Subject: [patch 07/10] unprivileged mounts: add sysctl tunable for "safe" property
Posted by Miklos Szeredi on Wed, 16 Jan 2008 12:31:54 GMT
View Forum Message <> Reply to Message

From: Miklos Szeredi <mszeredi@suse.cz>

Add the following:

  /proc/sys/fs/types/${FS_TYPE}/usermount_safe

Signed-off-by: Miklos Szeredi <mszeredi@suse.cz>

Index: linux/fs/filesystems.c
===
--- linux.orig/fs/filesystems.c 2008-01-16 13:24:52.000000000 +0100
+++ linux/fs/filesystems.c 2008-01-16 13:25:09.000000000 +0100
@@ -12,6 +12,7 @@
 #include <linux/kmod.h>
 #include <linux/init.h>
 #include <linux/module.h>
+#include <linux/sysctl.h>
 #include <asm/uaccess.h>

 /*
@@ -51,6 +52,57 @@ static struct file_system_type **find_fi
  return p;
 }

+#define MAX_FILESYSTEM_VARS 1
+
+struct filesystem_sysctl_table {
+ struct ctl_table_header *header;
+ struct ctl_table table[MAX_FILESYSTEM_VARS + 1];
+};
+
+/*
+ * Create /sys/fs/types/${FSNAME} directory with per fs-type tunables.
+ */
+static int filesystem_sysctl_register(struct file_system_type *fs)
+{
+ struct filesystem_sysctl_table *t;
+ struct ctl_path path[] = {
+ { .procname = "fs", .ctl_name = CTL_FS },
+ { .procname = "types", .ctl_name = CTL_UNNUMBERED },
+ { .procname = fs->name, .ctl_name = CTL_UNNUMBERED },
+ { }
+ };

```
+
+ t = kzalloc(sizeof(*t), GFP_KERNEL);
+ if (!t)
+  return -ENOMEM;
+
+
+ t->table[0].ctl_name = CTL_UNNUMBERED;
+ t->table[0].procname = "usermount_safe";
+ t->table[0].maxlen = sizeof(int);
+ t->table[0].data = &fs->fs_safe;
+ t->table[0].mode = 0644;
+ t->table[0].proc_handler = &proc_dointvec;
+
+ t->header = register_sysctl_paths(path, t->table);
+ if (!t->header) {
+  kfree(t);
+  return -ENOMEM;
+ }
+
+ fs->sysctl_table = t;
+
+ return 0;
+}
+
+static void filesystem_sysctl_unregister(struct file_system_type *fs)
+{
+ struct filesystem_sysctl_table *t = fs->sysctl_table;
+
+ unregister_sysctl_table(t->header);
+ kfree(t);
+}
+
 /**
  * register_filesystem - register a new filesystem
  * @fs: the file system structure
@@ -80,6 +132,13 @@ int register_filesystem(struct file_syst
  else
   *p = fs;
  write_unlock(&file_systems_lock);
+
+ if (res == 0) {
+  res = filesystem_sysctl_register(fs);
+  if (res != 0)
+   unregister_filesystem(fs);
+ }
+
  return res;
 }
```

```
@@ -108,6 +167,7 @@ int unregister_filesystem(struct file_sy
    *tmp = fs->next;
    fs->next = NULL;
    write_unlock(&file_systems_lock);
+   filesystem_sysctl_unregister(fs);
    return 0;
   }
   tmp = &(*tmp)->next;
Index: linux/include/linux/fs.h
===================================================================
--- linux.orig/include/linux/fs.h 2008-01-16 13:25:09.000000000 +0100
+++ linux/include/linux/fs.h 2008-01-16 13:25:09.000000000 +0100
@@ -1437,6 +1437,7 @@ struct file_system_type {
  struct module *owner;
  struct file_system_type * next;
  struct list_head fs_supers;
+ struct filesystem_sysctl_table *sysctl_table;

  struct lock_class_key s_lock_key;
  struct lock_class_key s_umount_key;
Index: linux/Documentation/filesystems/proc.txt
===================================================================
--- linux.orig/Documentation/filesystems/proc.txt 2008-01-16 13:25:07.000000000 +0100
+++ linux/Documentation/filesystems/proc.txt 2008-01-16 13:25:09.000000000 +0100
@@ -43,6 +43,7 @@ Table of Contents
   2.13 /proc/<pid>/oom_score - Display current oom-killer score
   2.14 /proc/<pid>/io - Display the IO accounting fields
   2.15 /proc/<pid>/coredump_filter - Core dump filtering settings
+  2.16 /proc/sys/fs/types - File system type specific parameters

 ------------------------------------------------------------------------------
 Preface
@@ -2283,4 +2284,21 @@ For example:
   $ echo 0x7 > /proc/self/coredump_filter
   $ ./some_program

+2.16 /proc/sys/fs/types/ - File system type specific parameters
+-----------------------------------------------------------
+
+There's a separate directory /proc/sys/fs/types/<type>/ for each
+filesystem type, containing the following files:
+
+usermount_safe
+--------------
+
+Setting this to non-zero will allow filesystems of this type to be
+mounted by unprivileged users (note, that there are other
```

+prerequisites as well).
+
+Care should be taken when enabling this, since most
+filesystems haven't been designed with unprivileged mounting
+in mind.
+

--

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers