. "man" komutu ile, bir komutun detaylı kullanımı ve özellikleri hakkında bilgi alabilir, parametreleri hakkında bilgi sahibi olabilir ve nasıl kullanılacağını öğrenebilirisiniz.

du komutu

"du" komutu dosya ve dizinlerin disk kullanımını hakkında bilgi verir. Komut, belirtilen dosya veya dizinin diskte kapladığı boyutunu veya belirtilen dosya veya dizinlere ait alt dosya ve dizinlerin diskte kapladığı toplam boyutu görüntüler.

Örnek kullanım:

```
$ du /path/to/directory
456  /path/to/directory/file1
123  /path/to/directory/file2
789  /path/to/directory/subdirectory
1368 /path/to/directory
```

Bu komut, "/path/to/directory" dizinindeki dosya ve dizinlerin toplam disk kullanımını 1368 kilobayt olarak görüntüler.

mkdir komutu

"mkdir" komutu "make directory" (dizin oluştur) anlamına gelir. Bu komut, kullanıcının belirlediği bir dizin ismiyle yeni bir dizin oluşturmasını sağlar.

```
mkdir my_directory
```

"mkdir" komutu, aynı anda birden fazla dizin oluşturmak için de kullanılabilir:

```
mkdir dir1 dir2 dir3
```

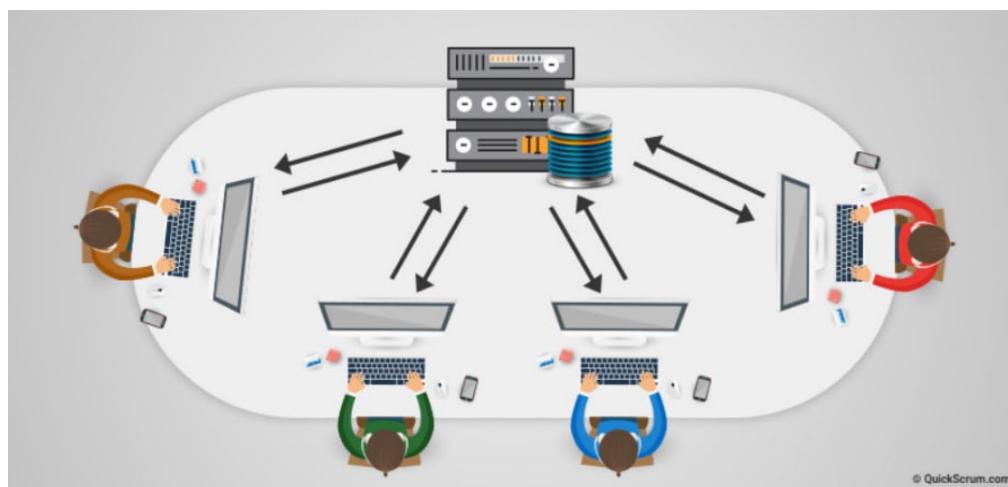
Unix 'job control' (bg, fg, jobs, &, CTRL-Z, ...)

Bir kullanıcının sisteme üzerinde yürüttüğü işlemleri yönetmesine olanak tanır. Aşağıdaki komutlar kullanılır:

1. bg: Arka planda çalışan bir işlemi tekrar çalışmaya başlatır.
2. fg: Arka planda çalışan bir işlemi öne çıkarır ve işlemin ekranını kaplamasına izin verir.
3. jobs: Sistem üzerinde çalışan işlemleri listeler.
4. &: Konsol üzerinde işlem çalıştırır ve konsol ekranından ayrırlır.
5. CTRL-Z: İşlemi durdurur ve arka planda bekletir.

GİT

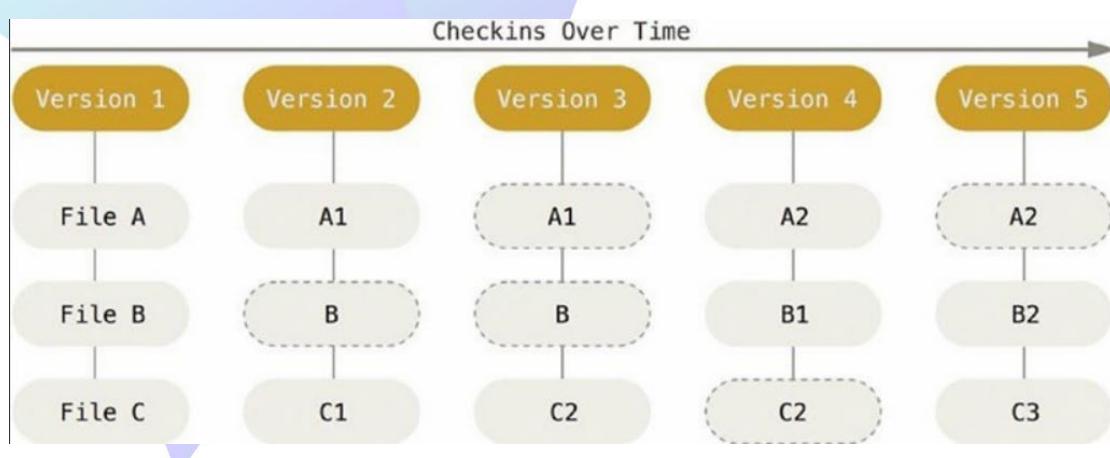
GIT NEDİR VE NASIL ÇALIŞIR



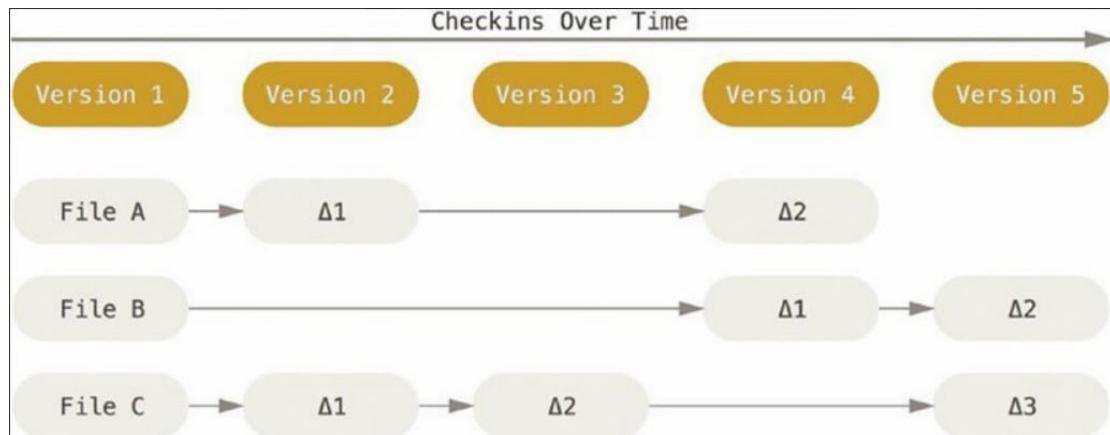
Git, yazılım geliştirme süreçlerinde kullanılan bir versiyon kontrol sistemi (VCS) yazılımıdır. VCS, bir projenin tüm değişikliklerini yönetmeye, izlemeye ve takip etmeye yardımcı olur. Bu, birden fazla kişi tarafından çalışılan projelerde çok önemlidir, çünkü farklı kişilerin farklı değişiklikler yapması ve bunları yönetmesi gerekebilir.

Diğer birçok VCS'ye kıyasla Git'in en büyük farkı, verileri nasıl depoladığıdır. Diğer VCS'ler değişiklikleri dosya tabanlı bir liste olarak depolarken, Git her bir dosyanın tamamını her bir değişiklik için kopyalar ve her bir kopyayı bir versiyon olarak saklar. Bu nedenle Git, diğer VCS'lerden daha hızlı ve daha esnek bir versiyon kontrolü sunar.

GİT



DİĞER VSC'LER



Ayrıca, Git'in dağıtılmış bir yapıya sahip olması da diğer VCS'lerden farklıdır. Her kullanıcının kendi kopyasına sahip olduğu ve değişikliklerin yerel olarak kaydedildiği bir yapıdır. Bu, kullanıcıların çevrimdışı çalışabilmesini, yerel değişiklikler yapabilmesini ve daha sonra ana depoya yeniden entegre edebilmesini sağlar.

Sonuç olarak, Git, hızlı, esnek ve dağıtılmış bir versiyon kontrol sistemi olarak diğer VCS'lerden ayrılır ve yazılım geliştirme süreçlerinde sıkça kullanılır.



Working Directory (Çalışma Alanı): Projenin kaynak dosyalarının bulunduğu alandır. Yapılan değişiklikler burada gerçekleştirilir.

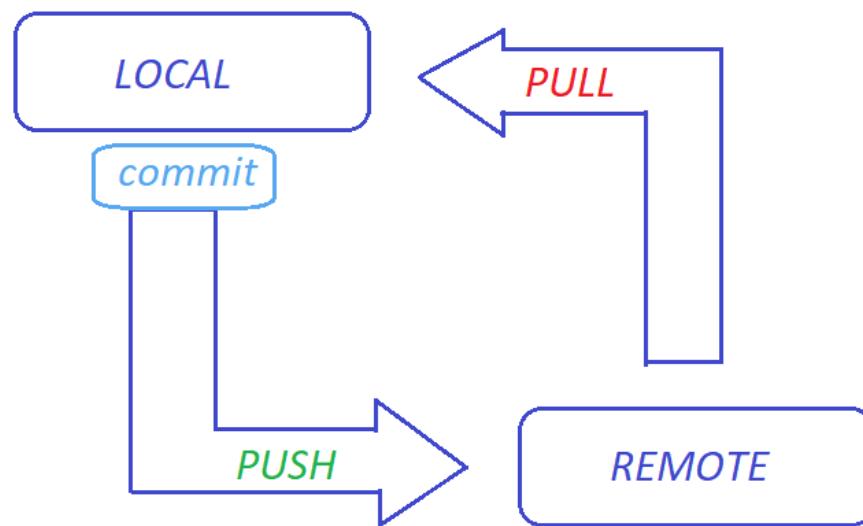
Staging Area (Hazırda Bekleyen Değişiklikler Bölgesi): Değişiklikleri geçici olarak yüklemek için kullanılan bir alandır. Burada, değişikliklerin hangi dosyalarda yapıldığı ve hangi değişikliklerin kaydedileceği seçilir.

Local repository: Git'in yerel bilgisayarda depolanan ve yönetilen projenin tam bir kopyasını içeren veritabanıdır. Bu veritabanı, projenin tüm dosyalarını, geçmişinde yapılan tüm değişiklikleri ve projenin geçmişini takip eden Git'in diğer özelliklerini içerir.

Remote repository: Git'te yerel bilgisayarın dışındaki bir sunucuda depolanan ve yönetilen projenin tam bir kopyasını içeren veritabanıdır. Bu sunucu, projenin tüm

dosyalarını, geçmişinde yapılan tüm değişiklikleri ve projenin geçmişini takip eden Git'in diğer özelliklerini içerir.

GİT KOMUTLARI



Push: Yerel depodaki değişiklikleri uzak depoya yüklemek için kullanılır. Yerelde yapılan değişikliklerin uzak depoda da görüntülenmesini sağlar. Bu işlem sırasında, Git önce yerel değişiklikleri sıkıştırır ve ardından uzak depoya gönderir.

Pull: Uzak depodaki değişiklikleri yerel depoya almak için kullanılır. Uzak depoda yapılan değişikliklerin yerelde de uygulanmasını sağlar. Bu işlem sırasında, Git önce uzak depodan değişiklikleri çeker ve ardından yerel depoda birleştirir.

Commit (Kayıt): Değişikliklerin tarihçesini saklar. Bir commit, bir veya daha fazla dosyada yapılan değişiklikleri temsil eder ve her commit, kim tarafından yapıldığı, ne zaman yapıldığı ve bir commit mesajı gibi diğer meta verileri içerir.

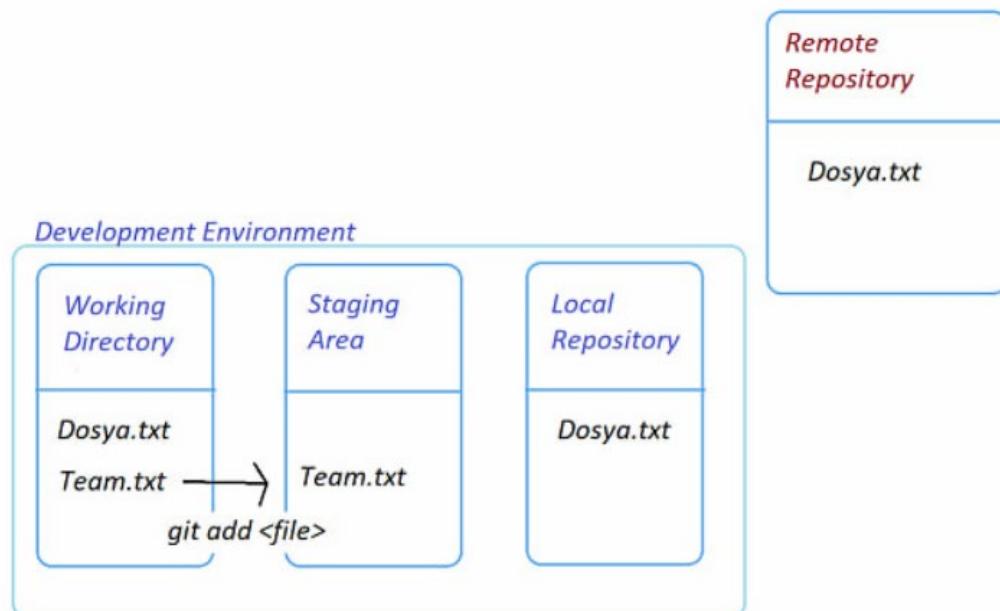
```

commit d020a517758e70b04d569d3e45f7974dff475ff7
Author: Lorna Jane Mitchell <lorna@lornajane.net>
Date:   Wed Aug 24 20:26:29 2016 +0100

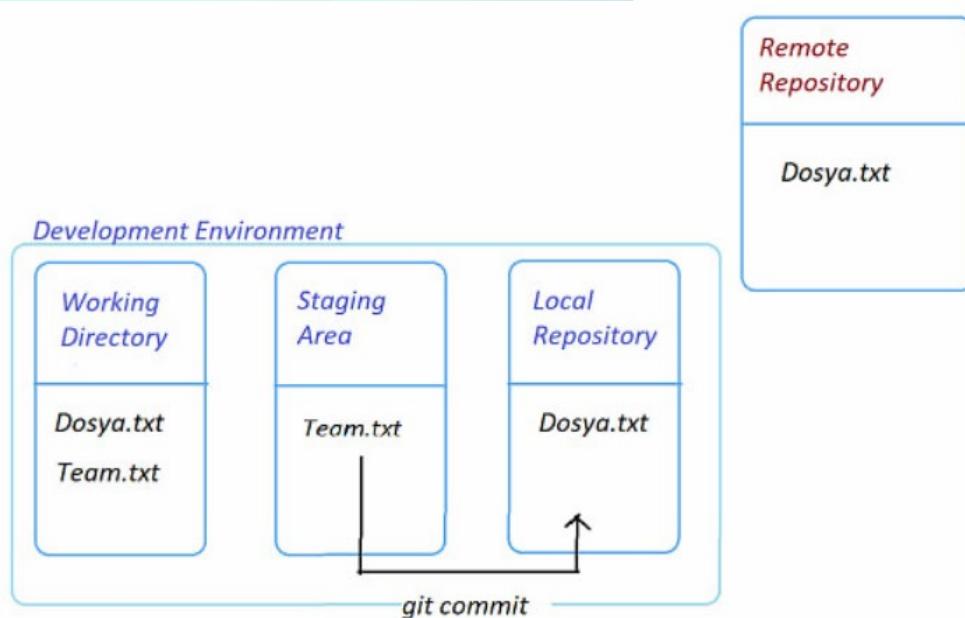
    A random change of 6088 to idea2.txt

commit c3f5f1c25a28941a11bf79b312d934861ceb6549
Author: Lorna Jane Mitchell <lorna@lornajane.net>
Date:   Wed Aug 24 20:26:29 2016 +0100

    A random change of 9844 to ideal.txt
  
```



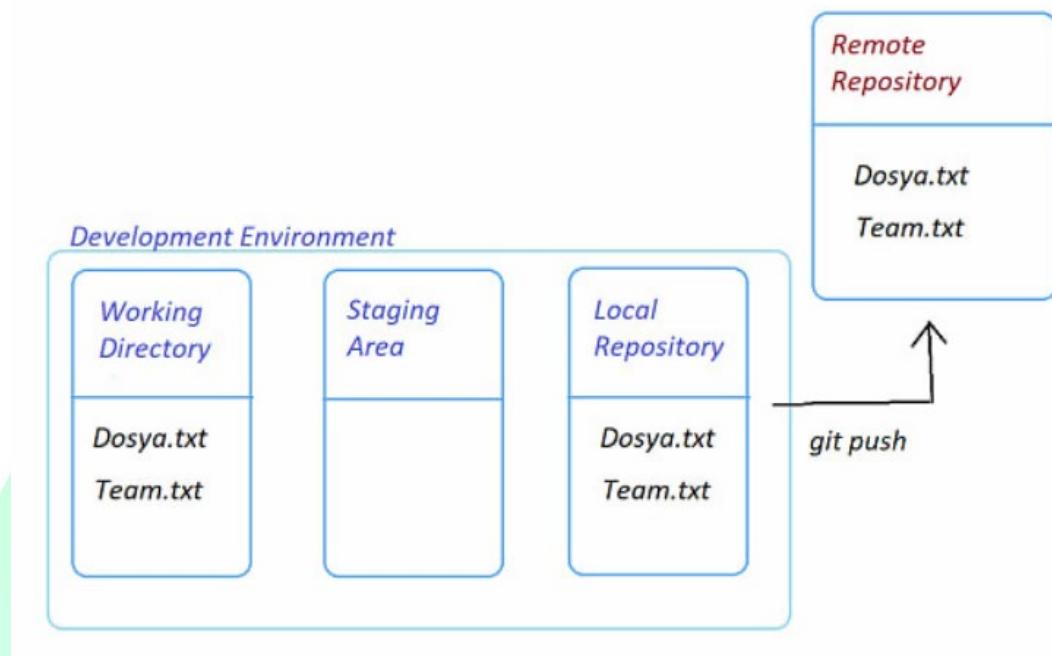
"git add" komutu, Git depolama alanında bulunan değişiklikleri takip etmek istediğiniz dosyaları "staging area" olarak adlandırılan geçici bir alana eklemek için kullanılır. "staging area" değişikliklerin daha sonra "commit" komutu ile kaydedilmesine izin verir.



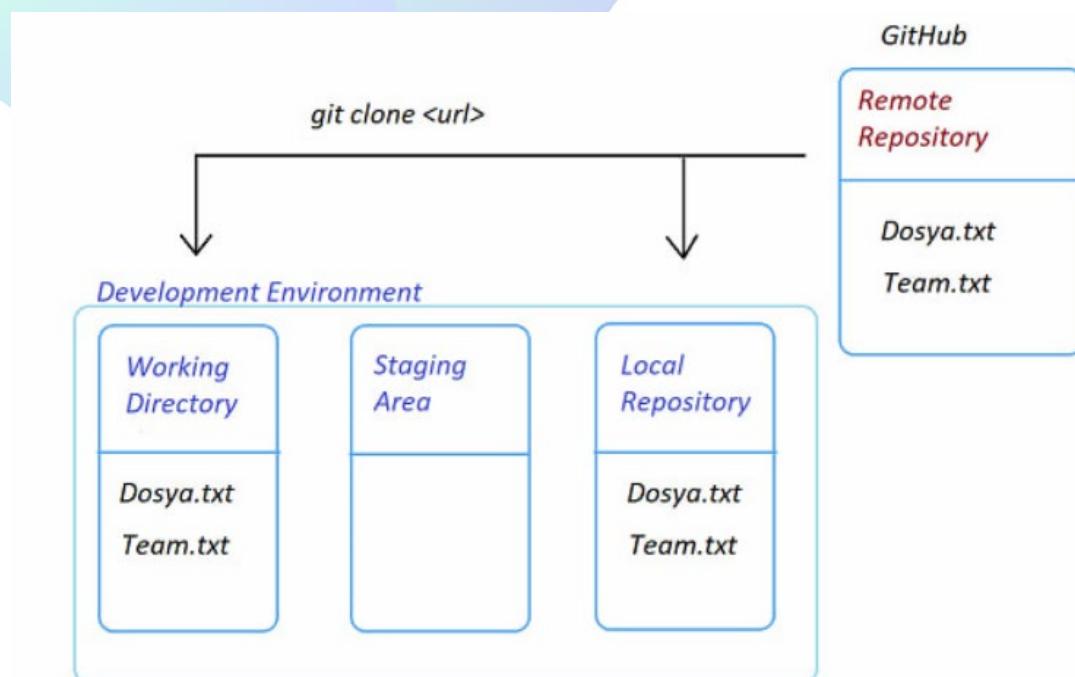
"git commit" komutu, Git depolama alanında bulunan değişiklikleri kalıcı olarak kaydetmek için kullanılır. Bu komut, "staging area" olarak adlandırılan geçici alanda bulunan değişiklikleri bir Git "commit" nesnesi olarak depolama alanında kalıcı hale getirir. Bu, bir projenin belirli bir noktasındaki kodun önceki sürümlerine geri dönebilmenizi sağlar.

```
git commit -m "commit mesajı"
```

"-m" parametresiyle de commiti kaydederken mesaj ekleyebiliriz.



"git push" komutu, yerel bir Git deposundaki değişiklikleri uzak bir Git deposuna göndermek için kullanılır. Bu komut, uzak depoda yer alan bir branche yerel depodaki bir branche eşitlemek için kullanılır.



"git clone" komutu, bir uzak Git deposundan yerel bir kopya oluşturmak için kullanılır. Bu komut, uzak depodaki tüm dosyaları ve geçmişlerini, yerel bir Git deposuna kopyalar.

"git clone" komutunu kullanmak için, aşağıdaki komutu çalıştırmanız yeterlidir:

```
git clone <uzak_depo_adresi>
```

Bu komut, belirtilen uzak Git deposundaki tüm dosyaları ve geçmişleri, yerel bir klasöre kopyalar ve bir yerel Git deposu oluşturur. Bu işlem, bir projenin kodunu indirip yerel olarak çalışmanızı sağlar.

"git log" komutu, Git depolama alanındaki commit geçmişini görüntülemek için kullanılır. Bu komut, Git deposundaki tüm commit'leri listeler ve her commit için aşağıdaki bilgileri gösterir:

commit kimliği, commit tarihi, commit mesajı, commit yazarı

Bu bilgiler, kodun sürüm kontrolünü yönetmek ve projenin geliştirme sürecini izlemek için önemlidir.

```
commit 2622d6439e21c0c24f3d83b27ec8af0c0f12d296
Author: John Doe <john.doe@example.com>
Date:   Tue Feb 15 14:38:23 2022 -0500

    Add new feature

commit 1e9c84e50e2f2a4d40237171f29e1c87ec6377e9
Author: Jane Smith <jane.smith@example.com>
Date:   Mon Feb 14 10:22:45 2022 -0500

    Update documentation

commit 3a1e0c2793175bbfe306aed1551d5f5b5a5c096d
Author: John Doe <john.doe@example.com>
Date:   Sun Feb 13 18:07:31 2022 -0500

    Fix bug in login form

commit 2c85e2014e8b4d4e2904bb754743c404630308d9
Author: Jane Smith <jane.smith@example.com>
Date:   Sat Feb 12 21:16:19 2022 -0500

    Refactor code for performance
```

Bu çıktıda her commit için ayrı bir blok görüntülenir. Her blok, commit kimliği, yazar adı ve e-posta adresi, commit tarihi ve commit mesajı gibi bilgiler içerir. En yeni commit en üstte görüntülenir.

"git tag" komutu, belirli bir commit veya kaynak kodu sürümü için bir etiket oluşturur. Bu etiketler, projenin belli bir sürümünün belirtilmesinde ve hatırlanmasında kullanılır. Örneğin, bir projenin "v1.0" sürümü için bir etiket oluşturabilirsiniz.

"git tag" komutu kullanımı oldukça basittir. Bir etiket oluşturmak için, aşağıdaki komutu kullanabilirsiniz:

```
git tag <tag-name>
```

Bu komut, mevcut HEAD (yani en son commit) için bir etiket oluşturur. Etiket adı olarak bir kelime veya sayı kullanabilirsiniz. Örneğin, "v1.0" gibi bir etiket oluşturmak için aşağıdaki komutu kullanabilirsiniz:

```
git tag v1.0
```

Bu komut, "v1.0" adlı bir etiket oluşturur ve mevcut HEAD için bu etiketi atar. Mevcut bir commit için etiket oluşturmak isterseniz, aşağıdaki komutu kullanabilirsiniz:

```
git tag <tag-name> <commit>
```

Bu komut, belirtilen commit için bir etiket oluşturur. Etiket oluşturulduktan sonra, "git tag" komutunu kullanarak tüm etiketleri listelemek mümkündür. Aşağıdaki komut, mevcut depodaki tüm etiketleri listeler:

```
git tag
```

"git tag" komutu, projenin belli bir sürümünün belirtilmesinde ve hatırlanmasında kullanılır. Özellikle büyük projelerde, farklı sürümler arasında geçiş yapmak veya belirli bir sürümü geri dönmek gerekiğinde, etiketler çok faydalı olabilir.

```
git status
```

"git status" komutu, Git deposundaki dosyaların durumunu gösterir. Bu komut, hangi dosyaların değiştirildiğini, hangi dosyaların eklenip silindiğini ve hangi dosyaların "stage" edildiğini (yani bir sonraki "commit" için hazır hale getirildiğini) gösterir.

"git checkout" Git'in en temel komutlarından biri olan git checkout, Git deposunda farklı dallar veya farklı commit'ler arasında geçiş yapmak için kullanılır.

```
git checkout feature-branch
```

Bu komut, mevcut dalınızdan "feature-branch" dalına geçiş yapar. Bu dalda yapılan değişiklikler, mevcut dalınızdan ayrılır.

```
git show abcd1234
```

Bu komut, "abcd1234" adlı commit'in değiştirildiği dosyaları ve değişikliklerin ne olduğunu ayrıntılı bir şekilde gösterir. Bu, bir commit'in içeriğini incelemek ve hata ayıklamak için kullanışlı olabilir.

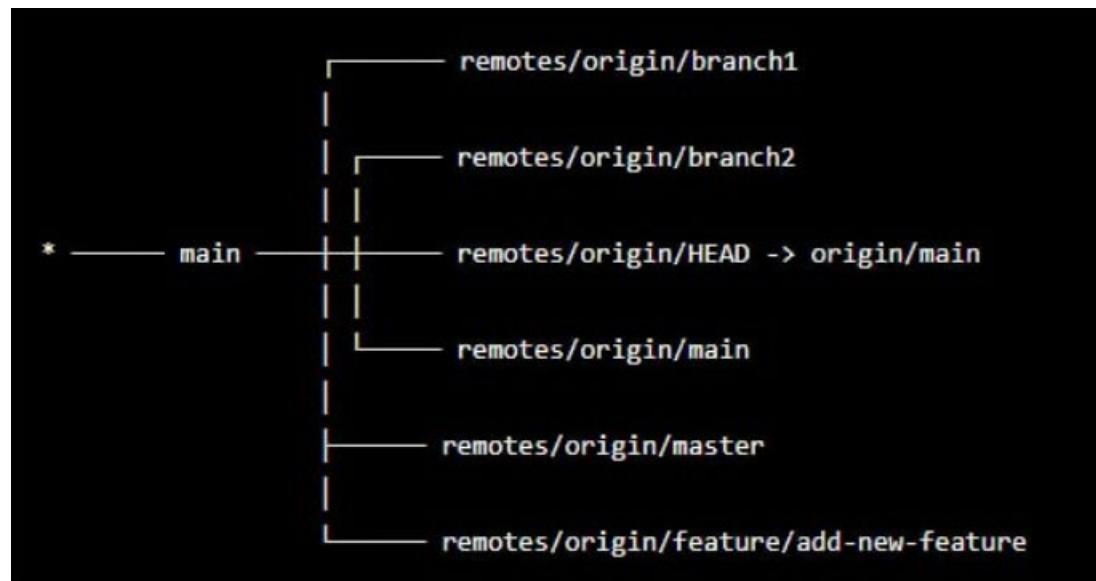
"git branch" komutu, mevcut Git deposundaki tüm yerel dalları listelemek için kullanılır.

```
branch1  
branch2  
* main  
master  
feature/add-new-feature
```

Bu çıktı, mevcut projede bulunan tüm dalların bir listesini göstermektedir. Yıldız işaretleri olan "main" dalı, mevcut bulunduğuuz dalı temsil etmektedir. Yukarıdaki örnekte, "main" dalında bulunuyoruz. "master" dalı ise "main" dalı ile aynı seviyede bulunan başka bir dal, "feature/add-new-feature" dalı ise "main" dalından çatallanarak oluşturulan başka bir dalı temsil eder. "branch1" ve "branch2" ise "main" dalından ayrılmak için oluşturulmuş iki başka dalı gösterir.

"git branch -a" komutu, hem yerel hem de uzak dalları listeler.

```
branch1  
branch2  
* main  
remotes/origin/HEAD -> origin/main  
remotes/origin/branch1  
remotes/origin/branch2  
remotes/origin/main  
remotes/origin/master  
remotes/origin/feature/add-new-feature
```



Bu çıktıda, "git branch" çıktısına ek olarak "remotes" bölümü de yer almaktadır. Bu bölüm, uzak sunucularda bulunan dalları gösterir. "remotes/origin/main" dalı, "origin" adlı uzak sunucudaki "main" dalını temsil eder. Aynı şekilde, "remotes/origin/master" dalı, "origin" adlı uzak sunucudaki "master" dalını temsil eder. "remotes/origin/feature/add-new-feature" dalı ise "origin" adlı uzak sunucudaki "feature/add-new-feature" dalını temsil eder.

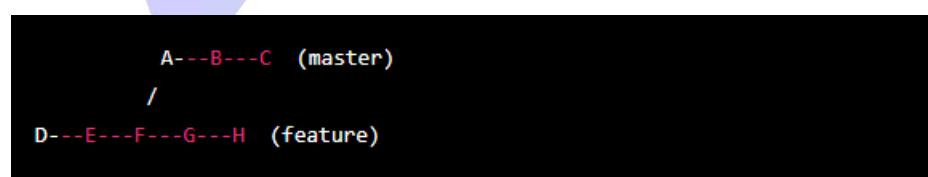
"remotes/origin/HEAD -> origin/main" satırı ise varsayılan olarak izlenen dalı gösterir. Yukarıdaki örnekte, "origin/main" dalı varsayılan olarak izlenen dal olarak ayarlanmıştır.

HEAD

Git dallanması, mevcut Git deposundaki kod tabanının belirli bir noktasından ayrılarak farklı bir yolda devam etmek anlamına gelir. Git'de her dal, bir Git işaretçisi olan HEAD ile belirtilir.

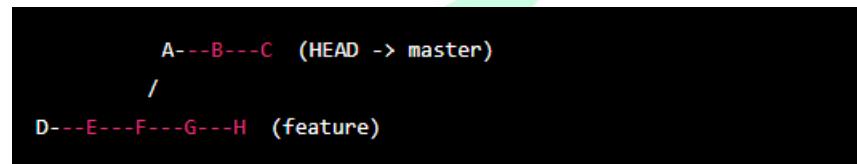
HEAD, mevcut dalda hangi commit'in işaret edildiğini gösteren bir işaretcidir. Bir daldaki son commit, HEAD işaretçisi tarafından gösterilir. Yani, HEAD işaretçisi, mevcut anda hangi dalda olduğumuzu ve en son hangi commit üzerinde çalıştığımızı belirlememize yardımcı olur.

Aşağıdaki şekilde, Git deposundaki dallanmayı ve HEAD işaretçisinin ne olduğunu gösteren bir örnek verilmiştir:

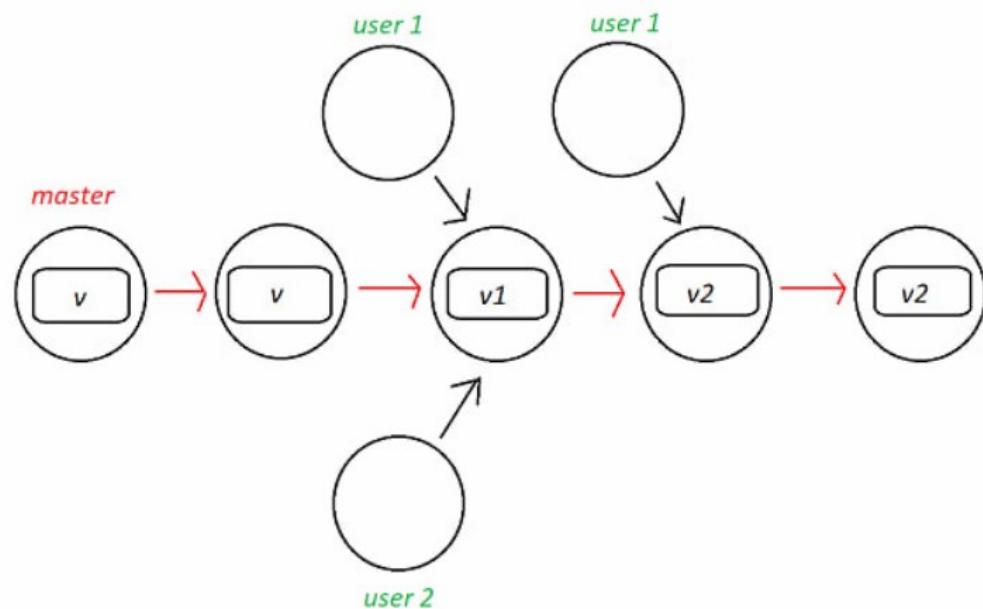


Yukarıdaki örnekte, master ve feature adında iki ayrı dal var. master dalında son commit, C commit'ıdır ve feature dalında son commit, H commit'ıdır. HEAD işaretçisi, mevcut olarak feature dalında olduğumuzu ve son commit'in H olduğunu gösterir.

Eğer git "checkout master" komutu çalıştırılsaydı, HEAD işaretçisi master dalına taşınacak ve son commit C olacaktı. Başka bir deyişle, HEAD işaretçisi şimdi master dalında konumlanacaktır:



Bu şekilde, HEAD işaretçisi, çalışma alanımızın hangi dalda olduğunu ve hangi commit üzerinde çalıştığımızı belirlememize yardımcı olur.



DİĞER KOMUT VE SERVİSLERİ

SSH NEDİR VE NASIL KULLANILIR

SSH (Secure Shell), bilgisayar ile sunucu arasında güvenli bir bağlantı oluşturmak için kullanılan bir ağ protokolüdür. Kullanıcılar terminal penceresi ile uzaktaki bir sunucuya bağlanarak SSH bağlantısı kurabilirler.

SSH, uzak makinelere güvenli giriş yapmak ve komutlar çalıştırmak için kullanılan bir ağ protokolüdür. Aynı zamanda, TCP portlarını ve X11 bağlantılarını iletmek, dosyaları aktarmak (SFTP veya SCP) ve tünel açmak gibi ek işlevler de destekler.

ÖNEMLİ: SSH, sadece bir makineye uzaktan giriş yapmanızına izin vermekle kalmaz, aynı zamanda ortak SSH ifadesinden sonra eklenen komutların uzaktan çalıştırmasına da olanak tanır.

SSH komutunu kullanmak için bir terminal veya komut satırı penceresine ihtiyacınız vardır. SSH komutunun temel kullanımı şu şekildedir:

```
ssh [username]@[hostname/IP]
```

[username] yerine bağlanmak istediğiniz uzak makinedeki kullanıcı adını, [hostname/IP] yerine ise uzak makineye ait host adını veya IP adresini girin. Eğer bağlanmak istediğiniz uzak makine üzerinde standart 22 portunu kullanıyorsa, bu komut port numarası belirtmeden çalışacaktır. Ancak, uzak makine üzerinde farklı bir port numarası kullanılıyorsa, port numarasını şu şekilde belirtmeniz gereklidir:

```
ssh -p [port] [username]@[hostname/IP]
```

Bağlantı kurulduktan sonra, uzak makine üzerinde komutları yazabilirsiniz. Bağlantıyı sonlandırmak için "exit" komutunu kullanabilirsiniz.

Telnet ve SSH FARKI nedir?

TELNET ve SSH, uzak makinelere bağlanmak için kullanılan ağ protokolleridir. Ancak, TELNET güvensiz bir bağlantı protokolüdür ve veri ve kullanıcı bilgileri şifrelenmez. SSH ise, güvenli ve şifreli bir bağlantı protokolüdür ve veri ve kullanıcı bilgileri şifrelerek gönderilir. Bunun sonucu olarak, SSH güvenli bir bağlantı kurmak isteyen kullanıcılar için daha güvenli bir seçenek sunar.

Suid bit

SUID (Set User ID) özelliği, bir dosyanın sahibinin kimliği yerine başka bir kullanıcının kimliğiyle çalıştırılmasına olanak tanıyan bir dosya özelliğidir. Bu özellik, özellikle sistem yöneticileri tarafından, bazı programların sistem düzeyinde ayrıcalıklar gerektirdiği durumlarda kullanılır.

SUID özelliği genellikle aşağıdaki dosyalar için kullanılır:

- 1- **/usr/bin/passwd**: Bu program, kullanıcıların şifrelerini değiştirmelerine izin verir ve SUID özelliğine sahiptir.
- 2- **/usr/bin/sudo**: Bu program, özel ayrıcalıklarla bir komutu veya bir dizi komutu çalıştırmak için kullanılır.
- 3- **/usr/bin/su**: Bu program, kullanıcıların başka bir kullanıcının kimliğiyle oturum açmalarına olanak tanır.
- 4- **/usr/sbin/postdrop**: Bu program, Postfix e-posta teslimi için kullanılır ve SUID özelliğine sahiptir.
- 5- **/usr/sbin/postqueue**: Bu program, Postfix posta kuyruğunu yönetmek için kullanılır ve SUID özelliğine sahiptir.

Örnek:

“ /usr/bin/passwd ” dosyası, kullanıcıların kendi şifrelerini değiştirmelerine izin veren bir programdır ve SUID özelliğine sahip olmalıdır. SUID ayarlamak için şu adımları izleyebilirsiniz:

- 1- “ ls -l /usr/bin/passwd ” komutuyla “ /usr/bin/passwd ” dosyasının mevcut izinlerini kontrol edin:

```
-rwxr-xr-x 1 root root 68232 Aug 6 2021 /usr/bin/passwd
```

- 2- “ sudo chmod u+s /usr/bin/passwd ” komutuyla “ passwd ” dosyasının SUID özelliğini ayarlayın:

```
sudo chmod u+s /usr/bin/passwd
```

- 3- “ ls -l /usr/bin/passwd ” komutuyla “ /usr/bin/passwd ” dosyasının yeni izinlerini kontrol edin:

```
-rwsr-xr-x 1 root root 68232 Aug 6 2021 /usr/bin/passwd
```

Gördüğünüz gibi, “ -rwxr-xr-x ” yerine “ -rwsr-xr-x ” yazıyor. “s” karakteri, “ passwd ” dosyasının SUID özelliğine sahip olduğunu gösterir.

Örnek:

Ping programı, ağ bağlantısının durumunu kontrol etmek için kullanılır ve bazı sistemlerde SUID özelliğine sahiptir. SUID ayarlamak için şu adımları takip edebilirsiniz:

- 1- “ls -l \$(which ping)” komutuyla “ping” programının mevcut izinlerini kontrol edin:

```
-rwxr-xr-x 1 root root 42K Jan 20 2022 /bin/ping
```

- 2- “sudo chmod u+s \$(which ping)” komutuyla “ping” programının SUID özelliğini ayarlayın:

```
sudo chmod u+s $(which ping)
```

- 3- “ls -l \$(which ping)” komutuyla “ping” programının yeni izinlerini kontrol edin:

```
-rwsr-xr-x 1 root root 42K Jan 20 2022 /bin/ping
```

Gördüğünüz gibi, “-rwxr-xr-x” yerine “-rwsr-xr-x” yazıyor. “s” karakteri, “ping” programının SUID özelliğine sahip olduğunu gösterir.

Not: “\$(which ping)” ifadesi, “which” komutunun çıktısını almak ve çıktıyı “ping” komutu ile değiştirmek için kullanılan bir bash özelliğidir.

“which” komutu, bir komutun tam yolunu çözümler.

chmod

chmod komutu, dosyaların izinlerini (permissions) değiştirmek için kullanılır. İzinler, dosyanın okunabilir (readable), yazılabilir (writable) ve çalıştırılabilir (executable) olup olmadığını belirler. Ayrıca, dosyaya kimlerin erişebileceğini de belirler.

chmod komutunun temel kullanımı şöyledir:

```
chmod [opsyonlar] izinler dosya veya dizin
```

Burada, “opsyonlar” chmod komutunun davranışını değiştiren seçeneklerdir. “izinler”, belirtilen dosya veya dizinin izinlerini belirler. “dosya veya dizin” ise izinleri değiştirecek dosyanın veya dizinin adıdır.

İzinler, semboller veya sayılar kullanılarak belirtilebilir. Semboller, "+" ve "-" işaretlerini kullanarak izinleri eklemek veya kaldırmak için kullanılır. "+" işaretini izinleri eklerken, "-" işaretini izinleri kaldırır. "=" işaretini ise, belirtilen izinleri atandır ve diğer tüm izinler kaldırılır.

Aşağıdaki tabloda sembolik izinlerin anlamları açıklanmaktadır:

Sembol	Anlamı
u	Dosya sahibi
g	Dosya grubu
o	Diğer kullanıcılar
a	Tüm kullanıcılar

Aşağıdaki tabloda, izin sembollerini ve anlamları açıklanmaktadır:

Sembol	Anlamı
r	Dosya okunabilir
w	Dosya yazılabilir
x	Dosya çalıştırılabilir

İzinlerin sayısal gösterimi, her bir izin türü için aşağıdaki değerlerin kullanılmasıyla hesaplanır:

r izni = 4

w izni = 2

x izni = 1

İzin yok = o

Örneklideelim

- 1- Dosya izinlerini değiştirmek için sembolik izinler kullanma

Dosyanın okunabilir, yazılabilir ve çalıştırılabilir olduğu bir senaryo düşünelim. Ancak, dosyaya sadece dosya sahibi tarafından yazılabilmesini istiyoruz. Bu durumda, şu komutu kullanabiliriz:

```
chmod u-w dosya_adi
```

Bu komut, "dosya_adi" adlı dosyanın dosya sahibinin yazma iznini (w) kaldırır (-) ve dosyaya sadece okuma ve çalışma izni bırakır.

- 2- Dosya izinlerini sayısal izinler kullanarak değiştirme

Dosyaya sadece dosya sahibinin okuma, yazma ve çalışma izni vermek istediğimiz bir senaryo düşünelim. Bu durumda, şu komutu kullanabiliriz:

```
chmod 700 dosya_adi
```

Bu komut, "dosya_adi" adlı dosyaya dosya sahibinin okuma, yazma ve çalışma izni (7) verir ve diğer tüm kullanıcıların (dosya grubu ve diğer kullanıcılar) hiçbir izni yoktur.

3- Dizin izinlerini değiştirme

Bir dizinin izinleri, dosya izinleriyle benzer şekilde değiştirilebilir. Dizine yazma izni vermek için şu komut kullanılabilir:

```
chmod +w dizin_adi
```

Bu komut, "dizin_adi" adlı dizine yazma izni ekler.

4- Tüm dosyaların ve dizinlerin izinlerini değiştirme

Bir dizindeki tüm dosyaların ve dizinlerin izinleri, "-R" seçeneği kullanılarak değiştirilebilir. Örneğin, aşağıdaki komut, "dizin_adi" adlı dizindeki tüm dosya ve dizinlerin izinlerini, dosya sahibine okuma, yazma ve çalışma izni vererek diğer tüm kullanıcılarından izni kaldırır:

```
chmod -R 700 dizin_adi
```

Bu komut, "dizin_adi" adlı dizindeki tüm dosya ve dizinlere dosya sahibinin okuma, yazma ve çalışma izni (7) verir ve diğer tüm kullanıcıların hiçbir izni yoktur.

openssl

openssl, güvenli internet üzerinde iletişim sağlamak için SSL ve TLS protokollerini uygulayan açık kaynak kodlu bir kriptografik kütüphanedir. Bu araç, özel ve genel anahtarlar oluşturmak, dijital sertifikalar imzalamak ve doğrulamak, verileri şifrelemek ve açmak, dijital imzalar oluşturmak ve yönetmek gibi fonksiyonlar için kullanılabilir. UNIX benzeri işletim sistemlerinde kullanılabilir ve yaygın olarak sistem yöneticileri, ağ güvenlik mühendisleri ve geliştiriciler tarafından kullanılır.

openssl'in bazı yaygın kullanıcıları şunlardır:

1. Şifreleme/açma: "openssl" verilerin şifrelenmesini ve açılmasını sağlar. Örneğin, bir dosya şifrelenebilir ve sonra açılabilir.
2. Anahtar oluşturma: "openssl" özel ve genel anahtarlar oluşturmak için kullanılabilir.
3. Dijital sertifikalar: "openssl" sertifikaların imzalanmasını, doğrulanmasını ve yönetilmesini sağlar.
4. Dijital imzalar: "openssl" dijital imzaların oluşturulmasını ve doğrulanmasını sağlar.

Komut satırında openssl'i kullanmak için, kullanmak istediğiniz fonksiyonun adını ve parametrelerini belirtmelisiniz. Örneğin, bir dosyanın şifrelenmesi için aşağıdaki komut kullanılabilir:

```
openssl enc -aes-256-cbc -salt -in plaintext_file.txt -out encrypted_file.enc
```

s_client

“s_client” OpenSSL komut satırı aracıdır ve SSL (Secure Sockets Layer) ve TLS (Transport Layer Security) protokollerini uygular. Bu araç, ağ bağlantılarını güvenli bir şekilde yapmanızı, SSL sertifikalarının doğruluğunu doğrulamanıza ve SSL yapılandırmalarını test etmenize olanak tanır.

Bir web sitesinin HTTPS sunucusuna nasıl bağlanacağını gösteren bir örnek:

```
openssl s_client -connect example.com:443
```

Bu, web sitesinin 443 numaralı bağlantı noktasındaki HTTPS sunucusuna güvenli bir bağlantı kuracak ve konu adı, veren ve sertifika zinciri dahil olmak üzere SSL sertifikası hakkında bilgi gösterecektir.

Ayrıca, -showcerts seçeneğini kullanarak tüm sertifika zincirini görüntüleyebilir ve -starttls seçeneğini kullanarak bir StartTLS yükseltilmesiyle güvenli bir TLS bağlantısı başlatabilirsiniz.

s_client ayrıca SSL yapılandırmalarını test etmek ve hata ayıklamak için kullanılabilir ve tarayıcı kullanmadan komut satırından SSL bağlantılarını test etme olağın sunar.

TELNET

“telnet” programı, UNIX ve Linux gibi işletim sistemlerinde, bir ağdaki uzak bir bilgisayarın konsol erişimine olanak tanıyan bir uygulamadır. Kullanıcılar, telnet programını kullanarak uzak bir sunucuda oturum açabilir ve bu sunucuda dosya yönetimi, sistem yapılandırması ve benzeri işlemler yapabilir.

Kullanım şekli şu şekildedir:

```
telnet [hostname/IP address] [port]
```

[hostname/IP address]: Telnet yapmak istenen sunucunun adı veya IP adresi.

[port]: Telnet bağlantısının kurulacağı port (varsayılan olarak 23).

```
telnet example.com
```

Bu komut, “example.com” sunucusuna bağlanmak için telnet bağlantısı kurar.

Ayrıca, telnet uygulaması günümüzde güvenlik açıkları nedeniyle yaygın olarak kullanılmaz ve tercih edilmez. Bunun yerine, daha güvenli ağ protokollerini olan “ssh” veya “ftp” gibi alternatifler kullanılır.

netcat

“nc” (Netcat) bir ağ araçtır. Ağlar arasında veri aktarımı yapmak, sunucular ve istemciler arasında bağlantılar kurmak veya sunucuların durumunu test etmek gibi çeşitli amaçlarla kullanılabilir. “nc” aracı; ağ protokollerini, veri gönderme ve alma işlemleri gibi çeşitli ağ işlemlerinde kullanılan bir betik diline benzer bir şekilde kullanılabilir.

```
nc [options] [hostname/IP address] [port]
```

[options]: nc'nin farklı özelliklerini aktifleştirmek veya yapılandırmak için kullanılan seçeneklerdir. Aşağıdaki seçeneklerden bazıları:

“-l”: Sunucu modunu aktifleştirir, belirtilen porta bağlı olarak istemci isteklerini dinler.

“-n”: Host adlarının çözümlemesini devre dışı bırakır, yalnızca IP adreslerini kullanır.

“-p”: Belirtilen kaynak portu kullanır.

“-u”: UDP protokolünü kullanır.

“-v”: Detaylı çıktı verir.

[hostname/IP address]: Bağlanmak istenen sunucunun adı veya IP adresi.

[port]: Bağlantının kurulacağı port numarası.

Örnek kullanımlar:

```
nc -l 8080
```

Bu komut, 8080 nolu porta bağlı olarak bir sunucu olarak çalışmaya başlar.

Başka bir makineye bağlanmak için:

```
nc example.com 80
```

Başka bir makineden gelen verileri görüntülemek için:

```
nc -l -p 8080
```

UDP protokolü kullanarak başka bir makineye veri göndermek için:

```
echo "Hello, World!" | nc -u example.com 8080
```

nmap

nmap, güçlü bir ağ keşif aracı ve güvenlik tarayıcısıdır. Bir ağı ana bilgisayarlar ve hizmetler için taramanızı, açık bağlantı noktalarını keşfetmenize ve bu ana bilgisayarlarda hangi işletim sistemlerinin ve uygulamaların çalıştığını belirlemenize olanak tanır.

Açık bağlantı noktaları için bir ağı taramak üzere nmap'in nasıl kullanılacağına ilişkin bir örnek:

```
nmap 192.168.1.0/24
```

Bu, “192.168.1.0/24” alt ağını tüm aktif hostlar için tarayacak ve açık portlar hakkında bilgi, bu portlarda çalışan hizmetler ve her hostun durumu hakkında bilgi görüntüleyecektir.

nmap'i taramayı özelleştirmek için kullanılabilen birçok seçenek ve parametre bulunur. En sık kullanılan seçeneklerden bazıları şunlardır:

- p: hedef port veya taranacak port aralığını belirtir.
- sS: gizli tarama yapar, bu tarama türü tespit etmek zor olması amaçlanmıştır.
- O: hedef hostların işletim sistemi belirlemeyi denemelidir.
- A: OS tanımlama, sürüm tanımlama, betik tarama ve izleme yapma işlevini etkinleştirir.

nmap'in bu seçeneklerle nasıl kullanılacağına dair bir örnek:

```
nmap -p 80,443 -sS -O -A example.com
```

Bu, host “example.com” u 80 ve 443 portları için tarayacak, gizli tarama yapacak, hedef hostun işletim sistemini belirlemeyi denemelidir ve OS tanımlama, sürüm tanımlama, betik tarama ve izleme yapma işlevini etkinleştirecektir.

“nmap” ağ keşfine ve güvenlik testi için kullanılabilecek güçlü bir araçtır, ancak ağır normal işletimi bozabilecek ve büyük miktarda ağ trafiği oluşturabileceği için dikkatli kullanılması gereklidir.

tmux

“tmux” terminal çalışmasını yöneten bir programdır. Tmux, birden fazla pencereyi, panelleri ve programları aynı anda yönetmenize olanak tanır. Tmux kullanımı, screen gibi komut satırından yapılır ve terminalde çalışan uygulamalarınızın çalışmasını sürdürmesine olanak tanır.

1. tmux başlatmak: "tmux" komutu ile tmux'u başlatabilirsiniz.
2. Pencere oluşturmak: "Ctrl + b + c" tuş kombinasyonunu kullanarak tmux içinde yeni bir pencere oluşturabilirsiniz.
3. Panel oluşturmak: "Ctrl + b + %" tuş kombinasyonunu kullanarak tmux içinde yeni bir panel oluşturabilirsiniz.

4. Çalışan programları kopyalamak: "Ctrl + b + [Space]" tuş kombinasyonunu kullanarak tmux içindeki çalışan programları kopyalayabilirsiniz.
5. tmux'dan çıkmak: "Ctrl + b + d" tuş kombinasyonunu kullanarak tmux programından çıkabilirsiniz.

cron

Cron, Unix ve Linux gibi işletim sistemlerinde kullanılan bir zamanlanmış görev planlayıcısıdır. Kullanıcılar, belirli bir zaman ve/veya periyodik olarak yapılması gereken görevleri belirleyebilir ve cron tarafından otomatik olarak çalıştırılabilir.

Cron kullanımı için **crontab** adı verilen bir dosya kullanılır. Bu dosya, hangi komutların hangi zamanlarda çalıştırılması gerektiğini belirler. Her görev için bir satır bulunur ve bu satır, zamanı belirlemek için bir dizi parametre içerir.

Örnek olarak, her gün saat 10:30'da belirli bir dosyayı çalıştırmak için aşağıdaki gibi bir görev tanımlanabilir:

```
30 10 * * * /path/to/command
```

Aşağıdaki örnekte, her saat başına /usr/bin/example_script.sh betiğinin çalıştırılmasını gösterir:

```
0 * * * * /usr/bin/example_script.sh
```

Bu satırın anlamı, her saat başında (o dakika) /usr/bin/example_script.sh betiğinin çalıştırılmasıdır.

Cron kullanımı, bir terminal penceresinde “crontab -e” komutunun çalıştırılmasıyla başlar ve kullanıcı, dosyayı düzenleyebilir veya yeni görevler ekleyebilir.

```
* * * * * [komut]
```

Bu satır, sütunlar şeklinde beş adet * işaretini ve sonunda çalıştırılması istenen komutu içerir. Her bir * işaretini aşağıdaki anlamı taşır:

1. Dakika (0-59)
2. Saat (0-23)
3. Gün (0-31)
4. Ay (0-12)
5. Haftanın günü (0-7, 0 veya 7 Pazar)

Örnek olarak, tüm hafta boyunca her gün saat 12:00'da belirli bir betiği çalıştırmak için aşağıdaki satırı ekleyebilirsiniz:

```
0 12 * * * [betik yolu]
```

Bu, her gün saat 12:00'da belirtilen betiği çalıştıracaktır. Crontab dosyasını kaydetmek ve değişiklikleri uygulamak için dosyayı kaydetmeniz ve çıkışınız gereklidir.

Cronjob belirli zamanlarda veya belirli aralıklarla otomatik olarak çalıştırılması gereken görevleri planlar ve yürütür. Örneğin, web sitenizin veritabanının günlük olarak yedeklenmesi, belirli aralıklarla e-postaların taranması gibi görevler cron tarafından yapılabilir.

Linux işletim sisteminde bu cronjobs'lar için birden fazla klasör bulunmaktadır, bunlar cron.d, cron.daily, cron.hourly, cron.monthly, crontab ve cron.weekly dir. Bu klasörler, nasıl çalıştırılması gerektigine dair programlar içeren dosyalar içermektedir.

1. **cron.d:** Sistem yöneticilerinin veya uygulamaların oluşturduğu özel cron görevlerinin yapılandırmasını içeren bir dizindir.
2. **cron.daily:** Günlük olarak tekrar eden görevleri yapılandırmak için kullanılan bir dizindir.
3. **cron.hourly:** Saatlik olarak tekrar eden görevleri yapılandırmak için kullanılan bir dizindir.
4. **cron.monthly:** Aylık olarak tekrar eden görevleri yapılandırmak için kullanılan bir dizindir.
5. **crontab:** Kullanıcıların kendi cron görevlerini yapılandırmasına olanak tanıyan bir dosyadır.
6. **cron.weekly:** Haftalık olarak tekrar eden görevleri yapılandırmak için kullanılan bir dizindir.

bash

Bash, bir komut satırı arabirimini (CLI) için kullanılan bir shelldir. Bu, UNIX benzeri işletim sistemlerinde terminalde kullanılan bir betik dildir. Bash kullanarak, sistemi yönetebilir ve tekrarlanabilir görevleri otomatik olarak yapabilirsiniz. Bash, bir dizi komutun bir dosyada toplanması ve otomatik olarak çalıştırılmasını sağlayan bash betiklerini (script) yazmak için de kullanılabilir. Bash, birçok UNIX benzeri işletim sisteminde varsayılan shelldir.

Bash'ın kullanımı, aşağıdaki gibi farklı yollarla yapılabilir:

1. Komut satırı: Terminalde komutları manuel olarak yazabilirsiniz. Örneğin, ls komutu kullanarak sistemdeki dosyaların listesini görüntüleyebilirsiniz.
2. Bash Script: Bir betik dosyası oluşturabilir ve bu dosyayı bash ile çalıştırabilirsiniz. Örneğin, aşağıdaki gibi bir bash betiği oluşturabilirsiniz:

```
#!/bin/bash
echo "Merhaba, Dünya!"
```

Bu betiği çalıştmak için terminalde ./script.sh yazmanız gereklidir.

3. Aliases: Bash, komutların kısaltılmış versiyonlarını oluşturmanıza olanak tanır. Örneğin, sık sık “ls -al” komutunu kullanıyorsanız, aşağıdaki gibi bir alias oluşturabilirsiniz:

```
alias l='ls -al'
```

Bu alias'ı oluşturduktan sonra, terminalde l yazmak yerine ls -al yazmanız gereklidir.

Bu örnekler, bash'in nasıl kullanılabileceğine dair sadece birkaç örnektir. Bash'in tam potansiyelini keşfetmek için, manuel olarak veya çeşitli kaynakları kullanmalısınız.

"sh" (Bash) komutu, Unix/Linux işletim sistemlerinde kullanılan bir kabuk programıdır. "sh" komutunun kullanımı, kullanıcıların işletim sistemi ile etkileşim kurmasına olanak tanır ve komut girdisi yaparak, dosya yönetimi, dizin değiştirme, süreç yönetimi gibi işlemleri gerçekleştirmelerine olanak tanır. Kullanımı için, bir terminal veya komut satırı penceresinde "sh" komutunu girin ve ardından kabuk üzerinde çalışmaya başlayabilirsiniz.

```
$ sh
$ # you are now in the sh shell
$ pwd
/home/user
$ ls
documents pictures music downloads
$ cd documents
$ ls
resume.pdf cover_letter.txt notes.txt
$ echo "Hello World"
Hello World
$ exit
$ # you have now exited the sh shell
```

Bu örnekte, "sh" komutu ile sh kabuğu başlatılır ve "pwd" ve "ls" komutları ile kullanıcının bulunduğu dizin ve içerisindeki dosyalar listelenir. "cd" komutu ile documents dizinine geçilir ve "ls" komutu ile bu dizin içerisindeki dosyalar listelenir. echo "Hello World" komutu ile ekrana "Hello World" yazdırılır ve son olarak "exit" komutu ile sh kabuğundan çıkarılır.

screen

Bir tek terminal penceresi içinde birden fazla terminal oturumunu çalıştırmanızı ve verimli bir şekilde yönetmenizi sağlar. Birden fazla terminal oturumu oluşturabilir, arasında geçiş yapabilir ve bu oturumların bağlantısını koparabilirsiniz. Screen ayrıca kopyala-yapıştır, kaydırma ve ağ bağlantısının kopmasından sonra oturumların tekrar başlatılması gibi özellikler sunar.

1. Screen programını başlatmak: "screen" komutu ile terminalde screen programını başlatabilirsiniz.
2. Screen içinde çalışan bir programı açmak: Başlatılan screen programı içinde, örneğin "top" komutunu çalıştırarak bir program açabilirsiniz.
3. Screen'den çıkmak: "Ctrl + a + d" tuş kombinasyonunu kullanarak screen programından çıkabilirsiniz.

- Screen'i tekrar başlatmak: "screen -r" komutunu kullanarak son çalışan screen programını tekrar başlatabilirsiniz.

tar komutu

"tar" komutu, bir veya daha fazla dosya veya dizinin bir arşiv halinde sıkıştırılmasını veya arşivden ayrılmamasını sağlar. Arşivlenen dosyalar, diskte daha az yer kaplar ve aynı zamanda verilerin daha kolay ve güvenli bir şekilde taşınmasına olanak tanır.

```
tar -czf archive.tar.gz dir1 dir2
```

Bu örnekte, "dir1" ve "dir2" dizinleri "archive.tar.gz" adlı bir arşiv içinde sıkıştırılır. "-c" seçeneği arşiv oluşturulmasını, "-z" seçeneği de arşivin gzip sıkıştırmasını belirtir. "-f" seçeneği de arşivin adını belirtir.

```
tar -xzf archive.tar.gz
```

Bu örnekte, "archive.tar.gz" adlı arşivdeki dosyalar ve dizinler ayrılr. "-x" seçeneği arşivden ayrılmamasını, "-z" seçeneği de arşivin gzip sıkıştırmasının çözülmesini belirtir. "-f" seçeneği de arşivin adını belirtir.

"tar" komutu, birçok farklı seçenek sahiptir ve ayrıntılı kullanımı için "man tar" ya da dokümantasyona bakılması önerilir.

bzip2 komutu

"bzip2" komutu, dosyaların sıkıştırılmasını ve sıkıştırılmış dosyaların açılmasını sağlar. Sıkıştırılmış dosyalar, diskte daha az yer kaplar ve aynı zamanda verilerin daha kolay ve güvenli bir şekilde taşınmasına olanak tanır. "bzip2" komutu, daha yüksek bir sıkıştırma oranı sunar, ancak aynı zamanda daha yavaş bir sıkıştırma ve açma hızına sahiptir.

```
bzip2 file.txt
```

Bu örnekte, "file.txt" dosyası "file.txt.bz2" adıyla sıkıştırılır.

```
bzip2 -d file.txt.bz2
```

Bu örnekte, "file.txt.bz2" adlı sıkıştırılmış dosya açılır ve orijinal "file.txt" dosyası oluşturulur. "-d" seçeneği dosyanın sıkıştırılmasının çözülmesini belirtir.

gzip komutu

"gzip" komutu, dosyaların sıkıştırılmasını ve sıkıştırılmış dosyaların açılmasını sağlar. Sıkıştırılmış dosyalar, diskte daha az yer kaplar ve aynı zamanda verilerin daha kolay ve güvenli bir şekilde taşınmasına olanak tanır.

```
gzip file.txt
```

Bu örnekte, "file.txt" dosyası "file.txt.gz" adıyla sıkıştırılır.

```
gzip -d file.txt.gz
```

Bu örnekte, "file.txt.gz" adlı sıkıştırılmış dosya açılır ve orijinal "file.txt" dosyası oluşturulur. "-d" seçeneği dosyanın sıkıştırılmasının çözülmesini belirtir.

BANDİT SORU ÇÖZÜMLERİ

BANDİT SEVİYE 0

(ssh)

Level Goal

The goal of this level is for you to log into the game using SSH. The host to which you need to connect is bandit.labs.overthewire.org, on port 2220. The username is **bandit0** and the password is **bandit0**. Once logged in, go to the Level 1 page to find out how to beat Level 1.

“ssh” ile bağlanmamız gereken ana bilgisayar ,2220 bağlantı noktasındaki bandit.labs.overthewire.org.’dur. Kullanıcı adı bandito ve parola bandito’dır. Bu bilgileri kullanarak bandito kullanıcısına giriş yaptık.

```
$ ssh bandit0@bandit.labs.overthewire.org -p2220
[...]
This is an OverTheWire game server.
More information on http://www.overthewire.org/wargames
bandit0@bandit.labs.overthewire.org's password:
```

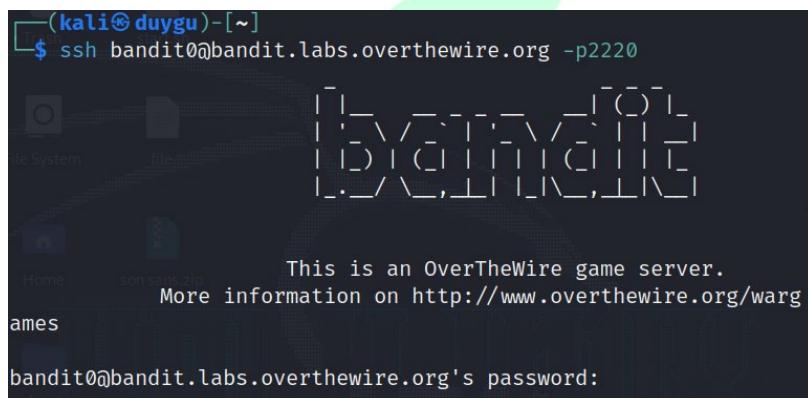
BANDİT SEVİYE 0 → SEVİYE 1

(ls , cd , cat , file , du , find)

Level Goal

The password for the next level is stored in a file called **readme** located in the home directory. Use this password to log into bandit1 using SSH. Whenever you find a password for a level, use SSH (on port 2220) to log into that level and continue the game.

Bağlanmanız gereken ana bilgisayar, 2220 bağlantı noktasındaki bandit.labs.overthewire.org'dur.



```
(kali㉿duygu)-[~]
$ ssh bandit0@bandit.labs.overthewire.org -p2220
[ _ | _ \ / _ _ | _ \ / _ | ( _ ) | _ |
| | _ | ( _ | | | | ( _ | | | | |
| . _ / \ _ , _ | | _ \ _ , _ | \ _ |

This is an OverTheWire game server.
More information on http://www.overthewire.org/wargames

bandit0@bandit.labs.overthewire.org's password:
```

Ls komutunu kullanarak içinde neler olduğuna bakalım.

```
bandit0@bandit:~$ ls
readme
```

readme 'yi okuyalım.

```
bandit0@bandit:~$ cat readme
NH2SXQwcBdpmTEzi3bvBHMM9H66vVXjL
```

İşte bir sonraki kullanıcının şifresini bulduk.

Exit komutunu kullanarak hesaptan çıkışlırsınız.

BANDİT SEVİYE 1 → SEVİYE 2

(ls , cd , cat , file , du , find)

Level Goal

The password for the next level is stored in a file called - located in the home directory

Önceki seviyede aldığımız şifreyi kullanarak giriş yapıyoruz.

```
$ ssh bandit1@bandit.labs.overthewire.org -p2220
[!]_ \ /_--\ /_([ )]_-
[ ]) ([ )][ ][( )][ ]
_./_ \_,_||| \|_,_||\|_
File System
Home son.sans.zip
ames
dosya
bandit1@bandit.labs.overthewire.org's password:

This is an OverTheWire game server.
More information on http://www.overthewire.org/wargame

bandit1@bandit:~$ ls
```

Ls komutu ile listelediğimizde görüyoruz ki “-” ile başlayan bir dosyamız var.

```
bandit1@bandit:~$ ls
-
```

Bunu cat komutu ile direkt okutamıyoruz fakat aşağıdaki yöntemle yazdırabiliriz.

```
bandit1@bandit:~$ cat < -
rRGizSaX8Mk1RTb1CNQoXTcYZWU6lgzi
```

BANDİT SEVİYE 2 → SEVİYE 3

(ls , cd , cat , file , du , find)

Level Goal

The password for the next level is stored in a file called **spaces in this filename** located in the home directory

Yeni aldığımız şifreyle bandit2 kullanıcısına giriş yapalım ve ls komutuyla içinde neler olduğunu bakalım.

```
bandit2@bandit:~$ ls
spaces in this filename
```

Dosyanın adı boşluklu bir ifadeden oluşmakta. Bu ifadeyi “cat” komutu ile okutabilmemiz için verilen dosyanın ismini çift tırnak içine yazmamız gereklidir.

```
bandit2@bandit:~$ cat "spaces in this filename"
aBZ0W5EmUfAf7kHTQeOwd8bauFJ2lAiG
```

BANDİT SEVİYE 3 → SEVİYE 4

(ls , cd , cat , file , du , find)

Level Goal

The password for the next level is stored in a hidden file in the **inhere** directory.

Bandit3 kullanıcısına giriş yapalım ve ls komutuyla içinde neler olduğuna bakalım.

```
bandit3@bandit:~$ ls
inhere
```

Bir dosyamız var. “cd” komutu ile içine girelim. Ve içindekileri listeleyelim.

```
bandit3@bandit:~$ cd inhere
bandit3@bandit:~/inhere$ ls
```

Herhangi bir şey çıkmadı.

Gizli bir dosya var mı yok mu diye bilmek için “ls -la” komutunu kullanalım.

```
bandit3@bandit:~/inhere$ ls -la
total 12
drwxr-xr-x 2 root      root      4096 Jan 11 19:19 .
drwxr-xr-x 3 root      root      4096 Jan 11 19:19 ..
-rw-r----- 1 bandit4 bandit3    33 Jan 11 19:19 .hidden
```

İşte “hidden” isimli gizli dosyamızı gördük. “cat” komutu ile içini okuyalım.

```
bandit3@bandit:~/inhere$ cat .hidden
2EW7BBsr6aMMoJ2HjW067dm8EgX26xNe
```

BANDİT SEVİYE 4 → SEVİYE 5

(ls , cd , cat , file , du , find)

Level Goal

The password for the next level is stored in the only human-readable file in the **inhere** directory. Tip: if your terminal is messed up, try the “reset” command.

Giriş yaptıktan sonra “ls” komutu ile listelediğimizde “inhere” adında bir dosya olduğunu görüyoruz. “cd” komutu ile içine girelim.

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

Burada dosyalar olduğunu görüyoruz. Soruda bize insan tarafından okunabilir olduğunu söylemiştı.

Hatırlatma: “file” komutu kullanarak dosyaların içeriğinin türüne bakabiliriz. Komutumuzda “ file ./* ” ifadesindeki “.” Bulduğumuz dizindeki tüm dosyaları kasteder. “/* ” ifadesi ise bütün dosyaları ele almak için kullanılan bir ifadedir. “ * ” ifadesi yerine özellikle öğrenmek istediğimiz dosyanın bilgisini yazarak sadece belirttiğimiz dosya hakkında da bilgi alabiliriz.

O zaman bu dosyaların içeriğinin türüne file komutu ile bakabiliriz.

```
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
```

Bu dosyaların içinde “ -file07 ” dosyasının ASCII text olduğunu görüyoruz.

“cat” komutu ile içindekini yazdırduğumızda bir sonraki kullanıcının şifresini bulmuş oluruz.

```
bandit4@bandit:~/inhere$ cat < -file07
lrIWWI6bB37kxfiCQZqUdOIYfr6eEeqR
```

BANDİT SEVİYE 5 → SEVİYE 6

([ls](#) , [cd](#) , [cat](#) , [file](#) , [du](#) , [find](#))

Level Goal

The password for the next level is stored in a file somewhere under the **inhere** directory and has all of the following properties:

- human-readable
- 1033 bytes in size
- not executable

Soruda bize “inhere” dizini altındaki bir dosyada saklı olduğunu söylüyor.

“cd” komutu ile “inhere” adlı dizine girdiğimizde soruda bizden istenilen şartlar doğrultusunda bir komut yazarak bulabiliyoruz.

Bilgi: “find” komutu aşağıdaki parametrelerle kullanarak sorunun bizden istediği kriterlere uygun saklı dosyayı bulabiliyoruz.

-type f: Sadece dosya tipindeki nesneleri (dosyaları) listelemek için.

-readable: Okunabilir dosyaları listelemek için.

! -executable: Çalıştırılabilir dosyaları hariç tutmak için. (-executable parametresi tek başına kullanıldığında “çalıştırılabilir” dosyaları bulacaktır. Başına koyduğumuz “!” işaretini bu parametrenin aksını istediğimiz için kullanılır.)

-size 1033c: 1033 baytlık dosyaları listelemek için.

```
bandit5@bandit:~/inhhere$ find . -type f -readable ! -executable -size 1033c
./maybehhere07/.file2
bandit5@bandit:~/inhhere$ cat ./maybehhere07/.file2
P4L4vucdmLnm8I7Vl7jG1ApGSfjYKqJU
```

Bize aradığımız şartları “./maybehhere07/.file2” dizinindeki dosyanın sağladığını söylüyor. Bunu “cat” komutuyla okuttuğumuzda bize şifreyi veriyor.

BANDİT SEVİYE 6 → SEVİYE 7

(ls , cd , cat , file , du , find, grep)

Level Goal

The password for the next level is stored **somewhere on the server** and has all of the following properties:

owned by user bandit7

owned by group bandit6

33 bytes in size

Soru bize dosyanın bandit7 kullanıcısına ve bandit6 grubuna ait olduğu aynı zamanda 33 bayt boyutunda olduğu bilgisini vermiş.

Option	Explanation
/	Root Directory
-user uname	File is owned by user uname (numeric user ID allowed)
-group gname	File belongs to group gname (numeric group ID allowed)
-size n c	File uses n units of space. 'c' refer to bytes.

Bu tablodan yararlanarak şartlara uygun bir komut yazabilirim.

```
" find / -user bandit7 -group bandit6 -size 33c "
```

Bu shell komutu, sistemdeki tüm dizinleri tarar ve aşağıdaki şartları karşılayan dosyaları listeler:

/: Sistemdeki tüm dizinleri taramak için.

-user bandit7: bandit7 kullanıcısı tarafından oluşturulan dosyaları listelemek için.

-group bandit6: bandit6 grubunun sahip olduğu dosyaları listelemek için.

-size 33c: 33 baytlik dosyaları listelemek için.

```
bandit6@bandit:~$ find / -user bandit7 -group bandit6 -size 33c
```

```
bandit6@bandit:~$ find / -user bandit7 -group bandit6 -size 33c
find: '/dev/mqueue': Permission denied
find: '/dev/shm': Permission denied
find: '/var/spool/bandit24': Permission denied
find: '/var/spool/rsyslog': Permission denied
find: '/var/spool/cron/crontabs': Permission denied
find: '/var/crash': Permission denied
find: '/var/snap/lxd/common/lxd': Permission denied
find: '/var/cache/ldconfig': Permission denied
find: '/var/cache/apt/archives/partial': Permission denied
find: '/var/cache/apparmor/e10c1cf9.0': Permission denied
find: '/var/cache/apparmor/c47eabf7.0': Permission denied
find: '/var/cache/private': Permission denied
find: '/var/cache/pollinate': Permission denied
find: '/var/log': Permission denied
find: '/var/lib/update-notifier/package-data-downloads/partial': Permission denied
find: '/var/lib/polkit-1': Permission denied
find: '/var/lib/chrony': Permission denied
find: '/var/lib/snapd/void': Permission denied
find: '/var/lib/snapd/cookie': Permission denied
find: '/var/lib/amazon': Permission denied
/var/lib/dpkg/info/bandit7.password
find: '/var/lib/apt/lists/partial': Permission denied
find: '/var/lib/private': Permission denied
find: '/var/tmp': Permission denied
find: '/drifter/drifter14_src/axTLS': Permission denied
```

İçerideki bütün dosyaları deniyor ve bize “/var/lib/dpkg/info/bandit7.password” adlı dosyanın uyuştuğunu gösteriyor. Bu dosyayı “cat” komutuyla okutalım.

```
bandit6@bandit:~$ cat /var/lib/dpkg/info/bandit7.password
z7WtoNQU2XfjmMtWA8u5rN4vzqu4v99S
```

BANDİT SEVİYE 7 → SEVİYE 8

(man, grep, sort, uniq, strings, base64, tr, tar, gzip, bzip2, xxd)

Level Goal

The password for the next level is stored in the file **data.txt** next to the word **millionth**

Soruda şifrenin “data.txt” dosyasında “millionth” kelimesinin yanında olduğunu söylemiş. Bunu “grep” komutuyla halledebiliriz. Grep komutunun yanına istenilen kelimeyi koyduğumuzda bize o kelimeyle ilgili başlayan satırları gösterecektir.

Bilgi: İki komut arasında “|” işaretini kullanıldığında, birinci komutun çıktısı ikinci komutun girdisi olarak kullanılır.

```
bandit7@bandit:~$ cat data.txt | grep millionth
millionth      TESKZC0XvTetK0S9xNwm25STk5iWrBvP
```

BANDİT SEVİYE 8 → SEVİYE 9

(grep, sort, uniq, strings, base64, tr, tar, gzip, bzip2, xxd)

Level Goal

The password for the next level is stored in the file **data.txt** and is the only line of text that occurs only once

Soru bize “data.txt” dosyasında yalnızca bir kere geçen bir satırda olduğunu söylüyor. Biz bunun için “uniq” komutunu kullanabiliriz. Aynı olan satırları değil benzersiz satırları yazdırırmak istiyorsak yanına “-u” parametresini eklemeliyiz. Aynı zamanda uniq komutu yinelenen satırların alt alta olmasını bekler ve bu yüzden önce onları sıralamamız gereklidir “sort” komutunu “uniq” komutundan önce kullanarak bu sorunu çözebiliriz.

```
bandit8@bandit:~$ cat data.txt | sort | uniq -u
EN632PlfYiZbn3PhVK3X0GSlNInNE00t
```

“ cat data.txt | sort | uniq -u ” komutunu kullanarak bizden istediği kriterleri sağladık.

1. cat data.txt: data.txt dosyasının içeriğini standart çıktıya yazdırır.
2. sort: Standart çıktıdaki satırları sıralar.
3. uniq -u: Sıralanmış standart çıktıdaki tekrar eden satırları filtreleyerek sadece tekil satırları standart çıktıya yazdırır.

Bu komut, “data.txt” dosyasındaki tekrar eden satırları filtreleyerek sadece tekil satırların listesini verir. Soru bize data.txt dosyasında yalnızca bir kere geçen bir satırda olduğunu söyler.

BANDİT SEVİYE 9 → SEVİYE 10

(grep, sort, uniq, strings, base64, tr, tar, gzip, bzip2, xxd)

Level Goal

The password for the next level is stored in the file **data.txt** in one of the few human-readable strings, preceded by several '=' characters.

Soruda şifrenin “data.txt” dosyasında, insan tarafından okunabilen birkaç diziden birinde saklanır ve önünde birkaç tane '=' karakteri bulunduğu söylenmiştir. Strings komutu ile grep komutunu kullanarak bunu yapabiliriz.

```
bandit9@bandit:~$ ls
data.txt
bandit9@bandit:~$ strings data.txt | grep =
c_____ the
h;_____ password
_____ isT
n.E_____ G7w8LIIi6J3kTb8A7j9LgrywtEULyyyp6s
```

“ strings data.txt | grep == ” komutunu kullandık.

1. "strings" komutu, özellikle belgelere, uygulamalara veya diğer benzer dosyalara gömülü olarak saklanan metin verilerinin bulunması ve incelenmesi gibi durumlarda kullanışlıdır. Soruda bize insan tarafından okunabildiği bilgisi verildiği için sadece bu satırları çıktı olarak almak adına strings komutu kullanıyoruz.
2. grep ==: Standart çıktıdaki satırlar içinde == içeren satırları filtreleyerek standart çıktıya yazdırır.

Bu komut, data.txt dosyasında == içeren ve insan tarafından okunabilen satırları listeler

BANDİT SEVİYE 10 → SEVİYE 11

(grep, sort, uniq, strings, base64, tr, tar, gzip, bzip2, xxd)

Level Goal

The password for the next level is stored in the file **data.txt**, which contains base64 encoded data

Soruda şifrenin data.txt dosyasının içinde base64 ile şifrelendiğini söylemiştir. “base64 -d” komutunu kullanarak bulabiliriz. “-d” parametresi verilen kodu çözmek için kullanılır.

```
bandit10@bandit:~$ cat data.txt | base64 -d
The password is 6zPeziLdR2RKNdNYFNb6nVCKzphlXHBM
```

“ cat data.txt | base64 -d ” komutunu kullandık.

1. cat data.txt: data.txt dosyasının içeriğini standart çıktıya yazdırır.
2. base64 -d: Standart çıktıdaki veriyi Base64 şifrelenmiş veri olarak yorumlar ve şifreyi çözer, çözülmüş veriyi standart çıktıya yazdırır.

Bu komut, data.txt dosyasındaki veriyi Base64 şifreli veriden çözülmüş veriye çevirir.

BANDİT SEVİYE 11 → SEVİYE 12

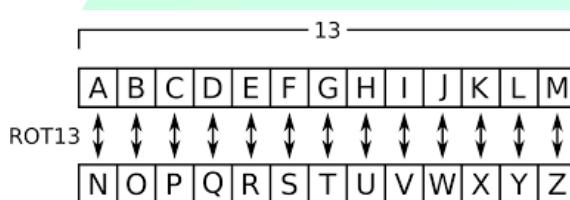
(grep, sort, uniq, strings, base64, tr, tar, gzip, bzip2, xxd)

Level Goal

The password for the next level is stored in the file **data.txt**, where all lowercase (a-z) and uppercase (A-Z) letters have been rotated by 13 positions

Soruda data.txt dosyasında rot13 ile şifrelenmiş bir parola olduğunu söylüyor. Bunu iki yol ile yapabiliyoruz. İnternetten hazır decoder kullanabiliriz ya da terminalde komut kullanarak yapabiliyoruz.

İşte rot13 kullanımını anlamamızı daha da kolaylaştıracak iki tablo.



Input	ABCDEFGHIJKLM NOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
Output	NOPQRSTUVWXYZ ABCDEFGHIJKLMnopqrstuvwxyz abcdefghijklm

" cat data.txt | tr 'A-Za-z' 'N-ZA-Mn-za-m' " komutunu kullanıyoruz ROT-13 algoritması için.

1. cat data.txt: data.txt dosyasının içeriğini standart çıktıya yazdırır.
2. tr 'A-Za-z' 'N-ZA-Mn-za-m': Standart çıktıdaki veriyi yer değiştirme işlemi yapar, tüm Latin alfabeteki harfleri 13 pozisyon (Caesar şifresi) sağa kaydırarak şifreler ve şifrelenmiş veriyi standart çıktıya yazdırır.

```
bandit11@bandit:~$ cat data.txt | tr 'A-Za-z' 'N-ZA-Mn-za-m'
The password is JVNBBFSmZwKKOP0XbFXOoW8chDz5yVRv
```

BANDİT SEVİYE 12 → SEVİYE 13

([grep](#), [sort](#), [uniq](#), [strings](#), [base64](#), [tr](#), [tar](#), [gzip](#), [bzip2](#), [xxd](#), [mkdir](#), [cp](#), [mv](#), [file](#))

Level Goal

The password for the next level is stored in the file `data.txt`, which is a hexdump of a file that has been repeatedly compressed. For this level it may be useful to create a directory under `/tmp` in which you can work using `mkdir`. For example: `mkdir /tmp/myname123`. Then copy the datafile using `cp`, and rename it using `mv` (read the manpages!)

Bu metin, bir dosyanın birden çok kez sıkıştırıldığı ve şifresinin "data.txt" dosyasında bulunduğu bir görev tarifi vermektedir. Görevin çözümünde "/tmp" altında çalışmak için bir dizin oluşturmanız önerilir.

Örneğin, "mkdir /tmp/ismim123". Daha sonra datafile dosyasını "cp" komutu ile kopyalayın ve "mv" komutu ile yeniden adlandırın.

Hatırlatma: Bir dosyadaki verileri görüntülemek için aşağıdaki komutlar kullanılabilir:

4. cat: Bu komut, dosyadaki verileri ekrana yazdırır. Örnek: "cat dosya.txt"
5. less: Bu komut, dosyadaki verileri sayfalama özelliği ile görüntüler. Örnek: "less dosya.txt"
6. head: Bu komut, dosyadaki ilk 10 satırı görüntüler. Örnek: "head dosya.txt"
7. tail: Bu komut, dosyadaki son 10 satırı görüntüler. Örnek: "tail dosya.txt"
8. hexdump: Bu komut, dosyadaki verileri onaltılık kodlamada görüntüler. Örnek: "hexdump dosya.txt"

Bu komutlar sadece birkaç örnek.

```
bandit12@bandit:~$ ls
data.txt
```

Dosyadaki verileri görüntüleyelim.

```
bandit12@bandit:~$ head data.txt
00000000: 1f8b 0808 8e0b bf63 0203 6461 7461 322e  .....c..data2.
00000010: 6269 6e00 013c 02c3 fd42 5a68 3931 4159  bin..< ... BZh91AY
00000020: 2653 598c b471 f700 0014 ffff fa59 c6c5 &SY..q.....Y..
00000030: af63 cfff af73 ffff bdb7 7c9f b1fb eafa .c ... s....|....
00000040: bfff fb9f f9fe bdbf ffef b001 3b2c .....;,
00000050: 5900 0341 a064 007a 8003 40d0 6869 a068 Y..A.d.z..@.hi.h
00000060: 3464 007a 81a0 0680 3401 90d0 6800 00d1 4d.z....4...h...
00000070: a0c9 a680 f51e 9a83 27a4 3d4f 4991 0000 .....'.=OI ...
00000080: 69a6 803d 4001 9001 a686 8c40 0d00 d3d2 i.=@.....@....
00000090: 0d00 0d06 08f5 0323 4034 069e a340 0da8 .....#@4 ... @...
```

Bilgi:

1. Onaltılık kodların tanımlanması: Onaltılık kodlamada, her bir bayt (8 bit) iki farklı onaltılık rakam ile ifade edilir. Bu rakamlar 0-9 arasında olabilir ve A-F harfleri ile temsil edilir.

2. Onaltılık kodların okunması: Eğer bir dosyanın içeriği onaltılık kodlama ile görüntülenirse, her bir bayt için iki farklı onaltılık rakam görünecektir. Bu rakamlar, bir dosyanın onaltılık verilerden oluştuğunu gösterir.

Verilere baktığımızda dosyanın onaltılık verilerden oluştuğunu görüyoruz.
 Binary (ikilik) veri, '0' ve '1' lerden oluşan bir veri türüdür. Gerçek dosyayı geri almak için bu onaltılık veriyi ikiliye dönüştürmeliyiz.
 Ancak bunları yapmadan önce geçerli konumda yeni dosyalar oluşturma iznimiz olmadığından dizide geçici bir çalışma dizini oluşturmak gereklidir.
 (/tmp dizini, sisteminizdeki geçici verileri saklamak için tasarlandı ve genellikle her kullanıcı için yazma izni verilir.)
 "mkdir" komutu ile bunu yapabiliriz.

Yeni dizine geçmek için de cd komutunu kullanabiliriz.

```
bandit12@bandit:~$ mkdir /tmp/sun
bandit12@bandit:~$ cd /tmp/sun
bandit12@bandit:/tmp/sun$
```

Artık data.txt bu yeni yere taşımamız gerekiyor. "cp" komutunu kullanarak bunu yapabiliriz. Ardından ".txt" dosyasının bir metin dosyası olmadığını bildiğimiz için uzantıyı kaldırmak için dosyayı yeniden adlandırmamız gerekiyor.

```
bandit12@bandit:/tmp/sun$ cp ~/data.txt .
bandit12@bandit:/tmp/sun$ ls
data.txt
bandit12@bandit:/tmp/sun$ mv data.txt data
bandit12@bandit:/tmp/sun$ ls
data
```

Artık veriler yeni bir dizin olduğuna göre verilenleri ikilik sisteme dönüştürmek için "xxd" komutunu kullanırız.

```
bandit12@bandit:/tmp/sun$ xxd -r data > binary
bandit12@bandit:/tmp/sun$ ls
binary data
```

Binary dosyasının türüne baktığımızda dosyanın gzip kullanılarak sıkıştırıldığını görebiliriz.

```
T*♦♦♦N$#-}♦9♦♦}<bandit12@bandit:/tmp/sun$ file binary
binary: gzip compressed data, was "data2.bin", last modified: Wed Jan 11 19
:18:38 2023, max compression, from Unix, original size modulo 2^32 572
```

Bir gzip dosyası adını açmaya çalışırken dosyanın doğru uzantıya sahip olması önemli bu yüzden dosyanın "mv" komutıyla uzantısını değiştireceğiz.

```
bandit12@bandit:/tmp/sun$ mv binary binary.gz
bandit12@bandit:/tmp/sun$ ls
binary.gz data
```

```

bandit12@bandit:/tmp/sun$ gunzip binary.gz
bandit12@bandit:/tmp/sun$ ls
binary data
bandit12@bandit:/tmp/sun$ file binary
binary: bzip2 compressed data, block size = 900k
bandit12@bandit:/tmp/sun$ bunzip2 binary
bunzip2: Can't guess original name for binary -- using binary.out
bandit12@bandit:/tmp/sun$ ls
binary.out data

```

Verilerin bzip2 kullanılarak sıkıştırıldığını görüyoruz bu bzip2 dosyasını açmak için “bunzip2” komutunu kullanabiliriz.

Not: “bzip2 -d” komutu “bunzip2” komutunun kısaltılmasıdır.

Şimdi de “binary.out” dosyasının gzip kullanılarak sıkıştırıldığını görebilirsiniz. Bunu bunzip2 komutu kullanarak açabiliriz.

```

bandit12@bandit:/tmp/sun$ file binary.out
binary.out: gzip compressed data, was "data4.bin", last modified: Wed Jan 1
1 19:18:38 2023, max compression, from Unix, original size modulo 2^32 2048
0
bandit12@bandit:/tmp/sun$ mv binary.out binary.gz
bandit12@bandit:/tmp/sun$ ls
binary.gz data
bandit12@bandit:/tmp/sun$ gunzip binary.gz
bandit12@bandit:/tmp/sun$ ls
binary data

```

File komutu kullanarak dosyada bulunan veri türüne bakarız. Verilerin bir tar arşivine kaydedildiğini görürüz. Bir tar dosyasını ayıklamak için “tar” komutunu kullanırız.

```

bandit12@bandit:/tmp/sun$ file binary
binary: POSIX tar archive (GNU)
bandit12@bandit:/tmp/sun$ tar -xf binary
bandit12@bandit:/tmp/sun$ ls
binary data data5.bin

```

Dosya tipini görmek için “file” komutunu kullanıyoruz. Parola dosyası “tar”, “gzip” ve “bzip2” kullanılarak ardışık olarak sıkıştırılmış gibi görünüyor. Yukarıdaki adımları tekrarlamaya devam ediyoruz.

```

bandit12@bandit:/tmp/sun$ file data5.bin
data5.bin: POSIX tar archive (GNU)
bandit12@bandit:/tmp/sun$ tar -xf data5.bin
bandit12@bandit:/tmp/sun$ ls
binary data data5.bin data6.bin
bandit12@bandit:/tmp/sun$ file data6.bin
data6.bin: bzip2 compressed data, block size = 900k
bandit12@bandit:/tmp/sun$ bunzip2 data6.bin
bunzip2: Can't guess original name for data6.bin -- using data6.bin.out
bandit12@bandit:/tmp/sun$ ls
binary data data5.bin data6.bin.out
bandit12@bandit:/tmp/sun$ file data6.bin.out
data6.bin.out: POSIX tar archive (GNU)
bandit12@bandit:/tmp/sun$ tar -xf data6.bin.out
bandit12@bandit:/tmp/sun$ ls
binary data data5.bin data6.bin.out data8.bin
bandit12@bandit:/tmp/sun$ file data8.bin
data8.bin: gzip compressed data, was "data9.bin", last modified: Wed Jan 11
19:18:38 2023, max compression, from Unix, original size modulo 2^32 49
bandit12@bandit:/tmp/sun$ gunzip data8.bin
gzip: data8.bin: unknown suffix -- ignored
bandit12@bandit:/tmp/sun$ ls
binary data data5.bin data6.bin.out data8
bandit12@bandit:/tmp/sun$ mv data8.bin data8.gz
bandit12@bandit:/tmp/sun$ gunzip data8.gz
bandit12@bandit:/tmp/sun$ ls
binary data data5.bin data6.bin.out data8
bandit12@bandit:/tmp/sun$ file data8
data8: ASCII text
bandit12@bandit:/tmp/sun$ cat data8
The password is wbWdlBxEir4CaE8LaPhauuOo6pwRmrDw

```

İşte şifreyi verdi.

BANDİT SEVİYE 13 → SEVİYE 14

([ssh](#), [telnet](#), [nc](#), [openssl](#), [s_client](#), [nmap](#))

Level Goal

The password for the next level is stored in `/etc/bandit_pass/bandit14` and can only be read by user **bandit14**. For this level, you don't get the next password, but you get a private SSH key that can be used to log into the next level. Note: **localhost** is a hostname that refers to the machine you are working on

Aşama 14'deki şifre, `/etc/bandit_pass/bandit14` dosyasında saklanır ve sadece bandit14 kullanıcı tarafından okunabilir. Sonraki aşamada şifreyi almazsınız ancak sonraki aşamaya giriş yapmak için kullanabileceğiniz bir özel SSH anahtarları alırsınız.

Bilgi: localhost çalıştığınız makineye atanmış bir makine adıdır.

Makinede oturum açıp "ls" komutunu çalıştırıldığında, ipucunun bahsettiği SSH özel anahtarını olması gereken "sshkey.private" adlı bir dosya görünüyor.

```
bandit13@bandit:~$ ls
sshkey.private
```

bandit14 kullanıcısının profiline girmek için

“ssh -i sshkey.private bandit14@bandit.labs.overthewire.org -p 2220” komutunu çalıştırıldım.

Bilgi: “ssh -i sshkey.private” komutunu açıklayalım.

1. ssh: SSH (Secure Shell) protokolünü kullanarak uzak makineye bağlanmak için kullanılan bir komut istemidir.
2. -i sshkey.private: -i seçeneği, SSH oturumu için kullanılacak bir anahtar dosyasını belirtir. sshkey.private dosyası, güvenli bir şekilde uzak makineye bağlanmak için kullanılacak bir SSH anahtarıdır.

Bu komut, “sshkey.private” anahtarını kullanarak güvenli bir şekilde uzak makineye bağlanmayı sağlar. SSH anahtarı kullanımı, şifreli bir şekilde uzak makineye giriş yapmak için güvenli bir alternatifdir ve kullanıcı adı ve şifre girişi yerine kullanılabilir. Biz de bu soruda bağlantımızı bu şekilde sağladık.

“-i” parametresi sunucudaki bandit14'te oturum açmak için bir kimlik dosyası kullandığım anlamına gelir.

```
bandit13@bandit:~$ ssh -i sshkey.private bandit14@bandit.labs.overthewire.org -p 2220
The authenticity of host '[bandit.labs.overthewire.org]:2220 ([127.0.0.1]:220)' 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'te oturum açtıktan sonra ipucunda belirtildiği gibi

“cat /etc/bandit_pass/bandit14” komutunu çalıştırıldım ve ardından bize bir sonraki şifreyi veren bandit14 dosyasını okudum.

```
bandit14@bandit:~$ cat /etc/bandit_pass/bandit14
fGrHPx402xGC7U7rXKDaxiWFT0iF0ENq
```

Bağlantıyı sağladıkten sonra dosyayı okuyabilmemizin sebebi sadece Bandit14 kullanıcının dosyayı okuma yetkisinin olmasıdır.

BANDİT SEVİYE 14 → SEVİYE 15

([ssh](#), [telnet](#), [nc](#), [openssl](#), [s_client](#), [nmap](#))

Level Goal

The password for the next level can be retrieved by submitting the password of the current level to **port 30000 on localhost**.

“Bu metin, şu anki seviyenin şifresinin, yerel makine olan localhost'un 30000 numaralı portuna gönderilmesi ile bir sonraki seviyenin şifresinin elde edilebileceğini ifade eder.”

Bir sonraki seviyenin şifresi, geçerli seviyenin şifresini localhost'ta 30.000 numaralı bağlantı noktasına göndererek alınabilir.

Hatırlatma: nc (Netcat) komutu, TCP ve UDP ağ protokollerini kullanarak veri gönderme ve alma işlemleri yapmak için kullanılan bir araçtır.

Bunun için “nc localhost 30.000 ” komutunu kullanacağız.

Bilgi:

1. localhost: Bu, bağlanmak istediğimiz hedef makinenin adı veya IP adresidir. localhost ifadesi, bulunduğuımız makinenin kendisini ifade eder. Veriyi bulunduğuımız makineden göndereceğimiz için bu ifadeyi kullandık.
2. 30000: Bu, hedef makinenin dinlediği TCP port numarasıdır. nc programı, localhost makinesinin 30000 numaralı portundaki bir TCP bağlantısı açmayı denemektedir.

Bu komut, localhost makinesinin 30000 numaralı TCP portundaki bir bağlantıyı açmayı denemektedir. Hedef makine 30000 numaralı portu dinliyorsa ve bağlantı kurulabiliyorsa, nc programı aracılığıyla veri okuma ve yazma işlemi için kullanılabilir.

```
bandit14@bandit:~$ nc localhost 30000
fGrHPx402xGC7U7rXKDaxiWFT0iF0ENq
Correct!
jN2kgmIXJ6fShzhT2avhotn4Zcka6tnt
```

BANDİT SEVİYE 15 → SEVİYE 16

([ssh](#), [telnet](#), [nc](#), [openssl](#), [s_client](#), [nmap](#))

Level Goal

The password for the next level can be retrieved by submitting the password of the current level to port 30001 on localhost using SSL encryption.

Helpful note: Getting "HEARTBEATING" and "Read R BLOCK"? Use `-ign_eof` and read the "CONNECTED COMMANDS" section in the manpage. Next to 'R' and 'Q', the 'B' command also works in this version of that command...

"Gelecek seviyenin şifresi, mevcut seviyenin şifresini localhost üzerindeki 30001 numaralı porta SSL şifrelemesi kullanarak göndermenizle elde edilebilir.

Yardımcı not: "HEARTBEATING" ve "Read R BLOCK" mesajlarını alıyorsanız "`-ign_eof`" seçeneğini kullanın ve man sayfasındaki "CONNECTED COMMANDS" bölümünü okuyun. Bu komutun bu sürümünde 'R' ve 'Q' yanında, 'B' komutu da çalışır."

"`-ign_eof`" seçeneği, girdi EOF'unun yok sayılmasını sağlar.

Bilgi: EOF (End of File) terimi, bir dosyanın ya da veri akışının sonunu ifade eder. EOF'un yok sayılması, "openssl s_client" komutunun, belirtilen veri akışındaki dosya sonunu gözardı etmesi anlamına gelir. Yani "openssl s_client" komutu veri akışının sonunda EOF işaretini görürse normal şartlarda işlemini sonlandırır. Ancak "`-ign_eof`" seçeneği ile EOF işaretinin yok sayılması istenebilir, böylece veri akışının sonunda EOF işaretini görülse bile komutun devam etmesi sağlanabilir.

SSL şifreleme protokolü, verinin güvenli bir şekilde gönderilmesini sağlamak için kullanılan bir şifreleme teknigidir. Veriyi göndermek istediğiniz porta (örneğin, 30001), OpenSSL s_client komutu kullanılarak bağlanabilirsiniz. Bağlantı, "`-ign_eof`" ve "`-connect localhost:30001`" seçenekleri ile yapılabilir. Bu seçenekler, bağlantının ne zaman sonlandırılacağını belirler ve verilerin şifreli bir şekilde gönderilmesini sağlar.

Bir sonraki seviyenin şifresi, mevcut seviyenin şifresini SSL şifrelemesi kullanılarak localhost'ta 30001 numaralı bağlantı noktasına göndererek alınabilir.

```
bandit15@bandit:~$ openssl s_client -connect localhost:30001
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN = localhost
```

OpenSSL açık kaynak kodlu güvenlik paketi içindeki bir araçtır ve SSL/TLS protokolüne dayalı bir bağlantı kurmak için kullanılır. Aşağıdaki işlemleri gerçekleştirir:

1. `openssl s_client`: OpenSSL araçları arasında bir bağlantı kurmak için kullanılan bir araçtır.
2. `-connect localhost:30001` : Bu, hedef makinenin adı veya IP adresi ile dinlediği TCP port numarasıdır. localhost ifadesi, bulunduğuuz makinenin kendisini ifade eder. 30001 numaralı port, hedef makinenin SSL/TLS bağlantıları için dinlediği porttur.

Bu komut, localhost makinesinin 3000ı numaralı TCP portundaki bir SSL/TLS bağlantısını açmayı denemektedir. Bağlantı kurulduysa, veri güvenli bir şekilde göndermek ve almak için kullanılabilir.

```
read R BLOCK
jN2kgmIXJ6fShzhT2avhotn4Zcka6tnt
Correct!
JQttfApK4SeyHwDlI9SXGR50qcl0Ail1

closed
```

BANDİT SEVİYE 16 → SEVİYE 17

([ssh](#), [telnet](#), [nc](#), [openssl](#), [s_client](#), [nmap](#))

Level Goal

The credentials for the next level can be retrieved by submitting the password of the current level to a port on localhost in the range 31000 to 32000. First find out which of these ports have a server listening on them. Then find out which of those speak SSL and which don't. There is only 1 server that will give the next credentials, the others will simply send back to you whatever you send to it.

“Gelecek seviyenin kimlik bilgileri, mevcut seviyenin şifresini localhost'daki 31000 ile 32000 arasındaki bir porta göndermenizle alınabilir. İlk olarak bu portlardan hangilerinin bir sunucu tarafından dinlendiğini bulun. Daha sonra bu portların hangilerinin SSL konuştuğunu ve hangilerinin konuşmadığını bulun. Sadece bir sunucu gelecek kimlik bilgilerini verecektir, diğerleri sadece size gönderdiğiniz her şeyi size geri gönderecektir.”

Hangi ana bilgisayarın SSL çalıştığını bulmamız gerekiyor. Bunu yapmak için, 31000'den 32000'e kadar her bağlantı noktasını kontrol edecek ve o bağlantı noktasında hangi hizmetlerin çalıştığını kontrol edecek bir nmap taraması çalıştırabiliriz.

```
bandit16@bandit:~$ nmap -sV localhost -p 31000-32000
Starting Nmap 7.80 ( https://nmap.org ) at 2023-02-02 14:35 UTC
Stats: 0:00:16 elapsed; 0 hosts completed (1 up), 1 undergoing Service scan
Timing: About 0.00% done
```

“nmap -sV localhost -p 31000-32000”

1. nmap: Bu açık kaynak kodlu bir ağ tarama aracıdır.
2. -sV: Bu seçenek, taranacak hedeflerin sunucu türlerini ve sürüm numaralarını belirlemek için kullanılır.
3. localhost: Bu, taranacak hedefin IP adresi veya anahtar kelime olarak belirtilir. localhost ifadesi, bulunduğuımız makinenin kendisini ifade eder.
4. -p 31000-32000: Bu seçenek, taranacak port aralığını belirtir. Bu komutta, 31000 ile 32000 arasındaki portlar taranacaktır.

Bu komut, localhost makinesinde 31000 ile 32000 arasındaki portları taramaya çalışır ve taranacak hedeflerin sunucu türlerini ve sürüm numaralarını belirlemek için -sV seçeneği

kullanılır. Sonuçta, hangi portlar açık olduğu ve bu portların hangi sunucuları ve sürümleri tarafından kullanıldığı görüntülenecektir.

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	

Bu görselde görüldüğü gibi nmap bize beş portun açık olduğunu söylüyor. Yalnızca iki bağlantı noktası (31518 ve 31790) SSL kullanır. Nmap ayrıca bize 31518 numaralı bağlantı noktasının yalnızca echo hizmetini çalıştırduğunu söyler. Gelecek vaat eden bağlantı noktası, bilinmeyen bir hizmeti çalıştırınan 31790 numaralı bağlantı noktası gibi görünüyor.

Şimdi yine OpenSSL kullanarak localhost üzerinde bu porta bağlanıp şifreyi gönderiyoruz.

```
openssl s_client -connect localhost:31790
```

```
JQttfApK4SeyHWDT19SXGR50qcL0A1t1
Correct!
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAvmOkuifmMg6HL2YPI0jon6iWfbp7c3jx34YkYWqUH57SUdyJ
imZzeyGC0gtZPGujUSxiJSWI/oTqexh+cAMTSMl0Jf7+BrJ0bArnxd9Y7YT2bRPQ
Ja6Lzb558YW3FZl870Ri0+rW4LCDNd2lUvLE/GL2GWyuKN0K5iCd5TbtJzEkQTu
DSt2mcNn4rhAL+JFr56o4T6z8WWAW18BR6vGrMa70/kALHYW30ekePOAzl0VUYbW
```

Sonuç, özel bir “ssh” anahtarıdır. Bu nedenle, anahtarını koymak için bir dosya oluşturuyoruz.

```
bandit16@bandit:~$ mkdir -p /tmp/bandit17
```

```
bandit16@bandit:~$ cd /tmp/bandit17
```

```
bandit16@bandit:/tmp/bandit17$ nano sshkey17
-----
```

Sshkey17 dosyasına aldığımız privatekey'i koyuyoruz.

Diger kullanıcıların dosya erişememesi için dosyanın izinlerini değiştirin.

(Sadece yönetici tarafından okunabilir yaptık çünkü sadece Bandit17 kullanıcısının okumasını istiyoruz.)

```
bandit16@bandit:/tmp/bandit17$ chmod 400 sshkey17
```

```
-r----- 1 bandit16 bandit16 1675 Feb 2 14:51 sshkey17
```

Artık bir sonraki seviyeye geçmek için anahtar dosyasını “ssh” komutuyla kullanabiliriz.

```
bandit16@bandit:/tmp/bandit17$ ssh -i sshkey17 bandit17@bandit.labs.overthewire.org -p 2220
The authenticity of host '[bandit.labs.overthewire.org]:2220 ([127.0.0.1]:220)' can't be established.
ED25519 key fingerprint is SHA256:C2ihUBV7ihN1wUXRb4RrEcLfXC5CXlhmAAM/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/bandit16/.ssh' (Permission denied).
Failed to add the host to the list of known hosts (/home/bandit16/.ssh/known_hosts).
```



Parolalar “/etc/bandit_pass” içinde saklandığı için de giriş yaptıktan sonra Bandit17’nin parolasına bakıyoruz.

```
bandit17@bandit:~$ cat /etc/bandit_pass/bandit17
VwOSWtCA7lRKkTfbr2IDh6awj9RNZM5e
```

BANDİT SEVİYE 17 → SEVİYE 18

([cat](#), [grep](#), [ls](#), [diff](#))

Level Goal

There are 2 files in the homedirectory: `passwords.old` and `passwords.new`. The password for the next level is in `passwords.new` and is the only line that has been changed between `passwords.old` and `passwords.new`.

NOTE: if you have solved this level and see 'Byebyel' when trying to log into bandit18, this is related to the next level, bandit19

Ana dizinde 2 dosya vardır: `passwords.old` ve `passwords.new`. Bir sonraki seviyenin parolası `passwords.new` içindedir ve `passwords.old` ile `passwords.new` arasında değiştirilen tek satırdır. Bunu “diff” komutuyla bulabiliriz.

Eğer her iki dosya da aynı değilse, “diff” komutu farkları gösterir.

```
bandit17@bandit:~$ ls
passwords.new  passwords.old
bandit17@bandit:~$ diff passwords.new passwords.old
42c42
< hga5tuuCLF6ffFzUpnagiMN8ssu9LFrdg
—
> 810zq8IK64u5A9Lb2ibdTGBtlcSZsoe8
```

İki şifre verdi ve yalnızca 1. Şifre çalıştı. Diğer kullanıcıya geçiş yapabilirsiniz.

BANDİT SEVİYE 18 → SEVİYE 19

(ssh, ls, cat)

Level Goal

The password for the next level is stored in a file **readme** in the homedirectory. Unfortunately, someone has modified **.bashrc** to log you out when you log in with SSH.

“Sonraki seviyenin şifresi, ev dizinindeki "readme" dosyasında saklanır. Ne yazık ki, biri SSH ile giriş yaptığınızda sizi oturumunuzu kapatacak şekilde “ .bashrc” yi değiştirdi.”

Not: “ .bashrc” kullanıcıların bash shell tarafından yüklenecek olan bir yapılandırma dosyasıdır. Bu dosya, kullanıcının bash shell'i başlatması sırasında çalıştırılan komutları, ayarları ve çevre değişkenlerini belirler. Kullanıcının bash shell'de yapmasını istediği her şeyi “ .bashrc” dosyasında tanımlayabilir ve bu dosya her shell başlatıldığında yüklenir.

Hatırlatma: SSH, sadece bir makineye uzaktan giriş yapmanızı izin vermekle kalmaz, aynı zamanda ortak SSH ifadesinden sonra eklenen komutların uzaktan çalıştırmasına da oylanır tanır.

Giriş yapar yapmaz oturumumuz sonlandığından dolayı SSH'ın bu özelliğinden faydalanaçagız.

Öncelikle “ls” komutuyla dosyamız orada mı kontrol ediyoruz.

```
(kali㉿duygu)-[~]
$ ssh bandit18@bandit.labs.overthewire.org -p 2220 ls
[...]
This is an OverTheWire game server.
More information on http://www.overthewire.org/wargames
bandit18@bandit.labs.overthewire.org's password:
```

Evet soruda da söylediğim gibi “readme” adlı dosyayı okumamız gerekiyor yine aynı şekilde dışarıdan bunu okuyabiliriz.

```
(kali㉿duygu) - [~]
$ ssh bandit18@bandit.labs.overthewire.org -p 2220 cat readme
This is an OverTheWire game server.
More information on http://www.overthewire.org/wargames

bandit18@bandit.labs.overthewire.org's password:
awhqfNnAbc1naukrpqDYcF95h7HoMTrc
```

BANDİT SEVİYE 19 → SEVİYE 20

Level Goal

To gain access to the next level, you should use the setuid binary in the homedirectory. Execute it without arguments to find out how to use it. The password for this level can be found in the usual place (/etc/bandit_pass), after you have used the setuid binary.

“Önceki seviyeden geçmek için ev dizinindeki setuid binary dosyasını kullanmanız gereklidir. Argüman vermeden çalıştırarak nasıl kullanacağınızı öğrenin. Bu seviyenin şifresi, setuid binary dosyasını kullandıktan sonra alışlageldiği yerde (/etc/bandit_pass) bulunabilir.”

Bilgi: “setuid binary” dosyası, kullanıcılar tarafından çalıştırılması durumunda, çalıştırılan dosyaya sahip olan kullanıcının haklarını kullanmasına izin veren bir tür dosyadır. Örneğin, root kullanıcısına ait bir “setuid binary” dosyası normal bir kullanıcı tarafından çalıştırıldığında root haklarıyla çalışmasına izin verir.

“ls” komutunu çalıştırıldım ve “bandit20-do” adlı ikili dosyayı gördüm.

Bandit20 kullanıcısının yetkilerine sahip olduğundan yalnızca Bandit20 kullanıcısının okuyabileceği “/etc/bandit_pass/bandit20” dizinini bu dosyayı çalıştırarak okuyabilirim.

```
bandit19@bandit:~$ ls
bandit20-do
```

İpucu, dosyayı çalıştırımadım gerektiğini belirtti, bu yüzden “./bandit20-do” komutıyla dosyayı çalıştırıldım. Çıktısı şu şekildeydi “Başka bir kullanıcı olarak bir komut çalıştırın. Örnek: “./bandit20-do id”. Daha sonra “./bandit20-do id” komutunu çalıştırıldım ve bandit20'nin kimlik bilgilerini görebildim.

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

Bu dosyanın bir “setuid binary” dosyası olduğunu ve Bandit20 kullanıcısının yetkilerine sahip olduğunu buradan anlayabiliriz.

Öyleyse yalnızca Bandit20 kullanıcısının okuyabileceği “/etc/bandit_pass/bandit20” dizinini bu dosayı çalıştırarak okuyabiliyim.

```
bandit19@bandit:~$ ./bandit20-do cat /etc/bandit_pass/bandit20
VxCazJaVykI6W36BkB0mJTCM8rR95XT
timed out waiting for input: auto-logout
```

Ve evet şifreyi okuyabiliyorum.

BANDİT SEVİYE 20 → SEVİYE 21

([ssh](#), [nc](#), [cat](#), [bash](#), [screen](#), [tmux](#), [Unix ‘job control’ \(bg, fg, jobs, &, CTRL-Z, ...\)](#))

Level Goal

There is a setuid binary in the homedirectory that does the following: it makes a connection to localhost on the port you specify as a commandline argument. It then reads a line of text from the connection and compares it to the password in the previous level (bandit20). If the password is correct, it will transmit the password for the next level (bandit21).

NOTE: Try connecting to your own network daemon to see if it works as you think

“Homedirectory’de bir setuid binary bulunmaktadır.

Aşağıdaki şeyleri yapar: komut satırı argümanı olarak belirttiğiniz porta localhost'a bağlantı kurar. Daha sonra bağlantıdan bir metin satırı okur ve önceki seviyedeki şifreye (bandit20) karşılaştırır. Şifre doğruysa, sonraki seviyedeki şifreyi (bandit21) iletecektir.

NOT: Çalışma şeklini düşündüğünüz gibi çalışıp çalışmadığını görmek için kendi ağınızdaki sunucuya bağlanmayı deneyin.”

'Netcat'i kullanarak, gelen bağlantıyı dinleyen sunucu modunda bir bağlantı oluşturabiliriz. Başlangıç olarak 1254 numaralı port üzerinde dinleyen bir TCP sunucusu oluşturuldu ve bu sunucunun arka planda çalışmasına izin verildi. (2.terminalle beraber çalışacağı için bu izne ihtiyacımız var.)

Netcat'in şifreyi göndermesini sağlamak için, “echo” komutunu ve onu netcat'e aktarırmı. -n, girişte yeni satır karakterlerini önlemek içindir. (Bu işlemi 1.terminalde gerçekleştiriyoruz.)

```
bandit20@bandit:~$ echo -n "VxCazJaVykI6W36BkB0mJTCM8rR95XT" | nc -l -p 1254 &
[1] 189710
```

“ echo -n “dosyaadi” | nc -l -p 1254 & ” komutunu açıklayalım.

1. “echo -n ‘dosyaadi’” kısmı, “dosyaadi” adında bir dosya adını ekrana yazdırır, ancak dosya satır sonu karakterini eklemez.
2. “nc -l -p 1254” kısmı, “netcat” (nc) programını çalıştırır. “-l” seçeneği sunucu modunda çalışmasını sağlar ve “-p 1254” seçeneği, dinlenecek port numarasını belirler.

3. "&" simbolü, komutun arka planda çalışmasını sağlar. Bu, kullanıcının ekrandaki işleme devam edebilmesini ve komutun arka planda çalışmasını sürdürmesini sağlar.

Bu komut, yerel ağ üzerinde veri aktarımı yapmak isteyen kullanıcılar için kullanışlıdır.

1234 numaralı bağlantı noktasıyla çalıştırılması, netcat sunucumuza bağlanacağı, girilen parolayı “echo” ile alacağı ve bir sonraki parolayı geri göndereceği anlamına gelir.

Not: Soruda bahsedilen setuid binary dosyası “./suconnect” dosyasıdır.

localhost'u 1254 bağlantı noktası ile bağlamak için setuid ikili dosyasını yürütüyoruz. (Bu işlemi 2. terminalde gerçekleştiriyoruz.)

```
bandit20@bandit:~$ ./suconnect 1254
Read: VxCazJaVykI6W36BkBU0mJTCM8rR95XT
Password matches, sending next password
NvEJF7oVjkddltPSrdKEF0llh9V1IBcq
```

BANDİT SEVİYE 21 → SEVİYE 22

([cron](#), crontab, crontab(5) (use “man 5 crontab” to access this))

Level Goal

A program is running automatically at regular intervals from **cron**, the time-based job scheduler. Look in **/etc/cron.d/** for the configuration and see what command is being executed.

“Bu bir cron tarafından düzenli aralıklarla çalıştırılan bir programdır. Yapılandırmayı **/etc/cron.d/** dizininde bulun ve çalıştırılan komutu görün.”

Bilgi: Cronjobs, düzenli aralıklarda otomatik olarak çalışan programları ifade eder. Linux işletim sisteminde bu cronjobs'lar için birden fazla klasör bulunmaktadır. Bunlardan **crond.d**: Sistem yöneticilerinin veya uygulamaların oluşturduğu özel cron görevlerinin yapılandırmasını içeren bir dizindir.

Bu seviye için ilk olarak, '/etc/cron.d' klasörünü bakıyoruz.

```
bandit21@bandit:~$ cd /etc/cron.d/
bandit21@bandit:/etc/cron.d$ ls
cronjob_bandit15_root  cronjob_bandit23      e2scrub_all
cronjob_bandit17_root  cronjob_bandit24      otw-tmp-dir
cronjob_bandit22       cronjob_bandit25_root  sysstat
bandit21@bandit:/etc/cron.d$ cat cronjob_bandit22
@reboot bandit22 /usr/bin/cronjob_bandit22.sh >& /dev/null
* * * * * bandit22 /usr/bin/cronjob_bandit22.sh >& /dev/null
```

Bu cronjob “/usr/bin/cronjob_bandit22.sh” dosyasını bandit22 kullanıcısı olarak çalıştırır.

Her gün her dakika çalıştırıldığını gösteren beş yıldızı(*****) görüyoruz, tam olarak nein yürütüldüğünü anlamak için bash dosyasına bakalım.

“ /usr/bin/cronjob_bandit22.sh” komutu, belirli bir yol üzerinde bulunan bir bash script dosyasını ifade etmektedir.

```
bandit21@bandit:/etc/cron.d$ cat /usr/bin/cronjob_bandit22.sh
#!/bin/bash
chmod 644 /tmp/t706lds9S0RqQh9aMcz6ShpAoZKF7fgv
cat /etc/bandit_pass/bandit22 > /tmp/t706lds9S0RqQh9aMcz6ShpAoZKF7fgv
```

“ /usr/bin/cronjob_bandit22.sh” komut dosyası, ‘tmp’ klasöründe yeni bir dosya oluşturur ve herkesin bu dosyayı okumasına izin verir. Aynı zamanda, bandit22 parola dosyasının içeriği de bu yeni oluşturulmuş dosyaya kopyalanır. (t7O6lds9SoRqQh9aMcz6ShpAoZKF7fgv dosyasına kopyalanır.)

İçine bir bakalım.

```
bandit21@bandit:/etc/cron.d$ cat /tmp/t706lds9S0RqQh9aMcz6ShpAoZKF7fgv
WdDozAdTM2z9DiFEQ2mGlwngMfj4EZff
```

BANDİT SEVİYE 22 → SEVİYE 23

([cron](#), crontab, crontab(5) (use “man 5 crontab” to access this))

Level Goal

A program is running automatically at regular intervals from [cron](#), the time-based job scheduler. Look in [/etc/cron.d/](#) for the configuration and see what command is being executed.

NOTE: Looking at shell scripts written by other people is a very useful skill. The script for this level is intentionally made easy to read. If you are having problems understanding what it does, try executing it to see the debug information it prints.

“**Cron**, zaman tabanlı görev planlayıcısından, bir program düzenli aralıklarla otomatik olarak çalışır. Yapılandırma dosyasını görüntülemek için [/etc/cron.d/](#) dizinine bakın ve çalıştırılan komutu görün.

NOT: Başka insanlar tarafından yazılmış shell betiklerine bakmak çok yararlı bir beceri. Bu seviyedeki betik, okumayı kolay hale getirmek amacıyla tasarlandı. Eğer ne yaptığıni anlamakta sorun yaşıyorsanız, hata ayıklama bilgileri görüntülemek için çalıştmayı deneyin.”

Soruda bize “[/etc/cron.d/](#) dosyasına bakın ve hangi komutun yürütülmekte olduğunu görün” denmiş.

```
bandit22@bandit:~$ cd /etc/cron.d/
bandit22@bandit:/etc/cron.d$ ls
cronjob_bandit15_root cronjob_bandit23      e2scrub_all
cronjob_bandit17_root cronjob_bandit24      otw-tmp-dir
cronjob_bandit22       cronjob_bandit25_root sysstat
bandit22@bandit:/etc/cron.d$ cat cronjob_bandit23
@reboot bandit23 /usr/bin/cronjob_bandit23.sh &> /dev/null
* * * * * bandit23 /usr/bin/cronjob_bandit23.sh &> /dev/null
```

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

```
bandit22@bandit:/etc/cron.d$ 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
```

Bu komut dosyası, değişken tanımlaması yapar. İlk tanımlanan değişken "myname" dir ve komut çıktısını saklar. Dosya, bandit23 olarak çalıştırılacağından, bandit23 değerini yazdırır.

Son satır ise, bandit23'den gelen şifrenin /tmp klasöründeki bir dosyaya yazılacağını belirtir. Dosya adını bu satır oluşturur.

```
bandit22@bandit:/etc/cron.d$ echo I am user bandit23 | md5sum | cut -d ' ' -f 1
8ca319486bfbb3663ea0fbe81326349
bandit22@bandit:/etc/cron.d$ cat /tmp/8ca319486bfbb3663ea0fbe81326349
QYw0Y2aiA672PsMmh9puTQuhoz8SyR2G
```

Bu komut, "I am user bandit23" metnini md5sum komutu ile hashler ve sonuçlarından sadece ilk değeri (hash değeri) cut komutu ile alır. Sonuç, hashlenmiş metnin ilk değeridir.

Bu kısım bittiğinde hash'i "mytarget" adlı bir değişkene kaydeder, hash "8ca319486bfbb3663ea0fbe81326349" dur.

Komut dosyasının bir sonraki kısmı, bandit23'ün parolasını "mytarget" dosyasındaki "/tmp/" klasörüne kopyalamasıdır. "cat /tmp/8ca319486bfbb3663ea0fbe81326349" komutunu kullandığımızda bir sonraki seviyenin şifresi karşımıza gelecektir.

BANDİT SEVİYE 23 → SEVİYE 24

([cron](#), crontab, crontab(5) (use “man 5 crontab” to access this))

Level Goal

A program is running automatically at regular intervals from **cron**, the time-based job scheduler. Look in **/etc/cron.d/** for the configuration and see what command is being executed.

NOTE: This level requires you to create your own first shell-script. This is a very big step and you should be proud of yourself when you beat this level!

NOTE 2: Keep in mind that your shell script is removed once executed, so you may want to keep a copy around...

Kendi oluşturduğunuz bir betiği (script) çalışan bir cron işine entegre etmenizi istiyor. Bu işlemde dizindeki dosyaları görüntülemek için "ls" komutunu kullanıyoruz ve bir sonraki seviye olan bandit24 için "cat" komutu ile "cronjob_bandit24" dosyasını okuyoruz.

Daha sonra, "/usr/bin/cronjob_bandit24.sh" dosyasını tespit ederek "cat" komutuyla script'i inceliyoruz.

```
bandit23@bandit:~$ cd /etc/cron.d
bandit23@bandit:/etc/cron.d$ ls -la
total 56
drwxr-xr-x  2 root root  4096 Feb 21 22:04 .
drwxr-xr-x 108 root root 12288 Feb 21 22:04 ..
-rw-r--r--  1 root root   62 Feb 21 22:02 cronjob_bandit15_root
-rw-r--r--  1 root root   62 Feb 21 22:02 cronjob_bandit17_root
-rw-r--r--  1 root root  120 Feb 21 22:03 cronjob_bandit22
-rw-r--r--  1 root root  122 Feb 21 22:03 cronjob_bandit23
-rw-r--r--  1 root root  120 Feb 21 22:03 cronjob_bandit24
-rw-r--r--  1 root root   62 Feb 21 22:03 cronjob_bandit25_root
-rw-r--r--  1 root root  201 Jan  8  2022 e2scrub_all
-rwx-----  1 root root   52 Feb 21 22:04 otw-tmp-dir
-rw-r--r--  1 root root  102 Mar 23  2022 .placeholder
-rw-r--r--  1 root root  396 Feb  2  2021 sysstat
bandit23@bandit:/etc/cron.d$ cat cronjob_bandit24
@reboot bandit24 /usr/bin/cronjob_bandit24.sh &> /dev/null
* * * * * bandit24 /usr/bin/cronjob_bandit24.sh &> /dev/null
```

```
bandit23@bandit:/etc/cron.d$ 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
```

Bu bir Bash betiği (script) ve aşağıdaki işlemleri yapıyor:

1. Mevcut kullanıcının adını (whoami) myname değişkenine atıyor.
2. /var/spool/\$myname/foo dizinine geçiyor.
3. /var/spool/\$myname/foo dizinindeki tüm dosyaları (gizli dosyalar dahil) işlemek üzere bir döngü başlatıyor.
4. Döngü içinde, her dosya için şu işlemleri yapıyor:
 - "." ve ".." dosyalarını atlayarak dosyanın adını ekrana yazdırıyor.
 - Dosyanın sahibini "owner" değişkenine atıyor.
 - Eğer dosyanın sahibi "bandit23" ise, dosyayı 60 saniyelik bir süre sınırı içinde (timeout) çalıştırıyor.
 - Dosyayı silmek için rm komutunu kullanıyor.

Bu script'in amacı, /var/spool/\$myname/foo dizinindeki tüm dosyaları sırayla çalıştmak ve sonra da silmek için tasarlanmıştır. Ancak yalnızca "bandit23" kullanıcısına ait dosyaları çalıştırır ve sınırlandırır.

Öncelikle /var/spool/bandit24 dizininde bir script oluşturarak /etc/bandit_pass/bandit24 dosyasından okunan parolayı /tmp/bursa dosyasına kopyalayabiliriz. Daha sonra chmod komutu ile dosya ve dizinlere gerekli izinleri verebiliriz. Son olarak, sleep komutu ile bir süre bekledikten sonra /tmp/bursa dosyasını cat komutu ile görüntüleyerek bir sonraki seviyenin parolasını elde edebiliriz.

```
bandit23@bandit:/tmp/bursa$ ls
antalya.sh
bandit23@bandit:/tmp/bursa$ cat antalya.sh
#!/bin/bash
cat /etc/bandit_pass/bandit24 > /tmp/bursa/password
bandit23@bandit:/tmp/bursa$
```

```
bandit23@bandit:/tmp/bursa$ chmod 777 /tmp/bursa
bandit23@bandit:/tmp/bursa$ chmod 777 /tmp/bursa/antalya.sh
bandit23@bandit:/tmp/bursa$ ls -la
total 124
drwxrwxrwx  2 bandit23 bandit23  4096 Mar  1 18:05 .
drwxrwx-wt 620 root    root     114688 Mar  1 18:08 ..
-rwxrwxrwx  1 bandit23 bandit23    64 Mar  1 18:05 antalya.sh
```

cp komutuyla komut dosyasını /var/spool/bandit24/foo dizinine kopyalıyoruz ve komut dosyasının bir dakika içinde otomatik olarak çalışmasını bekliyoruz.

```
bandit23@bandit:/tmp/bursa$ ls
antalya.sh
bandit23@bandit:/tmp/bursa$ cp antalya.sh /var/spool/bandit24/foo
bandit23@bandit:/tmp/bursa$ ls -la
total 128
drwxrwxrwx  2 bandit23 bandit23  4096 Mar  1 18:17 .
drwxrwx-wt 628 root    root     114688 Mar  1 18:17 ..
-rwxrwxrwx  1 bandit23 bandit23    65 Mar  1 18:13 antalya.sh
-rw-rw-r--  1 bandit24 bandit24   33 Mar  1 18:17 password
bandit23@bandit:/tmp/bursa$ cat password
VAfGXJ1PBSsPSnvsjI8p759leLZ9GGar
bandit23@bandit:/tmp/bursa$
```

Not: Çözerken "cp" komutunu kullandıkten sonra 1 dakika beklemeyi unutmayın.

BANDİT SEVİYE 24 → SEVİYE 25

Level Goal

A daemon is listening on port 30002 and will give you the password for bandit25 if given the password for bandit24 and a secret numeric 4-digit pincode. There is no way to retrieve the pincode except by going through all of the 10000 combinations, called brute-forcing.
You do not need to create new connections each time

“Bir daemon, bandit24’ün şifresi ve 4 basamaklı bir gizli sayısal pin kodu verildiğinde bandit25 şifresini veren 30002 nolu bağlantı noktasına dinlenmektedir. Pin kodunu sadece 10000 kombinasyonu geçerek alabileceğiniz tek yol vardır ve bu deneme yanılma yöntemi olarak adlandırılır. Her seferinde yeni bağlantılar oluşturmanıza gerek yoktur.”

Soruda Brute-force adı verilen 10000 kombinasyonun tümünü gözden geçirmek dışında pin kodunu almanın bir yolu olmadığını söylemiş. Öyleyse biz 4 haneli bütün şifreleri deneyen bir komut yazmalıyız.

```
bandit24@bandit:~$ for i in {0000 .. 9999}; do echo "VAfGXJ1PBSsPSnvsjI8p759leLZ9GGar $i"; done | 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.
Wrong! Please enter the correct pincode. Try again.
Wrong! Please enter the correct pincode. Try again.
Wrong! Please enter the correct pincode. Try again.
Wrong! Please enter the correct pincode. Try again.
Wrong! Please enter the correct pincode. Try again.
Wrong! Please enter the correct pincode. Try again.
Wrong! Please enter the correct pincode. Try again.
Wrong! Please enter the correct pincode. Try again.
Wrong! Please enter the correct pincode. Try again.
```

30002 numaralı bağlantı noktasını dinliyoruz ve soruda bize bandit24 parolası ve 4 haneli gizli bir sayısal pin kodu verildiğini söylemiş bu yüzden nc komutuyla dinliyoruz ve for döngüsüyle bütün şifreleri denemesini sağlıyoruz. Böylece en son eşleştiğinde bize bandit25’in şifresini veriyor.

```
wrong! Please enter the correct pincode. Try again.
Wrong! Please enter the correct pincode. Try again.
Wrong! Please enter the correct pincode. Try again.
Wrong! Please enter the correct pincode. Try again.
Correct!
The password of user bandit25 is p7TaowMYrmu230l8hiZh9UvD009hpx8d
Exiting.
```

BANDİT SEVİYE 25 → SEVİYE 26

(ssh, cat, more, vi, ls, id, pwd)

Level Goal

Logging in to bandit26 from bandit25 should be fairly easy... The shell for user bandit26 is not **/bin/bash**, but something else. Find out what it is, how it works and how to break out of it.

"Bandit25 kullanıcılarından bandit26'ya giriş yapmak oldukça kolay olmalıdır. Kullanıcı bandit26 için kabuk /bin/bash değil, başka bir şeydir. Ne olduğunu, nasıl çalıştığını ve nasıl kırılabileceğini bulun."

Öncelikle "ls" komutu ile listelediğimiz de görüyoruz ki elimizde bir ssh key var.

```
bandit25@bandit:~$ ls
bandit26.sshkey
```

Bu anahtarı kullanarak oturuma giriş yapmaya çalıştığımızda giriş yapacak fakat hemen oturumu kapatacaktır, bunun nedeni Bandit26 kullanıcı, /bin/bash gibi standart bir shell kullanmaz. Bunun yerine başka bir shell kullanır.

"bandit26" kullanıcısının kullandığı kabuk, sistemdeki "/etc/passwd" dosyasının içeriği "grep" ve "cat" komutlarını kullanarak görülebilir.

```
bandit25@bandit:~$ cat /etc/passwd | grep bandit26
bandit26:x:11026:11026:bandit level 26:/home/bandit26:/usr/bin/showtext
```

Bu, "/etc/passwd" dosyasındaki "bandit26" kullanıcısına ait satırı görüntüler ve kullanıcının kullandığı kabuk, son ":" sonrasında satırın sonunda görülebilir.

Bilgi: "/etc/passwd" dosyası işletim sistemindeki kullanıcılar hakkında bilgi içerir. Dosyadaki her satır tek bir kullanıcıyı temsil eder ve her kullanıcının bilgileri iki nokta üst üste (:) ile ayrılır. Aşağıdakiler, "/etc/passwd" dosyasındaki her satır için tipik bir biçimdir:

username:password:user ID (UID):group ID (GID):user info:home directory:Shell

- **username:** Kullanıcının kullanıcı adı.
- **password:** Kullanıcı parolasının şifrelenmiş bir gösterimi. Bazı sistemlerde, parolanın başka bir yerde depolandığını belirtmek için bu alan boş olabilir veya yıldız işaretleri (*) ile doldurulabilir.
- **user ID (UID):** Kullanıcı için benzersiz bir sayısal tanımlayıcı.
- **group ID (GID):** Kullanıcının ait olduğu grup için sayısal bir tanımlayıcı.
- **user info:** Kullanıcı hakkında, tam adı veya açıklaması gibi ek bilgiler içerebilen bir alan.
- **home directory:** Kullanıcının kişisel dosyalarının ve yapılandırma dosyalarının depolandığı giriş dizininin yolu.

- shell: Kullanıcının işletim sistemiyle etkileşim kurmak için kullandığı komut satırı arabirimini olan varsayılan kabuğunun yolu.

Bandit26'nın shell'i **/usr/bin/showtext**.

Bunun ne olduğunu okumamız gerekiyor.

```
bandit25@bandit:~$ cat /usr/bin/showtext
#!/bin/sh
dosya
export TERM=linux

exec more ~/text.txt
exit 0
```

Biz ne zaman bandit26 ya girsek bu script çalışacak ve bizi sürekli dışarıya atacak o zaman exit'e çarpmasını engellemeye çalışmalıyız. Yukarıdaki more komutu aşağıdaki metin olduğunu gösteriyor.

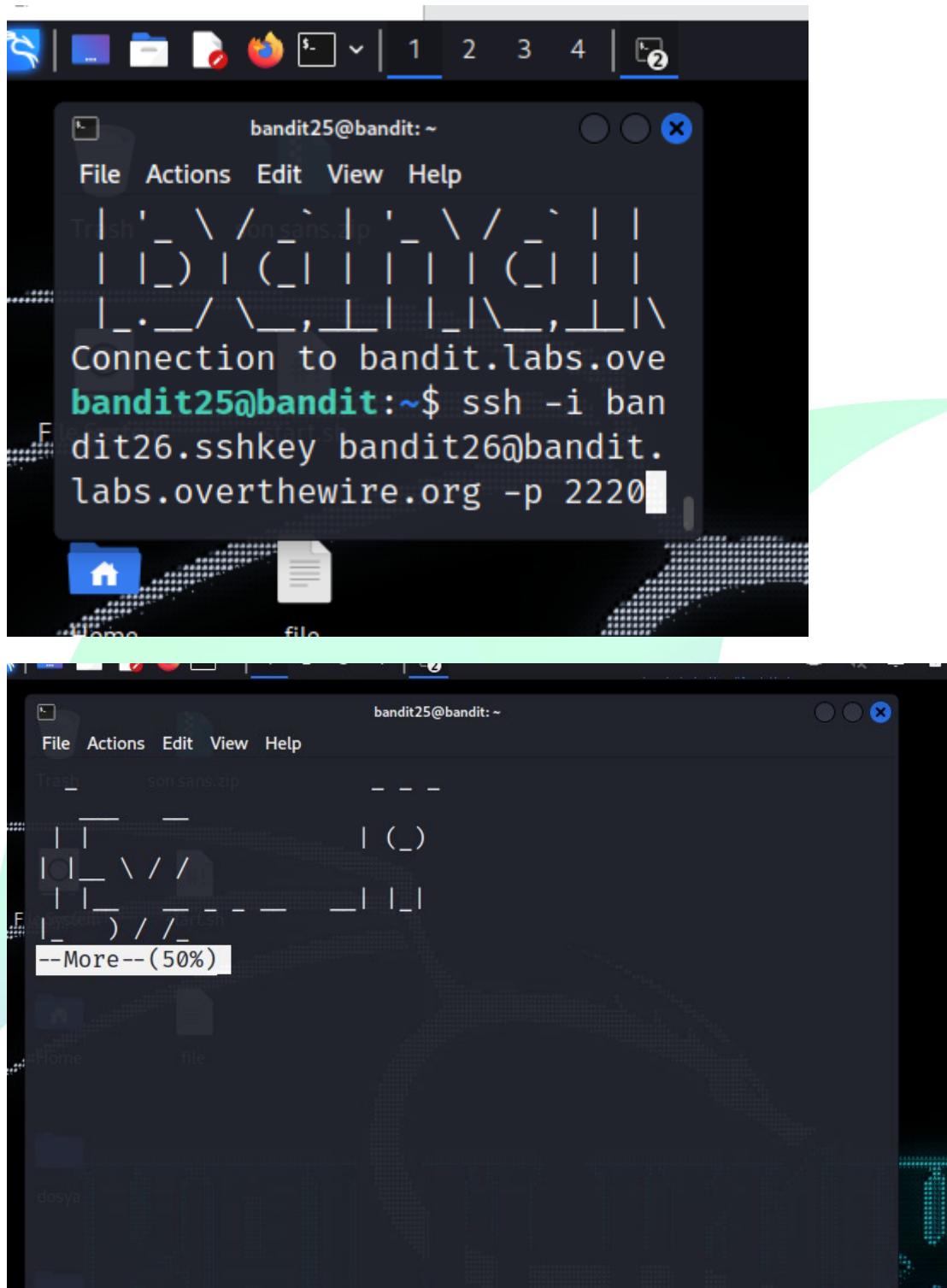
The terminal window shows the following output:

```
Home file
Enjoy your stay!

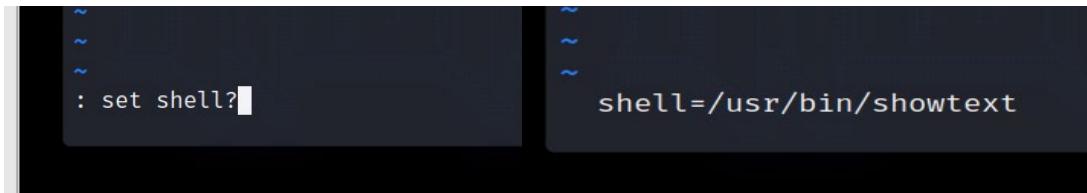
[REDACTED]
Connection to bandit.labs.overthewire.org closed.
bandit25@bandit:~$ █
```

The redacted area contains a large amount of binary or encoded data represented by various symbols like '|', '(', ')', '/', '\', etc., forming a complex pattern.

Bizden normalde beklediğimiz more komutu gibi sayfalar yapmamızı istememesinin nedeni bunun için sayfalamaya gerek olmaması çünkü terminalinize mükemmel bir şekilde sigar. O zaman exite çarpmaması için more komutunda kalmalıyız. Bunun için de ekranı küçülterek more komutunun sigamayacağı şeyle getireceğiz. Şimdi giriş yapalım.

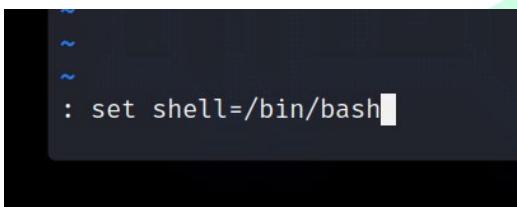


Bir Vim metin düzenleyicisi açabilir ve Vim metin düzenleyicisinde bulunan bir dosya açma komutunu kullanarak şifre dosyasını görüntüleyebiliriz ama bizim hedefimiz burada kabuğu değiştirmek. Düzenleyici olan v komutudur. V komutuna basınca vim açılacak ve burada kabuğu değiştireceğiz.



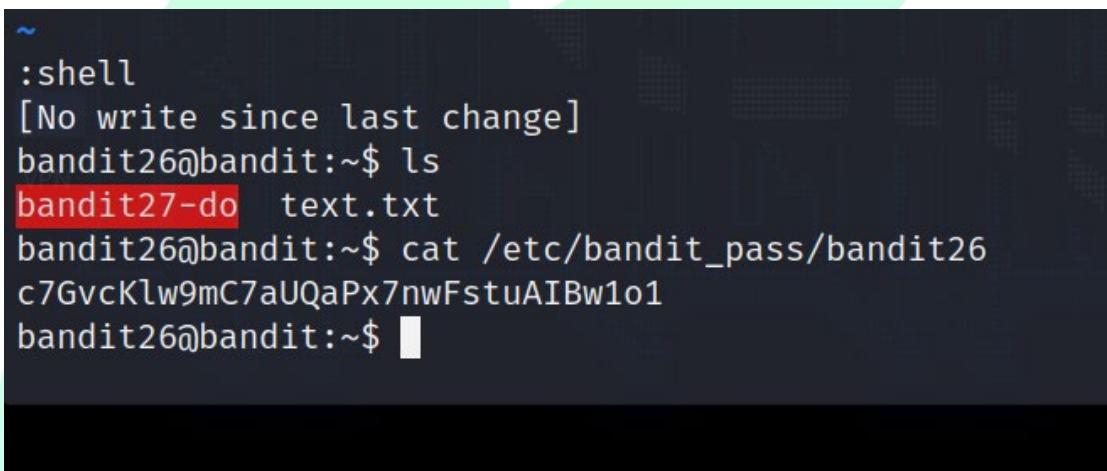
```
~  
~  
~  
: set shell?█  
~  
~  
shell=/usr/bin/showtext
```

Vim şimdi bandit26 olarak açıldı. Yapmamız gereken, kullanıcının vim'deki varsayılan kabuğunu yararlı bir kabuğa ayarlamaktır



```
~  
~  
~  
: set shell=/bin/bash█
```

Şimdi de :Shell yazdığımızda bandit26'nın kabuğundayız.



```
~  
:shell  
[No write since last change]  
bandit26@bandit:~$ ls  
bandit27-do text.txt  
bandit26@bandit:~$ cat /etc/bandit_pass/bandit26  
c7GvcKlw9mC7aUQaPx7nwFstuAIBw1o1  
bandit26@bandit:~$ █
```

İpucu: Bir sonraki seviye için kabuktan çıkış yapmayın :)

“bandit27-do” çalıştırılabilir bir dosyadır.

BANDİT SEVİYE 26 → SEVİYE 27

Level Goal

Good job getting a shell! Now hurry and grab the password for bandit27!

“Bir kabuk almak için iyi iş! Şimdi acele edin ve bandit27'nin şifresini alın!”

Önceki seviyede gördüğümüz çalıştırılabilir dosyayı kullanarak bandit27'nin şifresini alabiliyoruz.

Bunun için bandit26 kullanıcısının kabuğunda olmamız gerekiyor.

```
bandit26@bandit:~$ ls
bandit27-do  text.txt
bandit26@bandit:~$ ./bandit27-do cat /etc/bandit_pass/ba
ndit27
YnQpBuifNMas1hcUFk70ZmqkhUU2EuaS
bandit26@bandit:~$
```

BANDİT SEVİYE 27 → SEVİYE 28

Level Goal

There is a git repository at `ssh://bandit27-git@localhost/home/bandit27-git/repo`. The password for the user `bandit27-git` is the same as for the user `bandit27`.

Clone the repository and find the password for the next level.

İşlem yapabileceğimiz bir klasöre ihtiyacımız var çünkü git deposunu klonlamalıyız ve bulunduğuımız dizinde bu yetkiye sahip değiliz.

Hatırlatma: /tmp dizini, sistem açılışı sırasında kullanılan geçici dosyaların bulunduğu bir yerdir. Bu dosyalar ve dizinler genellikle (her zaman olmayabilir) bir uygulamanın şimdilik ihtiyaç duymadığı ancak ileride ihtiyaç duyabileceği verileri saklar.

Bu yüzden tmp altında kendi dosyamızı (book dosyası) oluşturup içine girdik. Şimdi de git komutu ile bize verdiği url'yi kullanarak klonluyoruz.

```

bandit27@bandit:~$ mkdir /tmp/book
bandit27@bandit:~$ cd /tmp/book
bandit27@bandit:/tmp/book$ git clone ssh://bandit27-git@localhost:2220/home/bandit27-git/repo
Cloning into 'repo' ...
The authenticity of host '[localhost]:2220 ([127.0.0.1]:2220)'
can't be established.
ED25519 key fingerprint is SHA256:C2ihUBV7ihnV1wUXRb4RrEcLfXC5
CXLhmAAM/urerLY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerpr
int])? yes
Could not create directory '/home/bandit27/.ssh' (Permission d
enied).
Failed to add the host to the list of known hosts (/home/bandi
t27/.ssh/known_hosts).

```

Şifremizi girerek devam ediyoruz.

Evet repo dosyası böylelikle inmiş oldu.
Yazdırarak şifremizi alıyoruz.

```

bandit27@bandit:/tmp/book$ ls
repo
bandit27@bandit:/tmp/book$ cd repo
bandit27@bandit:/tmp/book/repo$ ls
README
bandit27@bandit:/tmp/book/repo$ cat README
The password to the next level is: AVanL161y9rsbcJIsFHuw35rja0
M19nR

```

BANDİT SEVİYE 28 → SEVİYE 29

(git)

Level Goal

There is a git repository at `ssh://bandit28-git@localhost/home/bandit28-git/repo`. The password for the user `bandit28-git` is the same as for the user `bandit28`.

Clone the repository and find the password for the next level.

Yine aynı şekilde işlem yapabileceğimiz bir klasöre ihtiyacımız var. Tmp altında “sud” dosyası oluşturup içine girdik.

```

bandit28@bandit:~$ cd /tmp
bandit28@bandit:/tmp$ mkdir sud
bandit28@bandit:/tmp$ cd sud
bandit28@bandit:/tmp/sud$ git clone git clone ssh://bandit28-git@localhost:2220/home/bandit28-git/repo
fatal: Too many arguments.

usage: git clone [<options>] [--] <repo> [<dir>]

-v, --verbose          be more verbose
-q, --quiet            be more quiet
--progress             force progress reporting
--reject-shallow       don't clone shallow repository
-n, --no-checkout      don't create a checkout
--bare                 create a bare repository

```

“git clone” komutuyla repo dosyasını bulunduğu yere çektiğimiz.

İçine baktığımızda README.md var.

Okuduğumuzda parolanın şifrelenmiş olduğunu görüyoruz.

```

bandit28@bandit:/tmp/sud$ ls
repo
bandit28@bandit:/tmp/sud$ cd repo
bandit28@bandit:/tmp/sud/repo$ ls
README.md
bandit28@bandit:/tmp/sud/repo$ cat README.md
# Bandit Notes
Some notes for level29 of bandit.

## credentials

dosya    nano 5848 ...
- username: bandit29
- password: xxxxxxxxxxxx

```

Dosyanın depo geçmişini incelemek için “log” komutunu kullanmamız gerekiyor.

Not: Üstte en son taahhütler gösterilir.

```
bandit28@bandit:/tmp/sud/repo$ git log
commit 104db85a904e9691ff22aafe1a96124c88f75afa (HEAD → master, origin/master, origin/HEAD)
Author: Morla Porla <morla@overthewire.org>
Date:   Tue Feb 21 22:03:10 2023 +0000

    fix info leak

commit 6c3c5e485cc531e5d52c321587ce1103833ab7c3
Author: Morla Porla <morla@overthewire.org>
Date:   Tue Feb 21 22:03:10 2023 +0000

dosya add missing data

commit cd3b97ef95879ec34df0d6c82f2a96d552f0e56c
Author: Ben Dover <noone@overthewire.org>
Date:   Tue Feb 21 22:03:10 2023 +0000

initial commit of README.md
```

“git show” komutu kullanarak en son taahhütte bakıyoruz.

```
bandit28@bandit:/tmp/sud/repo$ git show 104db85a904e9691ff22aa
fe1a96124c88f75afa
commit 104db85a904e9691ff22aafe1a96124c88f75afa (HEAD → master,
origin/master, origin/HEAD)
Author: Morla Porla <morla@overthewire.org>
Date:   Tue Feb 21 22:03:10 2023 +0000

    fix info leak

dosya diff --git a/README.md b/README.md
index b302105..5c6457b 100644
— a/README.md
+++ b/README.md
@@ -4,5 +4,5 @@ Some notes for level29 of bandit.
## credentials

- username: bandit29
-- password: tQKvmcwNYcFS6vmPHIUSI3ShmsrQZK8S
+- password: xxxxxxxxxxxx
```

BANDİT SEVİYE 29 → SEVİYE 30

(git)

Level Goal

There is a git repository at `ssh://bandit29-git@localhost/home/bandit29-git/repo`. The password for the user `bandit29-git` is the same as for the user `bandit29`.

Clone the repository and find the password for the next level.

Çalışabileceğimiz bir dosya oluşturup, "git clone" ile veriyi alalım.

```
bandit29@bandit:~$ mkdir /tmp/duk
bandit29@bandit:~$ cd /tmp/duk
bandit29@bandit:/tmp/duk$ git clone ssh://bandit29-git@localhost:2220/home/bandit29-git/repo
Cloning into 'repo' ...
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/bandit29/.ssh' (Permission denied).
```

Ve repo dosyamız gelmiş içindeki README.md dosyasını okuduğumuzda parola kısmında şifre bulunmuyor.

```
bandit29@bandit:/tmp/duk$ ls
repo
bandit29@bandit:/tmp/duk$ cd repo
bandit29@bandit:/tmp/duk/repo$ ls
README.md
bandit29@bandit:/tmp/duk/repo$ cat README.md
# Bandit Notes
Some notes for bandit30 of bandit.

## credentials

- username: bandit30
- password: <no passwords in production!>
```

Git kullanarak bu deponun bundan önceki sürümlerinde neler olduğunu göreceğiz.

```
bandit29@bandit:/tmp/duk/repo$ git log
commit 8159c819f4d37d9491254035c9e74ffcb316652e (HEAD →
  master, origin/master, origin/HEAD)
Author: Ben Dover <noone@overthewire.org>
Date:   Wed Jan 11 19:18:54 2023 +0000

    fix username

commit 23706c87f70872af9f04744569f7b6273647fb14
Author: Ben Dover <noone@overthewire.org>
Date:   Wed Jan 11 19:18:54 2023 +0000

    initial commit of README.md
```

***Bilgi:** "head -> master" ifadesi Git log komutunun çıktısındaki bir commitin hangi branch'e (master) bağlandığını gösterir. "head" kelimesi, o anki en son commit'i veya branch'i işaret eder. Yani "head -> master" ifadesi, o commit'in en son master branch'ine eklendiğini gösterir.

" README.md " nin ilk hali kontrol edelim.

```
bandit29@bandit:/tmp/mük/repo$ git checkout 2370
Note: switching to '2370'.

You are in 'detached HEAD' state. You can look around, m
ake experimental
changes and commit them, and you can discard any commits
you make in this
state without impacting any branches by switching back t
o a branch.

If you want to create a new branch to retain commits you
create, you may
do so (now or later) by using -c with the switch command
```

Bu deponun geçmişine "log" ile bakıp "checkout" yaptığımda şifrenin olmadığını görüyoruz.

```
bandit29@bandit:/tmp/mük/repo$ cat README.md
# Bandit Notes
Some notes for bandit30 of bandit.

## credentials

- username: bandit29
- password: <no passwords in production!>
```

Biz varsayılan " master " dalındayız. " git branch -a " komutu, Git deposundaki tüm yerel (local) ve uzak (remote) dalları listeler. Bunu kullanarak diğer dalları inceleyelim.

```
bandit29@bandit:/tmp/mük/repo$ git branch
* (HEAD detached at 23706c8)
  master
bandit29@bandit:/tmp/mük/repo$ git checkout master
Previous HEAD position was 23706c8 initial commit of REA
DME.md
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
bandit29@bandit:/tmp/mük/repo$ git branch -a
* master
  remotes/origin/HEAD → origin/master
  remotes/origin/dev
  remotes/origin/master
  remotes/origin/sploits-dev
```

Dallanmalara baktığımızda son güncellenen geliştirici olan “ /dev ” dallanmasını görürüz. “ git checkout dev ” komutuyla “ dev ” dallanmasına geçiş yapıyoruz.

```
bandit29@bandit:/tmp/mük/repo$ git checkout dev
Branch 'dev' set up to track remote branch 'dev' from 'o
rigin'.
Switched to a new branch 'dev'
bandit29@bandit:/tmp/mük/repo$ git log
commit be91af8ed9847de549d0e3361cf675674b25867 (HEAD →
  dev, origin/dev)
  dev, origin/dev)
Author: Morla Porla <morla@overthewire.org>
Date:   Wed Jan 11 19:18:55 2023 +0000

    add data needed for development
```

“ls” komutuyla içeriyi listeliyoruz ve biraz önce erişemediğimiz şifreyi burada görebiliyoruz.

```
bandit29@bandit:/tmp/mük/repo$ ls
code  README.md
bandit29@bandit:/tmp/mük/repo$ cat README.md
# Bandit Notes
Some notes for bandit30 of bandit.

## credentials

- username: bandit30
- password: xbhV3HpNGLTIdnjUrdAlPzc2L6y9EOnS
```

BANDİT SEVİYE 30 → SEVİYE 31

(git)

Level Goal

There is a git repository at `ssh://bandit30-git@localhost/home/bandit30-git/repo`. The password for the user `bandit30-git` is the same as for the user `bandit30`.

Clone the repository and find the password for the next level.

Git depomuz var ve bu depoyu yine içinde işlem yapacağımız dizine (/tmp) alıyoruz. Dosya oluşturup “git clone” komutunu çalıştırıyoruz.

```
bandit30@bandit:~$ mkdir /tmp/fire
bandit30@bandit:~$ cd /tmp/fire
bandit30@bandit:/tmp/fire$ git clone ssh://bandit30-git@localhost:2220/home/bandit30-git/repo
Cloning into 'repo' ...
The authenticity of host '[localhost]:2220 ([127.0.0.1]:2220)' can't be established.
ED25519 key fingerprint is SHA256:C2ihUBV7ihnV1wUXRb4RrEcLfxC5CxLhmAAM/urerLY.
bandit30@bandit:/tmp/fire/repo$ ls
README.md
bandit30@bandit:/tmp/fire/repo$ cat README.md
just an empty file ... muahaha
```

“git log” ile baktığımızda bir şey bulamadık ve “ README.md ” de bize herhangi bir bilgi vermiyor.

Biz de “ git tag ” komutuyla git deposundaki etiketleri kontrol ediyoruz ve “secret” adlı bir etiket buluyoruz.

```
bandit30@bandit:/tmp/fire/repo$ git tag
secret
```

Etiketin işaret ettiği commit’i görüntüleyerek şifreye ulaşıyoruz.

```
bandit30@bandit:/tmp/fire/repo$ git show secret
OoffzGDLzhAlerFJ2cAiz1D41JW1Mhmt
```

BANDİT SEVİYE 31 → SEVİYE 32

Level Goal

There is a git repository at `ssh://bandit31-git@localhost/home/bandit31-git/repo`. The password for the user `bandit31-git` is the same as for the user `bandit31`.

Clone the repository and find the password for the next level.

Git depomuz var ve bu depoyu içinde işlem yapacağımız dizine (/tmp) alıyoruz. Dosya oluşturup “git clone” komutunu çalıştırıyoruz.

```
bandit31@bandit:~$ mkdir /tmp/bay
bandit31@bandit:~$ cd /tmp/bay
bandit31@bandit:/tmp/bay$ git clone ssh://bandit31-git@localhost:2220/home/bandit31-git/repo
Cloning into 'repo' ...
```

“repo” nun içine girelim ve “ README.md ” yi okutalım.

“ ls -la ” komutuyla içindeki tüm dosyaları listeliyoruz.

```
bandit31@bandit:/tmp/bay$ ls
repo
bandit31@bandit:/tmp/bay$ cd repo
bandit31@bandit:/tmp/bay/repo$ ls -la
total 20
drwxrwxr-x 3 bandit31 bandit31 4096 Feb  4 14:15 .
drwxrwxr-x 3 bandit31 bandit31 4096 Feb  4 14:15 ..
drwxrwxr-x 8 bandit31 bandit31 4096 Feb  4 14:15 .git
-rw-rw-r-- 1 bandit31 bandit31    6 Feb  4 14:15 .gitignore
-rw-rw-r-- 1 bandit31 bandit31 147 Feb  4 14:15 README.md
```

Gizlenmiş “ .gitignore ” dosyası dikkatimizi çekiyor.

Bilgi: “ .gitignore ” dosyası, git'in depoya göndermeyeceği dosya türlerini ve dosyaları listeler.

```
bandit31@bandit:/tmp/bay/repo$ cat README.md
This time your task is to push a file to the remote repository.

Details:
  File name: key.txt
  Content: 'May I come in?'
```

'README', depoya "key.txt" adında ve içeriği 'May I come in?' olan bir dosya göndermemiz gerektiğini belirtiyor. İlk önce dosyayı oluşturuyoruz:

```
bandit31@bandit:/tmp/bay/repo$ echo 'May I come in?' > key.txt
bandit31@bandit:/tmp/bay/repo$ ls
key.txt README.md
```

“.gitignore” dosyasını yazdırduğumızda “.txt” uzantılı hiçbir dosyanın depoya gönderilemediğini görüyoruz.

```
bandit31@bandit:/tmp/bay/repo$ cat .gitignore
*.txt
```

“.txt” uzantılı dosyamızı göndermemize engel olan “.gitignore” dosyasını silelim ve “.txt” uzantılı dosyamızı ekleyelim daha sonra ise “status” komutu ile durumuna bakalım.

```
bandit31@bandit:/tmp/bay/repo$ rm .gitignore
bandit31@bandit:/tmp/bay/repo$ git add key.txt
bandit31@bandit:/tmp/bay/repo$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   key.txt

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:   .gitignore
```

Bizden istenilen kriterlerdeki dosyamızı “git push” komutu ile gönderiyoruz.

```
bandit31@bandit:/tmp/bay/repo$ git push
The authenticity of host '[localhost]:2220 ([127.0.0.1]:2220)' can't
be established.
ED25519 key fingerprint is SHA256:C2ihUBV7ihnV1wUXRb4RrEcLfXC5CXlhmA
AM/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/bandit31/.ssh' (Permission denied)
.
Failed to add the host to the list of known hosts (/home/bandit31/.s
```

Bize geri döndürülen cevaptaki uzaktan gelen mesajlara(remote) baktığımızda şifreyi görüyoruz.

```

Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 315 bytes | 315.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: ### Attempting to validate files ... ####
remote:
remote: .o0o.o0o.o0o.o0o.o0o.o0o.o0o.o0o.o0o.
remote:
remote: Well done! Here is the password for the next level:
remote: rmCBvG56y58BXzv98yZGd07ATVL5dW8y
remote:
remote: .o0o.o0o.o0o.o0o.o0o.o0o.o0o.o0o.o0o.
remote:
To ssh://localhost:2220/home/bandit31-git/repo
 ! [remote rejected] master → master (pre-receive hook declined)
error: failed to push some refs to 'ssh://localhost:2220/home/bandit'

```

BANDİT SEVİYE 32 → SEVİYE 33

After all this **git** stuff its time for another escape. Good luck!

“Ssh” komutu ile girmek istediğimizde Shell her girilen küçük harfi büyük harfe çeviren bir kabuğa yönlendiriyor. Buradan çıkabilmek için bash kabuğunu çağırmalıyız.

Bilgi: “bash” komutunu kullanarak bir Bash kabuğu açığınızda, \$0 değişkeni, Bash kabuğunu çağırmak için kullanılan dosyanın adı olan bash olarak ayarlanacaktır. Terminal oluşturulurken bu terminalde çalıştırılacak olan Shell script in(bash) adı \$0 değişkenine atanır.

“\$0” komutunu yazdıktan sonra bash script’e girdik ve girmek istediğimiz Shell dizinini yazıyoruz.

Bandit33 ün şifresinin saklandığı dosyayı okuttuğumuzda ise şifre karşımıza çıkıyor.

```
For support, questions or comments, contact us on discord or IRC.
```

```
Enjoy your stay!
```

```

osya
WELCOME TO THE UPPERCASE SHELL
>> /bin/bash
sh: 1: /BIN/BASH: not found
>> dm
sh: 1: DM: not found
>> $0
$ /bin/bash
bandit33@bandit:~$ cat /etc/bandit_pass/bandit33
odHo63fHiFqcWWJG9rLiLDtPm45KzUKy
bandit33@bandit:~$ █
```

```
└─(kali㉿kali)-[~]
$ shutdown ┌─┐
```

:)

