FONTANA STEFANO

NETBACK rev. 3

Software di trasferimento e salvataggio file attraverso la rete.

Presentazione del software

Il software si presenta diviso in cinque eseguibili compilati, scritti in linguaggio C/C++ , ed uno script python, in quanto le sue funzionalità comprendono maggiormente lo scambio di dati in rete.

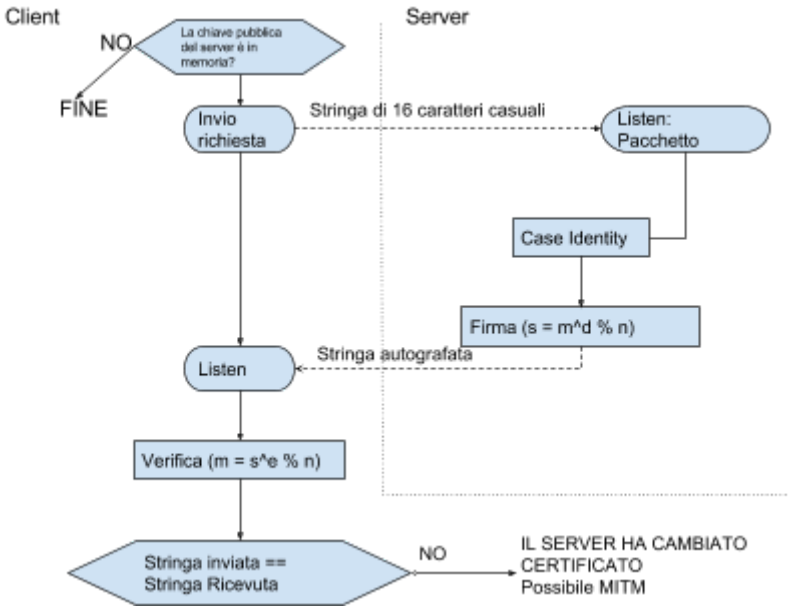
Lo scopo del software è il salvataggio di copie di backup di file e drive fisici attraverso la rete e la loro memorizzazione all'interno di un server preposto, senza utilizzare spazio sul disco fisico della macchina client.

La dotazione di un pannello di controllo con interfaccia web sicura permette l'utilizzo in reti di discrete dimensioni puntando sulla facilità di controllo dei client agli amministratori di rete.

Funzionalità

Controllo dell'identità del server mediante RSA

Il primo controllo del client nei confronti del server consiste nella verifica dell'identità della macchina mediante la firma digitale possibile grazie all'algoritmo RSA.



Al momento della configurazione del server verrà creata una coppia di chiavi (Pubblica e Privata) contenente le informazioni relative ai valori per compiere i passaggi di firmatura: questi dati sono contenuti nel file "`keys.rsacfg`" (Configurazione di default).

Come Visibile dall'immagine a lato, se il client possiede già una copia della chiave pubblica del server, procederà all'invio di una stringa di 16 caratteri che verrà firmata digitalmente dal server, per poi essere rispedita al mittente e verificata.

Se la verifica dovesse avere esito negativo, il server sarebbe giudicato non attendibile e di conseguenza la connessione non assicura la riservatezza dei dati.

Questo previene attacchi del tipo "Man In The Middle".

Crittografia AES ECB a 256 bit

Se definito nel file di configurazione del client, il file trasmesso sarà cifrato mediante `AES 256`.

La crittografie prende parte all'invio del file a blocchi di lunghezza configurabile ma multiplo di 16, dimensione del blocco AES.

Il server si limiterà a salvare il file così come viene inviato, e, mediante un eseguibile di utilità, i file potranno essere decifrati e salvati correttamente.

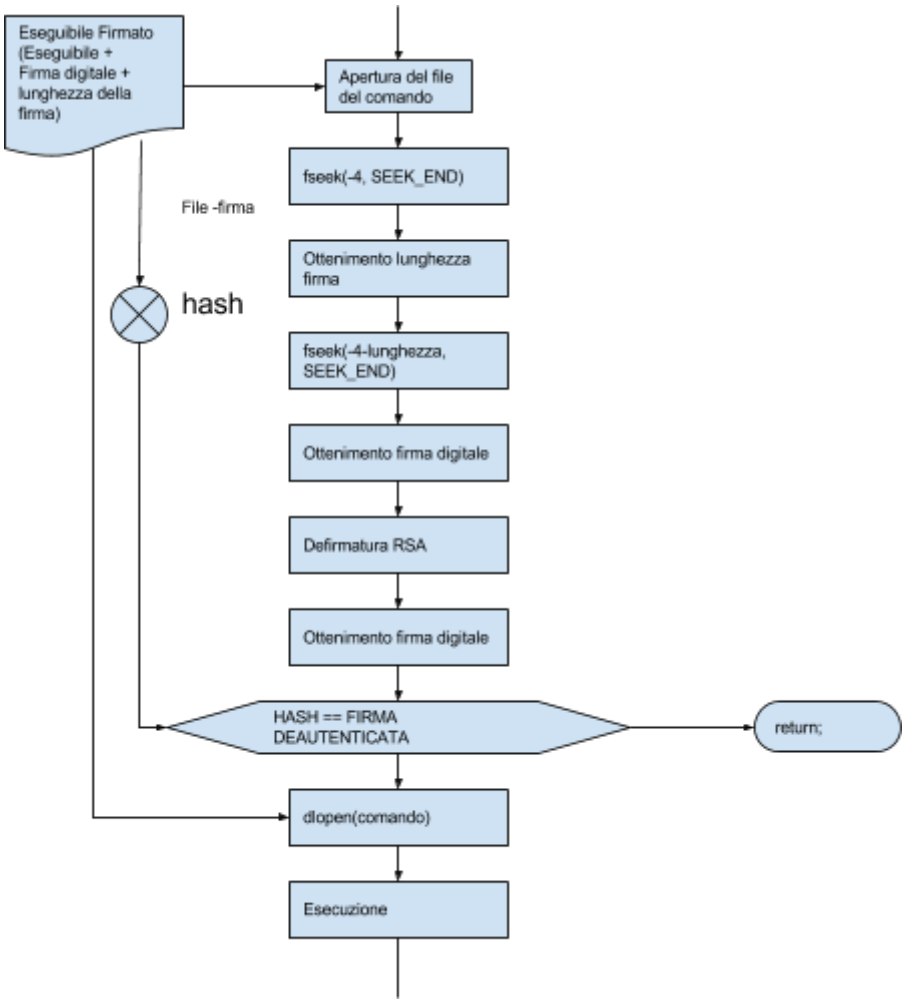
Integrità della trasmissione

La connessione TCP implementa nativamente la ricezione del pacchetto con risposta (ACK) e verifica di integrità dei dati tramite CRC32

Gestione dello spazio su disco

All'atto della ricezione del pacchetto, il server, salverà sul disco fisso i dati, e se specificato nel file di configurazione, il contenuto del pacchetto verrà compresso con l'algoritmo `BZip2`, mantenendo l'occupazione in memoria Ram limitata fissando un limite massimo di dimensione del file prima di essere zippato e scaricato sul disco.

Terminale di gestione interno



Un processo adibito alle funzioni di terminale filtra lo stream `STDIN` ed esegue i comandi caricandoli al momento come librerie dinamiche.
Prima di caricare il codice, per prevenire l'esecuzione di codice malevolo, il file viene verificato mediante il processo di firma digitale utilizzato per la verifica dell'identità del server.
In questo caso una CA (Certificate Authority) viene generata (set di chiavi) al momento della compilazione, per ragioni di sicurezza, la chiave privata e pubblica non dovranno mai esistere sullo stesso hard disk.

Shell SSH

Se impostato al momento della creazione, il server creerà un processo per la gestione delle connessioni SSHv1 e SSHv2 sulle quali verranno reindirizzati gli stream di I/O del terminale.

Questo soltanto se al momento della creazione riesce la creazioni di chiavi RSA e DSA atte alla gestione della shell sicura.

Pannello di controllo web

Lo script python distribuito consente l'apertura di una porta con web server per il controllo dello stato dei backup in corso sul server e la possibilità di gestire il traffico dati.

Implementato nel server vi è un controllo sugli host in grado di usufruire delle operazioni di gestione.

Se l'host su cui viene eseguito il web server è incluso nella suddetta lista, allora potrà accedere ai dati relativi allo stato dei backup.

Il web server è configurabile per richiedere l'autenticazione dell'utente, ed è possibile utilizzarlo sotto connessione protetta HTTPS.

L'utilizzo dell'API websocket richiede che la connessione venga effettuata da un client Google Chrome o derivati implementanti l'API.

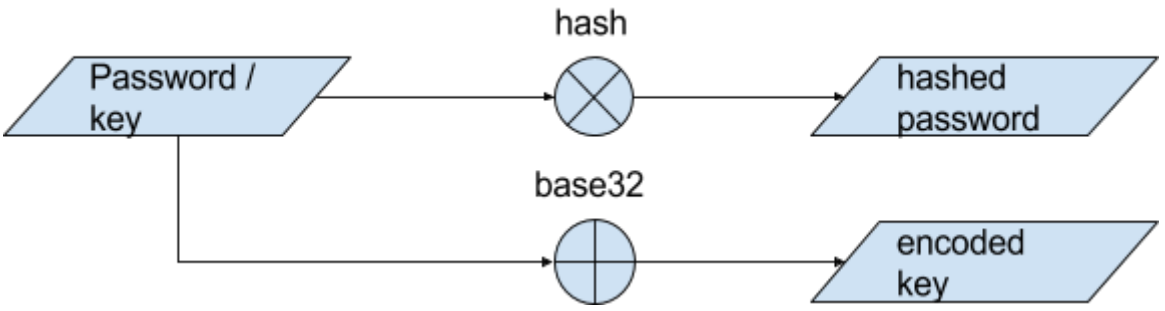
Utilità

All'atto della compilazione verranno generati tre eseguibili e un file di configurazione rsa che, **bensì situati in altre directory, per motivi di sicurezza è fortemente consigliata la separazione fisica degli eseguibili su diverse macchine.**

È di conseguenza consigliata la cancellazione degli eseguibili di utilità dal server e mantenerli su una seconda macchina possibilmente non raggiungibile dalla rete del server, in quanto il file di configurazione rsa (`CA.rsacfg`) contiene la chiave privata per la segnatura dei comandi.

`./genPasswd` - Codificatore di password

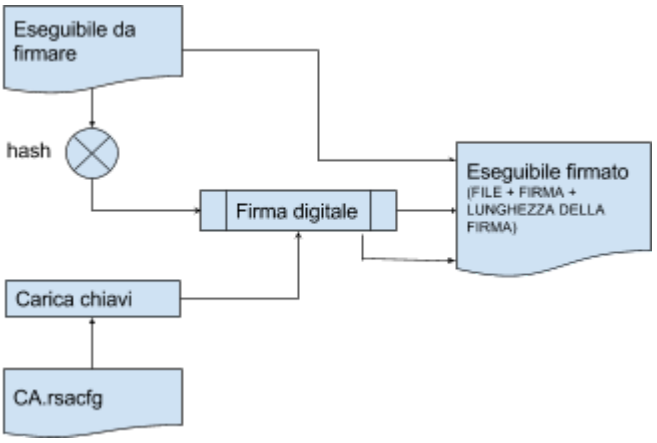
SINTASSI: ./genPasswd <password>
OUTPUT: l'hash generato relativo al parametro <password> inserito e la chiave codificata.



La sessione ssh verificherà l'autenticazione di un solo utente dati il suo nome utente e l'hash della password.
Al momento della convalida delle credenziali, il server verificherà le coincidenze del nome utente, l'hash salvato e fornito al momento della configurazione e l'hash della password inserita dall'utente generato runtime.
Al momento della configurazione, dopo aver generato il corrispondente hash sarà necessario trascrivere l'output del programma nel file di configurazione.

`./signElf` - Firma digitale dei comandi

SINTASSI: ./signElf <percorso/al/comando>
OUTPUT: none



Per verificare l'attendibilità prima dell'esecuzione di un programma il server verificherà la firma digitale del file generata da signElf.
La firma verrà aggiunta in coda all'eseguibile e consisterà nella rappresentazione esadecimale dell'hash del file segnato digitalmente con la chiave privata dell'autorità di certificazione.
Una volta svincolato il dato dalla firma digitale mediante la chiave pubblica salvata, il risultato verrà confrontato con l'hash del file (Firma esclusa).
Se gli hash variassero il comando non verrebbe eseguito perché modificato.

`./extract` - Estrazione dei file crittografati

SINTASSI: ./extract <input_file> <packet_lenght> <key> [output_file]
OUTPUT: none

Durante l'esecuzione, l'applicazione legge dal file `<input_file>` a blocchi di `<packet_lenght>`.

Quest'ultimo deve equivalere al valore (`transfer_block_size` x 16)

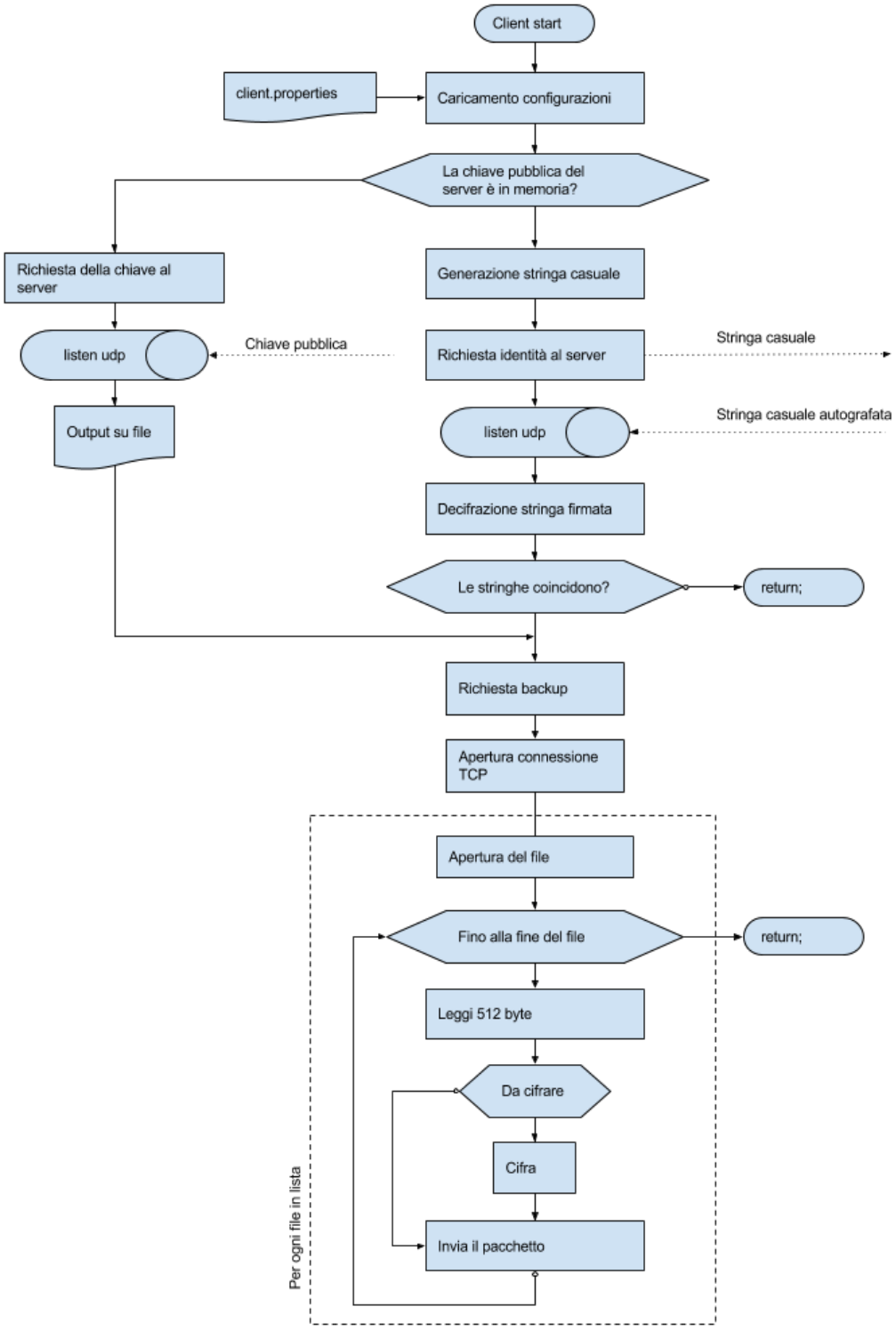
Una volta letto l'header del file, con definizione dei parametri quali la lunghezza del file e se quest'ultimo sia cifrato, verrà creato un file chiamato [output_file] oppure, se non definito, verrà creato un file con il nome definito nell'header di trasmissione nella directory di esecuzione dell'applicazione.

Nel caso il file sarà cifrato, il salvataggio avverà in chiaro.

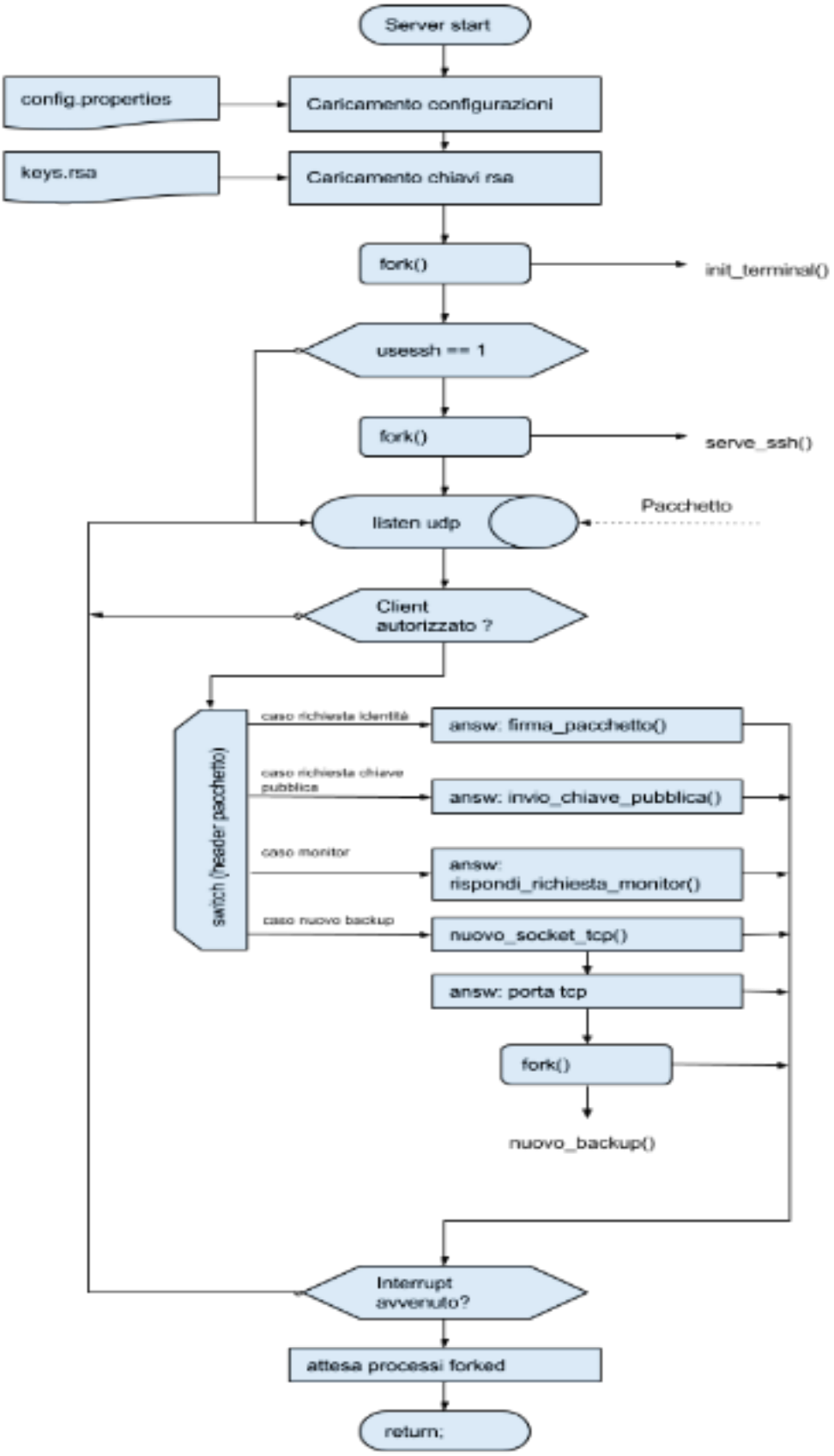
Nel caso in un backup fossero salvati più file con lo stesso nome, essi verranno salvati con il suffisso "`-_<variante>`".

WorkFlow

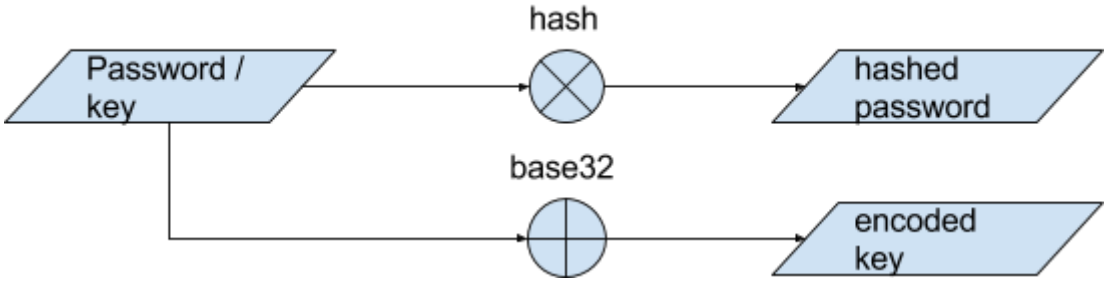
Client



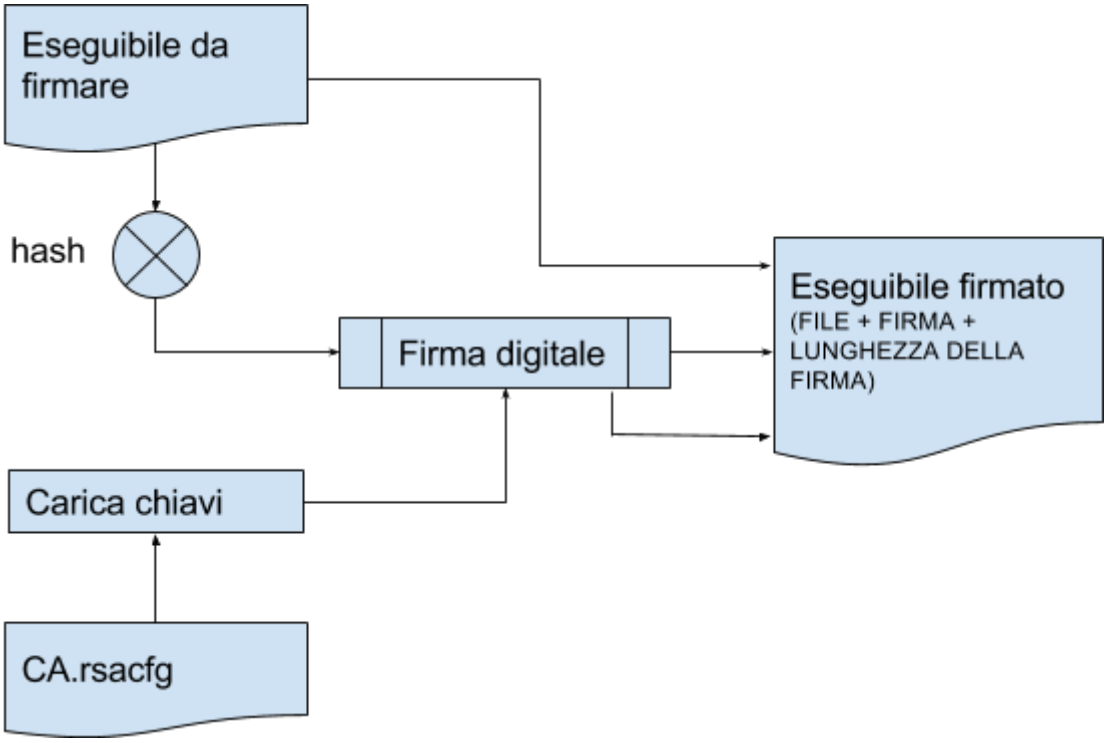
Server


Server start
config.properties
Caricamento configurazioni
keys.rsa
Caricamento chiavi rsa
fork()
init_terminal()
usessh == 1
fork()
serve_ssh()
listen udp
Pacchetto
Client autorizzato ?
switch (header pacchetto)
caso richiesta identità
answ: firma_pacchetto()
caso richiesta chiave pubblica
answ: invio_chiave_pubblica()
caso monitor
answ: rispondi_richiesta_monitor()
caso nuovo backup
nuovo_socket_tcp()
answ: porta tcp
fork()
nuovo_backup()
Interrupt avvenuto?
attesa processi forked
return;

Hashing delle password



Firma degli eseguibili



Gestione delle connessioni

Dopo le procedure di avvio, il server effettuerà le operazioni di binding sulla porta specificata nei parametri di configurazione.

La connessione con il server viene avviata come connessione UDP e i client comunicheranno con la macchina mediante pacchetti di 19 byte così definiti:

```
typedef struct{
    char magic_word[2];
    char data[17];
}__attribute__((packed));
```

dove i caratteri `magic_word` costituiscono l'identificatore del pacchetto e il valore semantico da attribuire ai dati presenti in `data`.

Una volta stabilita una connessione client/server e una volta gestita una richiesta di backup, il server creerà un processo con memoria condivisa, per la gestione del backup, tramite la chiamata `mmap` che aprirà una porta TCP nell'intervallo configurato e la comunicherà al client.

Le connessioni relative al terminale remoto utilizzato dall'interfaccia web vengono filtrate e gestite come pacchetti normali con identificatore ad hoc, mentre la gestione relativa all'interfaccia SSH viaggia su una connessione TCP over TSL con certificato ssh.

Processo di backup

Salvataggio

Una volta effettuata la richiesta di backup, in risposta verrà emanato un pacchetto contenente il numero di porta sulla quale iniziare una comunicazione TCP.

Una volta stabilita una connessione sulla porta indicata, il client invierà un header di backup contenente il client, il numero di file che verranno trasmessi e altre informazioni di utilità informativa per il flusso del programma.

Il backup sarà salvato in una directory all'interno della cartella predefinita, seguendo la nomenclatura `<client ip>_<unix timestamp>/`.

Dopo la trasmissione delle informazioni preliminari ha inizio la trasmissione dei file, con la trasmissione di un `file_header` con la lunghezza di trasmissione e informazioni riguardo se quest'ultimo sia cifrato, il nome originale e la data di invio.

I file vengono quindi salvati man mano che arrivano dalla rete per evitare sovraffollamento di dati nella ram.

Nel caso che vengano inviati due file con lo stesso nome, essi saranno salvati nello stesso file uno concatenato all'altro.

Ripristino

Una volta estratto il file (Se quest'ultimo presenta l'estensione `.bz2`) viene lanciato il software di estrazione.

Una volta letta l'intestazione del file, l'eseguibile scriverà il contenuto su un file specificato in modo binario.

Headers e strutture

Definiti in `Headers.h`

Struttura	Descrizione
`typedef struct{ char mw[2]; char data[17]; }__attribute__((packed)) packet;`	La struttura rappresenta il pacchetto ricevuto dal server UDP e comprende due campi `mw` (L'identificatore del tipo) e `data` contenente i dati scambiati da interpretare in conseguenza del valore `mw`.
`typedef struct{ int explen; int nlen; }__attribute__((packed)) pkey_identifier_t;`	La struttura rappresenta la sequenza di dati nello scambio di una chiave pubblica. Essendo quest'ultima composta da due valori E ed N, la struttura mantiene le lunghezze dei dati, mentre al momento dell'invio verrà inviata la struttura seguita da `explen+nlen` byte contenenti i valori.
`typedef union { uint8_t parts[4]; uint32_t ip; }ipAddr;`	`ipAddr` è l'unione utilizzata per la rappresentazione di un indirizzo ip e nel suo confronto: per la stampa viene utilizzato il vettore di interi a 8 bit rappresentanti i byte di un indirizzo ipv4, mentre per il confronto verrano utilizzati i numeri per incrementare le prestazioni su liste lunghe.
`typedef struct { ipAddr address; unsigned int netMask; }network ;`	Una rete verrà salvata con la coppia di dati `address` e `netmask`, quest'ultima vista come intero data la sua unica utilità nell'operazione logica di verifica rete.

```
typedef struct{
   int server_port;
   int starting_port;
   int port_interval;
   int ToZip;
   uint64_t maxRamAmount;
   int allowEqualsDeny_nets;
   int allowEqualsDeny_ips;
   int netsNo;
   network * networks;
   int ipsNo;
   ipAddr * ips;
   int manNo;
   ipAddr * mans;
}__attribute__((packed)) conf;
```

La struttura detiene i dati relativi alla configurazione del server e, una volta compilata in memoria, verrà passato il suo puntatore ai sottoprocessi di backup e di terminale per accedere alle configurazioni.

```
typedef struct {
   int socket;
   int port;
   sockaddr client;
   uint64_t dimension;
   uint64_t transferred;
   int numberOfFiles;
   int filesTransferred;
   time_t startedInTime;
   int status;
}__attribute__((packed))
backupThread;
```

La struttura rappresenta lo stato in un istante di lettura di un processo di backup.
Essa contiene il numero identificativo `socket` in caso di operazioni da terminale come la chiusura immediata o la pausa.
Contiene i dati utili alla rappresentazione degli stati come percentuali.

```
typedef struct{
   int port;
   char * user;
   char * password;
   char * rsa;
   char * dsa;
   int usepcap;
   char * pcap;
}__attribute__((packed)) sshconf;
```

La struttura mantiene le referenze per le configurazioni del terminale SSH

```
typedef struct{
   int isEncoded;
   sockaddr_in client;
   int numberOfFiles;
   time_t time;
   int packetSize;
   char foo[32];
}__attribute__((packed))
backupHead_t;
```

La struttura descrive l'intestazioni dei file `head.hd` relativi alle informazioni di backup.
le informazioni memorizzate sono:

- Se i file seguenti saranno cifrati,
- L'indirizzo del client,
- Il numero di file nel backup,
- Il timestamp di avvio del processo,
- La dimensione del pacchetto di trasmissione,
- 32 byte di padding per contenere future addizioni.

```
typedef struct{
   uint64_t dimension;
   uint64_t transfer_dimension;
   int isEncoded;
   time_t time;
   char name[50];
   char foo[32];
}__attribute__((packed)) file_h;
```

La struttura definisce le intestazioni dei file salvati.
Essa contiene informazioni quali:

- Dimensioni del file in chiaro
- Dimensioni del file scritto sul disco (non compresso), questo perché se il file fosse cifrato le dimensioni dovranno essere multiplo di 16 e, nel caso non lo fossero verranno aggiunti n byte di padding
- Se il file è cifrato
- La data di trasmissione come timestamp
- Il nome originale
- 32 byte di padding per contenere future addizioni.

Formati file

*.rsacfg

Il formato `rsacfg` è un formato testuale utilizzato per la descrizione di una coppia di chiavi RSA.

Siano nominati i parametri di una coppia di chiavi RSA

- “P” e “Q” la coppia di numeri primi generati
- “N” il modulo
- “E” l’esponente pubblico
- “D” l’esponente privato

I dati contenuti saranno separati dal carattere ‘`\n`’ e memorizzati nel seguente ordine in rappresentazione esadecimale:

```
P (\n)
Q (\n)
N (\n)
E (\n)
D (\n)
 (EOF)
```

*.public

Il formato `public` è un formato testuale utilizzato per la descrizione di una chiave RSA pubblica ed è una variante del formato `rsacfg`.

Siano nominati i parametri di una chiave RSA pubblica

- “N” il modulo
- “E” l’esponente pubblico

I dati contenuti saranno separati dal carattere '`\n`' e memorizzati nel seguente ordine in rappresentazione esadecimale:

```
N (\n)
E (\n)
 (EOF)
```

*.bak

Il formato bak è il formato con cui vengono memorizzati i file sul disco una volta trasferiti.

Viene memorizzato in formato binario ed è costituito da uno o più moduli concatenati fra loro e definiti in questo modo:

File Header (vedi `file_h` in *Header e Strutture*)
Contenuto inviato

Di conseguenza il file in memoria sarà così strutturato:

File Header Contenuto inviato
File Header Contenuto inviato
...
File Header Contenuto inviato
(EOF)

*.hd

I file in formato hd contengono le informazioni generali di un backup e contengono una sola struttura backupHeader_t (vedi *Header e Strutture*)

Crittografia

Gli algoritmi implementati per la cifratura e la verifica della sicurezza sono:

- AES ECB 256bit per la cifratura dei file
- RSA per la verifica delle identità dei file, utilizzando chiavi generate
- Una variante di md5 per motivi di semplicità di codice.

Compilazione

L’esecuzione del Makefile comprende le seguenti fasi:

1. Creazione dello scheletro directory per i programmi
2. Copia dei file generici e degli script
3. Generazione dei certificati e delle chiavi
4. Compilazione dei comandi
5. Compilazione

Comandi

Ogni comando compilato deve seguire le seguenti specifiche.

Modello

C

```c
#include "Headers.h"
#include <string.h>

int execute(char * args, backupThread * back, conf * cfgs,
                                        int (*print_f)(const char *, ...)){

return 0;
}
```

C++

```c
#include "Headers.h"
#include <string.h>
extern "C" int execute(char * args, backupThread * back, conf * cfgs,
                                        int (*print_f)(const char *, ...)){

return 0;
}
```

È obbligatoria la presenza di un metodo `execute` ritornante `int`.
L'inclusione di Headers.h è obbligatoria in quanto definisce tutte le strutture passate come argomenti alla funzione execute.
I parametri passati ad execute saranno:

- `args`: È una stringa contenente i parametri passati dalla linea di comando del terminale
- `back`: Un puntatore al vettore di lunghezza `cfgs->port_interval` contenente lo stato dei backup, chi è connesso è la quantità di dati trasferita.
- `cfgs`: Un puntatore alla struttura dove risiedono le configurazioni del server
- `print_f`: una funzione che permette la stampa sullo stream predefinito, sia il terminale in finestra o tramite SSH.
  La sintassi è la stessa che per printf e il ritorno è il numero di caratteri scritti.

Compilazione

Con il comando

```
gcc <file.c> --shared -fPIC -o <file>
```

Oppure

```
g++ <file.c> --shared -fPIC -o <file>
```

Rispettando il modello per il linguaggio adeguato

Configurazione

config.properties

`#Commento`	Le righe con '#' iniziale vengono ignorate
`port = 5577`	Porta dove sarà situato il server UDP
`startingPort = 11000`	Prima porta per i processi di backup
`interval = 100`	Numero massimo di processi contemporanei. verranno utilizzate le porte dalla `startingPort` alla `startingport+interval`.
`rsaConfig = keys.rsacfg`	File di configurazione delle chiavi RSA del server atte all'identificazione
`zipFiles = 1`	Se diverso da 0 i file ricevuti saranno compressi.
`maxRamAmount = 3000000`	La dimensione massima di un file memorizzato in memoria RAM prima della compressione: se quest'ultimo supera questa dimensione, il file non sarà compresso.
`networks_blacklist = 0`	Se diverso da 0 le reti elencate non saranno autorizzate alla connessione
`networks = :`	Inizia la lista di reti autorizzate (A meno di un valore positivo per `network_blacklist`)
`192.168.0.0/24`	
`135.144.1.0/24`	
`::`	Termina la lista
`ips_blacklist = 0`	Se diverso da 0 gli host elencati non saranno autorizzati alla connessione
`ips = :`	inizia la lista di host autorizzati (A meno di un

	valore positivo per `ips_blacklist`)
`127.0.0.1`	
`192.168.43.177`	
`::`	Termina la lista
`managers = :`	inizia la lista di host autorizzati all'utilizzo del terminale di rete
`127.0.0.1`	
`::`	Termina la lista
`use_ssh = 1`	Un valore diverso da 0 abilita la shell SSH
`ssh_user = stefano`	Indica il nome utente con cui sarà possibile connettersi alla shell ssh
`ssh_psw_md5 = a9d4eff9120382f59fb2d971ca96a85a`	Indica l'hash della password, generato con l'utility `./genPasswd`, prendendo la chiave "`HASH`", da utilizzare per l'accesso tramite SSH. Il confronto avverrà tra l'hash memorizzato e l'hash della stringa inserita dall'utente.
`ssh_port = 2222`	Numero della porta dove servire il servizio di SSH
`ssh_rsa = ./ssh_keys/ssh_host_rsa_key`	Directory relativa all'avvio dove risiede la chiave RSA per la connessione SSH
`ssh_dsa = ./ssh_keys/ssh_host_dsa_key`	Directory relativa all'avvio dove risiede la chiave DSA per la connessione SSH

client.properties

`#Commento`	Le righe con '#' iniziale vengono ignorate
`server_ip = 127.0.0.1`	l'indirizzo IPv4 del server a cui connettersi
`server_port = 5577`	Porta sulla quale contattare il server: dovrebbe corrispondere al parametro `port` in `config.properties` relativo al server
`connection_timeout = 10`	Timeout di connessione in secondi

`files = :`	inizia la lista di file da trasmettere
`/home/.../a.jpg`	File
`/home/.../b.pdf`	File
`/usr/bin/`	Directory
`::`	Termina la lista
`encrypt = 1`	Un valore diverso da 0 determina che i file verranno cifrati con algoritmo AES ECB 256bit prima dell'invio
`transfer_block_size = 32`	Indica il quantitativo in blocchi AES (16 byte l'uno) di byte da inviare per volta: il totale sarà dato dal valore moltiplicato per 16
`key = KFXU2SRTM5DXG6TLM5TEKVDQNJKHEWTLGRDDQYRZMVSXSYJXJVQWC===`	È la chiave di cifratura del file codificata come base32, generabile mediante l'utility `./genPasswd` prendendo il parametro "`Encoded key`"
`send_interval = 500`	Intervallo, in millisecondi, nella trasmissione dei pacchetti. Utile per la riduzione del carico della cpu

Librerie esterne

- libgmp (GNU MULTIPLE PRECISION arithmetic library) => https://gmplib.org/
- libbz2 (Compressione dei file) => http://www.bzip.org/
- libssh (Secure shell) => https://www.libssh.org/