THE UNIVERSITY
OF ADELAIDE
AUSTRALIA

Restricting access to websites for walk in users

Author: *Corey Wallis, Systems Librarian,*
University of Adelaide Library

Last Update: *9/10/2006 10:27 AM*

Abstract: *Describes the use of a Perl script to generate a list of domains for websites that can be accessed by walk in users of the Library.*

Problem

The University of Adelaide Library makes available a number of networked computers for use by Library patrons. These computers do not require a login and are therefore available to any user who "walks in" to the Library. This type of user is typically known as a "walk in user".

The main aim of these computers is to provide access to a limited set of resources that include:

1. The Library catalogue;
2. Websites listed in the Library catalogue; and
3. Electronic resources that the Library has a subscription to.

To achieve this aim it is therefore necessary to restrict the websites that can be accessed using these computers. A proxy auto-config file is used to restrict access by walk in users to a specific set of domains. More information on the format of this file can be found at the Netscape website listed in the references section of this document.

There are in excess of 88,000 links in the catalogue representing over 3,000 distinct domain names. With the large number of links, and more being added daily, it is not feasible to manage a proxy auto-config file manually. Therefore a Perl script was developed to generate the file, based on a small number of guidelines.

Solution

To automate the generation of the proxy auto-config file a small Perl script was developed. The script is comprised of the following five files:

1. walkin_pac_gen.pl – The Perl script that does all of the work;

2. walkin.pac.deny.txt – A list of domains of electronic resources that should not be made available to walk in users due to license restrictions;

3. walkin.pac.domains.txt – A list of domains that should be made available to walk in users, that are not listed in the Library catalogue;

4. walkin.pac.header – A file containing the header of the proxy auto-config file that is static and does not change often; and

5. walkin.pac.tlds.txt – A list of Top Level Domains, TLDs, for websites that should be made available to walk in users. For example all walk in users should be able to access Australian Government websites which have the .gov.au TLD.

Each file that lists domains can include comments about why the domain is included in the list, as well as any other notes that may be required.

Access to websites outside the University network is only possible via the University proxy servers. Therefore the guidelines that underpin the script are as follows.

1. If the requested domain ends in adelaide.edu.au, and is therefore on the University network, access the website directly and do not use the proxy server;

2. If the requested domain ends in one of the listed TLDS, access the website via the University proxy server;

3. If the requested domain ends in a domain retrieved from the catalogue, or listed in the walkin.pac.domains.txt file, access the website via the University proxy server; and

4. If the requested domain is not listed in the proxy auto-config file attempt to access the website directly. This access attempt will fail as all access to websites outside the University network must go through the University proxy servers.

Samples of the four supporting files are provided in the following sections.

Sample - walkin.pac.deny.txt

```
# A list of domains that should not be included in the walkin.pac file
# Place each domain on a new line
# To ensure correct operation, start each domain with a "."
# Lines beginning with a "#" are comments and are ignored
#
.westlaw.com
.example.com
.another-example.org
```

Sample - walkin.pac.domains.txt

```
# A list of domains that should included in the walkin.pac file
# These domains are typically required to access a resource but
# are not listed in the catalogue.
# Place each domain on a new line
# To ensure correct operation, start each domain with a "."
# Lines beginning with a "#" are comments and are ignored
#
.amazon.com
.google.com
.google.com.au
```

Sample - walkin.pac.header

```
function FindProxyForURL(url, host)
{
    //Deal with local Adelaide Uni addresses
    if (dnsDomainIs(host, "adelaide.edu.au"))
    {
        return "DIRECT";
    }
```

Sample - walkin.pac.tlds.txt

```
# A list of top level domains for inclusion in the walkin.pac file
# Place each top level domain on a new line.
# To ensure correct operation, start each domain with a "."
# Lines beginings with a "#" are comments and are ignored.
#
# Education Domains
.edu
.edu.au
.edu.tw
.ac.at
.ac.de
.ac.nz
.ac.be
.ac.uk
.ac.jp
# Government Domains
.gov
.gov.au
.gov.tw
.govt.nz
.gov.uk
.go.jp
```

Copy of the walkin_pac_gen.pl script

Note: This Perl script accesses the Voyager Oracle database and therefore uses the VGER Perl Module. This module is explained in the “A trivial Perl module improving Oracle access from Perl” document available from the University of Adelaide Digital Library.

```
#!/m1/shared/bin/perl
use File::Copy;
use DBI;

# Add the UALS lib dir to the list of lib dirs
# Use the VGER module from the UALS lib dir
use lib "/m1/uals/lib";
use VGER;

# Define global constants
$lib_dir = "/m1/uals/lib";
$header_file = $lib_dir . "/walkin.pac.header";
$tld_file = $lib_dir . "/walkin.pac.tlds.txt";
$other_domains_file = $lib_dir . "/walkin.pac.domains.txt";
$no_walkin_file = $lib_dir . "/walkin.pac.deny.txt";
$production_file = "/tmp/walkin.pac";
$email_from = 'xxx.yyy@adelaide.edu.au';
$email_to = 'xxx.yyy@adelaide.edu.au';
$sendmail = "/usr/lib/sendmail -t";

# Break up a string based on a set delimiter and reverse the elements
# e.g. "adelaide.edu.au" becomes "au.edu.adelaide"
sub reverse_string {
    if (@_ != 3) {
        print "WARNING! &reverse_string expects exactly three arguments.\n";
        return undef;
    } else {
        my ($string, $sep, $concat) = @_;
        my ($sep_to_join);
        @string_parts = split /$sep/, $string;
        @string_parts = reverse @string_parts;
        $sep_to_join = $sep;
        $sep_to_join =~ s/\\(.)/$1/g;
        $new_string = join $sep_to_join, @string_parts;

        if ($concat == 1) {
            if ($string =~ /$sep$/) {

                 $new_string = $sep_to_join . $new_string;
            }
        }

        return $new_string;
    }
}

# Remove any domains from the list that start with one of the domains that
# prohibit walk in access defined in the file $no_walkin
sub remove_no_walkin {

        my (@list) = @_;
        my (@saved, @no_walkin, $test);

        #read in contents of file
        if ( ! open NO_WALKIN, "<", $no_walkin_file) {
            die "Cannot open file: $!";
        }

        @no_walkin = <NO_WALKIN>;
        close NO_WALKIN;

        chomp(@no_walkin);

        # Remove any of the comment lines from the array
        while(@no_walkin) {
            $item = shift(@no_walkin);

            if($item !~ /^#/) {
                push(@saved, $item);
            }
        }

        @no_walkin = @saved;
        @saved = ();

        #reverse top level domains
        foreach $domain (@no_walkin) {
            $domain = &reverse_string($domain, "\\.", 1);
        }

        @no_walkin = sort @no_walkin;

        # Remove any domains from the list matching one of the domains for
        # resources that do not allow walk in users
        while(@no_walkin > 0) {
            $test = shift(@no_walkin);

            while(@list) {
                $item = shift (@list);

                if ($item !~ /^$test/) {
                    push(@saved, $item);
                }
            }

            @list = sort @saved;
            @saved = ();
        }

        return @list;
}

# Remove any domains from the list that start with one of the TLDs
# defined in the file $tld_file
sub remove_top_level {

        my (@list) = @_;
        my (@saved, @top_level, $test);

        #read in contents of file
        if ( ! open TOP_LEVEL, "<", $tld_file) {
            die "Cannot open file: $!";
        }
        @top_level = <TOP_LEVEL>;
        close TOP_LEVEL;

        chomp(@top_level);

        # Remove any of the comment lines from the array
        while(@top_level) {
            $item = shift(@top_level);

            if($item !~ /^#/) {
                push(@saved, $item);
            }
        }

        @top_level = @saved;
        @saved = ();

        #reverse top level domains
        foreach $domain (@top_level) {
            $domain = &reverse_string($domain, "\\.", 1);
        }

        @top_level = sort @top_level;

        # Remove any domains from the list starting with one of the
        # defined TLD's
        while(@top_level > 0) {
            $test = shift(@top_level);

            while(@list) {
                $item = shift (@list);

                if ($item !~ /^$test/) {
                    push(@saved, $item);
                }
            }

            @list = sort @saved;
            @saved = ();
        }

        return @list;
}

# Remove any domains that have a common stem
sub remove_duplicates {

    my (@list) = @_;
    my (@saved, @temp, $test, $item, $backup, $match_found, $short_stem);

    $match_found = 0;
    $short_stem = 0;

    # Take out the IP addresses first
    # They don't go through the common stem process well
    while(1) {
        $item = shift (@list);

        if ($item =~ /^(\d{1,3}\.){3}\d{1,3}$/) {
            push(@saved, $item)
        } else {
            unshift (@list, $item);
            last;
        }
    }

    # Get the first test case
    $test = shift(@list);

    # Remove any domains that have a common stem
    while(@list > 0) {

        # If a match has not been found,
        # Conver the domain to a stem if required
        if ($match_found == 0) {

            @temp = split /\./, $test;
            $backup = $test;
            # Most top level domains like .com .org .csiro etc
            if (@temp >= 3 && length $temp[1] > 3 && $temp[1] ne "info") {
                $short_stem = 1;
                $test = $temp[0] . "." . $temp[1] . ".";
            # Most of the other domains like .edu.au .com.au etc
            }elsif(@temp >= 3 && length $temp[2] >= 3) {
                $short_stem = 1;
                $test = $temp[0] . "." . $temp[1] . "." . $temp[2] . ".";
            }
        }

        # Get an item to test with, and make sure we actually got one
        # If we reach the end of the array at this point we can get an
        # undef value and not a domain
        #
        # If we do get an undef value, save the last test variable
        $item = shift(@list);

        if (!defined($item)) {
            if ($short_stem == 1) {
                push(@saved, $test);
                last;
            } else {
                push(@saved, $backup);
            }
        }

        # If there is a match, continuing test
        # If there is not a match, save the test variable and get a new one
        if ($item =~ /^\Q$test\E/) {
            $match_found = 1;
        } else {
            $match_found = 0;
            if ($short_stem == 1) {
                push(@saved, $test);
                $short_stem = 0;
                $test = $item;
            } else {
                push(@saved, $backup);
                $test = $item;
            }
        }

        # If we reach the end of the list at this point.
        # Save the test variable
        if (@list == 0) {
           if ($short_stem == 1) {
               push(@saved, $test);
               last;
           } else {
               push(@saved, $backup);
           }
        }
    }

    # Remove the www from the front of the domains
    # Remember the domains are reversed at this point!
    foreach $domain (@saved) {
        $domain =~ s/www$/\./;
        $domain =~ s/www\.$/\./;
    }

    return @saved;
}

# A helpful little subroutine to print out the values in an array
# Used during development debugging
sub print_list {
    print "\n";
    my (@list) = @_;
    foreach $domain (@list) {
        print "Domain:\t$domain\n";
    }
    print "\n";
}

# Get the list of domains to process
# Either from a file, or the DB
sub get_domain_list {
    my ($todo) = @_;
    my (@full_list, @other_domains);

    if ($todo eq "file") {
        # We need to operate on a file
        if ( ! open FULL_LIST, "<", "./cw_elink_hosts.txt") {
            die "Cannot open file: $!";
        }

        @full_list = <FULL_LIST>;
        close FULL_LIST;
        chomp(@full_list);
    } else {
        # We need to get info from the DB
        # Connect to the Oracle Database
        my $dbh = DBI->connect("dbi:Oracle:host=$db_host;sid=$db_sid",
                $db_username,
                $db_password
           ) or die "Could not connect: $DBI::errstr";

        # Prepare the SQL statement
        my $sth = $dbh->prepare(qq|
                SELECT DISTINCT(LOWER(elink_index.url_host)) AS hosts
                FROM elink_index
                WHERE elink_index.record_type = 'B'
                AND elink_index.url_host IS NOT NULL
                OR elink_index.record_type='M'
                AND elink_index.url_host IS NOT NULL
                ORDER BY hosts|
           ) or die $dbh->errstr;

        # Execute the statement
        $sth->execute or die $dbh->errstr;

        # Get the results
        while ($host = $sth->fetchrow_array()) {
            push(@full_list, $host);
        }

        # Close the statement and database handlers
        $sth->finish;
        $dbh->disconnect;
    }

    # We need to add any other domains to the list
    # read in contents of file
    if ( ! open OTHER, "<", $other_domains_file) {
        die "Cannot open file: $!";
    }

    @other_domains = <OTHER>;
    close OTHER;

    chomp(@other_domains);

    # Remove any of the comment lines from the array
    while(@other_domains) {
        $item = shift(@other_domains);

        if($item !~ /^#/) {
            push(@saved, $item);
        }
    }

    @other_domains = @saved;

    push (@full_list, @other_domains);
    return @full_list;
}

# Write the new pac file
sub write_pac_file {
    my (@list) = @_;
    my (@top_level, @saved);
    my ($proxy) = "return \"PROXY www-proxy.adelaide.edu.au:8080\";";

    # Write a production walkin.pac file
    # Copy the template to the production name
    copy($header_file, $production_file)
        or die "Unable to copy $header_file to $production_file: $!";
    # Get the top level domains
    if ( ! open TOP_LEVEL, "<", $tld_file) {
        die "Cannot open file: $!";
    }

    @top_level = <TOP_LEVEL>;
    close TOP_LEVEL;

    chomp(@top_level);

    # Remove any of the comment lines from the array
    while(@top_level) {
        $item = shift(@top_level);

        if($item !~ /^#/) {
            push(@saved, $item);
        }
    }

    @top_level = @saved;
    @saved = ();

    # Open the production file for output
    if ( ! open OUTPUT, ">>", $production_file) {
        die "Cannot open file: $!";
    }

    # Output the list of top level domains
    print OUTPUT "\n";

    $domain = shift(@top_level);
    print OUTPUT "    if (dnsDomainIs(host, \"$domain\") ||\n";

    while(@top_level) {
        $domain = shift(@top_level);
        if (@top_level != 0) {
            print OUTPUT "        dnsDomainIs(host, \"$domain\") ||\n";
        } else {
            print OUTPUT "        dnsDomainIs(host, \"$domain\"))\n";
        }
    }

    print OUTPUT "    {\n";
    print OUTPUT "        $proxy\n";
    print OUTPUT "    }\n\n";

    # Output the list of other domains
    $domain = shift(@list);
    print OUTPUT "    if (dnsDomainIs(host, \"$domain\") ||\n";

    while(@list) {
        $domain = shift(@list);
        if (@list != 0) {
            print OUTPUT "        dnsDomainIs(host, \"$domain\") ||\n";
        } else {
            print OUTPUT "        dnsDomainIs(host, \"$domain\"))\n";
        }
    }

    print OUTPUT "    {\n";
    print OUTPUT "        $proxy\n";
    print OUTPUT "    }\n\n";

    # Write the end of the file
    print OUTPUT "    return \"DIRECT\";\n";
    print OUTPUT "}";

    # Close the file
    close OUTPUT;
}

# Email the walkin.pac file to somebody
sub email_file {

    # Read in contents of the production walkin.pac file
    if ( ! open PROD, "<", $production_file) {
        die "Cannot open file: $!";
    }

    my(@walkin_pac) = <PROD>;
    close PROD;

    # Send the email
    open(SENDMAIL, "|$sendmail") or die "Cannot open $sendmail: #!";
    print SENDMAIL "From: $email_from\n";
    print SENDMAIL "Subject: New Walkin Pac file\n";
    print SENDMAIL "To: $email_to\n";
    print SENDMAIL "Content-type: text/plain\n\n";

    foreach $line (@walkin_pac) {
        print SENDMAIL $line;
    }

    close SENDMAIL;
}

# Main program block
# Get the list of domains
@domain_list = &get_domain_list("oracle");

# Reverse & sort the domains
foreach $domain (@domain_list) {
    $domain = &reverse_string($domain, "\\.", 0);
}
@domain_list = sort @domain_list;

# Remove any domains matching the list prohibiting walkin users
@domain_list = &remove_no_walkin(@domain_list);

# Remove any domains matching the list of TLDs
@domain_list = &remove_top_level(@domain_list);

# Remove domains with a common stem
@domain_list = &remove_duplicates(@domain_list);

# Put domains back into the right order
foreach $domain (@domain_list) {
    $domain = &reverse_string($domain, "\\.", 0);
}
```

```
@domain_list = sort @domain_list;

# Write the new walkin pac file
&write_pac_file(@domain_list);

# Email the new walkin.pac file
&email_file();
#print "Process Complete!\n";
```

References

Proxy auto-config file format
http://wp.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html

A trivial Perl module improving Oracle access from Perl – Available via the University of Adelaide Digital Library
http://hdl.handle.net/2440/14785