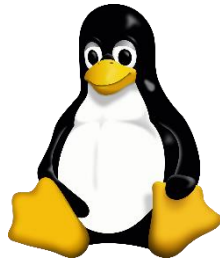


Linux Hardening



Furkan Enes Polatoğlu

furkanenes1160[at]icloud[.]com

05.10.2022

İÇİNDEKİLER

Güçlü Parola Yönetimi.....	3
Yeni Hesap Oluşturma.....	4
Yeni Kullanıcı Oluşturma ve Kullanıcılar Arası İzinsel İncelemeler.....	4
Hesap Geçerlilik Süreleri.....	6
Linux Hesaplarının Geçerlilik Süreleri ve Zaman Yönetimi.....	6
Kilitli Hesapların Yönetilmesi.....	7
ecryptfs Sisteminin Kullanımı.....	8
GPG ile Şifreleme Uygulamaları.....	11
GPG ile Simetrik Şifreleme Uygulaması.....	11
GPG ile Asimetrik Şifreleme Uygulaması.....	14
iptables Kullanımı.....	18
SSH Parolasız Erişim Yönetimi.....	24
sudo Komutunun Kullanımı.....	26

Güçlü Parola Yönetimi

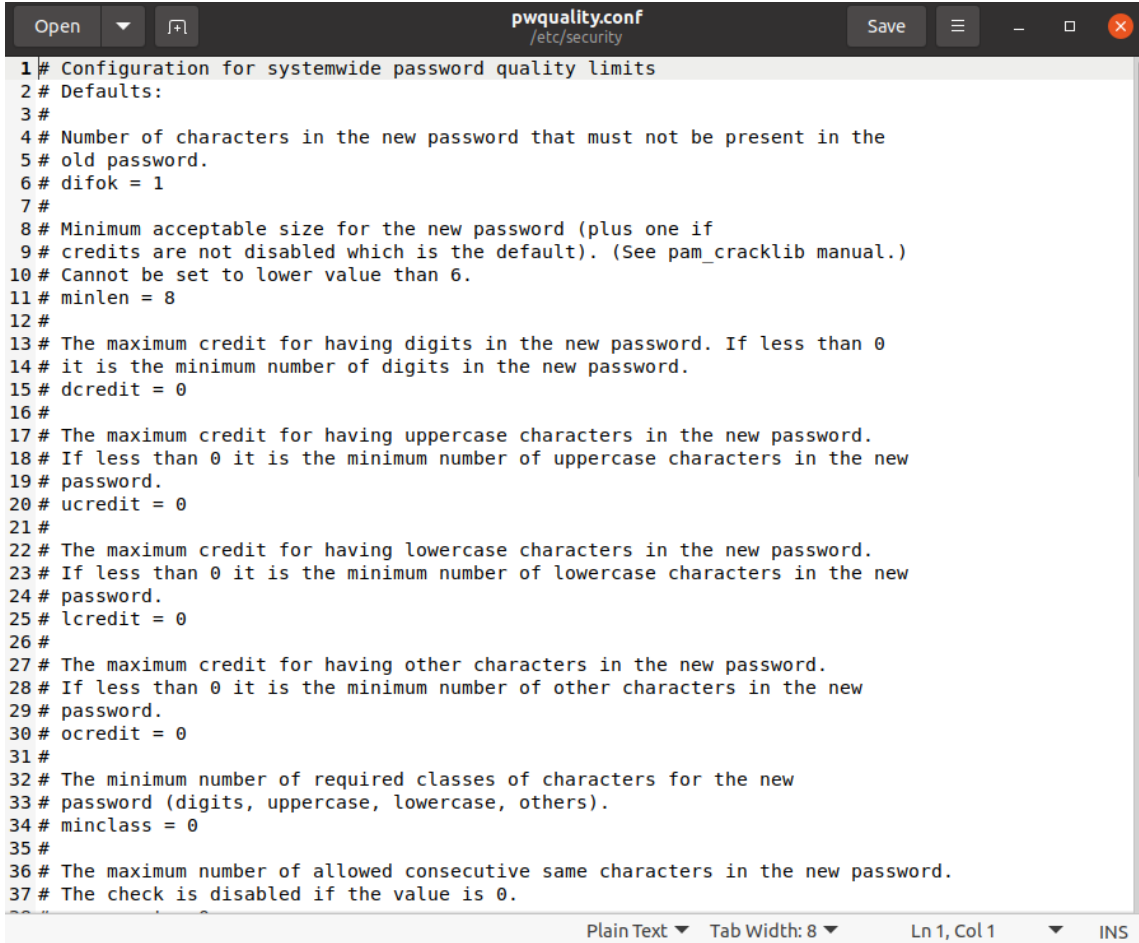
Bu başlık altında Linux sistemler üzerinde güçlü parola politikası uygulaması ile alakalı örnekler yapacağız. Biliyoruz ki ister Linux olsun ister Windows olsun ya da herhangi bir uygulama olsun, siber güvenlik anlamında almamız gereken en önemli önlem güçlü bir parola politikasıdır. Neden? Herhangi bir hacker veya pentester size saldırmak istediğinde kullanacağı en temel yöntemlerden bir tanesi Dictionary Attack (Sözlük Saldırısı) dediğimiz yöntemdir. Tahmini veya bizzat oluşturulmuş sözcüklerden oluşan bir sözlük üzerinden size deneme yanılma yöntemiyle bütün olası parolaları deneyip şifrenizi kırmaya çalışacaktır. Dolayısıyla basit bir şifre bizim rahatlıkla hacklenebileceğimiz anlamına gelmektedir. İşin aslı, hepimizin bildiği çok önemli bir konu olmasına rağmen gerçek dünyada bir çok sistem yöneticisinin hacklenmesinin temelinde parola politikasına yeterince özen göstermediği yatmaktadır.

Bu konuda bize yardımcı olacak paketi indirelim;

“sudo apt install libpam-pwquality”

Kurmuş olduğumuz paketin yapılandırma dosyasını inceleyelim;

“sudo gedit /etc/security/pwquality.conf”



```
1 # Configuration for systemwide password quality limits
2 # Defaults:
3 #
4 # Number of characters in the new password that must not be present in the
5 # old password.
6 # difok = 1
7 #
8 # Minimum acceptable size for the new password (plus one if
9 # credits are not disabled which is the default). (See pam_cracklib manual.)
10 # Cannot be set to lower value than 6.
11 # minlen = 8
12 #
13 # The maximum credit for having digits in the new password. If less than 0
14 # it is the minimum number of digits in the new password.
15 # dcredit = 0
16 #
17 # The maximum credit for having uppercase characters in the new password.
18 # If less than 0 it is the minimum number of uppercase characters in the new
19 # password.
20 # ucredit = 0
21 #
22 # The maximum credit for having lowercase characters in the new password.
23 # If less than 0 it is the minimum number of lowercase characters in the new
24 # password.
25 # lcredit = 0
26 #
27 # The maximum credit for having other characters in the new password.
28 # If less than 0 it is the minimum number of other characters in the new
29 # password.
30 # ocredit = 0
31 #
32 # The minimum number of required classes of characters for the new
33 # password (digits, uppercase, lowercase, others).
34 # minclass = 0
35 #
36 # The maximum number of allowed consecutive same characters in the new password.
37 # The check is disabled if the value is 0.
```

Dosya açıldığında göreceğiz ki bütün değerler disabled (#) yani kapalı durumdadır. Biz bunları açtıkça etkinleşmiş hale gelecektir. Konfigürasyon dosyası içerisindeki parola politikası özelliklerinden birkaç tanesinden bahsedecek olursak;

difok: Eski ve yeni şifreyi karşılaştırıp aralarında en az ne kadar fark olması gerektiği ayarlanır.

minlen: Parola uzunluğunun en az ne kadar olması gerektiği ayarlanır.

minclass: Küçük harf, büyük harf ve sayıdan oluşan bir şifre istediğimizi belirtebiliriz. Bu özelliğe 3 değerini verirsek, 3 farklı türden oluşan bir parola politikası uygulamış olacağız.

maxrepeat: Parola içerisinde aynı karakterlerin en fazla ne kadar kullanılabileceği ayarlanır.

Yeni Hesap Oluřturma

Yeni Kullanıcı Oluřturma ve Kullanıcılar Arası İzinler İncelemeler:

Bu bölümde yeni bir kullanıcı yaratıp o kullanıcı için bir ev dizini oluřturma ve ayrıca varsayılan kabuk tanımlama gibi özellikleri inceleyeceğiz. Ondan önce bazı izensel noktalara göz atmalıyız. Birkaç kullanıcı oluřturarak başlayalım;

“sudo useradd ilayda -m -d /home/ilayda -s /bin/bash”

komutu ile ilayda isimli bir kullanıcı tanımlayıp **-m -d** parametrelerini kullanarak bir ev dizini oluřturmak istediğimizi belirttik ve **-s** parametresiyle /bin/bash temel kabuk olarak belirlendi. Normal de /bin/sh şeklindedir ancak o biraz daha sadedir. Ardından;

“sudo passwd ilayda” komutu ile ilayda kullanıcısı için bir řifre tanımladık.

Aynı şekilde bir kullanıcı daha oluřturalım;

sudo useradd alp -m -d /home/alp -s /bin/bash

sudo passwd alp

Ařağıdaki görselde görüldüğü üzere dosya izinlerinden de anlaşılabilieceğı gibi ilayda kullanıcısındayken /home dizini altındaki diğerkullanıcıların ev dizini içerisinde dolaşabilmekteyim.

```
bulletproof@ubuntu:/home$ sudo useradd ilayda -m -d /home/ilayda -s /bin/bash
bulletproof@ubuntu:/home$ sudo passwd ilayda
New password:
Retype new password:
passwd: password updated successfully
bulletproof@ubuntu:/home$ sudo useradd alp -m -d /home/alp -s /bin/bash
bulletproof@ubuntu:/home$ sudo passwd alp
New password:
Retype new password:
passwd: password updated successfully
bulletproof@ubuntu:/home$ ls -al
total 24
drwxr-xr-x  6 root          root          4096 Dec 19 22:19 .
drwxr-xr-x 20 root          root          4096 Mar 11  2021 ..
drwxr-xr-x  2 alp          alp          4096 Dec 19 22:19 alp
drwxr-xr-x 20 bulletproof  bulletproof 4096 Nov 15 11:07 bulletproof
drwxr-xr-x  2 ilayda      ilayda      4096 Nov 11 02:07 furkan
drwxr-xr-x  2 ilayda      ilayda      4096 Dec 19 22:18 ilayda
```

Bu durumu önlemek için çalıştırabileceğimiz bir komuttan bahsedelim;

sudo chmod 700 * komutu ile aşağıdaki görselde olduğu gibi klasör yetkilendirmeleri daha kontrollü hale gelmiş oldu ve ben başka bir kullanıcının ev dizini içerisine girmeye çalıştığımda artık engellenmiş olacağım.

```
bulletproof@ubuntu:/home$ sudo chmod 700 *
[sudo] password for bulletproof:
bulletproof@ubuntu:/home$ ls -l
total 16
drwx----- 2 alp alp 4096 Dec 19 22:19 alp
drwx----- 20 bulletproof bulletproof 4096 Nov 15 11:07 bulletproof
drwx----- 2 ilayda ilayda 4096 Nov 11 02:07 furkan
drwx----- 2 ilayda ilayda 4096 Dec 19 22:21 ilayda
bulletproof@ubuntu:/home$ su ilayda
Password:
ilayda@ubuntu:/home$ ls -l
total 16
drwx----- 2 alp alp 4096 Dec 19 22:19 alp
drwx----- 20 bulletproof bulletproof 4096 Nov 15 11:07 bulletproof
drwx----- 2 ilayda ilayda 4096 Nov 11 02:07 furkan
drwx----- 2 ilayda ilayda 4096 Dec 19 22:21 ilayda
ilayda@ubuntu:/home$ cd alp/
bash: cd: alp/: Permission denied
ilayda@ubuntu:/home$
```

Güvenli bir Linux ortamında ev dizininin varsayılan yetkilerini düzenlemek çok önemlidir. Bunu bir konfigürasyon dosyası üzerinden yapmak da mümkün;

“**sudo nano /etc/login.defs**” komutu ile açılan login.defs dosyası içerisinde “UMASK” adlı bir değer bulunmaktadır. Bu değeri **022’den 077’ye** çevirdiğimiz zaman artık bundan sonra yeni bir kullanıcı oluşturulduğunda otomatik olarak başka kullanıcıların erişimine kapatılmış olarak oluşturulacaktır.

```
GNU nano 5.2 /etc/login.defs
# 027, or even 077, could be considered better for privacy
# There is no One True Answer here : each sysadmin must make up his/her
# mind.
#
# If USERGROUPS_ENAB is set to "yes", that will modify this UMASK default value
# for private user groups, i. e. the uid is the same as gid, and username is
# the same as the primary group name: for these, the user permissions will be
# used as group permissions, e. g. 022 will become 002.
#
# Prefix these values with "0" to get octal, "0x" to get hexadecimal.
#
ERASECHAR      0177
KILLCHAR       025
UMASK          022
#
# Password aging controls:
#
#      PASS_MAX_DAYS   Maximum number of days a password may be used.
#      PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location
^X Exit      ^R Read File ^\ Replace  ^U Paste    ^J Justify  ^_ Go To Line
```

Hesap Geçerlilik Süreleri

Linux Hesaplarının Geçerlilik Süreleri ve Zaman Yönetimi:

Bu yazımda Linux üzerinde bir kullanıcı hesabı oluşturacağız ve bu esnada oluşturduğumuz hesapların zaman yönetimi ile son kullanım tarihini belirleme konularından bahsedeceğim. Özellikle bir kurumda sistem yöneticiliği yaptığınızda personel sirkülasyonu anlamında çok ciddi bir hareket olacaktır ve yeni bir kullanıcı hesabı oluştururken bu hesabın son kullanım tarihini belirlememiz işlerimizi daha sağlam yürütme anlamında çok kıymetlidir. Bu işlemler sistem yöneticiliği yapan insanların günlük rutin işlerinden bir tanesi olduğunu söyleyebilirim. Örneğin kurumunuzda yeni bir personel işe başlıyor ve siz onun için bir hesap oluşturuyorsunuz;

sudo useradd enes -m -d /home/selim -s /bin/bash -e 2023-06-30

-e: Parametresi, expire, expiration date (Son kullanma tarihi) anlamına gelir. Parametre için belirttiğimiz “2023-06-30” tarihinde, kullanıcının hesap süresi dolacaktır ve hesabı kullanamaz hale gelecektir ve gerektiğinde tekrardan uzatmamız gerekecektir. Hesap kullanım süresini güncelleme işlemi şu şekilde yapılabilir;

sudo usermod selim -e 2021-05-29

Ben sistem yöneticisi olarak selim kullanıcısına bu hesabı tahsis ederken bir parola vermem gerekiyor. Bu parolayı ben sistem yöneticisi olarak biliyorum ancak selim kullanıcısı hesabını ilk giriş yaptığında bu parola yenilenmelidir. Peki bu işlemi nasıl gerçekleştirebiliriz?

sudo passwd selim komutu ile selim kullanıcısına bir parola belirlemiş oldum. Ardından;

sudo passwd -e selim komutunu çalıştırdığımda karşıma “**passwd: password expiry information changed**” bilgisi gelir yani, bu şifrenin son kullanım süresinin değiştiğini bize bildiriyor. Dolayısıyla selim kullanıcısı **su selim** komutu ile kendi hesabına ilk girişini yaptığında sistem yöneticisinin vermiş olduğu parolayı girip ardından yeni parola belirlemesi istenecektir. Bunun sonucunda yeni kullanıcımıza hesabını güvenli bir şekilde teslim etmiş oluyoruz.

Kilitli Hesapların Yönetilmesi

Bu yazımda Linux sistemlerde kullanıcı hesaplarını kitleme özelliğini inceleyeceğiz. Ne demek bu? Eğer bir kullanıcı, oturum açma anlamında bizim belirlediğimiz sınırın üzerinde bir oturum açma denemesi gerçekleştiriyorsa bu durumda otomatik olarak hesabın kitlenmesini sağlayabiliriz. Bunun için Linux'te "pam" denilen yani password manager anlamına gelen bir kütüphane mevcuttur. Orada bulunan bir yapılandırma dosyasına ek bir satır ekleyeceğiz;

sudo gedit /etc/pam.d/common-auth

komutu ile dosyayı açtıktan sonra "# here are the per-package modules (the "Primary" block)" yorum satırının hemen altına aşağıdaki yapılandırmayı ekliyoruz.,

auth required pam_tally2.so onerr=fail deny=3 unlock_time=600 even_deny_root audit

Yapılandırmayı soldan sağa inceleyecek olursak, **authentication required** (oturum açmak için yetkilendirme gereklidir), **pam_tally2.so** (kütüphane), **onerr=fail** (herhangi bir hata durumunda mutlaka tepki ver), **deny=3** (3 denemeden sonra tepki ver), **unlock_time=600** (10 dakika boyunca da bu kilitli olma durumu devam etsin) ve **even_deny_root** (root kullanıcısı olsa bile bu hesabı kilitler)

root hesabı bildiğimiz gibi önceden kilitlenmediyse ya da iptal edilmediyse bütün Linux ortamlarında varsayılan olarak mevcuttur. Dolayısıyla bu hesap tarafından bir saldırı düzenlendiyse bunu kaldırmak/kilitlemek önemlidir. Mümkünse de root hesabını tamamen sisteminizden kaldırmanız ve kendi özgün root hesabınızı yaratmanız Linux güvenliği açısından daha faydalı olacaktır.

sudo pam_tally2 komutu ile kullanıcıların son hatalı oturum denemelerini tarih ve saatleri ile birlikte görebiliriz.

Şimdi ise kilitlenmiş olan bir hesabı (selim olduğunu varsayalım) açmak istediğimizde ise;

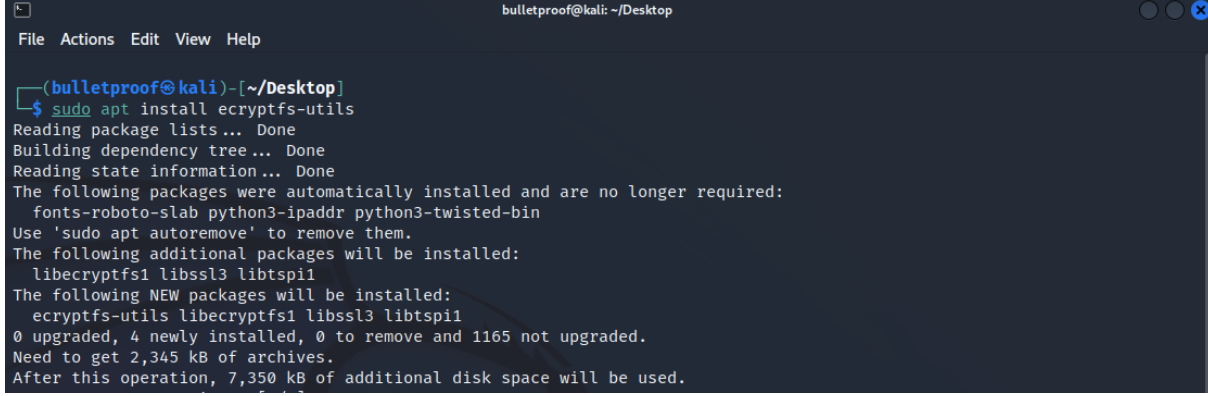
sudo usermod -u selim komutu kullanabiliriz. Buradaki -u parametresi "unlock" anlamı taşır.

sudo passwd -l selim komutunu çalıştırdığımızda ise bu hesabı kilitlemiş oluyorum. Aynı şekilde, **sudo passwd -u selim** komutunu çalıştırdığımda ise hesabın kilidini kaldırmış oluyorum.

ecryptfs Sisteminin Kullanımı

Linux Güvenliği ve Sıkılaştırması serisinde sizlere ecryptfs sisteminin kullanımından yani korumalı bir klasörün nasıl oluşturulabileceğinden bahsediyor olacağım. Oluşturacağımız korumaları klasör içerisinde dosyalarımız otomatik olarak şifrelenecek ve bir başkasının erişimi otomatik olarak engellenecektir. Bu maksatla öncelikle bir kütüphane indirmemiz gerekiyor;

“**sudo apt install ecryptfs-utils**” komutu ile aşağıda gördüğümüz şekilde kütüphanemizi indirmiş olacağız.



```
bulletproof@kali: ~/Desktop
File Actions Edit View Help

(bulletproof@kali)-[~/Desktop]
$ sudo apt install ecryptfs-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
 fonts-roboto-slab python3-ipaddr python3-twisted-bin
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
 libecryptfs1 libssl3 libtspi1
The following NEW packages will be installed:
 ecryptfs-utils libecryptfs1 libssl3 libtspi1
0 upgraded, 4 newly installed, 0 to remove and 1165 not upgraded.
Need to get 2,345 kB of archives.
After this operation, 7,350 kB of additional disk space will be used.
```

Kütüphaneyi indirdikten sonra akabinde bir klasör oluşturalım;

“**sudo mkdir /secrets**”

Ardından,

“**sudo mount -t ecryptfs /secrets /secrets**”

komutu ile bu klasörü (secrets) kendi üzerinden mount ediyorum ve sistemin onu tanımasını sağlayacak bir mekanizma kurmuş oluyorum. Resim-1’de gördüğümüz gibi;

- Bizden bir Passphrase veya TSPI seçmemiz isteniyor. Passpharase seçerek devam ediyoruz ve bir değer giriyoruz.
- Ardından hangi şifreleme algoritması kullanacağımızı belirtiyoruz. (Burada enter diyerek default olarak AES şifreleme algoritmasını seçmiş olduk.)
- Akabinde kaç bitlik bir şifreleme olacağını belirtmemiz gerekiyor. (Burada enter diyerek default olarak 16 bitlik seçmiş olduk.)
- Ardından “Enable plaintext passthrough” kısmına “no” diyerek devam ediyoruz.
- Sonraki adımda bize “Enable filename encryption” yani dosya ismi şifrlenmesi olsun mu diye soruluyor. Dosya isimlerinin de şifrlenmesini istiyorsak bu adımda “yes” diyerek devam edebiliriz.
- Son olarak “Filename Encryption Key (FNEK) Signature [c24d33bac5c66ca2]” yani Key olarak da “c24d33bac5c66ca2” değerini kullanayım mı diye soruluyor. Buna da “yes” diyerek işlemi tamamlamış oluyoruz.


```
bulletproof@kali: ~/Desktop
File Actions Edit View Help

(bulletproof@kali)-[~/Desktop]
$ sudo mount -t ecryptfs /secrets /secrets
Select key type to use for newly created files:
1) passphrase
2) tspi
Selection: 1
Passphrase:
Select cipher:
1) aes: blocksize = 16; min keysize = 16; max keysize = 32
2) blowfish: blocksize = 8; min keysize = 16; max keysize = 56
3) des3_ede: blocksize = 8; min keysize = 24; max keysize = 24
4) twofish: blocksize = 16; min keysize = 16; max keysize = 32
5) cast6: blocksize = 16; min keysize = 16; max keysize = 32
6) cast5: blocksize = 8; min keysize = 5; max keysize = 16
Selection [aes]:
Select key bytes:
1) 16
2) 32
3) 24
Selection [16]:
Enable plaintext passthrough (y/n) [n]:
Enable filename encryption (y/n) [n]: y
Filename Encryption Key (FNEK) Signature [c24d33bac5c66ca2]:
Attempting to mount with the following options:
ecryptfs_unlink_sigs
ecryptfs_fnek_sig=c24d33bac5c66ca2
ecryptfs_key_bytes=16
ecryptfs_cipher=aes
ecryptfs_sig=c24d33bac5c66ca2
```

Şimdi ise bir dosya şifreleme işlemi gerçekleştirebilmek için “secrets” adında bir klasör oluşturun ve içerisine “cok_gizli.txt” adlı bir metin dosyası açarak içine de “burada cok cok gizli bilgiler var...” yazarak kaydediyorum.

```
(bulletproof@kali)-[/secrets]
$ sudo nano cok_gizli.txt

(bulletproof@kali)-[/secrets]
$ ls
cok_gizli.txt

(bulletproof@kali)-[/secrets]
$ cat cok_gizli.txt
burada cok cok gizli bilgiler var ...
```

Ardından secrets klasörü ile işim bittiğinde “**sudo umount -l /secrets**” komutu ile sistemden umount ediyorum. Şimdi ise bu klasöre ve içerisinde verilere erişmeye çalışalım;

```
(bulletproof@kali)-[/secrets]
$ ls
ECRYPTFS_FNEK_ENCRYPTED.FWb0HHCuLQNgcUSxW2SgReBi0F1Gh8Rb4jeFb6PJXpoVl7yTxFVwpnQT.U--

(bulletproof@kali)-[/secrets]
$ cat ECRYPTFS_FNEK_ENCRYPTED.FWb0HHCuLQNgcUSxW2SgReBi0F1Gh8Rb4jeFb6PJXpoVl7yTxFVwpnQT.U--
%***I*:*^*
*3DUfw*`***Xf*Y***^"***_CONSOLE*M3***l*x[*L*4j*9*[*B*%=Cm0N*P****A)***A*0**\*=z*
qL*Ü**L
D*ejw*4*!A*8y***'***i***J**WW1*elC***[YqR*o***!g*dV***I_*g**w*5yyÇ***Ü*
e*p*f*`=*o'*
*`*Kr9*q**C/*11***
Ph**<zR*?E***#*5**Y**J**E**nu`
>n9*S8***$w *>*I**7N*u*`*@q**8=***x*FG**~**]*x**W*:u***`***l~*a[9**qbr**6**W**X*BUX**
]***,*l*:p***8=***<z*****C*RT*bM***m*k*.*
**[s7*j*~*u*k*\p*~*d 8***S*x**x4**C*8.*p*39*
T**f*5nh!*r*v*b***0>F**k7M*i***X**z*k{*}**
**B
**8a***яI*f!
o*8*****O"*H=**U**63vU 0[
Ö*!
]7**w*>*JX u`s>t$*****=***
```

Klasör adı ve içerisindeki metin dosyasının klasör şifreleme işlemi sonucu nasıl bir hal aldığını görmüş olduk.

Verilere tekrar erişmek ve değişiklik yapabilmek için tekrar **secrets** klasörünü **mount** etmemiz gerekir. Bu işlemi;

```
sudo mount -t ecryptfs /secrets /secrets -o  
key=passphrase,ecryptfs_cipher=aes,ecryptfs_key_bytes=16,ecryptfs_passthrough=no,ecryptfs_enable_filename_  
crypto=yes
```

komutu ile Resim-5'te gördüğümüz gibi gerçekleştirebiliriz. (Dikkat edilecek olursa, komut içerisindeki parametreler şifreleme yapısını inşa ederken belirlediğimiz değerlerden oluşuyor.)

```
(bulletproof@kali)-[/secrets]  
$ sudo mount -t ecryptfs /secrets /secrets -o key=passphrase,ecryptfs_cipher=aes,ecryptfs_key_bytes=16,ecryptfs_passthrough=no,ecryptfs_enable_filename_crypto=yes  
Passphrase:  
Filename Encryption Key (FNEK) Signature [c24d33bac5c66ca2]:  
Attempting to mount with the following options:  
  ecryptfs_unlink_sigs  
  ecryptfs_fnek_sig=c24d33bac5c66ca2  
  ecryptfs_key_bytes=16  
  ecryptfs_cipher=aes  
  ecryptfs_sig=c24d33bac5c66ca2  
Mounted eCryptfs
```

secrets klasörünü tekrar kontrol ettiğimizde verilerimiz açık ve anlaşılır bir biçimde karşımıza çıkacaktır.

```
(bulletproof@kali)-[/secrets]  
$ ls  
cok_gizli.txt  
  
(bulletproof@kali)-[/secrets]  
$ cat cok_gizli.txt  
burada cok cok gizli bilgiler var...
```

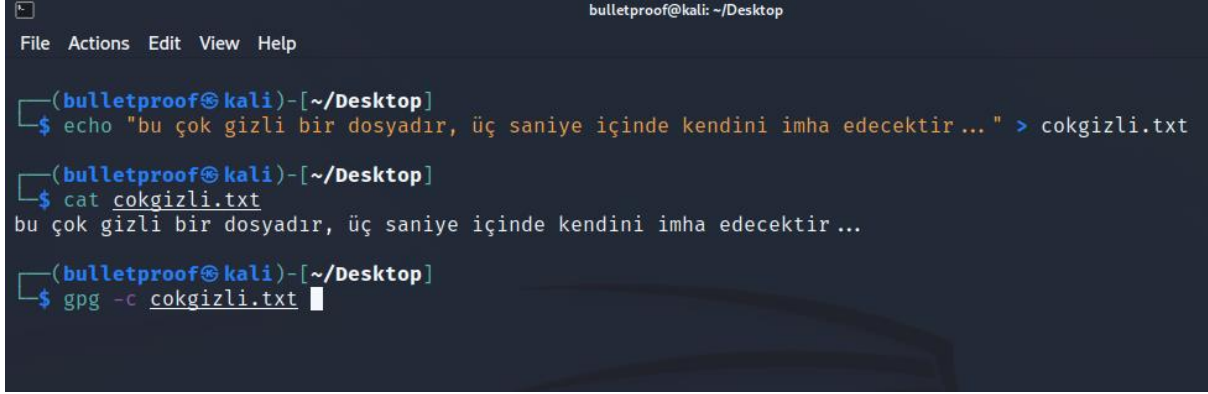
GPG ile Şifreleme Uygulamaları

Bu başlık altında yine şifreleme konusundan devam ediyor olacağız. Konuyla alakalı GPG isimli bir aracı kullanacağız. GPG temelinde, kökeni 1991’li yıllara kadar giden Pretty Good Privacy (PGP) isimli bir aracın yönteminin devamı niteliğindedir. Aşağıda konuyla alakalı olarak hem simetrik hem de asimetrik şifreleme örnekleri yapacağız.

GPG ile Simetrik Şifreleme Uygulaması

echo "bu çok gizli bir dosyadır, üç saniye içinde kendini imha edecektir..." > cokgizli.txt komutu ile cokgizli.txt adlı bir dosya oluşturarak içerisine bir metin yazdıktan sonra kaydettik.

Ardından **gpg -c cokgizli.txt** komutunu kullanarak dosyayı şifreleme işlemini gerçekleştireceğiz.



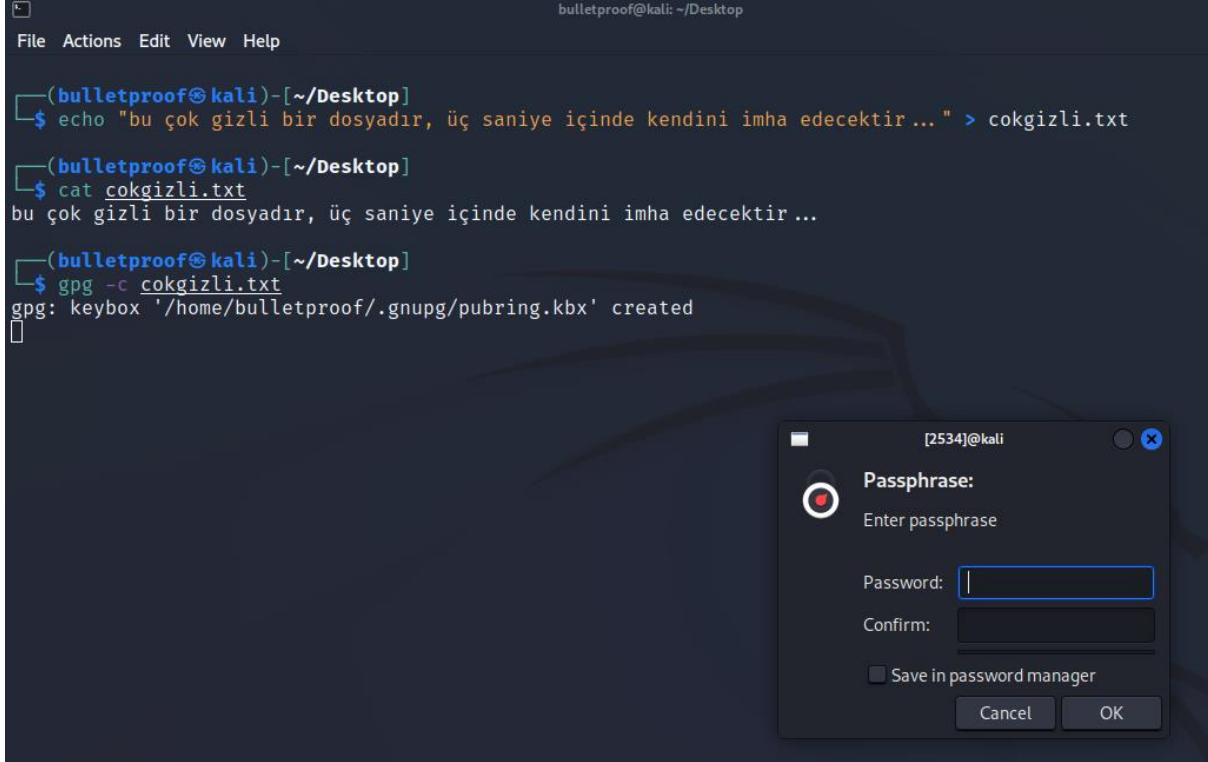
```
bulletproof@kali: ~/Desktop
File Actions Edit View Help

(bulletproof@kali)-[~/Desktop]
$ echo "bu çok gizli bir dosyadır, üç saniye içinde kendini imha edecektir..." > cokgizli.txt

(bulletproof@kali)-[~/Desktop]
$ cat cokgizli.txt
bu çok gizli bir dosyadır, üç saniye içinde kendini imha edecektir...

(bulletproof@kali)-[~/Desktop]
$ gpg -c cokgizli.txt
```

gpg -c cokgizli.txt komutunun ardından aşağıdaki resimde görüldüğü gibi bizden bir parola girilmesini istedi.



```
bulletproof@kali: ~/Desktop
File Actions Edit View Help

(bulletproof@kali)-[~/Desktop]
$ echo "bu çok gizli bir dosyadır, üç saniye içinde kendini imha edecektir..." > cokgizli.txt

(bulletproof@kali)-[~/Desktop]
$ cat cokgizli.txt
bu çok gizli bir dosyadır, üç saniye içinde kendini imha edecektir...

(bulletproof@kali)-[~/Desktop]
$ gpg -c cokgizli.txt
gpg: keybox '/home/bulletproof/.gnupg/pubring.kbx' created
```

[2534]@kali

Passphrase:

Enter passphrase

Password:

Confirm:

☐ Save in password manager

Cancel OK

Parolamızı girdikten sonra **ls** komutu ile oluşturulan şifrelenmiş dosyayı kontrol ettiğimizde **cokgizli.txt.gpg** adlı bir dosyanın oluştuğunu görüyoruz.

```
(bulletproof@kali)-[~/Desktop]
$ ls
cokgizli.txt  cokgizli.txt.gpg  rockyou.txt  Tools
```

Şifrelenmiş dosyamızı **cat cokgizli.txt.gpg** komutuyla okumaya çalıştığımızda anlamsız verilerle karşılaşyoruz.

```
(bulletproof@kali)-[~/Desktop]
$ cat cokgizli.txt.gpg
uv48T+oLn(o**S+3Y+y+k`9(g+mN+h*:WuQ` ;A1>Q6?e+3=U+hN2
S@ZÛyv
>@31F,IN0fd;

(bulletproof@kali)-[~/Desktop]
$
```

Bunun yanı sıra şifrelenmemiş olan **cokgizli.txt** dosyamızın da artık sistem üzerinde bulunmasını istemiyoruz diyelim. Bildiğimiz üzere Linux'ta bu klasörü silebilmemiz için **rm** komutundan yararlanabiliriz ancak bu gerçekten çok gizli bir dosya ise ben bu dosyayı **rm** komutu ile yok etmek istemem çünkü Forensics (Adli Bilişim) yöntemleri ile bu dosyayı geri getirmek, kısmi olarak dahi olsa okuyabilmek mümkün olacaktır. Bunun için direkt dosyayı ve verileri wipe eden yani silip süpüren özel bir komut kullanacağız. Aracımızın ismi **shred**.

shred -u -z cokgizli.txt komutu ile txt dosyasını sildikten sonra geriye yalnızca **pgp** uzantılı şifrelenmiş dosyamızın kaldığını görüyoruz.

```
File Actions Edit View Help

(bulletproof@kali)-[~/Desktop]
$ ls
cokgizli.txt  cokgizli.txt.gpg  Tools

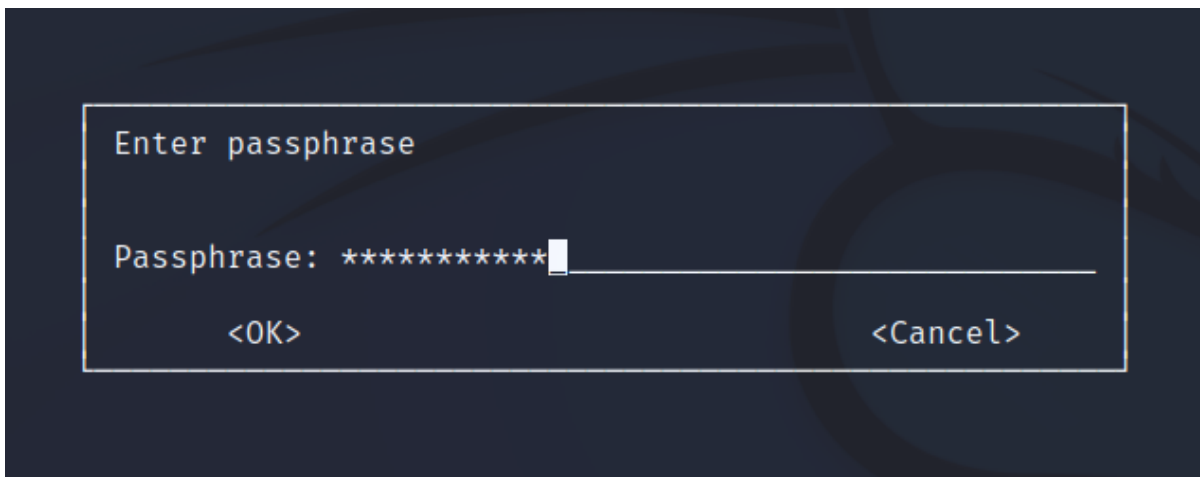
(bulletproof@kali)-[~/Desktop]
$ shred -u -z cokgizli.txt

(bulletproof@kali)-[~/Desktop]
$ ls
cokgizli.txt.gpg  Tools
```

Bu bilgisayarsa bir de **furkan** adlı bir kullanıcının olduğunu tasavvur edelim. Aşağıdaki resimde gördüğümüz gibi kullanıcı şifrelenmiş dosyayı okuyamıyor ve dosya hakkında **file** komutu ile bilgi almaya çalıştığında GPG ile simetrik olarak şifrelendiği bilgisi ile karşılaşılıyor.

“**sudo gpg -d cokgizli.txt.gpg**” komutunu kullanarak dosya decrypt edilmek istediğinde karşımıza dosyayı şifrelerken verilen parolayı isteyen bir ekran gelir.

```
furkan@kali: ~  
File Actions Edit View Help  
  
(furkan@kali)~  
$ ls  
cokgizli.txt.gpg  
  
(furkan@kali)~  
$ cat cokgizli.txt.gpg  
uv8T+oLn(o**S+3Y+y+k`9(♦♦♦♦g♦♦mN♦♦♦h*:W♦uQ♦ ♦`♦;♦A1♦>♦Q6♦?e♦♦3♦=♦U♦hN2  
♦S♦@♦Z0yy  
>@♦♦♦3♦♦@♦1♦♦♦♦F,IN0♦♦f♦d;♦♦  
  
(furkan@kali)~  
$ file cokgizli.txt.gpg  
cokgizli.txt.gpg: GPG symmetrically encrypted data (AES256 cipher)  
  
(furkan@kali)~  
$ sudo gpg -d cokgizli.txt.gpg
```



ve parolayı girdiğimizde şifreli dosya decrypt edilmiş bir şekilde karşımıza gelecektir.

```
(furkan@kali)~  
$ sudo gpg -d cokgizli.txt.gpg  
gpg: directory '/root/.gnupg' created  
gpg: keybox '/root/.gnupg/pubring.kbx' created  
gpg: AES256.CFB encrypted data  
gpg: encrypted with 1 passphrase  
bu çok gizli bir dosyadır, üç saniye içinde kendini imha edecektir ...  
  
(furkan@kali)~  
$
```

Gördüğümüz gibi simetrik şifreleme işlemini gerçekleştirmek oldukça kolaydır. Şimdi Asimetrik şifreleme ile işlemler gerçekleştirelim.

GPG ile Asimetrik Şifreleme Uygulaması

Burada ben, bulletproof kullanıcı olarak kendim için hem açık hem de gizli anahtarı üretmek istiyorum. Bunun için, “**sudo gpg --full-generate-key**” komutunu kullanıyoruz. Ardından aşağıdaki görselde gördüğümüz gibi bizden kullanmak istediğimiz şifreleme yöntemini belirlememizi istiyor. Ben burada 1 seçiyorum (RSA). Ardından istediğimiz anahtar uzunluğunu soruyor. Bu varsayılanda 3072’dir. Enter deyip devam ediyoruz. Akabinde bu oluşturacağımız anahtarın ne kadar geçerli olacağını soruyor. Ben burada bir süre kısıtı koymak istemiyorum ve enter diyerek varsayılan (0) seçerek devam ediyorum. En son doğrulama onayına “evet” diyerek devam ediyoruz.

```
bulletproof@kali: ~  
File Actions Edit View Help  
  
(bulletproof@kali)-[~]  
$ sudo gpg --full-generate-key  
gpg (GnuPG) 2.2.35; Copyright (C) 2022 g10 Code GmbH  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
  
Please select what kind of key you want:  
  (1) RSA and RSA (default)  
  (2) DSA and Elgamal  
  (3) DSA (sign only)  
  (4) RSA (sign only)  
  (14) Existing key from card  
Your selection? 1  
RSA keys may be between 1024 and 4096 bits long.  
What keysize do you want? (3072)  
Requested keysize is 3072 bits  
Please specify how long the key should be valid.  
  0 = key does not expire  
  <n> = key expires in n days  
  <n>w = key expires in n weeks  
  <n>m = key expires in n months  
  <n>y = key expires in n years  
Key is valid for? (0)  
Key does not expire at all  
Is this correct? (y/N) y
```

Benimle ilgili bir imza tanımlamak adına bilgilerimi girmemi istiyor ve bu işlemlerin ardından “Tamam” diyerek bu adımı tamamlamış oluyoruz. Benden bu işlemleri bir şifre ile şifrelememi istiyor.

```
bulletproof@kali: ~  
File Actions Edit View Help  
  
Please enter the passphrase to  
protect your new key  
Passphrase: *****  
  <OK>                                <Cancel>
```


Şifremizi girip onayladıktan sonra sistem bizden biraz hareket bekliyor. Bunun sebebi rastgele bir anahtar oluşturmak içindir. Faremiz ile birkaç sol tık yapmak bu durumda yeterli olacaktır. Bunun sonucunda açık anahtarımızı üretmiş olduk.

```
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: directory '/root/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/root/.gnupg/openpgp-revocs.d/8E3BD8D70399B00773EDB484E42D7AFFDEA6D69
A.rev'
public and secret key created and signed.

pub   rsa3072 2022-09-03 [SC]
      8E3BD8D70399B00773EDB484E42D7AFFDEA6D69A
uid           Furkan (a) <f@a.com>
sub   rsa3072 2022-09-03 [E]
```

```
(bulletproof@kali)-[~]
$
```

Peki bundan nasıl emin olabiliriz? “.gnupg” adlı bir klasör mevcut. Bu dizine “cd .gnupg” komutu ile gidelim. “ls” dediğimiz de burada gerekli olan dosya ve klasörlerin üretildiğini görüyoruz.

```
File Actions Edit View Help

(bulletproof@kali)-[~]
$ cd .gnupg

(bulletproof@kali)-[~/ .gnupg]
$ ls
private-keys-v1.d  pubring.kbx  random_seed

(bulletproof@kali)-[~/ .gnupg]
$
```

Şimdi, “gpg --export -a -o bulletproof_public_key.txt” komutunu yazdığımızda ve ardından “cat bulletproof_public_key.txt” dediğimizde, alışık olduğumuz açık anahtar tanımına uygun bir anahtarın üretildiğini görüyoruz.

```
File Actions Edit View Help

(root@kali)-[/home/bulletproof/.gnupg]
# gpg --export -a -o bulletproof_public_key.txt

(root@kali)-[/home/bulletproof/.gnupg]
# ls
bulletproof_public_key.txt  private-keys-v1.d  pubring.kbx  random_seed
```

```
(root@kali)-[/home/bulletproof/.gnupg]
# cat bulletproof_public_key.txt
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
mQGNBGMTnccBDADsWp8wpqcSDMPFbiTfGE3rTWL7/hlBTD/UYCSmpmQiw6EfWU2Q
C2N3RjBI6Pwg37Bng9CZyJ0g7I7cWTapMMWNSdbrwajKBPsKkc59aa1wsGln/Lfr
N3wpEYrbNmcCIzyQwnZ0nqH5LzFDKRI659cy7FJBoSzzdvYoKQ5ANjvhHPl1wbtm
PsaxCNLwg0yAiJfebpFXLhbmWn4viHM0PwPBGrkd00NIqabzwZ2DwCs0i6XmO+De
fsIFtLNS25xuwy+7/w3xWK00+0aBb1Nq1CCcZKnSK8lowWoewaTZlljAew+p0mY
eni+WI4V1fJBW80JyJX003hrS3P3P/k1l0o+mZQjEi10S8MeEWQwaU1ATmx8WFRl
OMc5T7ucoSmH9co0PTmZ1AVNGFgLFH8gdjyZHu60Ljk/y5IEcpz16oS76V84DtcR
G4S9hDBY7J3gkktzZWBV9kXphhC/SYs738hLDRR3RisZQm1OEZHdsyTdu9EHiv3B
yJfHKe37zU/WUpMAEQEAAbQURnVya2FuIChhKSA8ZkBlmNvbT6JAc4EEwEKADgW
IQS009jXA5mwB3PttITkLXr/3qbWmgUCYx0dxwIbAwULCQgHAgYVCgkICwIEFgID
AQIeAQIXgAAKCRDkLXr/3qbWmkMdC/0QhUAv30G4WPBedKoxmEIGTa1ermMZjcFS
kjleg09h7CMeV9GXAChImVrrNYyTJNr2Ts4gtFJ4B1kNxFX9fLGxRZDmrLjbCVyD
1cRggeoIIPbEez7W8z3Qn0HUIjfaBJhjLpTQGv0ZxjmW3/rOC1g/NfXYb132qKxG
WNT0A+23xPZlBzzwkRXdeehNcyv6jo+gyYkUGYV8W53MAalSx8wM0CEXDgs71NWz
kPja4jhzvS0rTxVOMcSckwAznAhjX+egwAMFlrvJU5u1tubTSkAAIvvA3jLU0n8
VaIOeMt+i/bVIznOfghhT2lI+wHM1PKE/rJ2mkpdktPZ9uZ09uxwdEMdWeQHMK6
ij284YbkKZlwYg+5A3lvBKysjHhLMJkvqzq9HaeZYh5eOFhdDiknqGKKJBw2VhiT
FosPcnrxjr6PNWD07wxFcVszexVu4Z+ukMLgXmoOP8dNrkpXL8d0S54+Bo5b/Ovk
7Q9y4+cqVMkML7nIOXKxkpUPWC/4YIK5AY0EYx0dxwEMANh6V5FJLcvEI/h7kNt7
4InAa5rpStv8gNYlgoDMog6weHF5nlmvZz4/yIJa3MrB4CZ7iENpSm3u51YvGzT8
paNs2F4cNQ07sJqKs0wqR+msD6kyeNceoM0fpAxkJP4w4LJD6/0Q5RTNQuGSK6LY
ufu8/iBj/TvRbr5Zm/3GOL1YXomovPKr217mZ1ZcCUfV7eMF9c9l8CCNz151Y3Rz
b/mHVFXtKC5wSq2C18LFSguQkIhu7I17UbI0sUrjgS8B00ytUae0aDoBf3BQSVZ/
M0rmKH7eeMGqt6rWFnmScZZv362z20T/mRRJaipWK/nOfgtyjAk9zDLV8kbZHOYi
PK2sgVmi03KLQzkbjgtNVtoNP2wH22KSdNX8DTkjMqbT1I9Xx7jmC1pS2JLlPyMM
4dlgPcETTS0VQtqQlMj57dJIOtDFHNHGcm9GSyFNFpmsCZLmJvX7a2WVHhD6WQB
5EeU0rNrPjsa5kgmHB35g/upZ71GeTqdPDJZFV0MiLu1+wARAQABiQG2BBgBCgAg
FiEEjYV1w0ZsAdz7bSE5C16/96m1poFAMMTnccCGwwACgkQ5C16/96m1prc5gv8
C3JpXIWmhUZJArHu0vJFC82repicXNRRJ21wxLgYKF6kELZCJYqcCCPkx0KAFque
cUalysa/HVfLKUqeqlUobc6JEubLZgWXPZekuwrt3sPwmBfsXGJG0k81YJgm/z
SQFQdrTaz5SjtSGy+JndIlaNAK9a3pDxTF800mvrMpn4x42P1pem4AXfhYPD+ri4
Ja1c0IyExJgQY9MWpaNXWY0cL+bzc70zAqOC8e4BY/e3n963b1g8xMq6f4sF20CD
o+W80QlthMpFZBU8xPxrcXQ27Xeqs8lN2sXCqhHWAD0coc/NNdul9QRMfV+vpRvS
fSkg4mGy4JN4Q5FZkFmqFDqvmvUSAns3Ewp66riGVovr3aW7VFepK7RSaMPuJ1X0
MR+DGnjt6gMNFKEfELJeyjNdE8+dVWNN+9nDL8mAsYs2DpUzp1Ew3VR0TrKYtZhS
VZOQZlTv+VwDVecwZJNJ6fzSQAx2KVjPhpDx4A+kkck+Gfg+LxjYA/MpK9c9K/kD
=jIjV
```

```
-----END PGP PUBLIC KEY BLOCK-----
```


Şimdi ise farklı bir kullanıcının (furkan) anahtar altyapısını burada tanımlamak istiyorum. Bu sayede bulletproof ve furkan kullanıcıları, birbirlerinin public key'lerini kullanarak şifrelenmiş dosyaların içeriğine erişebileceklerdir. Ben daha önceden furkan kullanıcısı için, bulletproof kullanıcısı için yaptığımız gibi, aynı şekilde bir anahtar tanımlaması yapmıştım. İstiyorum ki furkan kullanıcısının açık anahtar alt yapısını burada tanımlayalım. Bunun için bulletproof kullanıcısının .gnupg dizininde;

"sudo gpg --import /home/furkan/.gnupg/furkan_public_key.txt"

komutu ile furkan kullanıcısının public key'ini aşağıdaki resimde gördüğümüz gibi almış olduk.

```
(bulletproof@kali)-[~/gnupg]
$ sudo gpg --import /home/furkan/.gnupg/furkan_public_key.txt
gpg: key E42D7AFFDEA6D69A: "Furkan (a) <f@a.com>" not changed
gpg: key 41A8C1E08C7F63AB: "furkan (furkan) <info@furkan.com>" not changed
gpg: Total number processed: 2
gpg: unchanged: 2
```

Şimdi bir dosya üretelim ve bunu ve bunu kendi özel anahtarımızda encrypt edelim.

echo "bu çok çok gizli bir dosyadır..." > cokcokgizli.txt

```
(bulletproof@kali)-[~/Desktop]
$ echo "bu çok çok gizli bir dosyadır ..." > cokcokgizli.txt
```

komutu ile bir dosya oluşturduk ve şimdi bunu kendi gizli anahtarımızla encrypt etmek için;

sudo gpg -s -e cokcokgizli.txt

dedikten sonra bizden karşı tarafın kullanıcı ismini istiyor. Buraya furkan yazarak devam ediyoruz.

```
(bulletproof@kali)-[~/Desktop]
$ sudo gpg -s -e cokcokgizli.txt
You did not specify a user ID. (you may use "-r")

Current recipients:

Enter the user ID. End with an empty line: furkan

Current recipients:
rsa3072/8C10E499C669FEAA 2022-09-03 "Furkan (a) <f@a.com>"

Enter the user ID. End with an empty line: 
```

ve ardından parolamızı da girdikten sonra OK diyerek devam ediyoruz.

```
Please enter the passphrase to unlock the OpenPGP secret key:
"Furkan (a) <f@a.com>"
3072-bit RSA key, ID E42D7AFFDEA6D69A,
created 2022-09-03.
```

Passphrase: *****

<OK>

<Cancel>

Aşağıda gördüğümüz gibi gpg uzantılı encrypt edilmiş dosyamız oluşturulmuş oldu.

```
(bulletproof@kali)-[~/Desktop]
$ ls -l
total 16
-rw-r--r-- 1 bulletproof bulletproof 36 Oct 3 00:20 cokcokgizli.txt
-rw-r--r-- 1 root root 965 Oct 3 00:24 cokcokgizli.txt.gpg
```

Şimdi ise furkan kullanıcısına geçiş yaparak bulletproof kullanıcısının şifrelediği dosyayı açmaya çalışalım.

“sudo gpg -d cokcokgizli.txt.gpg” komutunu yazdıktan sonra bizden belirttiğimiz parolayı girmemizi istiyor.

```
(bulletproof@kali)-[~/Desktop]
$ su furkan
Password:
$ /bin/bash
(furkan@kali)-[/home/bulletproof/Desktop]
$ sudo gpg -d cokcokgizli.txt.gpg
```

```
Please enter the passphrase to unlock the OpenPGP secret key:
"Furkan (a) <f@a.com>"
3072-bit RSA key, ID 8C10E499C669FEAA,
created 2022-09-03 (main key ID E42D7AFFDEA6D69A).
```

Passphrase: *****

<OK>

<Cancel>

ve bu işlemin sonucunda gizli dosyanın içeriğine ulaşmış olduk.

```
(furkan@kali)-[/home/bulletproof/Desktop]
$ sudo gpg -d cokcokgizli.txt.gpg
[sudo] password for furkan:
gpg: encrypted with 3072-bit RSA key, ID 8C10E499C669FEAA, created 2022-09-03
"Furkan (a) <f@a.com>"
bu çok çok gizli bir dosyadır...
gpg: Signature made Mon 03 Oct 2022 12:23:57 AM +03
gpg: using RSA key 8E3BD8D70399B00773EDB484E42D7AFFDEA6D69A
gpg: Good signature from "Furkan (a) <f@a.com>" [ultimate]
```

Bu bölümde bir dosyayı hem simetrik hem de asimetrik yöntemle oldukça pratik bir araçla Linux ortamında şifrelemek ve koruma altına almak mümkün. Bu teknolojiyi günlük hayatımızda da kullanmamız gerektiğini düşünüyorum. Bu gibi uygulamaların dosyalarınızın çalınma riskine karşı, fidye yazılımlarına karşı çok daha güvenli olacağını göreceksiniz.

iptables Kullanımı

Bu başlık altında iptables ile güvenlik duvarını yapılandırmayı inceleyeceğiz. İlk olarak sistemimizdeki mevcut güvenlik duvarı yapılandırmasını görmemin yolu nedir bunu bir inceleyelim.

“**sudo iptables -L**” komutu ile aşağıdaki resimde de gördüğümüz gibi güvenlik duvarı yapılandırmamızda 3 adet ana kategori bulunmaktadır. Input, Forward ve Output. Bunların tamamı için ise genel politika Accept olarak tanımlanmıştır.

```
(bulletproof@kali)-[~/Desktop]
$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

Önemli bir noktaya değinecek olursak, buradaki kurallar IPv4 için geçerlidir. Eğer IPv4 için kurallar tanımlamak istiyorsak “**sudo ip6tables -L**” şeklinde aşağıdaki resimde de gördüğümüz gibi bir değişiklik yapmamız gerekir.

```
(bulletproof@kali)-[~/Desktop]
$ sudo ip6tables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

Şimdilik biz burada IPv4 üzerine çalışalım ve bir senaryo kurgulayarak başlayalım. Diyelim ki ben bir sunucu kurdum ve bu sunucuda adım adım sıkılaştırma işlemlerini gerçekleştirmek istiyorum. Kaynağı benim sunucum olan bir trafik karşı tarafla bir bağlantı kurduysa benim bu trafiği engellememin bir mantığı yok çünkü trafiğin kaynağı benim. Dolayısıyla ben bu tarz trafiklere müsaade eden bir kural ile başlayayım;

sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

-A parametresi append yani eklemek anlamına gelir. Hangi kategoriye ekleyeceğimiz burada belirtilir.

conntrack bağlantıyı takip et anlamına gelir.

-j parametresi jump anlamına gelir ve burada aksiyon olayı belirtilir.

ESTABLISHED, bağlantı kurulmuş.

RELATED ise ilişkili anlamlarına gelir.

Aşağıdaki resimde görüldüğü gibi yazdığımız kuralı kontrol ettiğimizde, kuralın başarılı bir şekilde eklenmiş olduğunu görüyoruz.

```
(bulletproof@kali)-[~/Desktop]
$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

(bulletproof@kali)-[~/Desktop]
$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination            ctstate RELATED,ESTABLISHED

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

Başka neler yapabiliriz? Ben kurduğum ve ayağa kaldırdığım bu sunucuya uzaktan SSH aracılığıyla ulaşmak istedim diyelim. O zaman SSH servisi yani 22 numaralı portu da açmamda fayda var. Bunun için ise;

```
sudo iptables -A INPUT -p tcp --dport ssh -j ACCEPT
```

şeklinde bir kural yazarak SSH bağlantısını da kabul eden bir kuralı tanımlamış olduk.

Sunucumuz ister istemez adlandırma sunucusu ile iletişime geçecek. Bir ad almak isteyecektir. Dolayısıyla, DNS hizmeti dediğimiz sunucuyla iletişime geçmek isteyecektir. DNS protokolü, 53 numaralı porttan giden TCP ve aynı zamanda UDP protokolüdür. Dolayısıyla biz bu kuralı;

```
sudo iptables -A INPUT -p tcp --dport 53 -j ACCEPT
```

```
sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT
```

şeklinde hem TCP hem de UDP için tanımlayarak ekleyebiliriz. Şu anda aşağıdaki resimde de gördüğümüz üzere hem DNS paketleri hem de SSH paketleri kabul edilebilecek hale gelmiş oldu.

En başta düşünmemiz gereken trafiklerden ya da kurallardan bir tanesi aslında sistemin kendi kendine çelme takmaması, yani Loopback Adapter dediğimiz trafik. (127.0.0.1 veya localhost gibi trafikler.) Loopback'ı aslında ilk tanımlanması gereken kurallar olarak düşünebiliriz. Bu duruma müsaade eden bir kural tanımlayacak olursak;

```
sudo iptables -I INPUT 1 -i lo -j ACCEPT
```

Burada -I parametresini, yazdığımız kuralı listenin en başına eklemek için kullandık.

Son durumu “**sudo iptables -L -n**” komutu ile kontrol edelim. Buradaki “-n” parametresi, nümerik olarak durumu gözlemlemek için kullanılabilir.

```
(bulletproof@kali)-[~/Desktop]
$ sudo iptables -A INPUT -p tcp --dport 53 -j ACCEPT

(bulletproof@kali)-[~/Desktop]
$ sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT

(bulletproof@kali)-[~/Desktop]
$ sudo iptables -I INPUT 1 -i lo -j ACCEPT

(bulletproof@kali)-[~/Desktop]
$ sudo iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
ACCEPT     all  --  0.0.0.0/0             0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0             0.0.0.0/0          ctstate RELATED,ESTABLISHED
ACCEPT     tcp  --  0.0.0.0/0             0.0.0.0/0          tcp dpt:53
ACCEPT     udp  --  0.0.0.0/0             0.0.0.0/0          udp dpt:53

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

Eğer güvenli bir sunucu kurmak istiyorsak, ICMP dediğimiz ping trafiği baştan kapatılmalıdır ki hackerlar sizi göremesin. Aksi durumda hackerlar sizi kesinlikle görür. Bu bir nmap komutu kadar kolay bir şeydir. Yeter ki herhangi bir portunuz açık olsun. Dolayısıyla bir sistem yöneticisi olduğumuzu düşünürsek, ICMP’nin faydalarından da istifade etmemiz gerekiyor. Burada health check yani sistemin sağlığının kontrolü açısından faydalı olacak ICMP paketlerine müsaade etmek bizim için mantıklı olacaktır.

Şimdi bu işlemleri yapalım;

```
sudo iptables -A INPUT -m conntrack -p icmp --icmp-type 3 --ctstate NEW, ESTABLISHED, RELATED -j ACCEPT
```

Yazdığımız kurala dikkat edecek olursak, ICMP Type 3’e müsaade etmiş olduk. Type 3, Unreachable yani ulaşılamaz mesajlardır. Eğer bir trafikte ICMP paketleri gerçekten bir problem nedeniyle hedefe ulaşmıyorsa benim bunu engellemem mantıklı bir şey değildir, bu durumun farkında olmamız gerekir.

Aynı biçimde Type 11 trafiğine de müsaade edelim;

```
sudo iptables -A INPUT -m conntrack -p icmp --icmp-type 11 --ctstate NEW,ESTABLISHED,RELATED -j ACCEPT
```

Type 11 ise TTL yani Time to Live yani zaman aşımına uğrayan paketleri tanımlayan bir tiptir. Dolayısıyla zaman aşımına uğramış paketleri de bizim yakalayıp görmemizde fayda var. Aynı biçimde

Type 12 trafiği için de bir tanımlama yapalım;

```
sudo iptables -A INPUT -m conntrack -p icmp --icmp-type 12 --ctstate NEW,ESTABLISHED,RELATED -j ACCEPT
```

Type 12, bozuk başlık yani Bad Header denilen değere sahip olan ICMP türlerini tanımlar. ICMP Type 0 (ICMP Echo Request) ve ICMP Type 8 (ICMP Echo Reply) en çok kullanılan paketlerdir. Sonraki politikalarımızda bunları zaten drop etmiş olacağız. Ek olarak ICMP Type 5 (ICMP Echo Redirect) paketlerini de tanımlamamış olacağız. Tanımlamış olduğumuz 3, 11 ve 12 bizim için yönetim anlamında kolaylık sağlayan türlerdir.

Şimdi tanımladığımız kuralları tekrar kontrol edelim;

```
(bulletproof@kali)-[~/Desktop]
$ sudo iptables -A INPUT -m conntrack -p icmp --icmp-type 3 --ctstate NEW,ESTABLISHED,RELATED -j ACCEPT

(bulletproof@kali)-[~/Desktop]
$ sudo iptables -A INPUT -m conntrack -p icmp --icmp-type 11 --ctstate NEW,ESTABLISHED,RELATED -j ACCEPT

(bulletproof@kali)-[~/Desktop]
$ sudo iptables -A INPUT -m conntrack -p icmp --icmp-type 12 --ctstate NEW,ESTABLISHED,RELATED -j ACCEPT

(bulletproof@kali)-[~/Desktop]
$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
ACCEPT     all  --  anywhere               anywhere
ACCEPT     all  --  anywhere               anywhere             ctstate RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere               anywhere             tcp dpt:domain
ACCEPT     udp  --  anywhere               anywhere             udp dpt:domain
ACCEPT     tcp  --  anywhere               anywhere             tcp dpt:ssh
ACCEPT     icmp --  anywhere               anywhere             ctstate NEW,RELATED,ESTABLISHED icmp destination-unreachable
ACCEPT     icmp --  anywhere               anywhere             ctstate NEW,RELATED,ESTABLISHED icmp time-exceeded
ACCEPT     icmp --  anywhere               anywhere             ctstate NEW,RELATED,ESTABLISHED icmp parameter-problem

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

Şu ana kadar sadece ACCEPT yani “Kabul Et” kurallarını tanımladık. Biliyoruz ki, firewall yapılandırılırken ilk önce kabul edilen bütün kurallar tanımlanır, en son reddedilen kurallar tanımlanır. Şimdi ise DROP (paketi tamamen kes/düşür ve hiçbir şey yapma) ya da REJECT (paketi düşür, karşı tarafa paketini düşürüldüğünü bildir) paketini tanımlayalım;

sudo iptables -A INPUT -j DROP komutunu yazabiliriz ancak bu komutun ufak bir sıkıntısı var. Kuralı bu şekilde yazdığımızda INPUT kategorisinin en altına ekleyecektir. Ben bu DROP kuralından sonra yeni bir ACCEPT kuralı tanımladığımda bunu algılamayacaktır. O yüzden bunun yerine, temel politika olarak DROP’u tanımlayalım. Yani -A yerine -P olarak tanımlama yapalım;

sudo iptables -P INPUT DROP

ve bu kuralın dışındaki bütün politika, paketleri düşürmeye yönelik olacak demiş olduk. Bundan sonra tanımlayacağımız kuralların politikası DROP olacak şekilde değiştirildi. Tanımladığımız kuralları tekrar görüntülediğimizde policy durumunun DROP olarak güncellendiğini aşağıdaki resimde de görebiliriz;

```
(bulletproof@kali)-[~/Desktop]
$ sudo iptables -P INPUT DROP

(bulletproof@kali)-[~/Desktop]
$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source                destination
ACCEPT     all  --  anywhere               anywhere
ACCEPT     all  --  anywhere               anywhere             ctstate RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere               anywhere             tcp dpt:domain
ACCEPT     udp  --  anywhere               anywhere             udp dpt:domain
ACCEPT     tcp  --  anywhere               anywhere             tcp dpt:ssh
ACCEPT     icmp --  anywhere               anywhere             ctstate NEW,RELATED,ESTABLISHED icmp destination-unreachable
ACCEPT     icmp --  anywhere               anywhere             ctstate NEW,RELATED,ESTABLISHED icmp time-exceeded
ACCEPT     icmp --  anywhere               anywhere             ctstate NEW,RELATED,ESTABLISHED icmp parameter-problem

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

Bu tanımladığımız kurallar restart işleminden sonra uçabilir. Bu durumun gerçekleşmemesi için bir paket mevcut;

sudo apt-get install iptables-persistent

komutuyla paketi sisteme kurduğumuzda tanımlamış olduğumuz kuralların sistemde kalıcılığını sağlamış olacağız.

Bazen yazdığımız kuralları sıfırlamak isteyebiliriz. Bunun için ise;

sudo iptables -F INPUT ACCEPT

sudo iptables -F OUTPUT ACCEPT

sudo iptables -F FORWARD ACCEPT

Politikaları yukarıdaki gibi sıfırdan ACCEPT politikasına çeviriyoruz ve son olarak;

“sudo iptables -F” diyerek tanımladığımız kuralları temizlenmiş oluyoruz.

SSH Parolasız Eriřim Yönetimi

Bu bölümde SSH servisini kullanıcı adı ve parola girmeden, oluşturduğumuz açık ve özel anahtarları kullanarak nasıl doğrudan karşı tarafa bağlantı sağlayabileceğimizi göreceğiz. Senaryo olarak bir Kali Linux makinesi, diğer tarafta da bir Ubuntu makinesinin olduğunu düşünelim. Bu konu hacking yöntemlerinde uzaktan sisteme ele geçirme sırasında da kullanılabilir. Hem güvenlik açısından hem de pentest açısından bu senaryoyu inceleyebiliriz.

Öncelikle, “ssh-keygen” komutunu kullanarak kendimiz için bir RSA anahtarı generate ediyoruz. Bu aşamada bizden bir şifre girmemizi istiyor burayı boş geçerek devam ediyoruz. En sonunda “/root/.ssh/” dizinindeki public RSA anahtarı olan “id_rsa.pub” ve “id_rsa” dosyaları belirtilen dizinde oluşturulmuş oluyor.

```
(bulletproof@kali)-[~]
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/bulletproof/.ssh/id_rsa):
Created directory '/home/bulletproof/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/bulletproof/.ssh/id_rsa
Your public key has been saved in /home/bulletproof/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:m9n5cildfj6Yx9BuNcvAHX+08SVvZmNrHjsPtokQQk bulletproof@kali
The key's randomart image is:
+--[RSA 3072]--+
|      E      o|
|      . .    .o|
|      o .    o*|
|      .      o.=+|
|      S.  o.=..|
|      *o. =o=+|
|      =.++ =o *|
|      +*. =o.o.|
|      .oo=. ooo|
+--[SHA256]--+
```

Bu işlemin sonucunda “.ssh” dizinine gidip dosyaları kontrol ettiğimizde gerekli dosyaların oluşturulduğunu görüyoruz. “id_rsa.pub” adlı dosya yani benim açık anahtarım olan dosyayı cat komutu ile ekrana basıp içeriğini kopyalıyorum.

```
(bulletproof@kali)-[~]
$ cd .ssh

(bulletproof@kali)-[~/ .ssh]
$ ls
id_rsa  id_rsa.pub

(bulletproof@kali)-[~/ .ssh]
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDehv5S5B6Ed+mNk8WxIKjmrOgpsBPm8CDzShZ3DKHwGYdtJOyy8PzB7aTFxawVb9lhj
uKc1tKpUUG/TpzzFXoSICKV8JbFfjY2R4VGJNRNChOfcv5hgANKn9D7H4ZRozyN5hN0dibbFrRqENLKQFUKbS9RXbRMITj8bHl8JQE4/X
Co090aG8isidmpdMxOeix3jOlPURJjR5/GroyAFT9zzr3HaTORahkTQUHF9A+Zq+pdgs5yPUnI3bOS4u1jzUKSk9J3470fhY4FqgYf9RP
Qu3lkUBIS8prz1tpLFxcKgOCUpyytFnSloaWtj5abxOs4BQ3LrOYvKWldjVpku4ayJg/anuIJogc1Uqbs5uBE3zDlsCKVygZvdQSC854y
+EWdNDdpG57oZrmvDg0QbFjiKZNfWqpMbqFnJBizwi3u4MBHnONHZuZP2oTT4zS+hwz0tHseYnEQnaLwv17m4TpVZ/Q1cMSkJSBntC4wM
itnRMylKbq2E0Q/rB0mhLNlXBU= bulletproof@kali
```

Şimdi diğer Linux makinesine gelemiz. Bu makinenin .ssh dizinine, Kali makinesinden kopyaladığımız açık anahtarı,

echo “ssh-rsa

AAAAB3NzaC1yc2EAAAADAQABAAQgQDehv5S5B6Ed+mNk8WxIKjmrOgpsBPm8CDzShZ3DKHwGYdtJOyy8PzB7aTFxawVb9lhjuKc1tKpUUG/TpzzFXoSICKV8JbFfjY2R4VGJNRNChOfcv5hgANKn9D7H4ZRozyN5hN0dibbFrRqENLKQFUKbS9RXbRMITj8bHl8JQE4/XCo090aG8isidmpdMxOeix3jOlPURJjR5/GroyAFT9zzr3HaTORahkTQUHF9A+Zq+pdgs5yPUnI3bOS4u1jzUKSk9J3470fhY4FqgYf9RPQu3lkUBIS8prz1tpLFxcKgOCUpyytFnSloaWtj5abxOs4BQ3LrOYvKWldjVpku4ayJg/anulJogc1Uqbs5uBE3zDlsCKVygZvdQSC854y+EWdNDdpG57oZrmvDg0QbFjiKZNfWqpMbqFnJBizwi3u4MBHnONHZuZP2oTT4zS+hwz0tHseYnEQnaLwv17m4TpVZ/Q1cMSkJSBntC4wMitnRMylKbq2E0Q/rB0mhLNlXBU= bulletproof@kali” >> authorized_keys

şeklinde “authorized_keys” adındaki bir dosyaya aşağıdaki resimde gördüğümüz biçimde yazıyoruz.

sudo Komutunun Kullanımı

sudo -l : komutu sudo grubuna üye olan bir kullanıcının hangi yetkilere sahip olduğunu gösterir. sudo komutunu ilk defa kullandığımızda bizden bir şifre isteyecektir ve 5 dakika boyunca bu yetki geçerli olacaktır ancak ben,

sudo -k : komutunu kullanırsam, ardından tekrar sudo -l dersem, benden tekrar bir şifre isteyecektir. Yani -k parametresi sudo yetkisini kill ediyor diyebiliriz.

sudo -i : komutu ile direkt olarak root olabilirim.

sudo useradd -m Furkan : komutu ile bir kullanıcı oluşturulur. [-m parametresi /home dizini oluşturması için] veya

sudo useradd Furkan -m -d /home/Furkan

sudo passwd Furkan : komutu ile oluşturulan kullanıcıya şifre verilir.

su Furkan : komutu ile Furkan kullanıcısına geçiş yapılır ve ardından,

“sudo -i” komutu ile root olmaya çalışırken şifre girdiğimizde, işletim sistemi bize bu kullanıcının henüz sudoers dosyası içerisinde olmadığına dair bir bilgi verecektir. Bu durumda,

sudo usermod -aG sudo Furkan : komutu ile Furkan kullanıcısını sudo grubuna eklemiş olduk ve bu kullanıcı artık sudo yetkilerine “(ALL : ALL) ALL” sahiptir.

Bir kullanıcıyı daha spesifik olarak yetkilendirebilmek için “**sudoers**” dosyasını düzenleyebiliriz.

“sudo cat /etc/sudoers” komutu ile sadece bu dosyayı okuyabiliriz. Bu dosya geçici bir dosyadır. Bu dosyayı düzenleyebilmek için özel bir komut mevcuttur;

sudo visudo : komutu ile sudoers dosyasını düzenleyebilirim. Örneğin en alt satıra inerek;

Furkan ALL=(ALL) ALL satırını yazdığımızda aslında **sudo usermod -aG sudo Furkan** komutunu manuel gerçekleştirmiş oldum.

Daha spesifik düşünelim. Ben bazı durumlarda belki de kullanıcının sadece belli başlı komutları çalıştırmasını da isteyebilirim. Bunun için bir örnek yapalım;

Furkan ALL=(ALL) /bin/systemctl status networking

Furkan ALL=(ALL) /sbin/fdisk

satırlarını sudoers dosyasına ekledim ve bu kullanıcıya başka bir yetki vermedik, bu durumda sadece bu iki komutu kullanabilecektir.

Şimdi ise başka bir güvenlik konusundan söz edelim. Ben tellme.sh diye bir bash dosyası oluşturdum ve içeriği;

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

```
echo “merhaba”
```

```
sudo -i
```

```
chmod +x tellme.sh
```

ardından sudoers dosyasına aşağıdaki gibi bir satır eklersem;

Furkan ALL=(ALL) /home/Furkan/tellme.sh

sudo ./tellme.sh komutunu çalıştırdığımızda root kullanıcısına atamış olur.

Burada bir yetki yükseltme saldırısı yapmış oluyoruz.