



```
#!/usr/bin/env perl # Stefan Tomanek stefan@pico.ruhr.de # updated by Wolfram Sang ninja@the-dreams.de on 21.10.06 and on 26.06.07 ## # pcf2vpnc <pcf file> [vpnc file] ## # This program is free software; you can redistribute it and/or modify # it under the terms of the GNU General Public License as published by # the Free Software Foundation; either version 2 of the License, or # (at your option) any later version. # # This program is distributed in the hope that it will be useful, # but WITHOUT ANY WARRANTY; without even the implied warranty of # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the # GNU General Public License for more details. # # You should have received a copy of the GNU General Public License # along with this program; if not, write to the Free Software # Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA # # $Id$ use IO::File; use strict; use warnings;
```

```
my %authmode = ( 1 => 'psk', 3 => 'cert', 5 => 'hybrid' ); my $needs_cert = 0; my $no_decrypt = 0; if (system("cisco-decrypt", "q") == -1) {
```

```
    $no_decrypt = 1;
    print STDERR "cisco-decrypt not in search path,\n";
    print STDERR "    adding passwords in obfuscated form\n";
```

```
}
```

```
sub readPCF($) {
```

```
    my ($file) = @_;
    my %config;
    while (<$file>) {
# Filter unnecessary chars at beginning & end of line
s/^\!*\///; s/[\r ]*$\///;
if (/^(.*?)(=.*?)$/) {
    # We don't need empty config strings
    next if ($2 eq "");
    if ($1 eq "Host") {
        $config{IPSec}{gateway} = $2;
    } elsif ($1 eq "GroupName") {
        $config{IPSec}{ID} = $2;
    } elsif ($1 eq "GroupPwd") {
        $config{IPSec}{secret} = $2;
    } elsif ($1 eq "enc_GroupPwd") {
        if ($no_decrypt) {
            $config{IPSec}{obfuscated} = "secret $2";
        } else {
            $config{IPSec}{secret} = `cisco-decrypt $2`;
        }
    } elsif ($1 eq "AuthType") {
        $config{IKE}{Authmode} = $authmode{$2};
        if ($2 == 3 || $2 == 5) {
            $needs_cert = 1;
        }
    } elsif ($1 eq "DHGroup") {
        $config{IKE}{DH} = "Group dh$2";
    } elsif ($1 eq "Username") {
        $config{Xauth}{username} = $2;
    } elsif ($1 eq "UserPassword") {
        $config{Xauth}{password} = $2;
    } elsif ($1 eq "enc_UserPassword") {
        if ($no_decrypt) {
            $config{Xauth}{obfuscated} = "password $2";
        } else {
            $config{Xauth}{password} = `cisco-decrypt $2`;
        }
    } elsif ($1 eq "NTDomain") {
        $config{Domain}{""} = $2;
    }
}
}
return \%config;
```

```
}
```

```
sub writeVPNC($) {
```



```
my ($config) = @_;  
my $text = "## generated by pcf2vpnc\n";  
foreach my $section (keys %$config) {  
foreach my $item (keys %{ $config->{$section} }) {  
    $text .= "$section ".$item ? "$item " : '' ).$config->{$section}{$item}."\n";  
}  
}  
unless (defined $config->{Xauth}) {  
$text .= "\n## To add your username and password,\n";  
$text .= "## use the following lines:\n";  
$text .= "# Xauth username <your username>\n";  
$text .= "# Xauth password <your password>\n";  
}  
return $text;
```

}

if (defined \$ARGV[0]) {

```
my $src = new IO::File($ARGV[0]) || die "Unable to open file ".$ARGV[0]."\n";  
if (defined $ARGV[1]) {  
my $dst = new IO::File($ARGV[1], "w") || die "Unable to open file ".$ARGV[1]."\n";  
$dst->write( writeVPNC(readPCF($src)) ) || die "Unable to write to file ".$ARGV[1]."\n";  
$dst->close() || die "Unable to close file ".$ARGV[1]."\n";  
printf STDERR "vpnc config written to '%s' with permissions '%04o'.\n", $ARGV[1],  
(stat($ARGV[1]))[2];  
print STDERR "Please take care of permissions.\n";  
} else {  
print writeVPNC(readPCF($src));  
}  
$src->close() || die "Unable to close file ".$ARGV[0]."\n";  
if ($needs_cert) {  
print STDERR "\nDon't forget to copy the needed certificate(s).\n\n";  
}  
}
```

} else {

```
print STDERR "$0 converts VPN-config files from pcf to vpnc-format.\n";  
print STDERR "Usage: $0 <pcf file> [vpnc file]\n";
```

}