Hack Android/Samsung

À l'attaque du noyau

SSTIC 2013

Etienne Comet
e.comet@lexfo.fr

Introduction

- **La sécurité des smartphones: un enjeu important**
  - Des systèmes de plus en plus répandus
  - … qui contiennent de plus en plus d'informations
- **Des attaquants acharnés**
  - Des cibles d'attaques très prisées
  - De multiples vecteurs d'attaque
  - De nombreux malwares

Les différents types d'exploits

Les exploits avec accès physique

- Bootloader unlock, flash d'une nouvelle ROM etc.
- Nécessite en général de se connecter en USB
- Utilisés généralement par les utilisateurs eux-même

Les attaques à distance

- Exploits « client-side » sur les navigateurs et autres
- Exploits sur les protocoles de communication
- Installation d'applications malveillantes (SE, corruption d'applications sur les points de distribution)
- **On obtient un accès au téléphone mais pas total**

Nécessité d'exploits en local pour élever ses privilèges

- Attaque d'applications mal sécurisées: obtention de permissions
- Attaque du système: failles de symlinks etc.: obtention du root
- **Attaque du noyau**

Le noyau android

C’est un noyau linux…

- **shell@android**:/ $ cat /proc/version
  Linux version 3.4.0-526204 (se.infra@SEP-68) (gcc version 4.6.x-google 20120106 (prerelease) (GCC)) #1 SMP PREEMPT Fri Apr 26 18:44:05 KST 2013

Avec des patchs en plus

- Améliorer la sécurité
- Fixer des bugs
- Améliorer les performances
- Mieux loguer les erreurs
- Mieux gérer l’énergie de l’appareil
- Supporter d’autres périphériques

Le noyau android (2)

- **Au total 249 patchs pour le 2.6.38, soit 3,3 Mo**
  - binder (mécanisme d'IPC spécifique à Android)
  - Ashmem (Android Shared Memory)
  - Pmem (processus memory allocator)
  - Logger (system logging facility)
  - Wakelocks
  - Paranoid network security
  - Oom handling
  - Posix Alarm Timers
  - *…*

Le noyau android (3)

- **Souvent compilé avec des options peu auditées**
- **Tourne sur ARM**
  - Une surface d'attaque importante

CVE-2011-1759:
Integer overflow in the sys_oabi_semtimedop function in arch/arm/kernel/sys_oabi-compat.c in the Linux kernel before 2.6.39 on the ARM platform, when CONFIG_OABI_COMPAT is enabled, allows local users to gain privileges or cause a denial of service (heap memory corruption) by providing a crafted argument and leveraging a race condition.

Le noyau android (4)

- **Donc le noyau est un vecteur d'attaque prometteur**

Cibler les constructeurs

- **De nombreux constructeurs se basent sur android**
- **Ils ajoutent des surcouches**
  - De nouvelles bibliothèques
  - Des applications spécifiques
- **Ils modifient le noyau**
- **Donc en ciblant des constructeurs particuliers la surface d'attaque est élargie**

Techniques de debug

- **Avec le root**
  - Le remote debugging avec un stub vmware n’est pas possible
  - Recompiler un noyau modifié
  - Utiliser SystemTap
  - Lire les informations dans /proc
  - KGTP (Linux Kernel debuggger and tracer)
- **Sans le root: à la recherche d’informations**
  - /proc/kallsyms est (presque) mort
  - dmesg est (presque) mort
  - /proc/slabinfo is still alive!
  - adb bugreport
  - Utiliser des infoleaks
  - Extraire le noyau des firmwares

Bug /dev/diag (1)

```
static int diagchar_write(struct file *file, const char __user *buf,
                          size_t count, loff_t *ppos)
{
...
        err = copy_from_user((&pkt_type), buf, 4);
        if (count < 4) {
          ... erreur et sortie de fonction ...
        }
        payload_size = count - 4;
        if (payload_size > USER_SPACE_DATA) {                          // [1]
          ... erreur et sortie de fonction ...
        }
        if (pkt_type == DCI_DATA_TYPE) {
              err = copy_from_user(driver->user_space_data, buf + 4,   // [2]
                                   payload_size);
...
              err = diag_process_dci_transaction(driver->user_space_data,  // [3]
                                   payload_size);
```

Bug /dev/diag (2)

```
int diag_process_dci_transaction(unsigned char *buf, int len)
{
…
        ret = diag_send_dci_pkt(entry, buf,len, index);                    // [4]
…
}
int diag_send_dci_pkt(struct diag_master_table entry, unsigned char *buf,
                      int len, int index)
{
    int i;
    /* remove UID from user space pkt before sending to peripheral */
    buf = buf + 4;
    len = len - 4;
    mutex_lock(&driver->dci_mutex);
    /* prepare DCI packet */
    driver->apps_dci_buf[0] = CONTROL_CHAR; /* start */
    driver->apps_dci_buf[1] = 1; /* version */
    *(uint16_t *)(driver->apps_dci_buf + 2) = len + 4 + 1; /* length */
    driver->apps_dci_buf[4] = DCI_PKT_RSP_CODE;
    *(int *)(driver->apps_dci_buf + 5) =
        driver->req_tracking_tbl[index].tag;
    for (i = 0; i < len; i++)                                              // [5]
        driver->apps_dci_buf[i+9] = *(buf+i);                              // [6]
…
}
```

Trigger le bug /dev/diag

- **L’exploit prend la forme d’une application android**
- **JNI: permet d’utiliser du code natif dans une classe JAVA**
- **Facteur mitigeant: ICS et Jelly Bean restreignent l’accès à /dev/diag au groupe qcom_diag**

Le bug tspdrv

```
--8<-- GT-I9300_JB_Opensource_Update9/Kernel/drivers/motor/tspdrv.c --8<--
static char g_cWriteBuffer[SPI_BUFFER_SIZE];
...
static ssize_t write(struct file *file, const char *buf, size_t count,
                     loff_t *ppos)
{
                     int i = 0;

                     *ppos = 0;  /* file position not used, always set to 0 */
                     if (file->private_data != (void *)TSPDRV_MAGIC_NUMBER) {
                                  DbgOut((KERN_ERR "tspdrv: unauthorized write.\n"));
                                  return 0;
                     }

                     /* Copy immediately the input buffer */
                     if (0 != copy_from_user(g_cWriteBuffer, buf, count)) {                    // [1]
                                  /* Failed to copy all the data, exit */
                                  DbgOut((KERN_ERR "tspdrv: copy_from_user failed.\n"));
                                  return 0;
                     }
                     /* Check buffer size */
                     if ((count <= SPI_HEADER_SIZE) || (count > SPI_BUFFER_SIZE)) {            // [2]
                                  DbgOut((KERN_ERR "tspdrv: invalid write buffer size.\n"));
                                  return 0;
                     }
                     ...
}
--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--
```

Le bug tspdrv: exploitation

- **Une success story bien courte…**

- **Exploitation: les grandes lignes**
  - Récupérer le bon groupe pour l'accès en écriture sur le périphérique
  - Trouver des données intéressantes à écraser
    - Pointeurs de fonction
    - Pointeurs vers des données à réécrire
  - Une possibilité: convertir le bug en infoleak

Le bug tspdrv (3)

```
shell@android:/ $ cat /proc/kallsyms | grep
    g_cwrite_buffer -A 6

00000000 b g_cwrite_buffer

00000000 B g_hap_data

00000000 b enable_count

00000000 b g_pdata

00000000 b en_mutex

00000000 b g_client
```

Conclusion

www.lexfo.fr @LexfoSecurite contact@lexfo.fr

Questions

Merci de votre attention

www.lexfo.fr

@LexfoSecurite

e.comet@lexfo.fr
contact@lexfo.fr