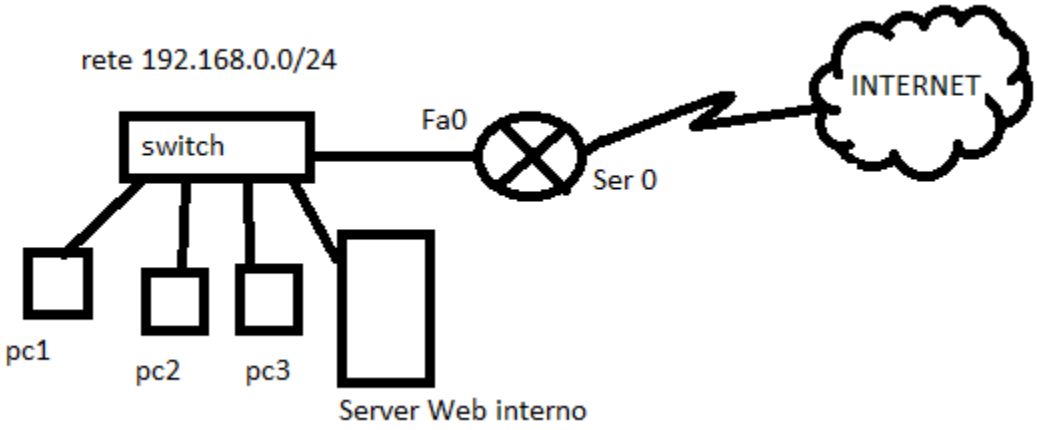
Script PHP per Configurare gli Accessi ad Internet di un router CISCO

Autore Roberto Bandiera – 9 dicembre 2014

Obiettivo: scrivere uno script PHP per poter controllare da remoto la configurazione di un router CISCO. In particolare l'obiettivo è quello di gestire i diritti di accesso ad Internet da parte degli utenti della rete locale. Le prove tecniche sono state fatte con un router CISCO modello 1721.

Per fissare le idee si pensi al seguente semplice schema di rete aziendale (che però potrebbe essere molto più articolata, comprendendo molteplici LAN con indirizzi privati del tipo 192.168.x.0/24):



Lo schema della rete di riferimento

dove l'indirizzo dell'interfaccia Fa0 del router ha indirizzo IP 192.168.0.1/24.

Per la configurazione iniziale del router si utilizza la porta Console, si configura l'interfaccia Fa0 e si abilita l'accesso (login) mediante telnet.

In seguito ci si può interfacciare al router mediante connessione Telnet sulla porta 23 oppure si può utilizzare una connessione cifrata con SSH sulla porta 22.

Configurazione iniziale tramite Console e poi configurazione con Telnet – mediante software Putty

Nel server della LAN viene installato XAMPP per avere il server Apache in grado di eseguire gli script PHP adibiti alla gestione del router.

Nella cartella C:\xampp\htdocs viene creata la cartella “router” contenente il file “router.php” per dimostrare le possibilità di agire sul router.

Si fa notare che poiché lo script invia comandi al router e riceve dallo stesso informazioni di configurazione, è opportuno che il dialogo con il router avvenga tramite protocollo SSH, anche se nell’esempio è stato usato Telnet. Inoltre, poiché da qualsiasi pc della rete è possibile attivare lo script di configurazione, è opportuno che il dialogo con il server Apache avvenga sulla porta 443, con il protocollo https, piuttosto che sulla porta 80 con il protocollo http, e che ovviamente l’accesso a tale script sia protetto da password.

Lo script PHP per la configurazione del router contiene la classe Router per eseguire la connessione con il router e per inviargli i comandi desiderati. Ogni metodo di configurazione assume di iniziare dalla “privileged mode” **R#** e al termine ci si ritrova nuovamente in tale modalità.

```
class Router
{
  // attributi pubblici
  public $connection = null;
  public $message = "";
```

Il metodo per effettuare la connessione riceve l'indirizzo IP del router, che sarà "192.168.0.1", la porta di accesso (23) e le password per l'accesso, che sono "cisco" e "cisco".

Il metodo effettua la connessione mediante la funzione **fsockopen()** che ritorna un valore booleano corrispondente al successo della stessa. In caso di fallimento, vengono assegnate alle due variabili passate per riferimento $errno e $errstr, rispettivamente, il numero e il messaggio dell'errore verificatosi.

Dopo aver effettuato la connessione, l'attributo $connession contiene il riferimento al canale di comunicazione con il server e su di esso si agisce come se fosse un file di testo:

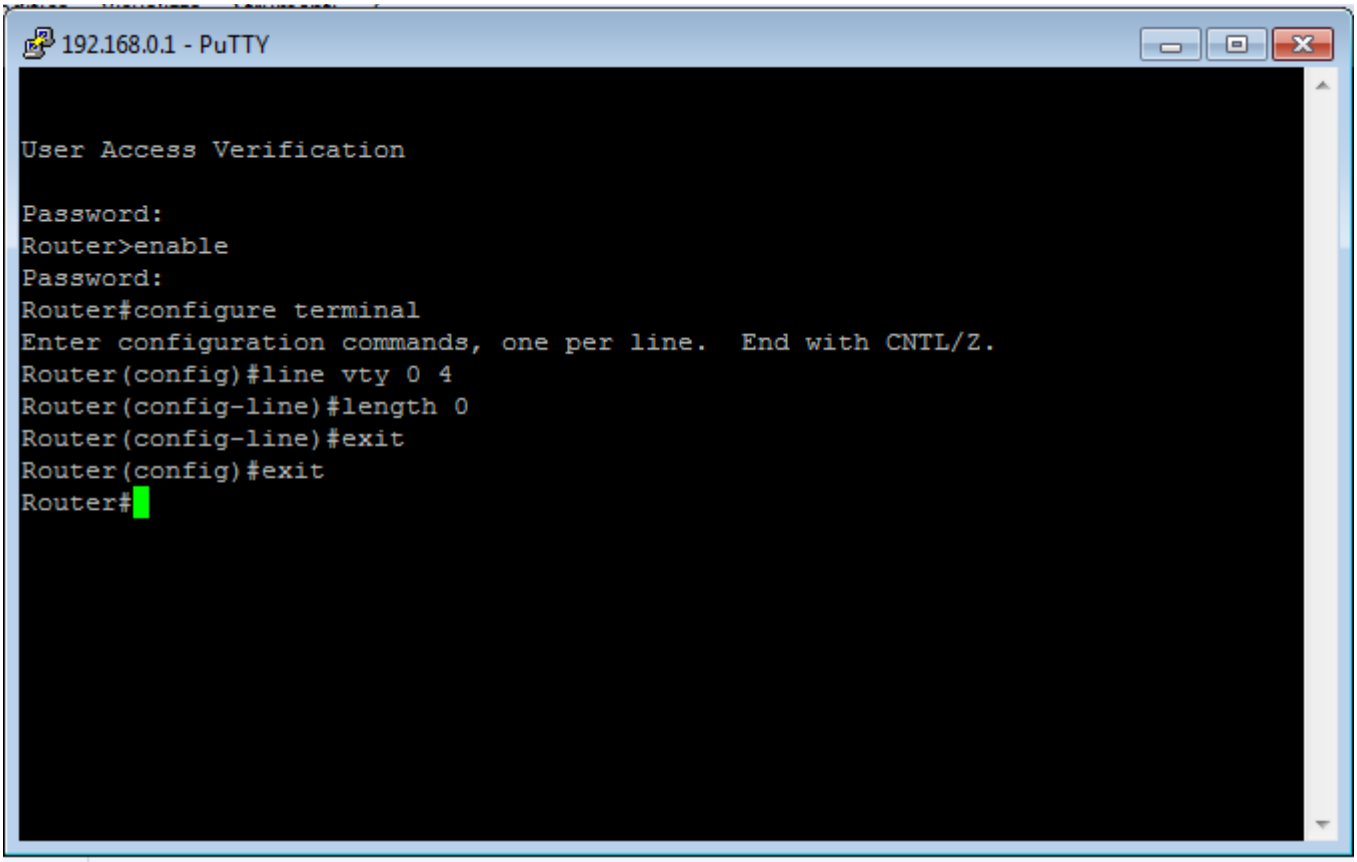
fgets() per leggere e **fputs()** per scrivere. In particolare, nella funzione fgets(), il parametro 128 specifica che si leggono al massimo 128 byte per riga (tale parametro si può anche omettere nelle ultime versioni di PHP).

All'inizio il router si aspetta la password di Telnet e poi, per entrare in privileged mode, viene inviato il comando "enable" e subito dopo la password corrispondente.

Tale metodo si occupa anche di configurare la linea vty (quella usata da telnet) per disabilitare la paginazione dell'output in modo da avere un flusso continuo di informazioni dal router: si tratta di dare il comando **length 0**

```
// metodo per effettuare la connessione con il router
public function connect($routerIP, $port, $passwordTelnet, $passwordEnable)
{
  // $port = 23 per telnet  -  $port = 22  per ssh
  $timeout = 10; // lo imposto < 30, che è il default, per poterlo intercettare!
  $success;
  $this->connection = fsockopen($routerIP, $port, $errno, $errstr, $timeout);
  if(!$this->connection) {
    $success = false;
    $this->message = "connection failed - $errno $errstr \n";
  }
  else {
        $success = true;
    $this->message = "connected \n";
  }
  // password di accesso per telnet
  fputs ($this->connection, "$passwordTelnet\r\n");
  // entro in privileged mode R#
  fputs ($this->connection, "enable\r\n");
  fputs ($this->connection, "$passwordEnable\r\n");
   // disabilito la paginazione dell'output
  fputs ($this->connection, "configure terminal\r\n");
  fputs ($this->connection, "line vty 0 4\r\n");
  fputs ($this->connection, "length 0\r\n");
  fputs ($this->connection, "exit\r\n");
  fputs ($this->connection, "exit\r\n");
  // nel frattempo ho ricevuto l'eco dei comandi dati e le relative risposte
  for ($i=0; $i<13; $i++){
        $nonMiInteressa = fgets($this->connection, 128);
   }
  // imposto un timeout per i gets
  stream_set_timeout($this->connection, $timeout);
  return $success;
}
```

Per capire meglio il funzionamento di tale metodo, si deve tenere presente che il router mi invia continuamente l'eco dei comandi inviati, oltre che le relative risposte: è sufficiente guardare la schermata interattiva di Putty per rendersi conto di tutte le righe di informazione che vengono ricevute dal terminale:



Connessione interattiva con Putty

ecco perché viene effettuato un ciclo per leggere le 13 righe fornite dal router.

Tale metodo, infine, si occupa anche di impostare un timeout in lettura con la funzione **stream_set_timeout()**. Per default il timeout è di 30 secondi, dopodiché la connessione con il router viene automaticamente chiusa. Pertanto, conviene impostare un timeout inferiore in modo da poter controllare il suo verificarsi, rimanendo connessi al router e potendo così continuare a dialogare con lo stesso.

I metodi che attivano l'uscita verso Internet ad un singolo dispositivo o per una intera LAN sono i seguenti; essi ricevono in input un indirizzo IP corrispondente, rispettivamente, alla lan o al singolo pc da abilitare.

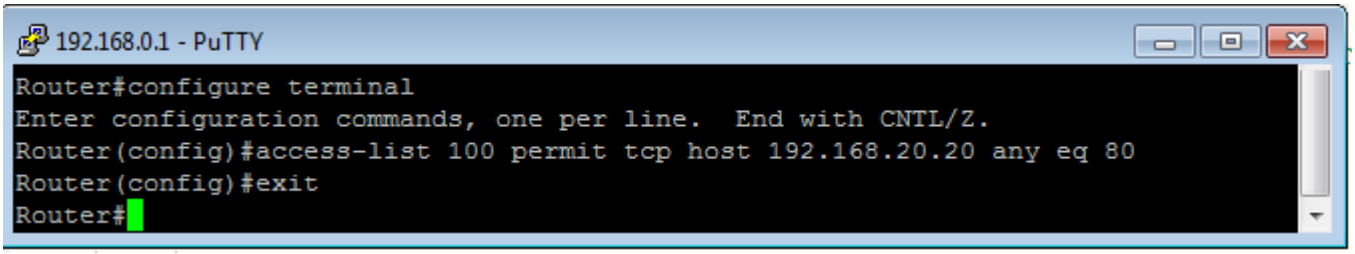
```
// attiva Internet per tutti i dispositivi di una LAN
public function enable_internet_lan($network_ip)
{
   if ($this->connection)
   {
      fputs ($this->connection, "configure terminal\r\n");
      fputs ($this->connection, "access-list 100 permit tcp $network_ip 0.0.0.255 any
eq 80\r\n");
      // ipotizzo che sia gia' impostato l'access-group 100  nella opportuna interfaccia
      fputs ($this->connection, "exit\r\n");
```

```
        // nel frattempo ho ricevuto l'eco dei comandi dati e le relative risposte
        for ($i=0; $i<5; $i++){
                $nonMiInteressa = fgets($this->connection, 128);
        }
    }
}

// attiva Internet per un singolo dispositivo
public function enable_internet_single($ip)
{
    if ($this->connection)
    {
        fputs ($this->connection, "configure terminal\r\n");
        fputs ($this->connection, "access-list 100 permit tcp host $ip any eq 80\r\n");
        fputs ($this->connection, "exit\r\n");
        // nel frattempo ho ricevuto l'eco dei comandi dati e le relative risposte
        for ($i=0; $i<5; $i++){
                $nonMiInteressa = fgets($this->connection, 128);
        }
    }
}
```

I due metodi sono del tutto analoghi. Essi inseriscono nel router una opportuna regola nella access list numero 100. Si suppone che sull'interfaccia di uscita del router sia attivata tale access list con un comando del tipo **ip access-group 100 out**

Ancora una volta si deve tener conto dell'eco dei comandi inviati, che consta di 5 righe, come si vede nella seguente schermata interattiva ottenuta con Putty:



Aggiunta di una regola per consentire al pc 192.168.20.20 di uscire in Internet

Un ulteriore metodo consente di visualizzare tutte le access list presenti nel router:

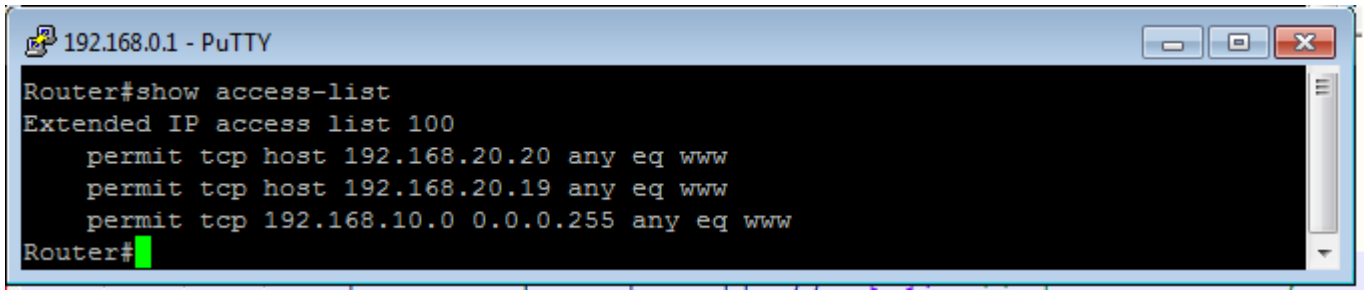
```
// metodo che legge le access list del router    -  versione 1
public function get_access_list()
{
    if ($this->connection)
    {
        fputs ($this->connection, "show access-list\r\n");
        // salto l'eco del comando
        $nonMiInteressa = fgets($this->connection, 128);
        while (!feof($this->connection))
        {
            $riga = fgets($this->connection, 128);
            echo $riga."<br>";
```

```
        // controllo se è andato in time out
         // $info è un array associativo
         $info = stream_get_meta_data($this->connection);
         if ($info['timed_out'])   {
                echo "Timed Out <br>";
                break; // esco dal ciclo
         }
      }
   }
}
```

Questa prima versione del metodo effettua un ciclo di letture finchè il router ha finito (end of file) ed effettua semplicemente un eco a video della risposta del router.

Si noti che dentro il ciclo si controlla se la lettura è andata in timeout, grazie al metodo **stream_get_meta_data()** che restituisce un array associativo dove la componente indicizzata da "timed_out" risulta true nel caso in cui si sia verificato il timeout. Apparentemente tale accorgimento sembra superfluo, ma in pratica si verifica che il router non mi comunica la fine del flusso di dati e pertanto si cade inevitabilmente in un timeout finale.

La figura seguente mostra la schermata ottenuta interattivamente con Putty:



Le access list del router

Una versione più interessante del suddetto metodo, consente di restituire un array di oggetti di tipo AccessRule, dove AccessRule è il nome di una classe PHP appositamente definita per rappresentare ciascuna singola regola.

```
// da includere all'inizio del file router.php
class AccessRule
{
   // solo per Extended Access List
    public $number;  // 100
    public $tipo;  // permit o deny
    public $protocollo;  // tcp  udp  icmp
    public $tipoOrigine;  // singolo  rete
    public $IPorigine;   //  192.168.0.0
    public $maskOrigine;   //  0.0.0.255
    public $tipoDestinaz;
    public $IPdestinaz;
    public $maskDestinaz;
    public $operatore;  //  eq  neq  gt   lt
    public $servizio;  // www  .....  o numero di porta
}
```

Ecco la seconda versione del metodo:

```
// metodo che legge le access list del router    -  versione 2
 // restituisce un array di AccessRule
 public function get_access_list()
 {
    $arr = array();   // l'array da restituire al chiamante
    if ($this->connection)
    {
      fputs ($this->connection, "show access-list\r\n");
      // salto l'eco del comando
      $nonMiInteressa = fgets($this->connection, 128);
      //  numero della access list corrente
      $numero;
      while (!feof($this->connection))
      {
        $riga = fgets($this->connection, 128);
        $riga = trim($riga);  // rimuove eventuali spazi iniziali
        // controllo se è andato in time out
        // $info è un array associativo
        $info = stream_get_meta_data($this->connection);
        if ($info['timed_out'])   {
                echo "Timed Out <br>";
                break; // esco dal ciclo
         }
       // uso una espressione regolare per verificare se la riga corrente contiene
"Extended"
        if (ereg("Extended", $riga))
        // Extended IP access list 100
        {
           $arrAppoggio = explode(" ", $riga);
           $numero = $arrAppoggio[4];  // estraggo il numero dell'access list
        }
        else     // singolo pc     permit tcp host 192.168.30.12 any eq www
                 // oppure rete    permit tcp 192.168.40.0 0.0.0.255 any eq www
        {
           $arrAppoggio = explode(" ", $riga);
           $rule = new AccessRule();
           $rule->number = $numero;
            $rule->tipo = $arrAppoggio[0];  // permit o deny
            $rule->protocollo = $arrAppoggio[1];  // tcp  udp  icmp
            if ($arrAppoggio[2] == "host")
             {
                    $rule->tipoOrigine = "singolo";  // singolo  rete
                    $rule->IPorigine = $arrAppoggio[3];   //  192.168.0.0
                    $rule->maskOrigine = "0.0.0.0";
                    $rule->tipoDestinaz = $arrAppoggio[4];  // any
                    $rule->IPdestinaz;
                    $rule->maskDestinaz;
                    $rule->operatore = $arrAppoggio[5];  //  eq  gt   lt
                    $rule->servizio = $arrAppoggio[6];   // www  .....
             }
            else   // è una rete
            {
```

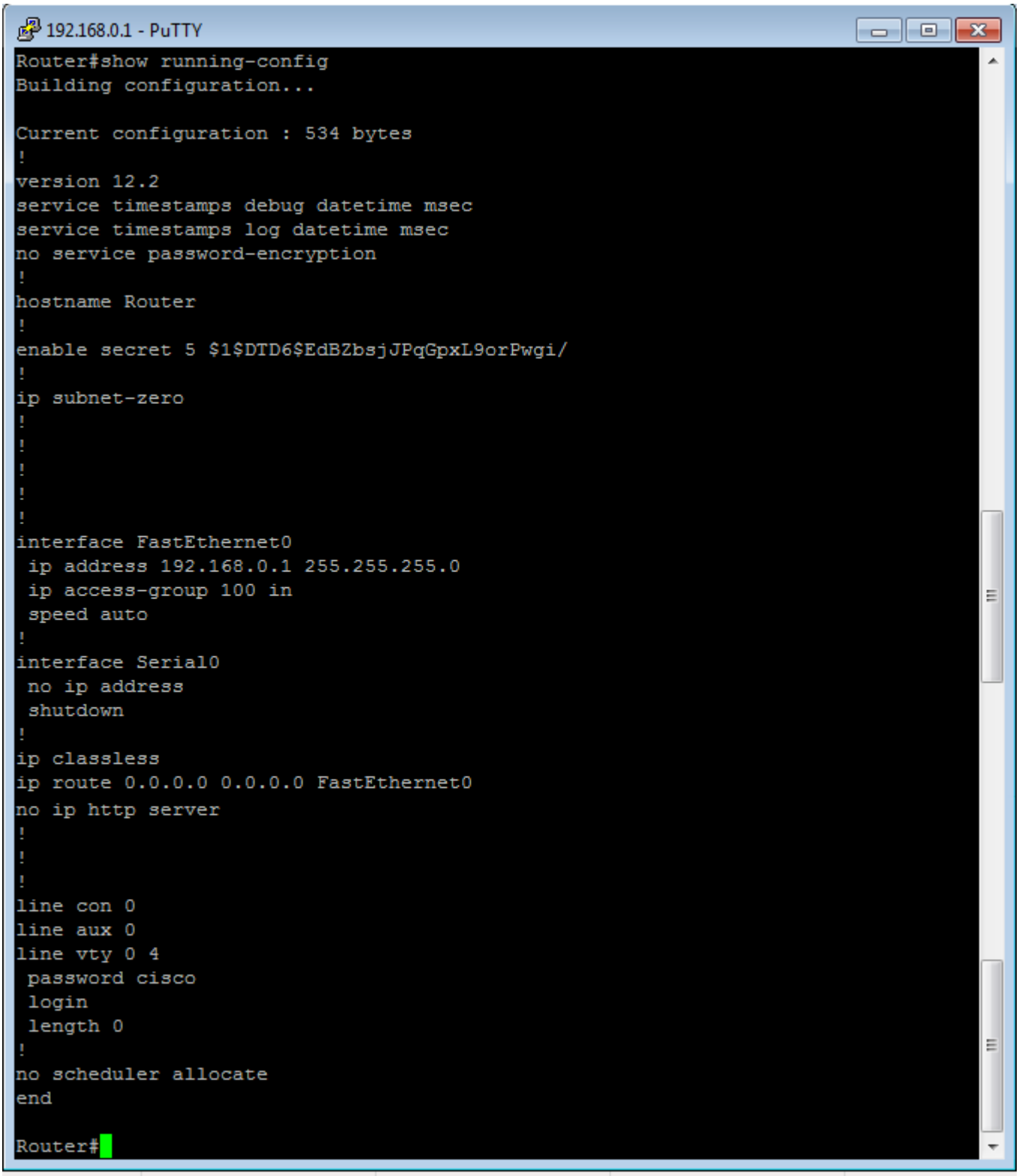
```
                    $rule->tipoOrigine = "rete";  // singolo  rete
                    $rule->IPorigine = $arrAppoggio[2];   //  192.168.0.0
                    $rule->maskOrigine = $arrAppoggio[3];
                    $rule->tipoDestinaz = $arrAppoggio[4];  // any
                    $rule->IPdestinaz;
                    $rule->maskDestinaz;
                    $rule->operatore = $arrAppoggio[5];  //  eq  gt   lt
                    $rule->servizio = $arrAppoggio[6];   // www  .....
                }
                $arr[] = $rule;
            }
        }
    }
    return $arr;
}
```

in definitiva si ottiene un array di oggetti che il chiamante di questo metodo potrà utilmente utilizzare in diversi modi.

Ora vediamo il metodo che legge la configurazione corrente del router (“running-config”) e la restituisce al chiamante come array di stringhe.

```
public function get_running_config()
  {
    if ($this->connection)
    {
                fputs ($this->connection, "show running-config\r\n");
                // salto l'eco del comando e le prime 2 righe della risposta
                for ($i = 0; $i<3; $i++){
                    $nonMiInteressa = fgets($this->connection, 128);
                }
      $arrConfig = array();
      while (!feof($this->connection))
      {
         $arrConfig[] = fgets($this->connection, 128);
        // controllo se è andato in time out
        // $info è un array associativo
                $info = stream_get_meta_data($this->connection);
                if ($info['timed_out'])
                {
                    echo "Timed out";
                    break;  // esco dal ciclo
                }
      }
      return $arrConfig;
    }
  }
```

La schermata di Putty illustra un possibile esito del comando **show running-config** (notare che non c’è nessuna paginazione dell’output):



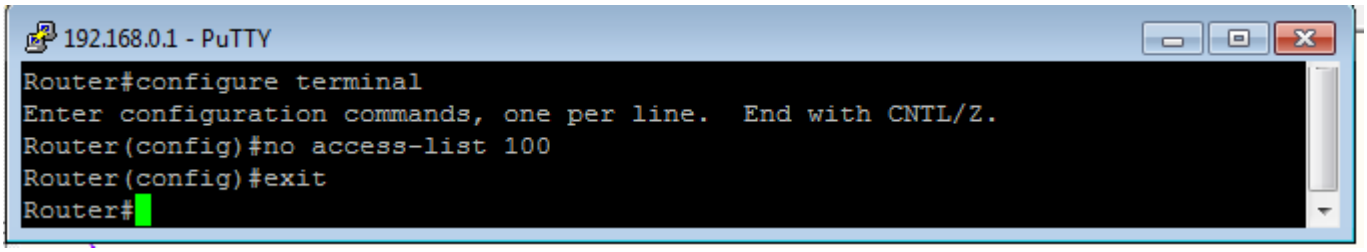
La configurazione del router

Per disabilitare l'accesso a Internet di tutti i dispositivi della LAN si utilizza il seguente metodo:

```
// cancella l'access list 100 cosicché nessuno potrà più andare in Internet
public function disable_internet()
{
  if ($this->connection)
  {
```

```
        fputs ($this->connection, "configure terminal\r\n");
        fputs ($this->connection, "no access-list 100\r\n");
        fputs ($this->connection, "exit\r\n");
        // salto l'eco del comando e la riga di risposta
        for ($i = 0; $i<5; $i++){
            $nonMiInteressa = fgets($this->connection, 128);
        }
    }
}
```

Ancora una volta è utile vedere quello che succede quando si agisce in modo interattivo:



Cancellazione dell'access list numero 100

Per disconnettersi dal router è sufficiente dare il comando **exit**:

```
// disconnessione dal router
public function disconnect()
{
    if ($this->connection)
    {
        fputs ($this->connection, "exit\r\n");
        fclose($this->connection);
    }
}
```

Tutti questi metodi della classe Router vengono chiamati da una pagina PHP a titolo esemplificativo (prova.php). Per chiamare questa pagina basta scrivere sul browser l'indirizzo:

http://localhost/router/prova.php

```
<?php
include_once("router.php");

// esempio di lavoro con il router
echo "---------CONNESSIONE----------<br>";
$router = new Router();
$success = $router->connect("192.168.0.1", 23, "cisco", "cisco");
echo $router->message."<br>";

echo "----------ENABLE-------------<br>";
$router->enable_internet_single("192.168.30.12");
$router->enable_internet_single("192.168.30.13");
$router->enable_internet_lan("192.168.40.0");
echo "------------------------------<br>";
```

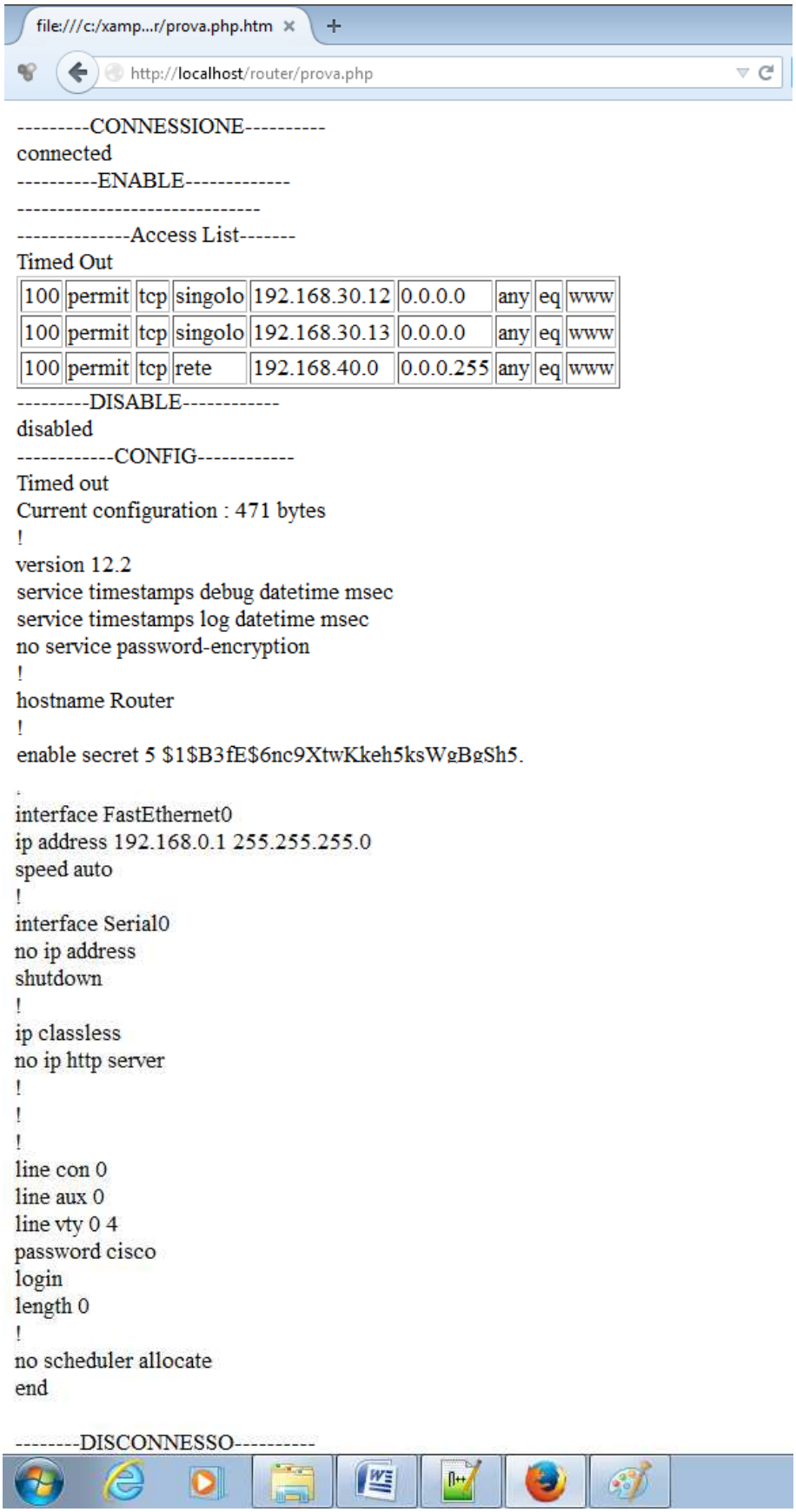
```
echo "--------------Access List-------<br>";
//sleep(2); // secondi
$arr = $router->get_access_list();
echo "<table border=1>";
foreach ($arr as $r):
    echo "<tr><td>$r->number</td><td>$r->tipo</td><td>$r->protocollo</td>";
        echo "<td>$r->tipoOrigine</td><td>$r->IPorigine</td><td>$r-
>maskOrigine</td>";
        echo "<td>$r->tipoDestinaz</td><td>$r->operatore</td><td>$r-
>servizio</td></tr>";
endforeach;
echo "</table>";

echo "---------DISABLE------------<br>";
$router->disable_internet();
echo "disabled<br>";

echo "------------CONFIG------------<br>";
$arr = $router->get_running_config();
foreach ($arr as $riga):
   echo $riga."<br>";
endforeach;
$router->disconnect();
echo "--------DISCONNESSO----------<br>";
?>
```

Se si teme che l'invio dei comandi risulti troppo rapido (caso che non si è verificato nelle prove tecniche effettuate), si possono inserire delle istruzioni **sleep()** per fare delle pause di alcuni secondi. Una esigenza di questo tipo si era verificata in un caso di dialogo con un dispositivo lento come il Robot Bosch SCARA che non era in grado di recepire input di molti caratteri in rapida successione.

Per rendersi conto dell'effetto di questa pagina di prova si veda il suo output visualizzato con il browser:



Lo script PHP chiamato con il browser

Dopo aver dimostrato la tecnica di dialogo e configurazione di un router CISCO mediante uno script PHP, si aprono ampie possibilità per la gestione centralizzata delle autorizzazioni all'accesso ad Internet degli utenti della rete aziendale, con l'eventualità di procedere anche alla contabilizzazione delle stesse:

Tabella Utenti

ID	Nome	Altri dati	Password
1	Antonio	Professore	615243yhqwoi
2	Michele	Studente	Ioyupo78
3	Gianni	Studente	Jjdsfjj99999

Tabella Accessi_Effettuati

ID	ID_utente	Indirizzo_IP	Timestamp_attivazione	Timestamp_disattivazione
1	1	192.168.10.29	08/12/2014 14:20	08/12/2014 16:08
2	2	192.168.10.30	08/12/2014 15:00	08/12/2014 15:30
3	2	192.168.20.23	08/12/2014 16.00	08/12/2014 17:01
4	1	192.168.10.29	09/12/2014 14:18	09/12/2014 15:59

Tabella Accessi_Correnti

ID	ID_utente	Indirizzo_IP	Timestamp_attivazione
1	1	192.168.10.29	10/12/2014 13:40

Poiché il comando **no access-list 100** determina la rimozione di tutte le regole dell'access list, si potrebbe pensare di definire una regola distinta per ciascun utente della rete: in questo modo si potrebbe semplificare l'attivazione e la disattivazione dell'accesso ad Internet del singolo utente.

Si ricorda che le Extended Access List per Cisco sono numerate da 100 a 199 e che pertanto ce ne sono al massimo 100.

Ovviamente si dovrà fare in modo di aggiungere le corrispondenti istruzioni **ip access-group** per associare le access-list all'interfaccia di uscita del router.