

## spamproc.pl

```
# spamproc.pl - Top Spam Senders Report Processor
# Revision 1.0 - Nov 21 2008
# Revision 1.1 - Dec 05 2008
# Revision 1.2 - Dec 15 2008

# NOTE: This software program is provided as-is without warranty. Use at your own risk.

# This Perl script was tested using ActivePerl 5.10 on x86 with proper environment variables set.

# Script Installation:
# 1) Copy spamproc.pl to an empty folder X.
# 2) Create empty subfolders of X named: NewReports, ProcessedReports, and Logs.

# Input:
# 1) In NewReports folder, one or more text files with .txt extensions,
# representing Top Spam Sender report emails Saved As Text files.
# After processing all .txt files in NewReports folder are moved to
# the ProcessedReports folder.
# 2) In folder X, text output from the spam filter Blocked IP Addresses
# bulk edit saved under filename blockedips.log.
# (CSV Format: IP Address, Netmask, Action, Comment)
# 3) In folder X, text output from spam filter Blocked Sender Domains
# bulk edit saved under filename blockednames.log.
# (CSV Format: Domain/Subdomain Name, Action, Comment)
# 4) The spamproc.pl script takes an optional threshold value as a command-line
# argument passed by the process_spam.bat. If no threshold value is specified
# or a non-integer value is specified, then the threshold is set to zero.
# The script will use the threshold to ignore sender domains whose total spam
# count is less than the threshold value.

# Output:
# 1) In Logs folder, consrptYMD_HMS.log contains the consolidated top spam
# senders report in text format. This consolidates all of the Top Spam Sender
# Reports found in the NewReports folder into one report. The consolidated
# report sorts out duplicate and new IP addresses and domains as checked
# the Barracuda log files (blockedips.log, blockednames.log).
# 2) In Logs folder, newipsYMD_HMS.log contains entries for new IP addresses with
# unknown domains in comma separated values format that can be appended to
# the bulk edit in the spam filter software Blocked IP Addresses.
# 3) In Logs folder, newnamesYMD_HMS.log contains entries for new domains in
# comma separated values format that can be appended to the bulk edit in
# the spam filter Blocked Sender Domains.
#
# Note: YMD_HMS denotes the system date/time of when spamproc.pl was executed.

# Running the script:
# 1) Copy one or more Top Spam Senders report emails in text format to
# the NewReports folder.
```

```

                                spamproc.pl
#       2) Update the input files: blockedips.log, blockednames.log to match the
#       the current state from the spam filter.
#       3) Run the process_spam.bat file, modified with the correct path names. Or
#       run the script from the command line. See the batch file for the steps
#       needed to run the spamproc.pl.
#       4) Review the output files.
#       5) If needed, edit the newipsYMD_HMS.log and newnamesYMD_HMS.log files.
#       Then append contents to appropriate bulk edit in the spam filter user interface.
#

```

use File::Copy;

```

# first get and format the timestamp as YMD_HMS
($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time);
$year +=1900;
$mon +=1;
$mytime = sprintf("%02d", $hour) . "h" . sprintf("%02d", $min) . "m" . sprintf("%02d", $sec) . "s";
$mydate = $year . sprintf("%02d", $mon) . sprintf("%02d", $mday);
$sptimestamp = "$mydate\_mytime";

print "\n=====\\n";
print "Top Spam Senders Report Processor executed on: $sptimestamp\\n";

# get the domain spam count threshold from the command line
$numArgs = $#ARGV + 1;

if ($numArgs == 0) {
    $threshold = 0;
}
else {
    if ($ARGV[0] =~ /^-?\d+$/ ) {
        # we have a number use it as the threshold
        $threshold = $ARGV[0];
    }
    else {
        $threshold = 0;
        print "WARNING: Illegal Threshold Value Found on Command Line.\\n";
    }
}

print "\nThe domain spam count threshold is: $threshold.\\n";

# input log files, located in current folder
$ipfile = "blockedips.log";
$namefile = "blockednames.log";

```

## spamproc.pl

```
# output log files to be written to Logs folder contain timestamp in file name
$rptlogfile = "consrpt" . $sptimestamp . ".log";
$newipslog = "newips" . $sptimestamp . ".log";
$newnameslog = "newnames" . $sptimestamp . ".log";

# initialize hash arrays
%consrpt = ();
%rptstarts = ();
%blockedips = ();
%blockednames = ();
%dupips = ();
%newips = ();
%dupnames = ();
%belowthresh = ();
%newnames = ();
%newipsentry = ();
%newnamesentry = ();

# open the blocked by ip log and save it in a hash array
open(IPLOG, "<$ipfile") or die "Can't open Blocked IPs input file $ipfile: $!\n";

while ($rptline = readline(IPLOG)) {
    # skip the header line and process all other lines
    if ($rptline !~ /^IP/) {
        # save the IP address as the key and the Subnet mask as the value
        @values = split(',', $rptline);
        $blockedips{$values[0]} = $values[1];
    }
}
close(IPLOG);

$nkeys = keys %blockedips;
print "Read $nkeys records from Blocked IPs file: $ipfile...\n";

# open the blocked sender domains log from and save it in a hash array
open(DOMLOG, "<$namefile") or die "Can't open Blocked Sender Domains input file $namefile: $!\n";

while ($rptline = readline(DOMLOG)) {
    # skip the header line and process all other lines
    if ($rptline !~ /^Domain/) {
        # just save the domain which is located before the first comma
        # the key and value in %blocked names will be equal
        @values = split(',', $rptline);
        $blockednames{$values[0]} = $values[0];
    }
}
}
```

spamproc.pl

```
close(DOMLOG);

$nkeys = keys %blockednames;
print "Read $nkeys records from Blocked Sender Domains file: $namefile...\n";

# get a list of all the spam report files
chdir("NewReports") or die "Couldn't change directory to NewReports folder: $!\n";
@rpts = glob("*.txt");

# process each spam report file and create the consolidated report data, weeding out duplicates
foreach $spamrpt (@rpts) {
    open(RPTFILE, "<$spamrpt") or die "Can't open file $spamrpt: $!";
    print "Processing file: $spamrpt...\n";

    $gottime = 0;
    $gotstart = 0;
    $dataline = 0;

    while ($rptline = readline(RPTFILE)) {
        # strip report line of trailing blanks and newlines
        $rptline =~ s/\s+$//;

        # skip empty lines in file
        if ($rptline) {
            # find the report date and save it in reports hash with file name

            if ($rptline =~ /^Start:/) {
                $gottime = 1;
                $rptline =~ s/^Start://; # get rid of the text at beginning
                $rptline =~ s/^\s+//; # and then get rid of leading spaces
                # save the report name and start time for further processing
                $rptstarts{$spamrpt} = $rptline;
            }

            # if we have the start time of report start looking for the spam data
            if ($gottime == 1) {
                # find the start of report data, parse out domain, ip, and count
                # the report item data is on 3 consecutive lines
                if ($gotstart == 1) {
                    # read all data lines for this item
                    if ($dataline == 0) {
                        # we discard the rank as this is report specific
                        $dataline++;
                    }
                }
                elsif ($dataline == 1) {
                    # save the domain-ip key for later processing
                    $domainip = $rptline;
                    $dataline++;
                }
            }
        }
    }
}
```



```

                                spamproc.pl
if ($blockedips{$myip}) {
    # we found this record in the log
    # save it in the duplicate ips list for consolidated report
    $dupips{$mydomip} = $mycount;
    $gotdup = 1;
}

# then process all subnets of IP address
if ($gotdup == 0) {
    @ipparts = split('\.', $myip);
    # build all possible subnets from this ip
    $ipsubnet1 = "$ipparts[0].0.0.0";
    $ipsubnet2 = "$ipparts[0].$ipparts[1].0.0";
    $ipsubnet3 = "$ipparts[0].$ipparts[1].$ipparts[2].0";

    if ($snmask = $blockedips{$ipsubnet1}) {
        # we found a match for the first subnet
        # now check if the subnet mask is correct
        if ($snmask eq "255.0.0.0") {
            $dupips{$mydomip} = $mycount;
            $gotdup = 1;
        }
    }

    if (($gotdup == 0) && ($snmask = $blockedips{$ipsubnet2})) {
        # we found a match for the second subnet
        # now check if the subnet mask is correct
        if ($snmask eq "255.255.0.0") {
            $dupips{$mydomip} = $mycount;
            $gotdup = 1;
        }
    }

    if (($gotdup == 0) && ($snmask = $blockedips{$ipsubnet3})) {
        # we found a match for the second subnet
        # now check if the subnet mask is correct
        if ($snmask eq "255.255.255.0") {
            $dupips{$mydomip} = $mycount;
            $gotdup = 1;
        }
    }
}

if (($gotdup == 0) && ($mydom eq "unknown")) {
    # the domain is UNKNOWN and
    # we didn't find this record in the Blocked IPs log so
    # save it in the new ips list for consolidated report

```

```

                                spamproc.pl
$newips{$mydomain} = $mycount;

# and write it to newips hash in CSV format:
#      IP/Network Address, Netmask, Action, Comment

$newentry = "$myip, 255. 255. 255. 255, Block, $mydomain $mydate";

if ($newipentry{$newentry}) {
    # handle rare case where the entry can be duplicated
    $newipentry{$newentry} += $mycount;
}
else {
    $newipentry{$newentry} = $mycount;
}
}

if (($gotdup == 0) && ($mydomain != "unknown")) {
    # we didn't find the IP address in Blocked IPs log
    # and the domain is something other than UNKNOWN
    # so process by domain

    if ($gotdup == 0) {
        if ($blockednames{$mydomain}) {
            # we found this exact domain in the log
            # save it in the duplicate names list for consolidated report
            $dupnames{$mydomain} = $mycount;
            $gotdup = 1;
        }
    }

    # if no exact match check each subdomain level for a match in the log
    if ($gotdup == 0) {
        # split the spam report domain name into subdomain parts
        @domparts = split('\.', $mydomain);

        # get count number of domain name parts
        $numsubdom = scalar(@domparts);

        $numrevdom = 0;

        # reverse the order of the domain parts and build the subdomain strings
        # we will check up to three levels of matching
        while (scalar(@domparts) > 0) {
            $revdom[$numrevdom] = pop(@domparts);
            $numrevdom++;
        }
    }
}

```

```

                                spamproc.pl
if ($numrevdom > 0) {
    # check rightmost subdomain level against blockednames.log
    if ($blockednames{$revdom[0]}) {
        # found this subdomain, mark as duplicate
        $dupnames{$mydomain} = $mycount;
        $gotdup = 1;
    }

    # check second from right subdomain level against blockednames.log
    if (($gotdup == 0) && ($numrevdom > 1)) {
        $mysub = "$revdom[1].$revdom[0]";
        if ($blockednames{$mysub}) {
            # found this subdomain, mark as duplicate
            $dupnames{$mydomain} = $mycount;
            $gotdup = 1;
        }
    }

    # check third from right subdomain level against blockednames.log
    if (($gotdup == 0) && ($numrevdom > 2)) {
        $mysub = "$revdom[2].$revdom[1].$revdom[0]";
        if ($blockednames{$mysub}) {
            # found this subdomain, mark as duplicate
            $dupnames{$mydomain} = $mycount;
            $gotdup = 1;
        }
    }
}
else {
    # program execution error write to STDOUT
    print "Unable to parse domain: $mydomain. Please process
manual ly. \n";
}

if ($gotdup == 0) {
    # ignore everything below the threshold
    if ($mycount < $threshold) {
        # count is below threshold
        # save it in the duplicate names list for consolidated report
        $belowthresh{$mydomain} = $mycount;
    }
    else {
        # we didn't find this record in the log and its above the threshold
        # save it in the new domains list for consolidated report

```

```

        spamproc.pl
        $newnames{$mydomip} = $mycount;

        # and write it to newnames hash in CSV format:
        #         IP/Network Address, Netmask, Action, Comment

        $newentry = "$mydom,Block,$myip,$mydate";
        if ($newnamesentry{$newentry}) {
            # handle rare case where the entry can be duplicated
            $newnamesentry{$newentry} += $mycount;
        }
        else {
            $newnamesentry{$newentry} = $mycount;
        }
    }
}

}

}

chdir("../Logs") or die "Couldn't change directory to Logs folder: $!\n";

# write the newips output file
open(NEWIPLOG, ">$newipslog") or die "Can't open New Unknowns Log file $newipslog: $!\n";
print "Writing new IPs output file to $newipslog...\n";

# sort data by entry in ascending order
@sortkeys = sort { $a cmp $b } keys %newipsentry;

foreach $mykey (@sortkeys) {
    print NEWIPLOG "$mykey\n";
}

close(NEWIPLOG);

# write the newnames output file
open(NEWDOMLOG, ">$newnameslog") or die "Can't open New Domains Log file $newnameslog: $!\n";
print "Writing new domains output file to $newnameslog...\n";

# sort data by entry in ascending order
@sortkeys = sort { $a cmp $b } keys %newnamesentry;

foreach $mykey (@sortkeys) {
    print NEWDOMLOG "$mykey\n";
}

close(NEWDOMLOG);

```

spamproc.pl

```
# finally write the consolidated report log
open(RPTLOG, ">$rptlogfile") or die "Can't open Report log file $rptlogfile: $!\n";
print "Writing consolidated report to $rptlogfile...\n";

# write the consolidated report to a file
print RPTLOG "==> Consolidated Top Spam Senders Report <==\n";
print RPTLOG "=====\n";
print RPTLOG "Report creation timestamp: $sptimestamp\n";

print RPTLOG "\nThe domain spam count threshold is: $threshold.\n";

print RPTLOG "\n\n$nfiles Top Spam Sender Reports Processed: \n\n";

print RPTLOG "Timestamp          \tFilename\n";
print RPTLOG "-----\t-----\n";
# sort data by date/time ascending order
@sortkeys = sort { $rptstarts{$a} cmp $rptstarts{$b} } keys %rptstarts;

foreach $mykey (@sortkeys) {
    print RPTLOG "$rptstarts{$mykey}\t$mykey\n";
}

print RPTLOG "\n\nNEW UNKNOWNNS SAVED IN $newipslog: \n\n";

print RPTLOG "Count\tDomain[IP]\n";
print RPTLOG "-----\t-----\n";

# sort data by count descending order
@sortkeys = sort { $newips{$b} <=> $newips{$a} } keys %newips;

foreach $mykey (@sortkeys) {
    print RPTLOG "$newips{$mykey}\t$mykey\n";
}

print RPTLOG "\n\nNEW DOMAINS SAVED IN $newnameslog: \n\n";

print RPTLOG "Count\tDomain[IP]\n";
print RPTLOG "-----\t-----\n";

# sort data by count descending order
@sortkeys = sort { $newnames{$b} <=> $newnames{$a} } keys %newnames;

foreach $mykey (@sortkeys) {
    print RPTLOG "$consrpt{$mykey}\t$mykey\n";
}
```

```

                                spamproc.pl
print RPTLOG "\n\nDUPLICATE IPS DISCARDED FROM CONSOLIDATED DATA: \n\n";

print RPTLOG "Count\tDomain[IP]\n";
print RPTLOG "-----\t-----\n";

# sort data by count descending order
@sortkeys = sort { $dupips{$b} <=> $dupips{$a} } keys %dupips;

foreach $mykey (@sortkeys) {
    print RPTLOG "$dupips{$mykey}\t$mykey\n";
}

print RPTLOG "\n\nDUPLICATE DOMAINS DISCARDED FROM CONSOLIDATED DATA: \n\n";

print RPTLOG "Count\tDomain[IP]\n";
print RPTLOG "-----\t-----\n";

# sort data by count descending order
@sortkeys = sort { $dupnames{$b} <=> $dupnames{$a} } keys %dupnames;

foreach $mykey (@sortkeys) {
    print RPTLOG "$dupnames{$mykey}\t$mykey\n";
}

print RPTLOG "\n\nBELOW THRESHHOLD DOMAINS DISCARDED FROM CONSOLIDATED DATA: \n\n";

print RPTLOG "Count\tDomain[IP]\n";
print RPTLOG "-----\t-----\n";

# sort data by count descending order
@sortkeys = sort { $belowthresh{$b} <=> $belowthresh{$a} } keys %belowthresh;

foreach $mykey (@sortkeys) {
    print RPTLOG "$belowthresh{$mykey}\t$mykey\n";
}

print RPTLOG "\n\n==> END OF REPORT FOR $sptimestamp <==\n";

close(RPTLOG);

# finally move the spam report files from NewReports to ProcessedReports folder
chdir("../NewReports") or die "Couldn't change directory to NewReports folder: $!\n";
foreach $spamrpt (@rpts) {
    move($spamrpt, ("..\ProcessedReports\\" . $spamrpt))
        or die "Can't move file $spamrpt to NewReports folder: $!\n";
}

```

spamproc.pl

```
    }  
    # go back to original directory  
    chdir("../");  
}  
print "DONE!\n";  
# end program
```