Subject: [patch 0/2] net namespace : L3 security patches
Posted by Daniel Lezcano on Fri, 26 Jan 2007 09:52:48 GMT
View Forum Message <> Reply to Message

The following patches make the L3 namespace more secure.
The IP addresses should be set into the L2 parent namespace and assigned to a L3 child.
The L3 namespaces can not do IP configuration anymore.

--

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: [patch 1/2] net namespace : assign IP address to specified nsproxy child
Posted by Daniel Lezcano on Fri, 26 Jan 2007 09:52:49 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

All IP configuration is done into the L2 namespace. The L3 namespace gains visibility to an IP address when this one is assigned from the L2 parent's namespace to the L3 child.
In order to be able to find a L3 child, the namespace identifier should be used to retrieve the namespace.

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>

---
 net/core/net_namespace.c | 24 ++++++++++++++++++------
 1 file changed, 18 insertions(+), 6 deletions(-)

Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
===
--- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
+++ 2.6.20-rc4-mm1/net/core/net_namespace.c
@@ -15,6 +15,7 @@
 #include <linux/inetdevice.h>
 #include <linux/skbuff.h>
 #include <linux/ip.h>
+#include <linux/capability.h>

 #include <net/ip_fib.h>
 #include <net/sock.h>
@@ -57,8 +58,8 @@

  kref_init(&ns->kref);
  if ((push_net_ns(ns)) != old_ns)

```
-
   BUG();
+
  if (level ==  NET_NS_LEVEL2) {
   ns->dev_base_p = NULL;
   ns->dev_tail_p = &ns->dev_base_p;
@@ -200,25 +201,36 @@
 {
  struct ifreq ifr;
  struct sockaddr_in *sin = (struct sockaddr_in *)&ifr.ifr_addr;
- struct net_namespace *net_ns = current_net_ns;
+ struct net_namespace *net_ns;
+ struct nsproxy *nsproxy;
  struct net_device *dev;
  struct in_device *in_dev;
  struct in_ifaddr **ifap = NULL;
  struct in_ifaddr *ifa = NULL;
  char *colon;
- int err;
+ int err, id = 0;

  if (!capable(CAP_NET_ADMIN))
   return -EPERM;

- if (net_ns->level != NET_NS_LEVEL3)
-  return -EPERM;
-
  if (copy_from_user(&ifr, arg, sizeof(struct ifreq)))
   return -EFAULT;

  ifr.ifr_name[IFNAMSIZ - 1] = 0;
+ id = sin->sin_port;

+ nsproxy = find_nsproxy_by_id(id);
+ if (!nsproxy)
+  return -ESRCH;
+
+ net_ns = nsproxy->net_ns;
+ put_nsproxy(nsproxy);
+
+ if (net_ns->parent != current_net_ns)
+  return -EPERM;
+
+ if (net_ns->level != NET_NS_LEVEL3)
+  return -EINVAL;

  colon = strchr(ifr.ifr_name, ':');
  if (colon)
```

--

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: [patch 2/2] net namespace : lose CAP_NET_ADMIN when being L3 namespace
Posted by Daniel Lezcano on Fri, 26 Jan 2007 09:52:50 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

Lose the CAP_NET_ADMIN privilege when being a L3 namespace.
That will forbid any IP configuration into the L3 namespace.

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>

---
 include/linux/net_namespace.h | 6 ++++++
 net/core/net_namespace.c | 20 ++++++++++++++++++++
 security/commoncap.c | 5 +++++
 security/dummy.c | 6 ++++++
 4 files changed, 37 insertions(+)

Index: 2.6.20-rc4-mm1/security/dummy.c
===
--- 2.6.20-rc4-mm1.orig/security/dummy.c
+++ 2.6.20-rc4-mm1/security/dummy.c
@@ -28,6 +28,8 @@
 #include <linux/hugetlb.h>
 #include <linux/ptrace.h>
 #include <linux/file.h>
+#include <linux/security.h>
+#include <linux/net_namespace.h>

 static int dummy_ptrace (struct task_struct *parent, struct task_struct *child)
 {
@@ -74,6 +76,8 @@

 static int dummy_capable (struct task_struct *tsk, int cap)
 {
+ if (net_ns_cap_net_admin(tsk, cap))
+ return -EPERM;
  if (cap_raised (tsk->cap_effective, cap))
   return 0;

```
  return -EPERM;
@@ -684,6 +688,8 @@

 static int dummy_netlink_recv (struct sk_buff *skb, int cap)
 {
+ if (net_ns_cap_net_admin(current, cap))
+  return -EPERM;
  if (!cap_raised (NETLINK_CB (skb).eff_cap, cap))
   return -EPERM;
  return 0;
Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
+++ 2.6.20-rc4-mm1/net/core/net_namespace.c
@@ -497,4 +497,24 @@
  else
   return sk->sk_net_ns == net_ns;
 }
+
+/*
+ * This function checks if the specified task is running into a L3 network
+ * namespace and avoid this one to do any IP configuration stuff inside it.
+ * @tsk : the specified task to be checked
+ * @cap : the specified capability
+ * Returns : -EPERM if the CAP_NET_ADMIN is set and task is running as L3, 0
+ * otherwise
+ */
+int net_ns_cap_net_admin(struct task_struct *tsk, int cap)
+{
+ struct net_namespace *net_ns;
+ net_ns = tsk->nsproxy->net_ns;
+
+ if (net_ns->level == NET_NS_LEVEL3 &&
+     cap == CAP_NET_ADMIN)
+  return -EPERM;
+
+ return 0;
+}
 #endif /* CONFIG_NET_NS */
Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
+++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
@@ -113,6 +113,8 @@
 extern int net_ns_sock_is_visible(const struct sock *sk,
       const struct net_namespace *net_ns);

+extern int net_ns_cap_net_admin(struct task_struct *tsk, int cap);
```

```
+
 #define SELECT_SRC_ADDR net_ns_select_source_address

 #else /* CONFIG_NET_NS */
@@ -203,6 +205,10 @@
   return 1;
 }

+static inline int net_ns_cap_net_admin(struct task_struct *tsk, int cap)
+{
+ return 0;
+}
 #define SELECT_SRC_ADDR inet_select_addr

 #endif /* !CONFIG_NET_NS */
Index: 2.6.20-rc4-mm1/security/commoncap.c
===================================================================
--- 2.6.20-rc4-mm1.orig/security/commoncap.c
+++ 2.6.20-rc4-mm1/security/commoncap.c
@@ -24,6 +24,7 @@
 #include <linux/xattr.h>
 #include <linux/hugetlb.h>
 #include <linux/mount.h>
+#include <linux/net_namespace.h>

 int cap_netlink_send(struct sock *sk, struct sk_buff *skb)
 {
@@ -35,6 +36,8 @@

 int cap_netlink_recv(struct sk_buff *skb, int cap)
 {
+ if (net_ns_cap_net_admin(current, cap))
+  return -EPERM;
  if (!cap_raised(NETLINK_CB(skb).eff_cap, cap))
   return -EPERM;
  return 0;
@@ -44,6 +47,8 @@

 int cap_capable (struct task_struct *tsk, int cap)
 {
+ if (net_ns_cap_net_admin(tsk, cap))
+  return -EPERM;
  /* Derived from include/linux/sched.h:capable. */
  if (cap_raised(tsk->cap_effective, cap))
   return 0;

--
_____________________________________________________
```

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers