Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Gestione automatica di namespace di rete privati in ambienti linux

Relatore:
Chiar.mo Prof.
Renzo Davoli

Presentata da:
Eduard Caizer

Sessione 2
Anno Accademico 2015-2016

Introduzione

Servizi come AWS (Amazon Web Services) sono basati su cloud e permettono agli utenti l'affitto di potenti e robuste infrastrutture ad ora o computazione. Si parla di software distribuito come servizio Saas (Software as a Service) e NaaS (Network as a Service) per quelli che costruiscono la propria infrastruttura sopra a cloud data center globali e utilizzano SDN (Software defined networking) e tecnologie di virtualizzazione.
Questo permette loro di avere sia una grande elasticità di configurazione che una buona robustezza del servizio.
Ognuno di questi servizi deve assicurarsi di isolare la rete destinata ai clienti e devono essere in grado di permettere ad ognuno di essi la gestione della propria sottorete senza mettere a repentaglio la sicurezza di quella degli altri.

La soluzione proposta da IBM è stata un framework unico per il networking in cloud, CloudNaaS. Nel processo di realizzazione di una richiesta di un nuovo customer viene creata una macchina virtuale (MV) apposita per le necessita del cliente, tuttavia questa MV deve essere posizionata in modo appropriato affinchè servizi simili e quindi macchine virtuali simili vengano gestite in modo efficiente dalla piattaforma cloud. É possibile mappare le richieste "logiche" dei clienti sulle risorse fisiche del cloud attraverso complessi algoritmi di ottimizzazione.

Ogni istanza di MV deve quindi essere creata, configurata adeguatamente, posizionata su un determinato server per sfruttare al meglio le risorse, deve

essere "connessa" alla rete, creare una sottorete per la specifica istanza e assegnarle un IP all'interno al cloud.

> "Our network aware VM placement strategy, in which VMs belonging to the same application deployment are placed topologically near each other, is better able to accommodate network service requests, particularly for applications with many VMs. Network-aware placement reduces the pathlength between VMs belonging to such applications by a factor of 3, and can support nearly 10% more applications than non network-aware algorithms."

Quindi al giorno d'oggi ci sono framework unificati come CloudNaas per il network isolation, custom adressing ecc.
Questi framework come problema hanno il fatto di dover ottimizzare le centinaia di migliaia di macchine virtuali che sono costretti a mantenere, una o più per cliente. Inoltre quando un host si danneggia o ha un errore bisogna migrare la MV del cliente su un altro host e ristabilire la connettività, ma se ci sono centinaia di migliaia di MV il processo potrebbe diventare estremamente costoso.
L'isolamento dei vari utenti viene quindi implementato creando una istanza di macchina virtuale per ogni cliente, con libpam-net, invece, si cerca di isolare i clienti tramite linux namespace di rete, permettendo ai vari clienti di avere la propria sottorete indipendente, privata.
Si vuole evitare la necessità di istanziare centinaia di migliaia di macchine virtuali, quando potrebbe bastare creare una MV per tipo di servizio offerto, e far accedere ogni cliente che necessita di quel servizio semplicemente collegandolo al proprio namespace di rete, isolandolo in tal modo dagli altri clienti.

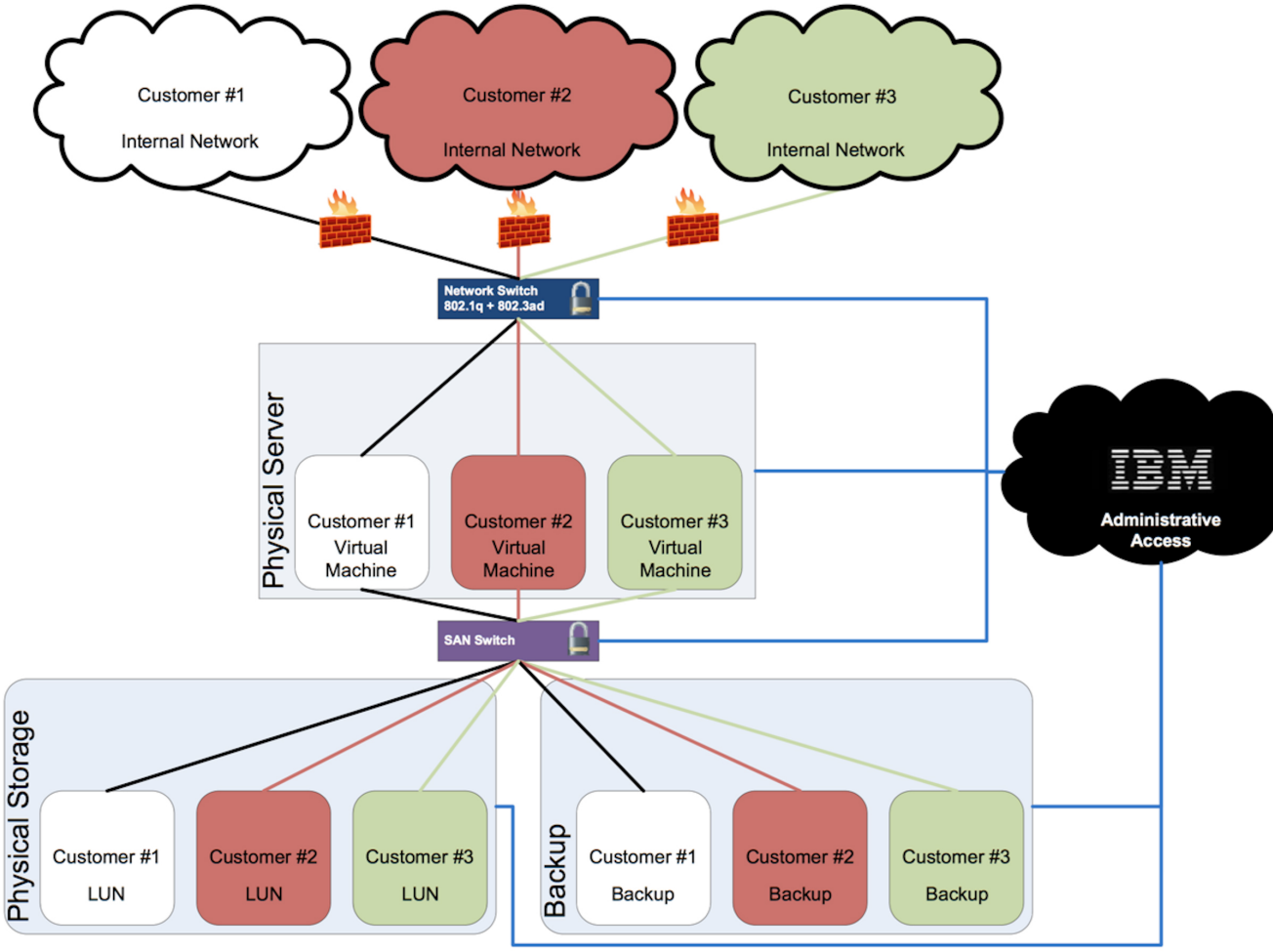

Figura 1: CloudNaaS

Indice

Elenco delle figure

Capitolo 1

Contesto

Creare una macchina virtuale per ogni utente che accede al sistema permette di avere una astrazione della rete indipendente da come viene costruita la rete fisica del sistema.
Un utente che si trova in un ambiente del genere non ha modo di sapere come è la rete sottostante, può utilizzare solamente le interfacce di rete che ha la macchina virtuale e di conseguenza non ha modo di avere accesso a risorse al di fuori di ciò che "vede".
Libpam-net si basa sullo stesso concetto cercando di estendere l'idea al networking con l'utilizzo di PAM e Linux network namespace, ma senza avere tutto l'overhead di una macchina virtuale.
Garantire sicurezza, isolamento ma allo stesso tempo anche performance e flessibilità è complesso.

1.1 Struttura con macchine virtuali

Come è possibile osservare in figura 1.1 fornendo una macchina virtuale o un set di macchine virtuali ad un utente A, queste si possono connettere in VLAN tra loro e creare, quindi, una sottorete perfettamente separata da quella di un utente B.

Non c'è modo per l'utente A di vedere i dati scambiati dall'utente B con il sistema e viceversa. Ognuna di queste sottoreti poi potranno essere connesse alla rete fisica del datacenter o cloud.
Naturalmente quando si parla di servizi NaaS, ci si aspettano centinaia di migliaia di utenti. Ora basti immaginare di configurare la rete anche soltanto ad una macchina virtuale per utente.
Diventa alquanto impegnativo riuscire a sostenere tale carico, in termini di saturazione della rete, consumo di risorse hardware e gestione dei fault.

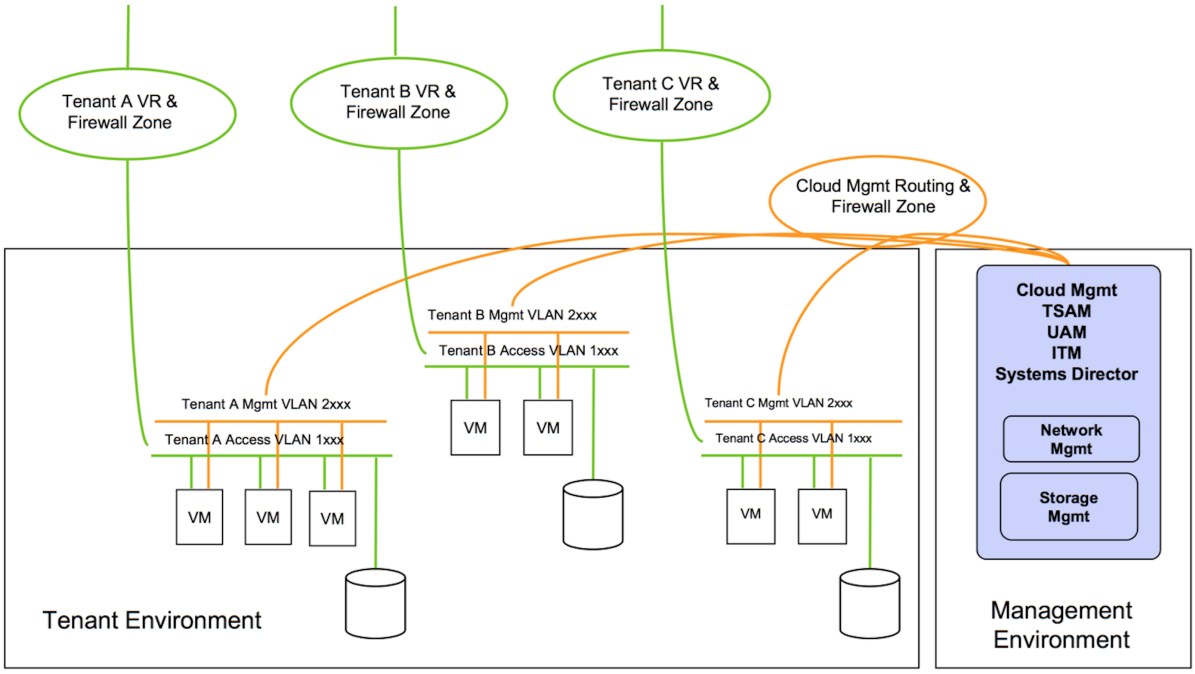

Figura 1.1: isolamento tramite macchine virtuali

1.2 Isolamento senza macchine virtuali

Si vorrebbe ottenere un risultato simile all'isolamento della rete ottenuto tramite l'istanziazione di macchine virtuali, ma senza effettivamente farlo.
In figura 1.2 denotiamo come:

1. "HOST" - una macchina fisica del sistema
2. namespace - un network namespace per un utente X
3. ssh login - un accesso alla macchina tramite ssh
4. TTY login - un accesso alla macchina utilizzando il login del sistema
5. gdm login - un accesso alla macchina utilizzando un client grafico

In figura è rappresentata l'idea di potersi connettere alla macchina nel modo "3", "4", o "5". Il sistema isola la sessione dell'utente X nel network namespace "2", nel quale l'utente vede solamente l'interfaccia esistente in quel namespace.
Qualsiasi altro accesso (anche cosiddetto "nasted", cioè anche all'interno di un'altra sessione attiva) dell'utente X alla macchina verr 'a trattato come il precedente, facendo join al namespace esistente.
Il sistema poi connetterà tale namespace alla rete fisica lasciando però l'utente ignaro di ciò che accade al di fuori del suo ambiente di rete.

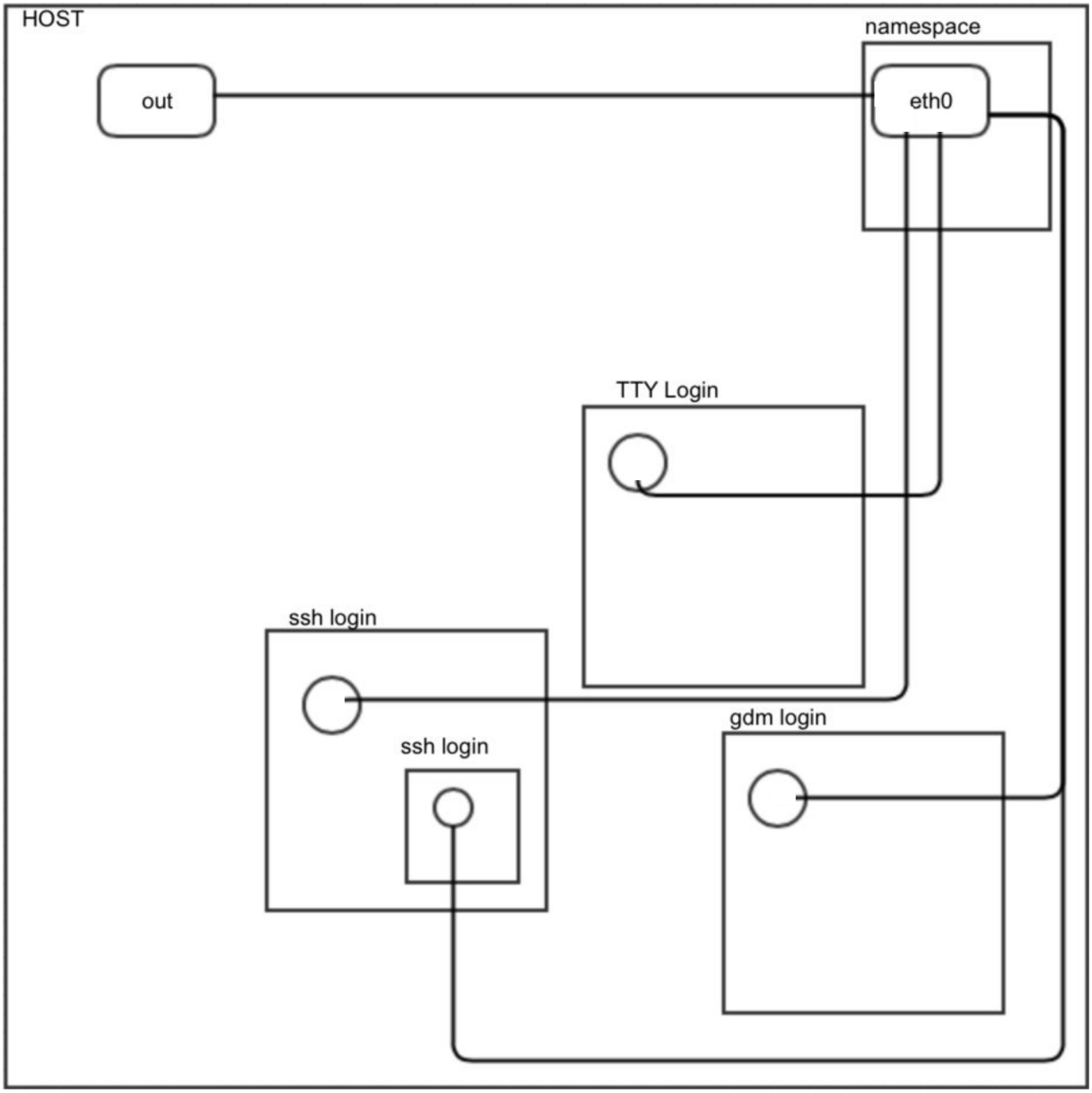

Figura 1.2: isolamento tramite namespace

1.3 PAM

Le applicazioni che necessitavano di autenticazione, prima di PAM, dovevano controllare gli username nel file passwd, confrontarli e quindi procedere con l'autenticazione. Ogni applicazione quindi doveva occuparsi personalmente della fase di autenticazione, autorizzazione, controllo dell'ambiente.
PAM è stato sviluppato da Red Hat Linux, nel 1996, per fornire una API di alto livello che permettesse ai programmi di essere scritti in maniera indipendente dallo schema di autenticazione del sistema operativo.
Grazie all'architettura modulare di PAM è, infatti, possibile specificare in appositi file di configurazione quali moduli debbano essere utilizzati dall'applicazione, che si tratti di autenticazione, inizializzazione, contenimento.
Il sistema si occupa di chiamare varie funzioni del lifecycle di PAM (figura 1.3), le quali vengono chiamate nei vari moduli inclusi nei file di configurazione dei software che ne fanno uso.
In libpam-net viene fatto uso, in particolare, della "pam_sm_open_session", chiamata durante la creazione della sessione utente.

Per approfondimenti su PAM [9, 10, 11].

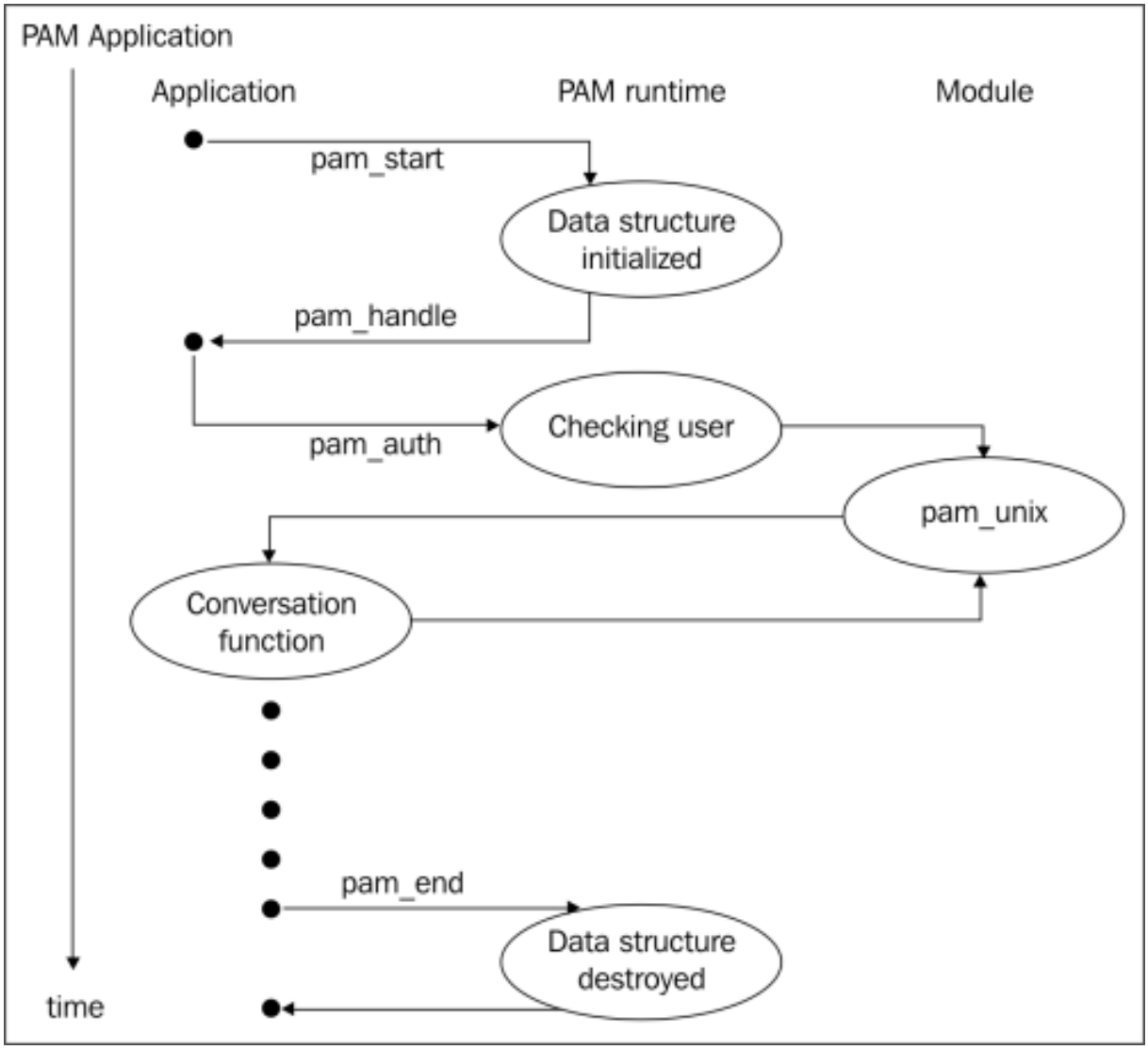

Figura 1.3: PAM lifecycle

1.4 Network namespaces

Un namespace di rete virtualizza la rete che vedono i processi al suo interno, cambia la visione del processo riguardo allo stack di rete e ai device, ai protocolli IPv4 e IPv6, alle routing tables , regole di firewall, /proc/net.
Per default dopo il boot tutte le interfacce fanno parte del init_net namespace e se si ha la capability CAP_SYS_ADMIN si possono creare altri network namespace.
Un network namespace appena creato avrà solo l'indirizzo loopback e nessun altro device, in questo modo si potranno aggiungere device virtuali per controllare il traffico di rete, costruire sottoreti, isolare tutti i processi dalla rete e molto altro.

Per approfondimenti [3, 14, 15]

Capitolo 2

Libpan-net

Come evidenziato precedentemente il problema che ci si pone e' come permettere ad un utente l'accesso ad una macchina (virtuale o non), limitandone o proteggendone l'accesso alla rete.
Con l'utilizzo di servizi come CloudNaaS il problema diviene triviale, infatti ogni utente avendo un'istanza di macchina virtuale non ha modo di avere accesso alla rete (eventualmente anche essa virtuale) sottostante.
L'idea è quindi di evitare la creazione di una macchina virtuale per ogni utente, ma di intervenire direttamente sulla visione della rete che hanno quell'utente o i suoi processi.
Soluzioni già esistenti come OpenVZ o Docker pur evitando parte dell'overhead introdotto da una macchina virtuale richiedono comunque la configurazione di un ambiente per ogni singolo utente, incorrendo in problemi simili a quelli affrontati da CloudNaas.

Inotify like In un primo momento si è pensato di cambiare la visione del networking del processo a runtime: non appena un processo veniva creato e quindi aggiunto in /proc si procedeva ad isolarlo.
Il problema consiste nel fatto che /proc è una interfaccia del kernel, quindi bisogna adottare un sistema di polling e anche solo il tempo necessario ad accorgersi della creazione del record in /proc può dar modo a tale processo

di compiere azioni pericolose.

Listing 2.1: unshareDaemon.c

```
1  static void daemonInit() {
2
3      pid_t pid = fork();
4
5      if (pid < 0) exit(EXIT_FAILURE);
6      // parent have to exit
7      if (pid > 0) exit(EXIT_SUCCESS);
8      //child have to become session leader
9      if (setsid() < 0) exit(EXIT_FAILURE);
10
11     // ------------------ Signal handling ------------------------ //
12     // TODO handle befor second fork
13
14     pid = fork();
15     if (pid < 0) exit(EXIT_FAILURE);
16     if (pid > 0) exit(EXIT_SUCCESS); // parent have to exit
17
18     umask(0); // new file permission
19     chdir("/proc"); // change working dir to /proc
20
21     int x;
22     // close all open file descriptors
23     for (x = sysconf(_SC_OPEN_MAX); x>0; x--) {
24         close(x);
25     }
26
27     openlog("unshareDaemon", LOG_PID, LOG_DAEMON);
28 }
29
```

```
30 // filter only pid directories
31 int toIntFilter (const struct dirent *entry) {
32    char path[256]={0};
33    struct stat st;
34    int pid = atoi(entry->d_name); //TODO atoi not secure
35
36    if ( !pid ) {
37       return 0;
38    }
39    strcat(path,"/proc/");
40    strcat(path, entry->d_name);
41    strcat(path,"/exe");
42
43    lstat (path, &st);
44    if ( st.st_gid == 1000 ) {
45       return 1;
46    }
47    else return 0;
48 }
49
50 int naturalSort (const struct dirent **entry1,
51                  const struct dirent **entry2) {
52    int a = atoi ((*entry1)->d_name);
53    int b = atoi ((*entry2)->d_name);
54    return a>b;
55 }
56
57 int main() {
58    struct dirent **proc_list = NULL;
59    int n=0, i=0;
60
```

```
61      daemonInit();
62      syslog (LOG_NOTICE, "daemon init done");
63
64      while (1) {
65
66          n = scandir ("/proc", &proc_list, toIntFilter, naturalSort);
67          if (n < 0) {
68              syslog (LOG_CRIT, "error scandir");
69              exit (EXIT_FAILURE);
70          }
71          for ( i=0; i<n; i++) {
72              syslog (LOG_NOTICE, "%s", proc_list[i]->d_name);
73              free (proc_list[i]);
74          }
75          n = 0;
76
77          free(proc_list);
78          sleep(5);
79      }
80
81
82      closelog();
83      return EXIT_SUCCESS;
84  }
```

Un processo alla sua creazione eredita la visione dell'ambiente che aveva il padre. Si è pensato quindi di intervenire a monte: bisognava intervenire prima che venissero rilasciati i diritti di root, ma nel momento in cui il primo processo di un determinato utente veniva creato ed eseguito.
Questo avviene nel momento in cui un utente fa accesso alla macchina. Si può quindi intervenire durante il Login, ssh ecc.

Login Un'altra possibilità poteva, quindi, consistere nel modificare direttamente il codice del login, facendo in modo che il processo dell'utente che voleva fare l'accesso erediti un network namespace creato dall'amministratore e quindi di conseguenza ogni altro processo dell'utente creato da quel momento in poi abbia ereditato il suddetto ambiente di rete. Il problema di questa soluzione è stato il rendersi neccessaria la modifica del codice del login di ogni distribuzione linux, non che il dover ricompilare l'esegiubile e quindi anche il dover mantenere la versione modificata nell'eventualità che esso venga aggiornato o modificato dalle varie repository.

Listing 2.2: login.c shadow-4.2

```
1     /* Drop root privileges */
2 #ifndef USE_PAM
3     if (setup_uid_gid (pwd, is_console))
4 #else
5     /* The group privileges were already dropped.
6      * See setup_groups() above.
7      */
8     if (change_uid (pwd))
9 #endif
10    {
11        exit (1);
12    }
13
14    setup_env (pwd);   /* set env vars, cd to the home dir */
```

Ovviamente una soluzione del genere oltre ad essere poco elegante è anche poco attuabile nella realta'. Serviva una soluzione modulare, qualcosa che potesse essere usato da un processo prima dell'accesso alla macchina, qualcosa che fosse indipendente da che tipo di accesso l'utente eseguiva, ssh, grafico, tty ...

È stato quindi valutata la possiblità di utilizzare PAM. Un modulo personalizzato di PAM infatti può essere eseguito in fase di autenticazione, può verificare che l'utente che sta tentando di accedere ad una macchina o un servizio appartenga ad una certa categoria e quindi configurare di conseguenza il suo ambiente.

2.0.1 Idea

Libpam-net entra in gioco al momento in cui deve essere creata una sessione per un utente, durante il suo accesso ad una macchina. Qualsiasi software che si occuperà dell'autorizzazione dell'utente alla macchina chiamerà i vari moduli PAM specificati nel proprio file di configurazione, in questo file verrà anche specificato di usare il modulo qui spiegato.

Listing 2.3: login.c shadow-4.2

```
1  /* Open the PAM session */
2      get_pam_user (&pam_user);
3      retcode = pam_open_session (pamh, hushed (pam_user) ? PAM_SILENT : 0)
4      PAM_FAIL_CHECK;
5
6      /* Grab the user information out of the password file for future usag
7       * First get the username that we are actually using, though.
8       *
9       * From now on, we will discard changes of the user (PAM_USER) by
10      * PAM APIs.
11      */
12     get_pam_user (&pam_user);
13     if (NULL != username) {
14         free (username);
15     }
```

```
16      username = xstrdup (pam_user);
17      failent_user = get_failent_user (username);
18
19      pwd = xgetpwnam (username);
20      if (NULL == pwd) {
21          SYSLOG ((LOG_ERR, "cannot find user %s", failent_user));
22          fprintf (stderr,
23                   _("Cannot find user (%s)\n"),
24                   username);
25          exit (1);
26      }
```

L'idea è stata quella di intervenire sul processo che esegue l'autenticazione, verificare se l'utente appartiene agli utenti a cui si vogliono applicare restrizioni alla rete o comunque se è uno degli utenti a cui si vuole modificare la visione dell'ambiente in cui vive.
Una volta verificata l'autenticazione, se si interviene sul processo in questo momento, qualsiasi fork/clone effettuata successivamente farà in modo che il processo abbia, in questo caso, la visione del networking desiderata. Una volta terminata la configurazione dell'ambiente attorno al processo, se tutto sarà andato per il meglio, il software originale potrà proseguire con l'autorizzazione dell'utente o qualsiasi altra cosa debba fare.
Da questo momento in poi ogni processo dell'utente sarà isolato e confinato nella rete per lui preparata, non potrà avere informazioni sulla rete fisica sottostante.
Eventualmente l'utente avrà la possibilità di creare le proprie sottoreti e interfacce grazie a CADO [16].

2.1 Struttura

libpam-net permette la configurazione della rete di un utente in due modi: si può specificare che un certo utente non deve avere alcun accesso alla rete

aggiungendolo al gruppo newnet (i processi dell'utente potranno comunicare tra loro solo se sono nella stessa sessione, avendo abilitata solo l'interfaccia di localhost), oppure si può specificare che deve avere un ambiente privato aggiungendolo a usernet

Listing 2.4: groups

```
1 e.g. in /etc/group:
2 newnet:x:148:usernonet
3 usernet:x:149:usernet
```

Nella directory di pam /etc/pam.d/ bisogna modificare i file di configurazione login, sshd ecc. aggiungendo

Listing 2.5: config

```
1 session    required   pam_newnet.so
2 session    required   pam_usernet.so
```

Nel caso in cui un utente faccia multipli accessi e quindi esista già un namespace per lui, ad ogni nuovo accesso verrà inserito nel namespace a suo nome che già esiste.

L'amministratore di sistema può creare namespace adhoc per i vari utenti, configurarli secondo necessità e quindi preparare un ambiente a cui ogni sessione dell'utente X si unirà.
Grazie a cado (link referenze), inoltre si può fornire ad un utente la capability CAP_NET_ADMIN con la quale questo sarà in grado di modificare a proprio piacimento la rete in cui si trova, naturalmente rimanendo confinato nel namespace a cui ha fatto join e quindi senza poter arrecare danno agli altri utenti presenti nel sistema.
Potrà ad esempio creare proprie interfacce, assegnarne l'IP, creare anche nuovi namespace all'interno di quello in cui si trova.

2.2 Dettagli implementativi

Newnet Nel caso in cui l'utente faccia parte del gruppo newnet

Listing 2.6: pam_newnet.c

```
1    if ((rv=pam_get_user(pamh, &user, NULL) != PAM_SUCCESS)) {
2        syslog (LOG_ERR, "get user: %s", strerror(errno));
3        goto close_log_and_exit;
4    }
5
6    isnewnet = checkgroup(user, "newnet");
```

bisogna isolare l'utente dal resto della rete, lasciandogli visibile solo una propria interfaccia di loopback.
Bisogna quindi disassociare il processo dal suo contesto di esecuzione.
Come evidenziato in precedenza, parte del contesto di esecuzione di un processo, come ad esempio il mount dei namespace è condiviso in maniera implicita quando viene creato un nuovo processo tramite fork o clone.
La system call unshare() permette di gestire il proprio contesto di esecuzione: nel nostro caso permette di disassociare il processo corrente dal network namespace, col flag CLONE_NEWNET il processo che la chiama viene "spostato" in un nuovo network namespace, che non e' condiviso con i processi in esecuzione fino a quell'istante.
Questo vuol dire che una volta che si uscirà dal modulo pam e verrà fatta una fork per creare un processo utente, quest' ultimo sarà all'interno di un namespace che non avrà nulla di condiviso con i processi in esecuzione sulla macchina, quindi sarà all'interno di una rete completamente isolata dal resto.

Listing 2.7: pam_newnet.c

```
1    if (isnewnet > 0) {
2        if (unshare(CLONE_NEWNET) < 0) {
3            syslog (LOG_ERR, "Failed to create a new netns: %s",
```

```
4                       strerror(errno));
5               goto close_log_and_abort;
6           }
7       }
```

Usernet Se invece l'utente fa parte del gruppo usernet

Listing 2.8: pam_usernet.c

```
1       if ((rv=pam_get_user(pamh, &user, NULL) != PAM_SUCCESS)) {
2           syslog (LOG_ERR, "get user: %s", strerror(errno));
3           goto close_log_and_exit;
4       }
5
6       isusernet = checkgroup(user, "usernet");
```

Bisogna creare un network namespace per l'utente, se questo non esiste già, creando un file con il nome dell'utente in /var/run/netns/

Listing 2.9: pam_usernet.c

```
1   if (isusernet > 0) {
2           int nsfd;
3           size_t ns_pathlen=sizeof(NSDIR)+strlen(user)+1;
4           char ns_path[ns_pathlen];
5
6           if (mkdir(NSDIR, S_IRWXU|S_IRGRP|S_IXGRP|S_IROTH|S_IXOTH)) {
7               if (errno != EEXIST) {
8                       syslog (LOG_ERR, "cannot create netns dir %s: %s",
9                                       NSDIR, strerror(errno));
10                      goto close_log_and_abort;
11              }
12          }
13          if (mount("", NSDIR, "none", MS_SHARED | MS_REC, NULL)) {
```

```
14              if (errno != EINVAL) {
15                  syslog (LOG_ERR, "mount --make-shared %s: %s",NSDIR, strerr
16                  goto close_log_and_abort;
17              }
18              if (mount(NSDIR, NSDIR, "none", MS_BIND, NULL)) {
19                  syslog (LOG_ERR, "mount --bind %s: %s",NSDIR, strerror(err
20                  goto close_log_and_abort;
21              }
22              if (mount("", NSDIR, "none", MS_SHARED | MS_REC, NULL)) {
23                  syslog (LOG_ERR, "mount --make-shared after bind %s: %s",N
24                  goto close_log_and_abort;
25              }
26          }
27  }
```

successivamente bisogna collegare /var/run/netns/usernamespace al network namespace /proc/self/ns/net tramite una bind mount.
Una volta creato il namespace non rimane che riassociare il processo al namespace creato fornendogli l'opportuno file descriptor e successivamente fare una unshare.

Listing 2.10: pam_usernet.c

```
1  if ((nsfd = open(ns_path, O_RDONLY)) < 0) {
2              if (errno == ENOENT) {
3                  if ((nsfd = open(ns_path, O_RDONLY|O_CREAT|O_EXCL, 0)) < 0
4                      syslog (LOG_ERR, "cannot create netns %s: %s",ns_path,
5                      goto close_log_and_abort;
6                  }
7                  close(nsfd);
8                  if (unshare(CLONE_NEWNET) < 0) {
9                      syslog (LOG_ERR, "Failed to create a new netns %s: %s",
10                     goto close_log_and_abort;
```

```
11              }
12              if (mount("/proc/self/ns/net", ns_path, "none", MS_BIND, NUI
13                  syslog (LOG_ERR, "mount /proc/self/ns/net -> %s failed: %
14                  goto close_log_and_abort;
15              }
16          } else {
17              syslog (LOG_ERR, "netns open failed %s",ns_path);
18              goto close_log_and_abort;
19          }
20      } else {
21          if (setns(nsfd, CLONE_NEWNET) != 0) {
22              syslog (LOG_ERR, "cannot join netns %s: %s",ns_path, strerro
23              close(nsfd);
24              goto close_log_and_abort;
25          }
26          close(nsfd);
27          if (unshare(CLONE_NEWNS) < 0) {
28              syslog (LOG_ERR, "unshare failed: %s", strerror(errno));
29              goto close_log_and_abort;
30          }
31      }
```

Conclusioni

Circoscrivere l'accesso alla rete ai processi è essenziale per garantire la protezione dell'ambiente in cui essi ed altri processi vivono. Serve a proteggere errori intenzionali e non, da processi che tentano di eseguire codice malevolo o che tentano di carpire informazione confidenziali dalla rete.
Con libpam-net si riescono ad isolare tutti i processi di un utente sia per proteggere essi che per proteggere da essi.
La genericità di questo progetto ne permette in realtà l'utilizzo in più ambiti, anche al di fuori del contesto dei datacenter o cloud di cui si è parlato. Lo si può usare per esempio nell'IoTh [17] o qualunque altro ambito in cui è necessario isolare e controllare l'accesso alla rete.
In futuro si potrebbe modificare il modo in cui specificare per quali utenti aggiungere o cambiare namespace di rete, creando un apposito file di configurazione al posto di specificare gruppi speciali di utenti.
Sarebbe necessario inoltre pacchettizare libpam-net e renderlo disponibili nelle repository delle principali distribuzioni linux.

Libpam-net vuole essere un tool generico che permetta in maniera semplice ed intuitiva la gestione della rete per un determinato utente, può essere utilizzato in molti contesti differenti, secondo la necessità e in qualche modo la fantasia di chi ne usufruisce.

Appendice A

Codice libpam-net

Listing A.1: pam_newnet.c

```
1  /*
2   * pam_newnet.
3   * Copyright (C) 2016  Renzo Davoli, Eduard Caizer University of
4     Bologna
5   *
6   * pam_newnet module
7   *   create a new network namespace at each login
8   *   (for users belonging to the "newnet" group)
9   *
10  * Cado is free software; you can redistribute it and/or
11  * modify it under the terms of the GNU General Public License
12  * as published by the Free Software Foundation; either version 2
13  * of the License, or (at your option) any later version.
14  *
15  * This program is distributed in the hope that it will be useful,
16  * but WITHOUT ANY WARRANTY; without even the implied warranty of
17  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
    See the
18  * GNU General Public License for more details.
```

```
19  *
20  * You should have received a copy of the GNU General Public License
21  * along with this program; If not, see <http://www.gnu.org/licenses/>.
22  *
23  */
24
25 #define _GNU_SOURCE
26 #include <stdio.h>
27 #include <syslog.h>
28 #include <errno.h>
29 #include <string.h>
30 #include <sched.h>
31 #include <security/pam_appl.h>
32 #include <security/pam_modules.h>
33
34 #include <pam_net_checkgroup.h>
35
36 /**
37  * init_log: log initialization with the given name
38  */
39 void init_log(const char * log_name)
40 {
41     setlogmask (LOG_UPTO (LOG_NOTICE));
42     openlog (log_name, LOG_CONS | LOG_PID | LOG_NDELAY, LOG_LOCAL1);
43 }
44
45 /**
46  * end_log: closes the log previously initialized
47  */
48 void end_log()
49 {
```

```
50      closelog ();
51  }
52
53  /*
54   * PAM entry point for session creation
55   */
56  int pam_sm_open_session(pam_handle_t *pamh, int flags,
57          int argc, const char **argv)
58  {
59      const char *user;
60      int rv;
61      int isnewnet;
62
63      init_log ("pam_newnet");
64      if ((rv=pam_get_user(pamh, &user, NULL) != PAM_SUCCESS)) {
65          syslog (LOG_ERR, "get user: %s", strerror(errno));
66          goto close_log_and_exit;
67      }
68
69      isnewnet = checkgroup(user, "newnet");
70
71      if (isnewnet > 0) {
72          if (unshare(CLONE_NEWNET) < 0) {
73              syslog (LOG_ERR, "Failed to create a new netns:
74                  %s", strerror(errno));
75              goto close_log_and_abort;
76          }
77      } else
78          rv=PAM_IGNORE;
79  close_log_and_exit:
80      end_log();
```

```
81     return rv;
82 close_log_and_abort:
83     rv = PAM_ABORT;
84     end_log();
85     return rv;
86 }
87
88 /*
89  * PAM entry point for session cleanup
90  */
91 int pam_sm_close_session(pam_handle_t *pamh,
92     int flags, int argc, const char **argv)
93 {
94     return(PAM_IGNORE);
95 }
```

Listing A.2: pam_usernet.c

```
1  /*
2   * pam_usernet.
3   * Copyright (C) 2016  Renzo Davoli, Eduard Caizer University of
4     Bologna
5   *
6   * pam_usernet module
7   *     provide each user with their own network
8   *    (for users belonging to the "usernet" group)
9   *
10  * Cado is free software; you can redistribute it and/or
11  * modify it under the terms of the GNU General Public License
12  * as published by the Free Software Foundation; either version 2
13  * of the License, or (at your option) any later version.
14  *
```

```
15  * This program is distributed in the hope that it will be useful,
16  * but WITHOUT ANY WARRANTY; without even the implied warranty of
17  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
    See the
18  * GNU General Public License for more details.
19  *
20  * You should have received a copy of the GNU General Public License
21  * along with this program; If not, see <http://www.gnu.org/licenses/>
22  *
23  */
24
25 #define _GNU_SOURCE
26 #include <stdio.h>
27 #include <syslog.h>
28 #include <errno.h>
29 #include <string.h>
30 #include <unistd.h>
31 #include <fcntl.h>
32 #include <sched.h>
33 #include <sys/stat.h>
34 #include <sys/types.h>
35 #include <sys/mount.h>
36 #include <security/pam_appl.h>
37 #include <security/pam_modules.h>
38
39 #include <pam_net_checkgroup.h>
40
41 #define NSDIR "/var/run/netns/"
42
43 /**
44  * init_log: log initialization with the given name
```

```
45  */
46 void init_log(const char * log_name)
47 {
48     setlogmask (LOG_UPTO (LOG_NOTICE));
49     openlog (log_name, LOG_CONS | LOG_PID | LOG_NDELAY, LOG_LOCAL1);
50 }
51
52 /**
53  * end_log: closes the log previously initialized
54  */
55 void end_log()
56 {
57     closelog ();
58 }
59
60 /*
61  * PAM entry point for session creation
62  */
63 int pam_sm_open_session(pam_handle_t *pamh,
64     int flags, int argc, const char **argv)
65 {
66     const char *user;
67     int rv;
68     int isusernet;
69
70     init_log ("pam_usernet");
71     if ((rv=pam_get_user(pamh, &user, NULL) != PAM_SUCCESS)) {
72         syslog (LOG_ERR, "get user: %s", strerror(errno));
73         goto close_log_and_exit;
74     }
75
```

```
76      isusernet = checkgroup(user, "usernet");
77
78      if (isusernet > 0) {
79          int nsfd;
80          size_t ns_pathlen=sizeof(NSDIR)+strlen(user)+1;
81          char ns_path[ns_pathlen];
82
83          if (mkdir(NSDIR, S_IRWXU|S_IRGRP|S_IXGRP|S_IROTH|S_IXOTH)) {
84              if (errno != EEXIST) {
85                  syslog (LOG_ERR, "cannot create netns dir %s:
86                      %s",NSDIR, strerror(errno));
87                  goto close_log_and_abort;
88              }
89          }
90          if (mount("", NSDIR, "none", MS_SHARED | MS_REC, NULL)) {
91              if (errno != EINVAL) {
92                  syslog (LOG_ERR, "mount --make-shared %s:
93                      %s",NSDIR, strerror(errno));
94                  goto close_log_and_abort;
95              }
96              if (mount(NSDIR, NSDIR, "none", MS_BIND, NULL)) {
97                  syslog (LOG_ERR, "mount --bind %s:
98                      %s",NSDIR, strerror(errno));
99                  goto close_log_and_abort;
100             }
101             if (mount("", NSDIR, "none", MS_SHARED | MS_REC, NULL)) {
102                 syslog (LOG_ERR, "mount --make-shared after bind %s:
103                     %s",NSDIR, strerror(errno));
104                 goto close_log_and_abort;
105             }
106         }
```

```
107
108         snprintf(ns_path,ns_pathlen,NSDIR "%s",user);
109         if ((nsfd = open(ns_path, O_RDONLY)) < 0) {
110             if (errno == ENOENT) {
111                 if ((nsfd = open(ns_path, O_RDONLY|O_CREAT|O_EXCL, 0))
112                         < 0) {
113                     syslog (LOG_ERR, "cannot create netns %s:
114                         %s",ns_path, strerror(errno));
115                     goto close_log_and_abort;
116                 }
117                 close(nsfd);
118                 if (unshare(CLONE_NEWNET) < 0) {
119                     syslog (LOG_ERR, "Failed to create a new netns %s:
120                         %s",ns_path, strerror(errno));
121                     goto close_log_and_abort;
122                 }
123                 if (mount("/proc/self/ns/net", ns_path, "none",
124                     MS_BIND, NULL)
125                         < 0) {
126                     syslog (LOG_ERR, "mount /proc/self/ns/net -> %s failed:
127                         %s",ns_path, strerror(errno));
128                     goto close_log_and_abort;
129                 }
130             } else {
131                 syslog (LOG_ERR, "netns open failed %s",ns_path);
132                 goto close_log_and_abort;
133             }
134         } else {
135             if (setns(nsfd, CLONE_NEWNET) != 0) {
136                 syslog (LOG_ERR, "cannot join netns %s:
137                     %s",ns_path, strerror(errno));
```

```
138                 close(nsfd);
139                 goto close_log_and_abort;
140             }
141             close(nsfd);
142             if (unshare(CLONE_NEWNS) < 0) {
143                 syslog (LOG_ERR, "unshare failed: %s", strerror(errno));
144                 goto close_log_and_abort;
145             }
146         }
147     } else
148         rv=PAM_IGNORE;
149 close_log_and_exit:
150     end_log();
151     return rv;
152 close_log_and_abort:
153     rv = PAM_ABORT;
154     end_log();
155     return rv;
156 }
157
158 /*
159  * PAM entry point for session cleanup
160  */
161 int pam_sm_close_session(pam_handle_t *pamh, int flags, int argc, cons
162 {
163     return(PAM_IGNORE);
164 }
```

Listing A.3: pam_net_checkgroup.c

```
1 /*
2  * pam_net_common.
```

```
3   * Copyright (C) 2016  Renzo Davoli, Eduard Caizer University of
4     Bologna
5   *
6   * pam_net common code.
7   *
8   * Cado is free software; you can redistribute it and/or
9   * modify it under the terms of the GNU General Public License
10  * as published by the Free Software Foundation; either version 2
11  * of the License, or (at your option) any later version.
12  *
13  * This program is distributed in the hope that it will be useful,
14  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
    See the
16  * GNU General Public License for more details.
17  *
18  * You should have received a copy of the GNU General Public License
19  * along with this program; If not, see <http://www.gnu.org/licenses/>.
20  *
21  */
22
23 #define _GNU_SOURCE
24 #include <stdio.h>
25 #include <string.h>
26 #include <grp.h>
27 #include <pwd.h>
28
29 /* check if "user" belongs to "group" */
30 int checkgroup(const char *user, const char *group) {
31     struct passwd *pw=getpwnam(user);
32     int ngroups=0;
```

```
33      if (pw == NULL) return -1;
34      if (getgrouplist(user, pw->pw_gid, NULL, &ngroups) < 0) {
35          gid_t gids[ngroups];
36          if (getgrouplist(user, pw->pw_gid, gids, &ngroups) == ngroups) {
37              struct group *grp;
38              int i;
39              while ((grp=getgrent()) != NULL) {
40                  for (i=0; i<ngroups; i++) {
41                      if (grp->gr_gid == gids[i] &&
42                          strcmp(grp->gr_name,group) == 0) {
43                          endgrent();
44                          return 1;
45                      }
46                  }
47              }
48              endgrent();
49              return 0;
50          }
51      }
52      return -1;
53  }
```

Bibliografia

[1] https://www.ibm.com/blogs/cloud-computing/2014/02/how-does-cloud-computing-work/

[2] https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/

[3] Benson, Theophilus, et al. "CloudNaaS: a cloud networking platform for enterprise applications." Proceedings of the 2nd ACM Symposium on Cloud Computing. ACM, 2011.

[4] Anshu Kak fCLOUD Computing - IBM Cloud Computing Reference Architecture Distinguished Engineer CTO SWG Tech Sales Cloud Computing

[5] https://www.ibm.com/blogs/cloud-computing/2013/05/ibm-cloud-computing-reference-architecture-v3-stefan/

[6] https://www.ibm.com/blogs/cloud-computing/2013/12/ibm-ccra-release-3-0-how-does-it-affect-me-no-9/

[7] http://www.ibm.com/developerworks/library/l-pam/

[8] Rosen, Rami. Linux kernel networking: Implementation and theory. Apress, 2014.

[9] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Managing_Smart_Cards/index.html

[10] http://www.linux-pam.org

[11] https://wiki.archlinux.org/index.php/PAM

[12] http://man7.org/linux/man-pages/man2/setns.2.html

[13] http://man7.org/linux/man-pages/man2/unshare.2.html

[14] https://lwn.net/Articles/580893/

[15] http://man7.org/linux/man-pages/man7/namespaces.7.html

[16] https://github.com/rd235/cado

[17] Davoli, Renzo. "Internet of threads: Processes as internet nodes." International Journal on Advances in Internet Technology Volume 7, Number 1 & 2, 2014 (2014).