小伙伴们的智能之旅

ANDROID、BRILLO、RPI 2B

BRILLO: SECURITY – SELINUX & CAPABILITY

2016-04-11 23:21:56 ~ 2016-11-03 22:49:50 | HZAK | 发表回复

我们知道在Linux系统中，可以建立不同的用户、组来进行权限管理。一个应用程序需要特定的用户才能执行，一个文件夹、文件可以设定用户及用户组，使得只有特定的用户组才能访问。而selinux与capability可以对权限做进一步的限制。

一般来说，我们对一个文件比较关注的是它的mode, ownership(user, group), timestamp(modify time/access time), size。对于cp命令来说，在copy文件时默认情况下只保留了mode, ownership, timestamp这些信息，对于如ext4支持额外属性的文件系统来说，在copy的过程中就会出现信息丢失的情况。

```
NAME
       cp - copy files and directories
...
DESCRIPTION
       Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.
...
       --preserve[=ATTR_LIST]
              preserve the specified attributes (default: mode,ownership,timestamps), if possible additional attributes: context, links, xattr, all
```

附一份代码用于查看ext4 xattr信息：2016_04_12_chkcap.cc

Brillo项目中与security相关的目录结构如下（以brillo-m10-dev-rpi3b为例）：

```
/local/brillo-m10-dev-rpi3b
+-- build
|   `-- tools
|       `-- fs_config
+-- device
|   +-- generic
|   |   `-- brillo
|   |       `-- sepolicy
|   `-- hzak
|       `-- rpi3b
|           +-- base
|           |   `-- sepolicy
|           `-- fs_config
|               `-- android_filesystem_config.h
+-- external
|   +-- libcap
|   |   `-- progs
|   |       +-- getcap.c
|   |       +-- getpcap.c
|   |       `-- setcap.c
|   +-- libcap-ng
|   +-- libselinux
|   |   `-- src
|   |       `-- android.c
|   +-- minijail
|   +-- selinux
|   `-- sepolicy
`-- system
    +-- core
    |   +-- include
    |   |   `-- private
    |   |       +-- android_filesystem_capability.h
    |   |       `-- android_filesystem_config.h
    |   `-- libcutil
    |       `-- fs_config.c
    `-- extra
        `-- ext4_utils
            `-- contents.c
```

- security – SELinux

Android官方参考文档：

1. https://source.android.com/security/selinux/
2. https://source.android.com/security/selinux/concepts.html
3. https://source.android.com/security/selinux/implement.html
4. https://source.android.com/security/selinux/customize.html
5. https://source.android.com/security/selinux/validate.html

官方推荐文档：

1. http://seandroid.bitbucket.org/PapersandPresentations.html
2. https://www.codeproject.com/Articles/806904/Android-Security-Customization-with-SEAndroid

3. https://events.linuxfoundation.org/sites/events/files/slides/abs2014_seforandroid_smalley.pdf
4. https://www.internetsociety.org/sites/default/files/02_4.pdf
5. http://freecomputerbooks.com/books/The_SELinux_Notebook-4th_Edition.pdf
6. http://selinuxproject.org/page/ObjectClassesPerms
7. https://www.nsa.gov/research/_files/publications/implementing_selinux.pdf
8. https://www.nsa.gov/research/_files/publications/selinux_configuring_policy.pdf
9. https://www.gnu.org/software/m4/manual/index.html

由此可以看出关于selinux这方面内容已经很全面了。

- 相关的一些命令：

a. 通过ls -Z 命令查看系统中文件的SELinux security context：

```
$ adb shell ls -Z /system/bin
u:object_r:system_file:s0               acpi
u:object_r:apmanager_exec:s0            apmanager
u:object_r:system_file:s0               audio_hal_playback_test
u:object_r:system_file:s0               audio_hal_record_test
u:object_r:system_file:s0               avahi-browse
u:object_r:avahi_exec:s0                avahi-daemon
u:object_r:system_file:s0               base64
u:object_r:system_file:s0               basename
...
```

b. 通过ps -Z命令查看当前系统进程的SELinux security context：

```
$ adb shell ps -Z
LABEL                          USER      PID   PPID  VSIZE  RSS   WCHAN            PC  NAME
u:r:init:s0                    root      1     0     6672   1508  SyS_epoll_ 00089cec S /init
u:r:kernel:s0                  root      2     0     0      0       kthreadd 00000000 S kthreadd
...
u:r:weaved:s0                  system    122   1     13304  7484  SyS_epoll_ 76b811c0 S /system/bin/weaved
u:r:webservd:s0                webserv   123   1     10592  6436  SyS_epoll_ 768431c0 S /system/bin/webservd
u:r:apmanager:s0               system    129   1     6864   4584  SyS_epoll_ 76b5c1c0 S /system/bin/apmanager
u:r:shill:s0                   root      131   1     10460  6684  SyS_epoll_ 76a161c0 S /system/bin/shill
u:r:tlsdated:s0                root      138   117   4744   368    pipe_wait 76ea9354 S /system/bin/tlsdated
...
```

c. 通过id命令查看当前shell的uid, gid, groups和SELinux security context：

```
$ adb shell id
uid=0(root) gid=0(root)
groups=0(root),1004(input),1007(log),1011(adb),1015(sdcard_rw),1028(sdcard_r),3001(net_bt_admin),3002(net_bt),3003(inet),3006(net_bw_stats),3009(readproc)
context=u:r:su:s0
```

d. 通过chcon命令修改文件的SELinux security context：

```
$ adb shell chcon
usage: chcon [-hRv] CONTEXT FILE...

Change the SELinux security context of listed file[s].

-h change symlinks instead of what they point to.
-R recurse into subdirectories.
-v verbose output.

chcon: Need 2 arguments
```

e. 还原文件默认的SELinux security context：

```
$ adb shell restorecon
usage: restorecon [-D] [-F] [-R] [-n] [-v] FILE...

Restores the default security contexts for the given files.

-D      apply to /data/data too
-F      force reset
-R      recurse into directories
-n      don't make any changes; useful with -v to see what would change
-v      verbose: show any changes

restorecon: Needs 1 argument
```

f. 通过runcon命令使程序运行在指定的SELinux security context：

```
$ adb shell runcon
usage: runcon CONTEXT COMMAND [ARGS...]

Run a command in a specified security context.

runcon: Need 2 arguments
```

- 系统文件SELinux security context创建过程

在创建system image时指定了系统文件的file context：

```
make_ext4fs -T -1 -S out/target/product/rpi3b/root/file_contexts.bin -L system -l 268435456 -a system
out/target/product/rpi3b/obj/PACKAGING/systemimage_intermediates/system.img out/target/product/rpi3b/system out/target/product/rpi3b/system
```

而file_contexts.bin是一个二进制文件，是这么创建出来的：

```
[ 35% 17/48] /bin/bash -c "(out/host/linux-x86/bin/checkfc out/target/product/rpi3b/obj/ETC/sepolicy_intermediates/sepolicy
out/target/product/rpi3b/obj/ETC/file_contexts.bin_intermediates/file_contexts.concat.tmp) && (out/host/linux-x86/bin/sefcontext_compile -o
out/target/product/rpi3b/obj/ETC/file_contexts.bin_intermediates/file_contexts.bin
out/target/product/rpi3b/obj/ETC/file_contexts.bin_intermediates/file_contexts.concat.tmp)"
```

file_contexts.concat.tmp内容如下：

```
#line 1 "out/target/product/rpi3b/obj/ETC/file_contexts.bin_intermediates/file_contexts.local.tmp"
#line 1 "external/sepolicy/file_contexts"
###########################################
# Root
/                   u:object_r:rootfs:s0

# Data files
/adb_keys           u:object_r:adb_keys_file:s0
/build\.prop        u:object_r:rootfs:s0
/default\.prop      u:object_r:rootfs:s0
/fstab\..*          u:object_r:rootfs:s0
/init\..*           u:object_r:rootfs:s0
/res(/.*)?          u:object_r:rootfs:s0
...
```

可以看出sefcontext_compile将file_contexts.concat.tmp文本文件转换成了二进制文件，这个二进制文件包含了系统文件的SELinux security context。

而make_ext4fs创建system image的时候会去读取这个文件，将文件系统中的文件设置SELinux security context。

- Security – capability

在make_ext4fs创建system image设置SELinux security context的同时，还会去设置文件的capability：

system/extra/ext4_utils/make_ext4fs.c

```
static u32 build_directory_structure(const char *full_path, const char *dir_path, const char *target_out_path,
        u32 dir_inode, fs_config_func_t fs_config_func,
        struct selabel_handle *sehnd, int verbose, time_t fixed_time)
{
        if (fs_config_func != NULL) {
#ifdef ANDROID
            unsigned int mode = 0;
            unsigned int uid = 0;
            unsigned int gid = 0;
            int dir = S_ISDIR(stat.st_mode);
            fs_config_func(dentries[i].path, dir, target_out_path, &uid, &gid, &mode, &capabilities);
            dentries[i].mode = mode;
            dentries[i].uid = uid;
            dentries[i].gid = gid;
            dentries[i].capabilities = capabilities;
#else
            error("can't set android permissions - built without android support");
#endif
        }
        // ...
        /*
         * It's important to call inode_set_selinux() before
         * inode_set_capabilities(). Extended attributes need to
         * be stored sorted order, and we guarantee this by making
         * the calls in the proper order.
         * Please see xattr_assert_sane() in contents.c
         */
        ret = inode_set_selinux(entry_inode, dentries[i].secon);
        if (ret)
            error("failed to set SELinux context on %s\n", dentries[i].path);
        ret = inode_set_capabilities(entry_inode, dentries[i].capabilities);
        if (ret)
            error("failed to set capability on %s\n", dentries[i].path);
    // ...
}
```

system/extras/ext4_utils/contents.c：

```
int inode_set_capabilities(u32 inode_num, uint64_t capabilities) {
    if (capabilities == 0)
        return 0;

    struct vfs_cap_data cap_data;
    memset(&cap_data, 0, sizeof(cap_data));

    cap_data.magic_etc = VFS_CAP_REVISION | VFS_CAP_FLAGS_EFFECTIVE;
    cap_data.data[0].permitted = (uint32_t) (capabilities & 0xffffffff);
    cap_data.data[0].inheritable = 0;
    cap_data.data[1].permitted = (uint32_t) (capabilities >> 32);
    cap_data.data[1].inheritable = 0;

    return xattr_add(inode_num, EXT4_XATTR_INDEX_SECURITY,
        XATTR_CAPS_SUFFIX, &cap_data, sizeof(cap_data));
}
```

而文件的capability属性是从下面这两个文件来的：

a. device/hzak/rpi3b/fs_config/android_filesystem_config.h

```
static const struct fs_path_config android_device_files[] = {
    { 00755, AID_SYSTEM,  AID_SHELL, CAP_MASK_LONG(CAP_NET_BIND_SERVICE) | CAP_MASK_LONG(CAP_NET_ADMIN) |
                                     CAP_MASK_LONG(CAP_NET_RAW),          "system/bin/dnsmasq" },

    { 00700, AID_SYSTEM,  AID_SHELL, CAP_MASK_LONG(CAP_BLOCK_SUSPEND),    "system/bin/nativepowerman" },
    { 00700, AID_SYSTEM,  AID_SHELL, CAP_MASK_LONG(CAP_SYS_TIME),         "system/bin/tlsdated" },
    { 00700, AID_WEBSERV, AID_SHELL, CAP_MASK_LONG(CAP_NET_BIND_SERVICE), "system/bin/webservd" },
    // ...
    { 00755, AID_WIFI,    AID_SHELL, CAP_MASK_LONG(CAP_NET_ADMIN) |
                                     CAP_MASK_LONG(CAP_NET_RAW),          "system/bin/apmanager" },
    // ...
};
```

b. system/core/libcutils/fs_config.c

```
static const struct fs_path_config android_files[] = {
    { 00440, AID_ROOT,      AID_SHELL,     0, "system/etc/init.goldfish.rc" },
    { 00550, AID_ROOT,      AID_SHELL,     0, "system/etc/init.goldfish.sh" },
    { 00550, AID_ROOT,      AID_SHELL,     0, "system/etc/init.ril" },
    { 00550, AID_DHCP,      AID_SHELL,     0, "system/etc/dhcpcd/dhcpcd-run-hooks" },
    // ...
};
```

而struct fs_path_config包含这几个字段：

```
struct fs_path_config {
    unsigned mode;
    unsigned uid;
    unsigned gid;
    uint64_t capabilities;
    const char *prefix;
};
```

NOTE: 系统在查找文件capability时，采用的是first match, 并且先查找device/hzak/rpi3b/fs_config/android_filesystem_config.h中的配置。

相关工具(external/libcap/progs)：

a. getcap – 查看文件的capability

```
$ adb shell getcap /system/bin/dnsmasq
/system/bin/dnsmasq = cap_net_bind_service,cap_net_admin,cap_net_raw+ep
```

b. getpcap – 查看进程的capability

```
$ adb shell getpcap `pid init`
Capabilities for `1': =
cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_linux_immutable,cap_net_bind_service,cap_net_broa
dcast,cap_net_admin,cap_net_raw,cap_ipc_lock,cap_ipc_owner,cap_sys_module,cap_sys_rawio,cap_sys_chroot,cap_sys_ptrace,cap_sys_pacct,cap_sys_admin,cap_sys_boot,cap_sy
s_nice,cap_sys_resource,cap_sys_time,cap_sys_tty_config,cap_mknod,cap_lease,cap_audit_write,cap_audit_control,cap_setfcap,cap_mac_override,cap_mac_admin,cap_syslog,c
ap_wake_alarm,cap_block_suspend,37+ep
```

c. setcap – 设置文件的capbility

NOTE: external/libcap/progs/setcap.c：

> Under Linux, effective file capabilities must either be empty, or exactly match the union of selected permitted and inheritable bits

附相关的编译脚本（external/libcap/progs/Android.mk)：

```
LOCAL_PATH:= $(call my-dir)

define my-build-caps
include $$(CLEAR_VARS)
LOCAL_MODULE_TAGS := debug
LOCAL_MODULE := $(1)
LOCAL_SRC_FILES := $(2)
LOCAL_SHARED_LIBRARIES := libcap
include $$(BUILD_EXECUTABLE)
endef

$(eval $(call my-build-caps,getcap,getcap.c))
$(eval $(call my-build-caps,getpcap,getpcaps.c))
$(eval $(call my-build-caps,setcap,setcap.c))
```

其他请参考这篇文档：POSIX 文件能力：分配根用户的能力

http://www.ibm.com/developerworks/cn/linux/l-posixcap.html?ca=drs-cn

- adb sync system

adb sync system命令可以很方便地将本地修改的文件同步到设备中。由于在做adb sync system时，adbd系统服务只是去重新设置了SELinux security context, 对于文件的capability并没有去重新设置，这使得原有文件的capability信息丢失，程序运行时会出现访问某些资源会没有权限。如你不小心修改了apmanager的代码，而又去做了adb sync system，那么Brillo系统中的apmanager就失去了CAP_NET_ADMIN和CAP_NET_RAW的权限，apmanager就无法去设置wlan0的IP地址。

在system/core/adb/file_sync_service.cpp中做file sync只是更新了SELinux security context：

```
static bool handle_send_file(int s, const char* path, uid_t uid,
                             gid_t gid, mode_t mode, std::vector<char>& buffer, bool do_unlink) {
    syncmsg msg;
    unsigned int timestamp = 0;

    __android_log_security_bswrite(SEC_TAG_ADB_SEND_FILE, path);
    int fd = adb_open_mode(path, O_WRONLY | O_CREAT | O_EXCL | O_CLOEXEC, mode);
    if (fd < 0 && errno == ENOENT) {
        if (!secure_mkdirs(adb_dirname(path))) {
            SendSyncFailErrno(s, "secure_mkdirs failed");
            goto fail;
        }
        fd = adb_open_mode(path, O_WRONLY | O_CREAT | O_EXCL | O_CLOEXEC, mode);
    }
    if (fd < 0 && errno == EEXIST) {
        fd = adb_open_mode(path, O_WRONLY | O_CLOEXEC, mode);
    }
    if (fd < 0) {
        SendSyncFailErrno(s, "couldn't create file");
        goto fail;
    } else {
        if (fchown(fd, uid, gid) == -1) {
            SendSyncFailErrno(s, "fchown failed");
            goto fail;
        }

        // Not all filesystems support setting SELinux labels. http://b/23530370.
        selinux_android_restorecon(path, 0);

        // fchown clears the setuid bit - restore it if present.
        // Ignore the result of calling fchmod. It's not supported
        // by all filesystems. b/12441485
        fchmod(fd, mode);
    }
    // ...
}
```

NOTE：由于RPi 2B使用的Linux kernel为4.1的版本，而DragonBoard 410c使用的kernel版本为3.10，也许由于版本的改变，kernel在做security – capability check的行为也改变：

a. DragonBoard 410c使用firewalld service punch TCP hole之所以会成功，是因为修改了qcom-msm-3.10/security/commoncap.c文件，包含如下patch(在进行capability check的时候，通过检查当前进程的用户组）：

```
commit 775d748ba7a4c68d02c1410c39bce8750394344c
Author: Chia-chi Yeh <chiachi@android.com>
Date:   Fri Jun 19 07:15:05 2009 +0800

    security: Add AID_NET_RAW and AID_NET_ADMIN capability check in cap_capable().

    Signed-off-by: Chia-chi Yeh <chiachi@android.com>

diff --git a/security/commoncap.c b/security/commoncap.c
index c44b6fe..3e81aa9 100644
--- a/security/commoncap.c
+++ b/security/commoncap.c
@@ -31,6 +31,10 @@
 #include <linux/binfmts.h>
 #include <linux/personality.h>

+#ifdef CONFIG_ANDROID_PARANOID_NETWORK
+#include <linux/android_aid.h>
+#endif
+
 /*
  * If a non-root user executes a setuid-root binary in
  * !secure(SECURE_NOROOT) mode, then we raise capabilities.
@@ -78,6 +82,11 @@ int cap_capable(const struct cred *cred, struct user_namespace *targ_ns,
 {
        struct user_namespace *ns = targ_ns;

+       if (cap == CAP_NET_RAW && in_egroup_p(AID_NET_RAW))
+               return 0;
+       if (cap == CAP_NET_ADMIN && in_egroup_p(AID_NET_ADMIN))
+               return 0;
+
        /* See if cred has the capability in the target user namespace
         * by examining the target user namespace and all of the target
         * user namespace's parents.
```

b. 同样的brillo-m8-dev分支中，brilloemulator_x86使用的kernel为android-3.18, 也包含上述patch，所以也能正常punch TCP hole。

[**2016-11-03 22:48:43**] 看来google也发现的这个问题，在master分支上已经有相关的patch来fix这个问题：

```
commit 32c60b4ced8ca0d0eaa3e32f04436a7246ad18f8
Author: Elliott Hughes <enh@google.com>
Date:   Tue Jun 7 17:18:25 2016 -0700

    Set file capabilities on adb sync/push.

    Bug: http://b/29180022
    Change-Id: Ia21ebf0972af41b0a3becc1189ed836fd74ae5c8
---
 adb/file_sync_service.cpp | 53 ++++++++++++++++++++++++++++++++++++++---------------
 1 file changed, 38 insertions(+), 15 deletions(-)
```

- 关于external/minijail

由于kernel 4.3+又定义了一个新的secure bit, 而brillo-m10-dev-rpi3b使用的kernel版本为4.1， 所以在使用minijail的时候，可能会出现kernel crash的情况：

```
04-10 05:33:20.103   206   206 E chkcap  : libminijail: prctl(PR_SET_SECUREBITS): Operation not permitted
--------- beginning of crash
04-10 05:33:20.103   206   206 F libc    : Fatal signal 6 (SIGABRT), code -6 in tid 206 (chkcap)
04-10 05:33:20.115   207   207 F DEBUG   : *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
04-10 05:33:20.116   207   207 F DEBUG   : Build fingerprint: 'Brillo/rpi3b/rpi3b:6.0.1/MASTER/hzak04082209:eng/test-keys'
04-10 05:33:20.116   207   207 F DEBUG   : Revision: '0'
04-10 05:33:20.116   207   207 F DEBUG   : ABI: 'arm'
04-10 05:33:20.116   207   207 F DEBUG   : pid: 206, tid: 206, name: try_to_jail  >>> try_to_jail <<<
04-10 05:33:20.117   207   207 F DEBUG   : signal 6 (SIGABRT), code -6 (SI_TKILL), fault addr --------
04-10 05:33:20.117   207   207 F DEBUG   :     r0 00000000  r1 000000ce  r2 00000006  r3 00000008
04-10 05:33:20.117   207   207 F DEBUG   :     r4 76f7f56c  r5 00000006  r6 76f7f514  r7 0000010c
04-10 05:33:20.117   207   207 F DEBUG   :     r8 7e988d70  r9 00000000  sl 00000000  fp 7e988ddc
04-10 05:33:20.117   207   207 F DEBUG   :     ip 00000001  sp 7e987c70  lr 76bb028f  pc 76bb2adc  cpsr 20000010
```

附代码：

```
// ...
int main(int argc, char *argv[]) {
    brillo::Minijail* m = brillo::Minijail::GetInstance();
    minijail* jail = m->New();

    m->DropRoot(jail, "system", "system");

    m->UseCapabilities(jail, kIpTablesCapMask);
    m->Enter(jail);

    return 0;
}
```

如果出现这个情况，可以将这个patch合入brillo-dev-m10分支中：

```
commit f783b5273d66d19a78705276a38ae68ef2e3e165
Author: Jorge Lucangeli Obes <jorgelo@google.com>
Date:   Mon Mar 14 14:34:10 2016 -0700

    Fix use of SECURE_ALL_BITS/SECURE_ALL_LOCKS.

    Kernels 4.3+ define a new securebit (SECURE_NO_CAP_AMBIENT_RAISE),
    so using the SECURE_ALL_BITS and SECURE_ALL_LOCKS masks from newer
    kernel headers will return EPERM on older kernels. Detect this, and
    retry with the right mask for older (2.6.26-4.2) kernels.

    Also add a compile-time assert to make sure we identify these changes
    sooner going forward.

    Bug: 27632733

    Change-Id: I6cf9c56fec222347575bd0d1147287aac6572e67
```

- 相关的参考文档：

1. http://www.chromium.org/chromium-os/chromiumos-design-docs/system-hardening
2. https://lwn.net/Articles/211883/ (File-based capabilities)
3. https://lwn.net/Articles/632520/ (Inheriting capabilities)
4. http://www.ibm.com/developerworks/cn/linux/l-posixcap.html?ca=drs-cn

相关文档：

1. **Android: 超级好用的adb forward命令**
2. **Brillo: Android客户端开发 - 查找服务，调用API**
3. **Android/Brillo selinux domain/context**
4. **Brillo: 使用iw命令设置无线网卡工作模式**
5. **Brillo开发: 关于google breakpad – macrodump**

评论

Be the First to Comment!

提醒我 当有新的评论时 邮箱 ›

Start the discussion