Android « 0-dayz hunting »

Référence : LEXSI-SSTIC12

Date : 30/05/2012

Version : V 1.0

Android 4.0 (Ice Cream Sandwich)

- Disponibilité du code source le 14/11/2011
- Possibilité d'auditer le code source à la recherche de vulnérabilités dans la 4.0
- Cibles privilégiées :
  - Les démons vold et netd (root, interactions utilisateur)
  - Le démon adbd (accès USB), les libs spécifiques à Android (libsysutils.so …)

Greppons joyeusement !

- Téléchargement des dépôts vold.git, netd.git et core.git contenant tous ces composants
- Et on grep …

```
$ grep -R system\( * 2> /dev/null
...
netd/NatController.cpp:    res = system(buffer);
netd/ThrottleController.cpp:    res = system(buffer);
netd/BandwidthController.cpp:        res = system(fullCmd.c_str());
...
```

- Wait … what ?

A la loupe …

- Vérifions dans NatController.cpp …

```
int NatController::runIptablesCmd(const char *cmd) {
    char *buffer;
...
    int res;
...
    asprintf(&buffer, "%s %s", IPTABLES_PATH, cmd);
    res = system(buffer);
    free(buffer);
    return res;
}
```

=> Pas de vérification de l'input.

A la loupe … (2)

- Où sont les appels à runIptablesCmd() ?
- Entre autres :

```
snprintf(cmd, sizeof(cmd),
            "-%s FORWARD -i %s -o %s -m state --state ESTABLISHED,RELATED -j ACCEPT",
            (add ? "A" : "D"),
            extIface, intIface);
   if (runIptablesCmd(cmd) && add) {
       // only bail out if we are adding, not removing nat rules
       return -1;
   }
```

Injection potentielle de commandes shell via « extIface » et « intIface » !

Exploitation path

- Vérifications sur ces variables ?

Exploitation path

- Vérifications sur ces variables ?
- Oui ! (ouf)

```
if (!interfaceExists(intIface) || !interfaceExists (extIface)) {
        LOGE("Invalid interface specified");
        errno = ENODEV;
        return -1;
    }
```

Exploitation path

- Vérifications sur ces variables ?
- Oui ! (ouf)

```
if (!interfaceExists(intIface) || !interfaceExists (extIface)) {
        LOGE("Invalid interface specified");
        errno = ENODEV;
        return -1;
    }
```

- Ou pas … (fail)

```
bool NatController::interfaceExists(const char *iface) {
    // XXX: Implement this
    return true;
}
```

Exploitation path (2)

- Commande « nat enable <arg2> <arg3> » dans le démon netd
  - arg2 == intIface
  - arg3 == extIface
- Envoi sur /dev/socket/netd
  - Accessible en écriture pour l'utilisateur system seulement
  - Peut-on abuser d'un service pour y écrire ?

Exploitation path (3)

- Le service NetworkManagementService dispose d'une méthode enableNat()

```
public void enableNat(String internalInterface, String externalInterface)
          throws IllegalStateException {
      mContext.enforceCallingOrSelfPermission(
              android.Manifest.permission.CHANGE_NETWORK_STATE, "NetworkManagementService");
      try {
          mConnector.doCommand(
                  String.format("nat enable %s %s", internalInterface, externalInterface));
      } catch (NativeDaemonConnectorException e) {
          throw new IllegalStateException(
                  "Unable to communicate to native daemon for enabling NAT interface");
      }
  }
```

=> Nécessite la permission CHANGE_NETWORK_STATE

Comment exploiter ?

- Application malveillante disposant de la permission CHANGE_NETWORK_STATE
- Contacte le service NetworkManagementService et appelle sa méthode enableNat(), via le mécanisme d'IPC « binder »
- Il « suffit » d'utiliser comme argument « ;/path/to/my/executable; » pour exécuter du code en tant que root.

Origine

```
commit 11b4e9b26fe7b878992162afb39f5a8acfd143ed
Author: JP Abgrall <jpa@google.com>
Date:   Thu Aug 11 15:34:49 2011 -0700
netd: all: use system() instead of logwrap() for now.
The logwrapper uses a blocking read() which does not always correctly
   detect when the child process at the other end is gone. This is a quick
   workaround ...
```

- Qui vérifie les commits ?

Correction

```
commit 8a12dd0851cc2aaa1f6ffb27e5d7616733200c36
Author: JP Abgrall <jpa@google.com>
Date:   Wed Dec 14 15:20:59 2011 -0800
netd: fix argument interpretation bug
While working around the logwrap() issue, it was replaced with system()
   which could lead to various commands getting misinterpreted.
We now use a system() equivalent that doesn't use "sh -c".
```

- [+] Correction assez rapide (1 mois post-ICS)
- [+] Seul smartphone vulnérable : Galaxy Nexus (à sa sortie)
- [-] Qui a relu le correctif ?

Correctif

- Nouvelle fonction « system_nosh() »

```
int system_nosh(const char *command)
{
...
    char buffer[255];
...
    if (strnlen(buffer, sizeof(buffer) - 1) == sizeof(buffer) - 1) {
        LOGE("command line too long while processing: %s", command);
        errno = E2BIG;
        return -1;
    }
    strcpy(buffer, command); // Command len is already checked.
...
```

- This is not the variable you are looking for …

Correctif du correctif ☺

- Des vérifications en amont empêchaient toutefois l'exploitation
- Le 7 Mars 2012, un nouveau correctif a été publié

```
commit 1babab9febdbd05b7b6d72e0728f49ee8d4b9bd2
Author: Nick Kralevich <nnk@google.com>
Date:   Wed Mar 7 12:54:41 2012 -0800
    netd: do check on command, not buffer.
```

</android>