Gömülü Linux Sistemleri

Login'e Kadar Linux

Loop Cihazlar ile İmaj Analizi

Giriş

Bu yazıda herhangi bir disk imajının diske kurulum yapılmadan analiz edilmesinden bahsedilecektir. Disk imajının ARM veya x86 veya belirli bir makineye ait olmasının bir önemi yoktur. Analiz tekniği tamamen aynıdır.

Böyle bir analize niye gerek vardır?

Genelde bir bord satın alındığı zaman, bord ile gelen CD içinde veya bord ile ilgili internet sitesinde, bord için hazırlanmış imajlar mevcuttur. Kullanıcı bu imajı MMC kart veya USB bellek veya Flash Disk gibi ortama dump ederek hemen bordu boot edip, içindeki programları veya dosyaları inceleyebilir. Peki elimizde bord yoksa ya da borda kurmadan ön inceleme yapmak istiyorsak ne yapmalıyız? Bu yazının konusu işte budur.

1. Raspberry Pi veya Kısaca RPi

Çok uzun zamandan beri Raspberry Pi isimli cihazın piyasaya çıkmasını bekledik. Bu bekleyiş sırasında “Acaba nasıl bir Linux dağıtımı kurmuşlar?” diye de içten içe çok merak ettik ve bu cihazla ilgili bir imajı indirip aşağıdaki gibi inceledik. Bu incelemenin RPi ile ilgili tek tarafı imajın RPi’ye ait olmasıdır. Herhangi bir imaj için de aynı inceleme hiç değiştirilmeden yapılabilir.

RPi üzerinde çalışan pek çok disk imajı mevcuttur. Bizler analiz için 2012-10-28-wheezy-raspbian.zip isimli imajı kullandık. Disk imajı aşağıdaki gibi indirilip açılabilir.

```
# wget http://files.velocix.com/c1410/images/raspbian/
     2012-10-28-wheezyraspbian/2012-10-28-wheezy-raspbian.zip

# unzip 2012-10-28-wheezy-raspbian.zip
```

2. file/fdisk İle İnceleme

Download ve unzip işleminden sonra nihai disk imajı 2012-10-28-wheezy-raspbian.img ismi ile elde edilir. İlk yapılacak iş file komutu ile imajın cinsini, aşağıdaki gibi, tespit etmektir.

```
# file 2012-10-28-wheezy-raspbian.img

2012-10-28-wheezy-raspbian.img:
    x86 boot sector;
    partition 1: ID=0xc, starthead 130, startsector 8192,
            114688 sectors;
    partition 2: ID=0x83, starthead 165, startsector 122880,
            3665920 sectors,
    code offset 0xb8
```

Kolay okunması için file komutunun çıkışına boşluklar ve satırlar eklenmiştir. Çıkıştan da görüleceği gibi imajın içinde 2 adet disk bölümü (disk partition) mevcuttur.

Buradaki img dosyası doğrudan mount edilemez. Sebebi basittir. mount komutu dosya sistemine muhtaçtır. img dosyası içinde 2 adet dosya sistemi mevcuttur ama bu dosya sistemlerinin başında x86 boot sektörü ve boot kodu için ayrılan sektörler vardır ve bu fazlalık sektörler yüzünden mount komutu yukarıdaki img dosyasını doğrudan bağlayamaz.

file komutu bize img dosyası hakkında özet bilgi vermektedir. img dosyasının çok ayrıntılı bilgileri, aşağıdaki gibi, fdisk komutu ile elde edilebilir.

```
# fdisk 2012-10-28-wheezy-raspbian.img

Command (m for help): p

Disk 2012-10-28-wheezy-raspbian.img: 1939 MB, 1939865600 bytes
255 heads, 63 sectors/track, 235 cylinders, total 3788800 sect

Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
Disk identifier: 0x000f06a6

                          Device   Boot  Start  End Blocks  Id
2012-10-28-wheezy-raspbian.img1 8192 122879 57344 c W95FAT32(L
2012-10-28-wheezy-raspbian.img2 122880 3788799 1832960 83 Linu
```

Yukarıdaki çıkıştan da görüleceği gibi, disk imajı içinde 2 adet disk bölümü mevcuttur. Bu bölümlere sıra ile 2012-10-28-wheezy-raspbian.img1 ve 2012-10-28-wheezy-raspbian.img2 isimleri verilmiştir. Eğer fdisk komutunu /dev/sda ile başlatmış olsaydık, bu isimler yerine /dev/sda1 ve /dev/sda2 olacaktı. Diğer bir deyişle fdisk komutu 1. ve 2. disk bölümlerine img1 ve img2 uzantılarını eklemiştir. Bu sadece bir isimlendirmedir.

Yukarıdakı çıkışın Id sütununda c ve 83 sayıları mevcuttur. Bu sayılar 1. bölümün W95 ve 2. bölümün Linux bölümü olduğunu gösterir.

3. Bölümlerin Bağlanması

Önce W95 bölümü üzerinde çalışalım. Bu bölüm, start ve end adreslerinde gösterildiği gibi 8,192. sektörden başlar ve 122,879. sektörde son bulur. Bu durumda W95 bölümünde toplam olarak 122,879 – 8,192= 114, 687 adet sektör bulunur. Her sektör 512 bayt olduğuna göre, 114,687 adet sektörde 114,687* 512= 58,719,744 bayt bulunur. Bu değeri 1024′e bölüp yuvarlarsak, 58,719,744 / 1024 = 57,344 KB elde ederiz.

Diğer bir deyişle W95 bölümü yaklaşık 57 MB yer kaplamaktadır. 57,344 değeri, aynı zamanda fdisk çıkışının Blocks sütununda raporlanmaktadır. Blocks sütunu, 1K'lık değerleri gösterir. Sektörler de 512 bayt olduğuna göre, (end- start)/2 hesabı ile de 57,344 değeri elde edilebilir.

W95 bölümünün başlangıç sektörü start adresinde, 8,192 olarak verilmiştir. Diğer bir deyişle ilk 8,192 sektör yani ilk 4K'lık alan boot sistemleri için ayrılmıştır. Bu ilk 4K'lık alanın 1. sektörüne MBR denir.

Boot için başlangıçta kaç sektörün ayrılacağı tamamen sistemi kurana bağlıdır. Eski fdisk sistemleri boot kodları için başlagıçta 63 sektör ayırmaktaydı. Yeni fdisk sistemleri ise 2K veya 4K yer ayırmaktadır. fdisk komutunda x ile expert moda girilip b girişi yapılarak, boot kodu için gerekli alanın değeri verilebilir.

Buradaki basit hesaptan da görüleceği gibi, 1. disk bölümünde bulunan dosya sistemi, başlangıçtan itibaren 8,192*512= 4K'lık alandan sonra başlar. mount komutunda, aşağıdaki gibi offset değeri verilerek ilk 4K'lık alan atlanabilir. İlk 4K'lık alandan sonra W95 dosya sistemi bulunacağı için, mount komutu bu bölümü sorunsuz olarak bağlayacaktır.

```
# mkdir /mnt/img1
# mount -o loop -o offset=$((8192*512))
        2012-10-28-wheezy-raspbian.img  /mnt/img1
```

-o seçeneği, option anlamındadır.

-o loop ile, mount edilecek dosyanın fiziksel bir blok cihaz olmayacağını, mount edilecek dosya sisteminin sıradan bir dosya içinde olacağını belirtir. Diğer bir deyişle, mount komutuna loop cihaz kullanması gerektiği söylenir.

-o offset ile mount edilecek dosyanın, burada verilen değer kadar ilerden mount edilmesi sağlanır. Diğer bir deyişle burada verilen bayt kadar ile gidilir ve mount işlemi gelinen bayttan itibaren yapılır.

$((…)) ifadesi mount komutu ile ilgili değildir. Bash kabuğu bir komutu işletmeden evvel $((…)) ile verilen aritmatik işlemi yapar, elde ettiği değeri yerine koyar ve sonra komutu işletir. Örneğin "$ echo $((8192*512))" girişi için 4194304 değeri üretilir.

mount edilmiş 1. bölüm aşağıdaki gibi, çeşitli komutlarla incelenebilir.

```
# df /mnt/img1
# Filesystem 1K-blocks Used  Available Use% Mounted on
  /dev/loop0 57288     16872 40416      30% /mnt/img1
```

/dev/loop0 olarak gösterilen cihaz, disk imajının 1. bölümünü temsil eder. Loop cihazının burada esas görevi mount komutuna, gerçekte fiziksel olarak olmayan bir cihazı, fiziksel bir blok cihaz gibi göstermektir.

mount point olarak gösterilen /mnt/img1 dizinine girdiğimiz zaman, ilgili disk bölümünün içine de, aşağıdaki gibi, girmiş oluruz.

```
# cd /mnt/img1
# ls -l
 total 16872
 -rwxr-xr-x 1 root root 17764 Oct 28 17:40 bootcode.bin*
 -rwxr-xr-x 1 root root 142 Oct 28 22:11 cmdline.txt*
 -rwxr-xr-x 1 root root 1180 Oct 28 22:11 config.txt*
 -rwxr-xr-x 1 root root 5282 Oct 28 17:40 fixup.dat*
 -rwxr-xr-x 1 root root 2020 Oct 28 17:40 fixup_cd.dat*
 -rwxr-xr-x 1 root root 137 Oct 28 23:01 issue.txt*
 -rwxr-xr-x 1 root root 2695192 Oct 28 17:40 kernel.img*
 -rwxr-xr-x 1 root root 2104952 Oct 28 17:40 kernel_cutdown.im
```

```
-rwxr-xr-x 1 root root 9522048 Oct 28 17:40 kernel_emergency.
-rwxr-xr-x 1 root root 2347828 Oct 28 17:40 start.elf*
-rwxr-xr-x 1 root root 523144 Oct 28 17:40 start_cd.elf*
```

Buradaki dosyaların biri hariç diğerleri bizi ilgilendirmemektedir. Okuyucu buradaki herhangi bir dosyayı cp komutu ile kendi diskine alarak inceleme yapabilir. file komutu ile dosyaların cinsi, aşağıdaki gibi, hızlıca incelenebilir.

```
# file *
bootcode.bin: data
cmdline.txt: ASCII text
config.txt: ASCII English text
fixup.dat: data
fixup_cd.dat: data
issue.txt: ASCII text
kernel.img: data
kernel_cutdown.img: data
kernel_emergency.img: data
start.elf: ELF 32-bit LSB executable, version 1 (SYSV),
           statically linked, stripped
start_cd.elf: ELF 32-bit LSB executable, version 1 (SYSV),
           statically linked, stripped
```

4. Çekirdeğin .config Dosyasının Elde Edilmesi

Şu anda bu dosyalardan sadece çekirdek imajı bizi ilgilendirmektedir. Aşağıdaki basit yöntem ile çekirdek imajı içinden config dosyası elde edilebilir. Bu dosyanın incelenmesi bize sistem hakkında pek çok bilgi verecektir.

```
01 # cd /tmp
02 # dd if=/mnt/img1/kernel.img of=Image bs=64 skip=1
03 # cp /usr/src/linux/scripts/extract-ikconfig .
04 # chmod +x extract-ikconfig
05 # ./extract-ikconfig Image > kernel.config
```

Yukarıdaki 5 komutla, çekirdeğin config dosyası, kernel.config ismi ile /tmp altına çıkarılacaktır.

02. satırda kernel.img dosyası dd komutu ile Image isimli dosyaya kopyalanmaktadır. Çekirdeğin en tepesinde 64 baytlık özel bir başlık bilgisi vardır. skip=1 seçeneği ile dd komutunun, kopyalamaya başlamadan evvel, 1 blok ileri atlaması sağlanır. Blok boyu ise bs=64 ile (bs: block size) verilir. Böylece bs ve skip komutları sayesinde ilk 64 bayt atlandıktan sonra kopyalama yapılır.

Masaüstü makinede çekirdeğin kaynak kodunun /usr/src/linux altında olduğu kabul edilmektedir. Kaynak kodunun içinde scripts/ dizini altında extract-ikconfig isimli bir betik bulunmaktadır. Bu betik çekirdek imajı içinden config dosyasını elde eder.

03 ve 04. satırlarda extract-ikconfig dosyası tmp altına alınmakta +x seçeneği ile çalışabilir moduna getirilmektedir.

05. satırda ise bu betik işletilmekte ve çekirdek içindeki config dosyası elde edilmektedir. Bu dosya çekirdek kaynak kodunun köküne .config adı ile kopyalanıp, çapraz derleme yapılabilir.

5. Kökün İncelenmesi

Kök dosya sistemi 2. bölümde oturmaktadır. İkinci bölüm, aynen 1. bölümdeki mantıkla, aşağıdaki gibi bağlanabilir ve her iki bölüm umount ile serbest bırakılabilir. mount komutundaki 122,880 sayısının, 2. bölümün start sektörü olduğunu hatırlatalım. Bu değer fdisk tarafından raporlanmıştı.

```
06 # mount -o loop -o offset=$((122880*512))
              2012-10-28-wheezy-raspbian.img /mnt/img2

07 # df /mnt/img2
 Filesystem 1K-blocks Used    Available Use% Mounted on
 /dev/loop1 1828192   1280068 456476     74% /mnt/img2

08 # umount /mnt/img1
09 # umount /mnt/img2
```

Yukarıdaki çıkışta dikkat edilirse, 2. bölümü mount etmek için loop1 cihazı kullanılmıştır. mount komutu boşta duran ilk loop cihazını kullanır. Okuyucu kendi denemelerinde farklı cihaz numaraları ile karşılaşabilir. Bu durumda, muhtemelen başka programlar, kendileri için loop cihazı ayırmışlardır. losetup komutu, kullanımda olan loop cihazlar konusunda ayrıntılı bilgi verir. Örnek çıkış aşağıda verilmiştir.

```
# losetup -a
/dev/loop0: [0808]:17 (/opt/rpi/2012-10-28-wheezy-raspbian.img
          offset 4194304
/dev/loop1: [0808]:17 (/opt/rpi/2012-10-28-wheezy-raspbian.img
          offset 62914560
```

Yukarıdaki çıkışta, mount komutunda verdiğimiz offset değerleri açıkça görülmektedir.

Kök dosya sistemi /mnt/img2 noktasına bağlıdır. Kök dosya sistemine aşağıdaki gibi

girilip incelenebilir. Özellikle /etc, /etc/inittab, /etc/rc* gibi dizin ve dosyaların incelenmesi tavsiye edilir.

```
# cd /mnt/img2
# ls -l
total 92
drwxr-xr-x 2  root root  4096 Oct 29 00:39 bin/
drwxr-xr-x 2  root root  4096 Oct 29 00:12 boot/
drwxr-xr-x 3  root root  4096 Oct 28 23:59 dev/
drwxr-xr-x 89 root root  4096 Oct 29 00:52 etc/
drwxr-xr-x 3  root root  4096 Oct 29 00:09 home/
drwxr-xr-x 12 root root  4096 Oct 29 00:31 lib/
drwx------ 2  root root 16384 Oct 28 23:50 lost+found/
drwxr-xr-x 2  root root  4096 Oct 28 23:54 media/
drwxr-xr-x 2  root root  4096 Sep  2 20:08 mnt/
drwxr-xr-x 3  root root  4096 Oct 29 00:11 opt/
drwxr-xr-x 2  root root  4096 Sep  2 20:08 proc/
drwx------ 2  root root  4096 Oct 28 23:53 root/
drwxr-xr-x 7  root root  4096 Oct 29 00:50 run/
drwxr-xr-x 2  root root  4096 Oct 29 00:39 sbin/
drwxr-xr-x 2  root root  4096 Jun 20 2012 selinux/
drwxr-xr-x 2  root root  4096 Oct 28 23:53 srv/
drwxr-xr-x 2  root root  4096 Sep 13 03:17 sys/
drwxrwxrwt 2  root root  4096 Sep  2 20:08 tmp/
drwxr-xr-x 10 root root  4096 Oct 28 23:54 usr/
drwxr-xr-x 11 root root  4096 Oct 28 23:54 var/
```

6. Bir Bash Özelliği

Bash kabuğunun ilginç bir “yerine koyma” özelliği mevcuttur. Düzenli ifadelerde (regular expressions) olduğu gibi, [...] içinde yazılan her karakter, önünde bulunan ifadeye eklenir. Örneğin

```
# echo img[12]
```

girişinde, “img1 img2″ çıkışı elde edilir. Yani 1 ve 2 karakterleri, img katarının sonuna eklenmişlerdir. Diğer bir deyişle “$ echo img[12]” girişi ile “$ echo img1 img2″ girişi tamamen aynı anlamdadır. Bu durumda yukarıda 08 ve 09. satırlarda verilen umount komutları aşağıdaki gibi, tek satırda yeniden yazılabilir.

```
# umount /mnt/img1 /mnt/img2
```

Yukarıdaki ifadede görüleceği gibi aynı katarın önünde 1 ve 2 vardır. O halde umount komutu daha da kısa olarak aşağıdaki gibi yazılabilir.

```
# umount /mnt/img[12]
```

Kullanım Hakları

— yazı sonu —