Remote X Apps mini-HOWTO

Vincent Zweije , *zweije@xs4all.nl* v, 14 luglio 1998

Questo mini-HOWTO descrive come eseguire applicazioni X in remoto. Ovvero, come avere un programma X che scrive su un computer diverso da quello su cui sta girando. O viceversa: come far sí che un programma X giri su un computer diverso da quello a cui uno è seduto. L'accento in questo mini-HOWTO è sulla sicurezza.

Indice

1 Introduzione **1**

2 Letture Scelte **2**

3 Lo Scenario **2**

4 Un Poco di Teoria **3**

5 Dirlo al Client **3**

6 Dirlo al Server **4**

6.1 Xhost 4

6.2 Xauth 5

6.2.1 Costruire i Cookie 5

6.2.2 Transportare il Cookie 6

6.2.3 Usare il Cookie 7

6.3 Ssh 7

7 Risoluzione dei Problemi **7**

1 Introduzione

Questo mini-HOWTO è una guida a come far andare le applicazioni X in remoto. È stato scritto per parecchie ragioni.

1. Sono apparse su usenet molte domande su come far andare applicazioni X in remoto.
2. Vedo molti, molti suggerimenti di "usare `xhost +hostname`" o perfino "`xhost +`" per permettere le connessioni X. **Questo è ridicolmente insicuro**, e ci sono metodi migliori.
3. Non conosco un documento semplice che descriva le opzioni che uno *davvero* ha. Per favore informatemi *zweije@xs4all.nl* se ne sapete di più.

Questo documento è stato scritto avendo in mente sistemi di tipo unix. Se il vostro sistema locale o remoto è di un altro tipo, potrete trovare qua come vanno le cose. Nonostante ciò, dovrete tradurre da soli gli esempi per adattarli al vostro sistema/i.

La versione [inglese] più recente di questo documento è sempre disponibile sul WWW all'indirizzo *http://www.xs4all.nl/~zweije/xauth.html* . È anche disponibile come il Linux Remote X Apps mini-HOWTO all'indirizzo *http://sunsite.unc.edu/LDP/HOWTO/mini/Remote-X-Apps* . I Linux (mini-)HOWTO sono disponibili via http o ftp da *sunsite.unc.edu* . La traduzione italiana può essere trovata sul sito *http://pluto.linux.it* e mirror.

Questa è la versione 0.5.1. Non ci sono garanzie, solo buone intenzioni. Sono disponibile a suggerimenti, idee, aggiunte, indicazioni utili, correzioni tipografiche, ecc... Tuttavia vorrei che questo rimanesse un documento semplice e leggibile, nello stile HOWTO inteso al meglio. I flame finiscono in `/dev/null`.

Contenuto aggiornato l'ultima volta il 14 Luglio 1998 da *Vincent Zweije* . Traduzione italiana a cura di *Nicola Pero* .

2 Letture Scelte

Un documento correlato sul WWW è “Che cosa fare quando Tk dice che il tuo display è insicuro” [in inglese], *http://ce-toolkit.crd.ge.com/tkxauth/* . È stato scritto da *Kevin Kenny* . Suggerisce una soluzione all'autenticazione X simile a quella suggerita in questo documento (xauth). Tuttavia, Kevin ha più come obiettivo di usare xdm per guidare xauth per voi.

X System Window System Vol. 8, “Guida dell'Amministratore di X Window System” [in inglese] da *O'Reilly and Associates* è stato anche portato alla mia attenzione come una buona fonte di informazione. Sfortunatamente, non ho potuto provarla.

Tuttavia un altro documento molto simile a quello che state leggendo, intitolato “Rendere X Windows Sicuro” [in inglese], è disponibile presso *http://ciac.llnl.gov/ciac/documents/ciac2316.html* .

Potete anche provare usenet newsgroup, come `comp.windows.x`, `comp.os.linux.x`, e `comp.os.linux.networking`.

3 Lo Scenario

State usando due computer. State usando l'X window system del primo per scrivere e guardare. State usando il secondo per fare qualche importante lavoro grafico. Volete far sí che il secondo mostri il suo output sul display del primo. X window system lo rende possibile.

Naturalmente, avete bisogno di una connessione di rete per fare ciò. Preferibilmente una connessione veloce; il protocollo X mangia molte risorse di rete. Ma con un poco di pazienza e un protocollo di compressione adatto, potete eseguire applicazioni perfino attraverso un modem. Per la compressione del protocollo X, potreste voler provare dxpc *http://ccwf.cc.utexas.edu/~zvonler/dxpc/* o LBX

http://www.ultranet.com/~pauld/faqs/LBX-HOWTO.html `http://www.ultranet.com/~pauld/faqs/LBX-HOWTO.html`

(anche noto come *LBX mini-HOWTO*).

Dovete fare due cose per ottenere tutto ciò

1. Dire al display locale (il server) di accettare connessioni dal computer remoto.
2. Dire all'applicazione (il client) di redirigere il suo output verso il display locale.

4 Un Poco di Teoria

La parola magica è `DISPLAY`. Nell'X window system, un display consiste (semplificando) di una tastiera, un mouse e uno schermo. Un display è gestito da un programma server, conosciuto come server X. Il server mette a disposizione le capacità di visualizzazione agli altri programmi che si connettono a lui.

Un display è indicato con un nome, per esempio:

- `DISPLAY=light.uni.verse:0`
- `DISPLAY=localhost:4`
- `DISPLAY=:0`

Il display consiste di uno hostname (come `light.uni.verse` e `localhost`), due punti (`:`), e un numero di sequenza (come `0` e `4`). L'hostname del display è il nome del computer dove gira il server X. Se l'hostname è omesso si intende il local host [computer locale]. Il numero di sequenza è solitamente 0 – può variare se ci sono più di un display connessi ad un solo computer.

Se mai vi capita di incontrare una indicazione di display con un `.n` in più attaccato, si tratta del numero dello schermo. Un display può in realtà avere più di uno schermo. Di solito tuttavia c'è un solo schermo, che ha numero `n=0`, per cui questo è il default.

Esistono altre forme di `DISPLAY`, ma le precedenti sono sufficienti per i nostri scopi.

5 Dirlo al Client

Il programma client (per esempio, la vostra applicazione grafica) sa a quale display deve connettersi da un esame della variabile di ambiente `DISPLAY`. Tuttavia questo settaggio può essere cambiato, dando al client l'argomento di linea di comando `-display hostname:0` quando viene fatto partire. Alcuni esempi potrebbero rendere le cose più chiare.

Il nostro computer è noto al mondo esterno come light, e siamo nel dominio uni.verse. Se stiamo facendo andare un server X normale, il display è conosciuto come `light.uni.verse:0`. Vogliamo far partire il programma di disegno xfig su un computer remoto, chiamato `dark.matt.er`, e stampare il suo output qua su light.

Supponete di avere già fatto un telnet dentro al computer remoto, `dark.matt.er`.

Se avete csh che sta andando sul computer remoto:

```
dark% setenv DISPLAY light.uni.verse:0
dark% xfig &
```

o alternativamente:

```
dark% xfig -display light.uni.verse:0 &
```

Se avete sh che sta andando sul computer remoto:

```
dark$ DISPLAY=light.uni.verse:0
dark$ export DISPLAY
dark$ xfig &
```

o, alternativamente,

```
dark$ DISPLAY=light.uni.verse:0 xfig &
```

o, naturalmente, anche:

```
dark$ xfig -display light.uni.verse:0 &
```

Sembra che alcune versioni di telnet trasportino automaticamente la variabile `DISPLAY` all'host remoto. Se avete una di queste, siete fortunati, e non dovete settarlo a mano. Altrimenti, la maggior parte delle versioni di telnet trasportano la variabile d'ambiente `TERM`; con qualche hacking giudizioso è possibile piggyback [lett., portare indietro a maialino] la variabile `DISPLAY` sulla variabile `TERM`.

La idea con il piggybacking è di prepare degli script per ottenere le cose seguenti: prima di fare il telnet, si attacca il valore di `DISPLAY` a `TERM`. Quindi si fa il telnet. All'estremità remota, nel file `.*shrc` appropriato, si legge il valore di `DISPLAY` da `TERM`.

6 Dirlo al Server

Il server non accetterà connessioni da dovunque come niente fosse. Non volete che tutti possano visualizzare finestre sul vostro schermo. O leggere quello che state scrivendo – ricordate che la tastiera è parte del vostro display!

Troppa poca gente sembra realizzare che permette di accedere al proprio display pone a rischio la sicurezza. Qualcuno che ha accesso al vostro display può leggere e scrivere sui vostri schermi, leggere che tasti premete, e leggere quello che fa il vostro mouse.

La maggior parte dei server conosce due modi di autenticare le connessioni verso di lui: con la lista di host (xhost) e con i magic cookie (xauth). Infine c'è ssh, la shell sicura, che può trasportare le connessioni X.

6.1 Xhost

Xhost permette l'accesso sulla base degli hostname. Il server mantiene una lista di host che hanno il permesso di connettersi a lui. Può anche disabilitare completamente il controllo degli host. Attenzione: questo significa che non viene fatto nessun controllo, per cui *qualunque* host può connettersi!

Potete controllare la lista di host del server con il programma xhost. Per usare questo meccanismo nell'esempio precedente, fate:

```
light$ xhost +dark.matt.er
```

Questo permette tutte le connessioni dall'host `dark.matt.er`. Non appena il vostro client X ha fatto la sua connessione e ha visualizzato una finestra, per sicurezza, revocate i permessi di altre connessioni con:

```
light$ xhost -dark.matt.er
```

Potete disabilitare il controllo degli host con:

```
light$ xhost +
```

Questo disabilita il controllo di accesso degli host e perciò permette a *chiunque* di connettersi. Non dovete *mai* fare questo in una rete in cui non vi fidate di *tutti* gli utenti (come nel caso di Internet). Potete ri-abilitare il controllo degli host con:

```
light$ xhost -
```

xhost - da solo *non* rimuove tutti gli host dalla lista di accesso (il che sarebbe abbastanza inutile - non potreste connettervi da nessun host, nemmeno dal vostro host locale).

Xhost è un meccanismo molto insicuro. Non distingue fra utenti diversi sull'host remoto. Ancora, gli hostname (in realtà gli indirizzi) possono essere spoofati [=falsificati]. Questo è male se vi trovate in una rete di cui non fidarsi (per esempio già con un accesso ad Internet con PPP, via rete telefonica).

6.2 Xauth

Xauth permette l'accesso a chiunque conosca il segreto giusto. Un tale segreto è chiamato codice di autorizzazione, o magic cookie [lett, biscottino magico]. Questo schema di autorizzazione è formalmente chiamato MIT-MAGIC-COOKIE-1.

I cookie per display differenti sono memorizzati insieme nel file `~/.Xauthority`. Il vostro `~/.Xauthority` deve essere inaccessibile al gruppo/ad altri utenti. Il programma xauth amministra questi cookies, da cui il nomignolo xauth per questo schema di autorizzazione.

Iniziando una sessione, il server legge un cookie dal file che è indicato dall'argomento `-auth`. Fatto questo, il server permette connessioni solo da client che conoscono questo stesso cookie. Quando il cookie in `~/.Xauthority` cambia, *il server non si accorgerà del cambiamento.*

Server più recenti possono generare al volo cookies per i client che lo richiedono. Tuttavia i cookie sono ancora mantenuti dentro il server; non finiscono in `~/.Xauthority` a meno che un client non li metta là. Secondo David Wiggins:

> Un ulteriore espediente che vi potrebbe interessare è stato aggiunto in X11R6.3. Attraverso la nuova estensione SECURITY, il server X stesso può generare e restituire nuovi cookie al volo. Per di più, i cookie possono essere designati come "untrusted" [di cui non fidarsi] in modo che applicazioni che fanno connessioni con tali cookie avranno delle restrizioni nelle operazioni. Per esempio, non potranno rubare input di tastiera/mouse, o contenuti di finestre, da altri client di cui ci si fida. C'è ora un sottocomando "genera" di xauth per rendere questa funzionalità almeno possibile da usare, se non semplice.

Xauth ha un chiaro vantaggio di sicurezza sopra xhost. Potete limitare l'accesso a utenti specifici su computer specifici. Non soffre per lo spoofing [falsificazione] di indirizzi come fa xhost. E se volete, potete ancora usare xhost insieme a xauth per permettere connessioni.

6.2.1 Costruire i Cookie

Se volete usare xauth, dovete far partire il server X con l'argomento `-auth authfile`. Se usate lo script startx, è il posto giusto per farlo. Create una registrazione di autorizzazione, come mostrato sotto, nel vostro script startx.

Passi scelti da `/usr/X11R6/bin/startx`:

```
mcookie|sed -e 's/^/add :0 . /'|xauth -q
xinit -- -auth "$HOME/.Xauthority"
```

Mcookie è un programma minuscolo del pacchetto util-linux, sito primario *ftp://ftp.math.uio.no/pub/linux/* . In alternativa, potete usare md5sum per rielaborare dei dati casuali (per esempio, presi da `/dev/urandom` o `ps -axl`) in formato cookie:

```
dd if=/dev/urandom count=1|md5sum|sed -e 's/^/add :0 . /'|xauth -q
xinit -- -auth "$HOME/.Xauthority"
```

Se non potete editare il file startx (perché non siete root), fate sistemare per bene startx al vostro amministratore di sistema, o fategli invece mettere su xdm. Se non può o non vuole, potete fare uno script `~/.xserverrc`. Se avete questo script, xinit lo esegue al posto del vero server X. Poi potete far partire il vero server X da questo script con gli argomenti adeguati. Per fare questo, fate usare al vostro `~/.xserverrc` la linea per i magic cookie vista prima per creare un cookie e quindi eseguire il vero server X:

```
#!/bin/sh
mcookie|sed -e 's/^/add :0 . /'|xauth -q
exec /usr/X11R6/bin/X "$@" -auth "$HOME/.Xauthority"
```

Se usate xdm per controllare le vostre sessioni X, potete usare xauth facilmente. Definite la risorsa .authDir del DisplayManager in `/etc/X11/xdm/xdm-config`. Xdm passerà l'argomento `-auth` al server X server quando parte. Quando poi voi fate un log in sotto xdm, xdm mette il cookie nel vostro `~/.Xauthority` per voi. Si veda xdm(1) per maggiori informazioni. Per esempio, il mio `/etc/X11/xdm/xdm-config` contiene la seguente linea:

```
DisplayManager.authDir: /var/lib/xdm
```

6.2.2 Transportare il Cookie

Ora che avete incominciato la vostra sessione X sull'host server `light.uni.verse` e che avete il vostro cookie in `~/.Xauthority`, dovrete trasferire il cookie all'host client, `dark.matt.er`.

La cosa più semplice è quando le vostre directory su light e dark sono condivise. I file `~/.Xauthority` sono gli stessi, per cui il cookie è trasportato simultaneamente. Tuttavia, c'è un inganno: quando mettete un cookie per `:0` in `~/.Xauthority`, dark penserà che sia un cookie per sé stesso invece che per light. Dovete usare un host name esplicito quando create un cookie; non potete tralasciarlo. Potete installare lo stesso cookie sia per `:0` che per `light:0` con:

```
#!/bin/sh
cookie=`mcookie`
xauth add :0 . $cookie
xauth add "$HOST:0" . $cookie
exec /usr/X11R6/bin/X "$@" -auth "$HOME/.Xauthority"
```

Se le home directory non sono condivise, potete trasportare il cookie per mezzo di rsh, la shell remota:

```
light$ xauth nlist :0 | rsh dark.matt.er xauth nmerge -
```

1. Estrae il cookie dal vostro `~/.Xauthority` locale (`xauth nlist :0`).
2. Lo trasferisce a dark.matt.er (`| rsh dark.matt.er`).
3. Lo mette nel `~/.Xauthority` là (`xauth nmerge -`).

È possibile che rsh non vada bene per voi. A parte questo, rsh ha anche un incoveniente per quanto rigurda la sicurezza (host spoofati [falsificati] di nuovo, se non ricordo male). Se non potete o non volete usare rsh, potete anche trasferire il cookie manualmente, tipo:

```
light$ echo $DISPLAY
:0
light$ xauth list $DISPLAY
light/unix:0 MIT-MAGIC-COOKIE-1 076aaecfd370fd2af6bb9f5550b26926
light$ rlogin dark.matt.er
Password:
dark% setenv DISPLAY light.uni.verse:0
dark% xauth add $DISPLAY . 076aaecfd370fd2af6bb9f5550b26926
dark% xfig &
[15332]
dark% logout
light$
```

Si vedano anche rsh(1) e xauth(1x) per maggiori informazioni.

Potrebbe essere possibile fare un “piggyback” del cookie nella variabile `TERM` o `DISPLAY` quando fate un telnet all'host remoto. Questo funzionerebbe nello stesso modo in cui si fa il piggyback della variabile `DISPLAY` sulla variabile `TERM`. Si veda la sezione 5: Dirlo al Client. Dal mio punto di vista qua sono fatti vostri, ma sono interessato se qualcuno potesse confermarlo o negarlo.

6.2.3 Usare il Cookie

Una applicazione X su dark.matt.er, come la xfig di prima, guarderà automaticamente in `~/.Xauthority` là per il cookie con cui autenticarsi.

6.3 Ssh

Le registrazioni di autorità sono trasmesse senza crittografia. Se siete anche solo impensieriti dall'idea che qualcuno possa snoopare [annusare] le vostre connessioni, usate ssh, la shell sicura. Andrà bene per trasportare X sopra connessioni crittate. E inoltre, è grande anche per molti altri motivi. È un buon miglioramento strutturale del vostro sistema. Visitate semplicemente *http://www.cs.hut.fi/ssh/* , la home page di ssh.

Chi conosce qualcosa d'altra sugli schemi di autenticazione o di crittografia delle connessioni X? Forse kerberos?

7 Risoluzione dei Problemi

La prima volta che cercate di lanciare una applicazione X remota, di solito non funziona. Ecco qua qualche comune messaggio di errore, le sue probabili cause, e soluzioni per aiutarvi.

```
xterm Xt error: Can't open display:
```

Non c'è una variabile `DISPLAY` nell'ambiente, e non avete neanche parlato all'applicazione con il flag `-display`. L'applicazione assume una stringa vuota, ma questa è sintatticamente invalida. Per risolvere questo problema, assicuratevi di aver settato correttamente la variabile `DISPLAY` nell'ambiente (con `setenv` o `export` a seconda della vostra shell).

```
_X11TransSocketINETConnect: Can't connect: errno = 101
xterm Xt error: Can't open display: love.dial.xs4all.nl:0
```

L'Errno 101 è "Network is unreachable" [rete irraggiungibile]. L'applicazione non ha potuto fare una connessione di rete al server. Controllate di avere settato correttamente `DISPLAY`, e che la macchina server sia raggiungibile dal vostro client (lo deve essere, dopotutto siete probabilmente loggati nel server e state facendo un telnet al client).

```
_X11TransSocketINETConnect: Can't connect: errno = 111
xterm Xt error: Can't open display: love.dial.xs4all.nl:0
```

L'Errno 111 è "Connection refused" [Connessione rifiutata]. La macchina server a cui state cercando di connettervi è raggiungibile, ma là il server indicato non esiste. Controllate di stare usando l'host name giusto e il numero di display giusto.

```
Xlib: connection to ":0.0" refused by server
Xlib: Client is not authorized to connect to Server
xterm Xt error: Can't open display: love.dial.xs4all.nl:0.0
```

Il client ha potuto fare una connessione al server, ma il server non permette al client di usarlo (non è autorizzato). Assicuratevi di avere trasportato il magic cookie corretto al client, e che non sia espirato (il server usa un nuovo cookie quando incomincia una nuova sessione).