配置与脚本的电子邮件通知IDS戒备的使用CiscoWorks Monitoring Center for Security

目录

简介

当事件规则被触发时，安全监视器有能力发送电子邮件通知。能在电子邮件通知内使用每个事件的内藏的变量不包括事例如签名ID，警报的源和目的，等等。本文提供您能使用配置安全监视器包括这些变量的说明(和许多)在电子邮件通知消息内。

先决条件

要求

本文档没有任何特定的要求。

使用的组件

本文档不限于特定的软件和硬件版本。然而，请务必使用根据什么的适当的Perl脚本传感器版本在您的环境运行。

规则

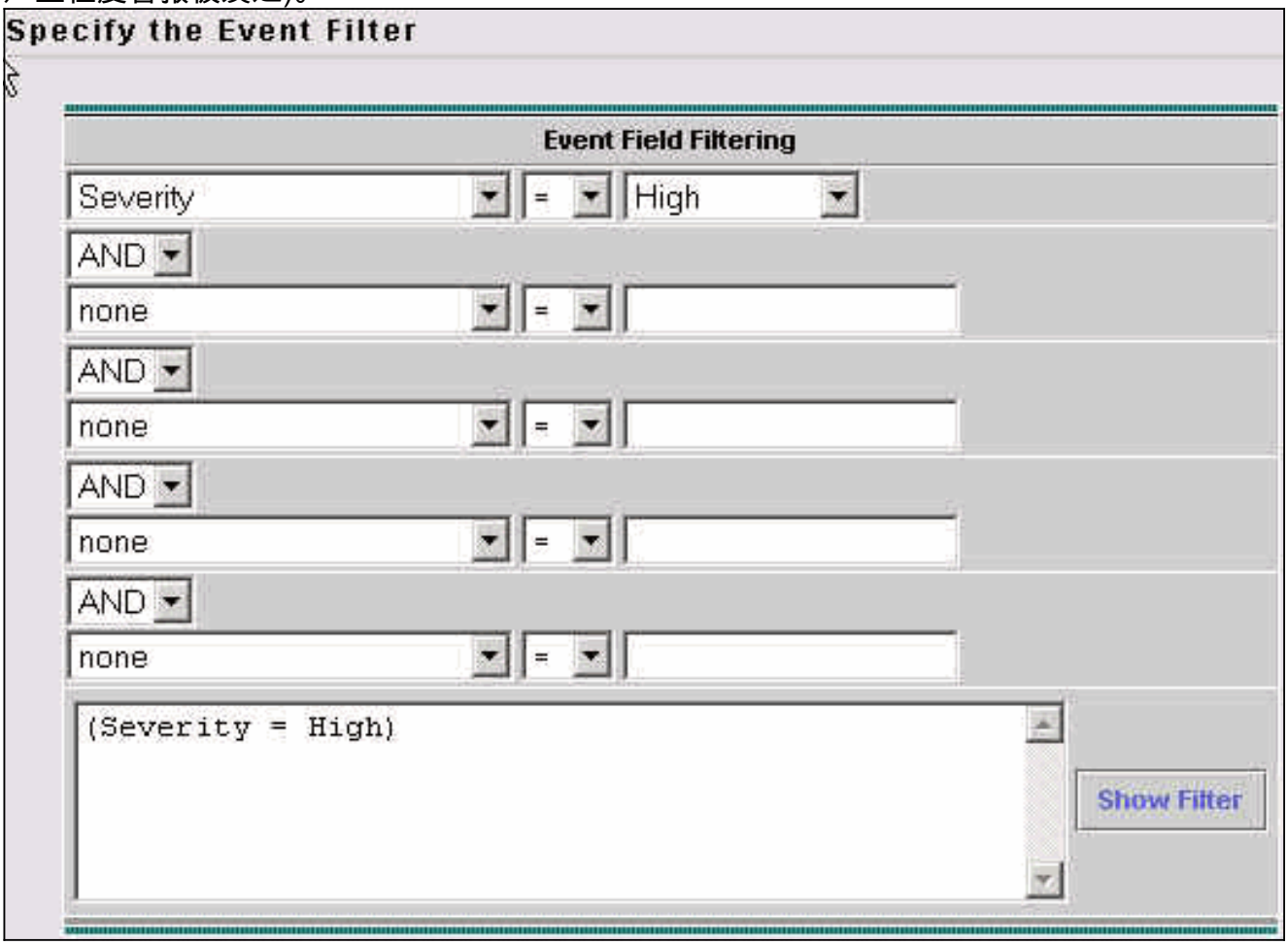
有关文档规则的详细信息，请参阅 Cisco 技术提示规则。

电子邮件通知配置程序

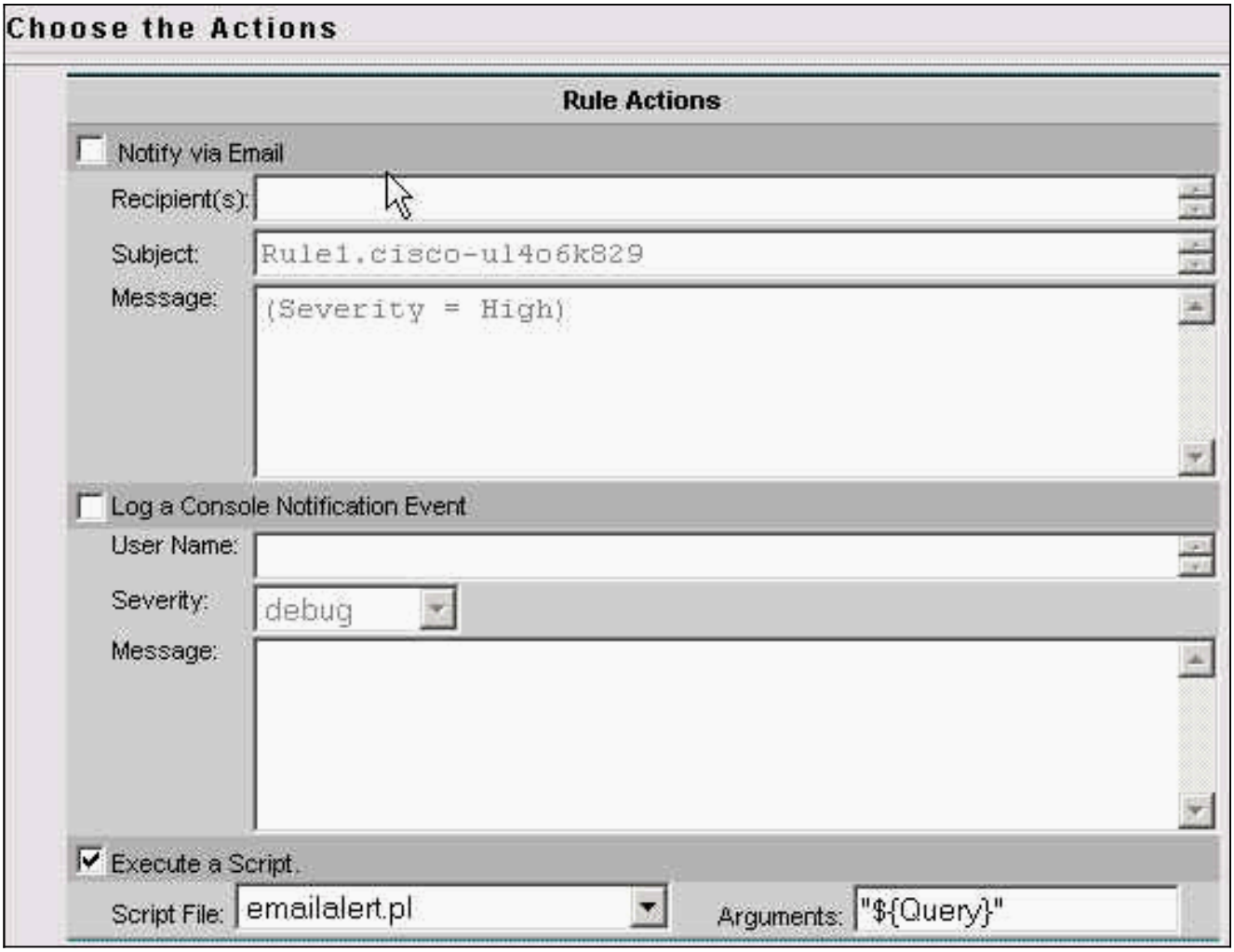
使用此步骤配置电子邮件通知。

注意： 为了发送电子邮件到正确电子邮件地址，请务必更改在脚本的电子邮件地址。

1. 复制这些脚本之一到$BASE \ CSCOpx \ MDC \等\ ID \脚本目录在VPN/安全管理解决方案(VM)服务器。当您定义了事件规则时，这允许您选择它以后在进程。保存脚本作为**emailalert.pl。注意：** 如果使用一不同的名称，请保证您在这些步骤定义的事件规则命名的参考。对于版本3.x传感器，请使用3.x传感器脚本对于版本4.x传感器，请使用4.x传感器脚本对于版本5.x传感器，请使用5.x传感器脚本如果有传感器版本的组合，思科推荐您升级，以便他们全都在同一个版本级别。这是因为仅一这些脚本可以随时运行。
2. 脚本包含解释每个部分和所有需要的输入的注释。特别是，请修改`$EmailRcpt`变量(在文件的顶部附近)是将收到警报人的电子邮件地址。
3. 定义在安全监视器内的一个事件规则调用一个新的Perl脚本。从主要安全监视器页，请选择**Admin > Event Rules**并且添加一个新的事件。
4. 在指定事件过滤器窗口，添加过滤器您要触发电子邮件警报(在此处示例，电子邮件为所有高严重程度警报被发送)。



5. 在选择操作窗口，检查方框执行脚本和选择从下拉框的脚本名。
6. 在Arguments部分，输入**"$ {查询}"**如显示此处。**注意：** 必须正确地输入这，当在这里，包括双引号。它也区分大小写。



7. 当一警报，如对您的事件过滤器定义(在本例中，高严重程度警报)时接收，呼叫emailalert.pl的脚本呼叫与$ {}参数包含关于警报的其他信息。脚本解析所有独立的字段并且使用呼叫的程序“咩咩叫”发送电子邮件对最终用户。
8. blat是在windows系统用于的免费软件电子邮件程序发送从批处理文件或Perl脚本的电子邮件。它包括作为VMS安装一部分在$BASE \ CSCOpx \二进制文件目录。为了验证您的路径设置，打开在VMS服务器的一个命令提示符窗口和类型请**咩咩叫**。如果收到错误，请复制blat.exe文件到winnt\system32目录或者查找它并且从查找的目录打开它。为了安装此，请运行：

   `blat -install <SMTP server address> <source email address>` 一旦此程序安装，您执行。

脚本

这些是在配置程序的step1是指的脚本：

- 3.x传感器脚本
- 4.x传感器脚本
- 5.x传感器脚本

3.x传感器脚本

请使用此脚本版本3.x传感器。

3.x传感器

```
#!/usr/bin/perl
#*****************************************************
***************
#
# FILE NAME : emailalert.pl
#
# DESCRIPTION : This file is a perl script that will be
executed as an
# action when an IDS-MC Event Rule triggers, and will
send an
# email to $EmailRcpt with additional alert parameters
(similar to
# the functionality available with CSPM notifications)
#
# NOTE:   this script only works with 3.x sensors,
alarms from 4.0
#         sensors are stored differently and cannot be
represented
#         in a similar format.
#
# NOTE:   check the "system" command in the script for
the correct
#         format depending on whether you're using
IDSMC/SecMon
#         v1.0 or v1.1, you may need the "-on" command-
line option.
#
# NOTE :  This script takes the ${Query} keyword from
the
#         triggered rule, extracts the set of alarms
that caused
#         the rule to trigger. It then reads the last
alarm of
#         this set, parses the individual alarm fields,
and
#         calls the legacy script with the same set of
command
#         line arguments as CSPM.
#
# The calling sequence of this script must be of the
form:
#
#         emailalert.pl "${Query}"
#
# Where:
#
#         "${Query}" - this is the query keyword
dynamically
#         output by the rule when it triggers.
#         It MUST be wrapped in double quotes when
specifying it in the Arguments
#         box on the Rule Actions panel.
#
#
#*****************************************************
***************
##
## The following are the only two variables that need
changing. $TempIDSFile can be any
## filename (doesn't have to exist), just make sure the
directory that you specify
## exists. Make sure to use 2 backslashes for each
directory, the first backslash is
## so the Perl interpretor doesn't error on the
```

```
pathname.
##
## $EmailRcpt is the person that is going to receive the
email notifications. Also
## make sure you escape the @ symbol by putting a
backslash in front of it, otherwise
## you'll get a Perl syntax error.
##
$TempIDSFile = "c:\\temp\\idsalert.txt";
$EmailRcpt = "nobody\@cisco.com";

##
## pull out command line arg
##

$whereClause = $ARGV[0];

##
## extract all the alarms matching search expression
##

$tmpFile = "alarms.out";

## The following line will extract alarms from 1.0
IDSMC/SecMon database, if
## using 1.1 comment out the line below and un-comment
the other system line
## below it.

## V1.0 IDSMC/SecMon version
system("IdsAlarms -s\"$whereClause\" -f\"$tmpFile\"");

## V1.1 IDSMC/SecMon version.
## system("IdsAlarms -on -s\"$whereClause\" -
f\"$tmpFile\"");
##

# open matching alarm output

if (!open(ALARM_FILE, $tmpFile)) {
    print "Could not open ", $tmpFile, "\n";
    exit -1;
}

# read to last line

while (<ALARM_FILE>) {
    $line = $_;
}

# clean up

close(ALARM_FILE);
unlink($tmpFile);

##
## split last line into fields
##

@fields = split(/,/, $line);

$eventType = @fields[0];
$recordId = @fields[1];
$gmtTimestamp = 0; # need gmt time_t
```

```
$localTimestamp = 0; # need local time_t
$localDate = @fields[4];
$localTime = @fields[5];
$appId = @fields[6];
$hostId = @fields[7];
$orgId = @fields[8];
$srcDirection = @fields[9];
$destDirection = @fields[10];
$severity = @fields[11];
$sigId = @fields[12];
$subSigId = @fields[13];
$protocol = "TCP/IP";
$srcAddr = @fields[15];
$destAddr = @fields[16];
$srcPort = @fields[17];
$destPort = @fields[18];
$routerAddr = @fields[19];
$contextString = @fields[20];


## Open temp file to write alert data into,

open(OUT,">$TempIDSFile") || warn "Unable to open output
file!\n";

## Now write your email notification message. You're
writing the following into
## the temporary file for the moment, but this will then
be emailed. Use the format:
##
## print (OUT "Your text with any variable name from the
list above \n");
##
## Again, make sure you escape special characters with a
backslash (note the : in between $sigId
## and $subSigId has a backslash in front of it)

print(OUT "\n");
print(OUT "Received severity $severity alert at
$localDate $localTime\n");
print(OUT "Signature ID $sigId\:$subSigId from $srcAddr
to $destAddr\n");
print(OUT "$contextString");
close(OUT);

## then call "blat" to send contents of that file in the
body of an email message.
## Blat is a freeware email program for WinNT/95, it
comes with VMS in the
## $BASE\CSCOpx\bin directory, make sure you install it
first by running:
##
## blat -install <SMTP server address> <source email
address>
##
## For more help on blat, just type "blat" at the
command prompt on your VMS system (make
## sure it's in your path (feel free to move the
executable to c:\winnt\system32 BEFORE
## you run the install, that'll make sure your system
can always find it).

system ("blat \"$TempIDSFile\" -t \"$EmailRcpt\" -s
\"Received IDS alert\"");
```

4.x传感器脚本

请使用此脚本版本4.x传感器。

4.x传感器

```
#!/usr/bin/perluse
Time::Local;#*****************************************
**************************
#
# FILE NAME : emailalert.pl
#
# DESCRIPTION : This file is a perl script that will be
executed as an
# action when an IDS-MC Event Rule triggers, and will
send an
# email to $EmailRcpt with additional alert parameters
(similar to
# the functionality available with CSPM notifications)
#
# NOTE: this script only works with 4.x sensors. It will
# not work with 3.x sensors.
#
# NOTES : This script takes the ${Query} keyword from
the
# triggered rule, extracts the set of alarms that caused
# the rule to trigger. It then reads the last alarm of
# this set, parses the individual alarm fields, and
# calls the legacy script with the same set of command
# line arguments as CSPM.
#
# The calling sequence of this script must be of the
form:
#
# emailalert.pl "${Query}"
#
# Where:
#
# "${Query}" - this is the query keyword dynamically
# output by the rule when it triggers.
# It MUST be wrapped in double quotes
# when specifying it in the Arguments
# box on the Rule Actions panel.
#
#
#*****************************************************
***************
##
## The following are the only two variables that need
changing. $TempIDSFile can be any
## filename (doesn't have to exist), just make sure the
directory that you specify
## exists. Make sure to use 2 backslashes for each
directory, the first backslash is
## so the Perl interpretor doesn't error on the
pathname.
##
## $EmailRcpt is the person that is going to receive the
email notifications. Also
## make sure you escape the @ symbol by putting a
backslash in front of it, otherwise
## you'll get a Perl syntax error.
##
```

```
$TempIDSFile = "c:\\temp\\idsalert.txt";
$EmailRcpt = "yourname\@yourcompany.com";

# subroutine to add leading 0's to any date variable
that's less than 10.
sub add_zero {
my ($var) = @_;
if ($var < 10) {
$var = "0" .$var
}
return $var;
}

# subroutine to find one or more IP addresses within an
XML tag (we can have multiple
# victims and/or attackers in one alert now).
sub find_addresses {
my ($var) = @_;
my @addresses = ();
if (m/$var/) {
$raw = $&;
while ($raw =~ m/(\d{1,3}\.){3}\d{1,3}/) {
push @addresses,$&;
$raw = $';
}
$var = join(', ',@addresses);
return $var;
}
}

# pull out command line arg

$whereClause = $ARGV[0];

# extract all the alarms matching search expression

$tmpFile = "alarms.out";

# Extract the XML alert/event out of the database.

system("IdsAlarms -s\"$whereClause\" -f\"$tmpFile\"");

# open matching alarm output

if (!open(ALARM_FILE, $tmpFile)) {
print "Could not open $tmpFile\n";
exit -1;
}

# read to last line

while (<ALARM_FILE>) {
chomp $_;
push @logfile,$_;
}

# clean up

close(ALARM_FILE);
unlink($tmpFile);

# Open temp file to write alert data into,
```

```
open(OUT,">$TempIDSFile");

# split XML output into fields

$oneline = join('',@logfile);
$oneline =~ s/\<\/events\>//g;
$oneline =~ s/\<\/evAlert\>/\<\/evAlert\>,/g;
@items = split(/,/,$oneline);

# If you want to see the actual database query result in
the email, un-comment out the
# line below (useful for troubleshooting):
# print(OUT "$oneline\n");

# Loop until there's no more alerts

foreach (@items) {

if (m/\<hostId\>(.*)\<\/hostId\>/) {
$hostid = $1;
}

if (m/severity="(.*?)"/) {
$sev = $1;
}

if (m/Zone\=".*"\>(.*)\<\/time\>/) {
$t = $1;
if ($t =~ m/(.*)(\d{9})/) {
($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) =
localtime($1);

# Year is reported from 1900 onwards (eg. 2003 is 103).
$year = $year + 1900;

# Months start at 0 (January = 0, February = 1, etc), so
add 1.
$mon = $mon + 1;

$mon = add_zero ($mon);
$mday = add_zero ($mday);
$hour = add_zero ($hour);
$min = add_zero ($min);
$sec = add_zero ($sec);
}
}

if (m/sigName="(.*?)"/) {
$SigName = $1;
}

if (m/sigId="(.*?)"/) {
$SigID = $1;
}

if (m/subSigId="(.*?)"/) {
$SubSig = $1;
}

$attackerstring = "\<attacker.*\<\/attacker";
if ($attackerstring = find_addresses ($attackerstring))
{
}
```

```
$victimstring = "\<victim.*\<\/victim";
if ($victimstring = find_addresses ($victimstring)) {
}

if (m/\<alertDetails\>(.*)\<\/alertDetails\>/) {
$AlertDetails = $1;
}

@actions = ();
if (m/\<actions\>(.*)\<\/actions\>/) {
$rawaction = $1;
while ($rawaction =~ m/\<(\w*?)\>(.*?)\</) {
$rawaction = $';
if ($2 eq "true") {
push @actions,$1;
}
}
if (@actions) {
$actiontaken = join(', ',@actions);
}
}
else {
$actiontaken = "None";
}

## Now write your email notification message. You're
writing the following into
## the temporary file for the moment, but this will then
be emailed.
##
## Again, make sure you escape special characters with a
backslash (note the : between
## the SigID and the SubSig).
##
## Put your VMS servers IP address in the NSDB: line
below to get a direct link
## to the signature details within the email.

print(OUT "\n$hostid reported a $sev severity alert at
$hour:$min:$sec on $mon/$mday/$year\n");
print(OUT "Signature: $SigName \($SigID\:$SubSig\)\n");
print(OUT "Attacker: $attackerstring ---> Victim:
$victimstring\n");
print(OUT "Alert details: $AlertDetails \n");
print(OUT "Actions taken: $actiontaken \n");
print(OUT "NSDB: https\://<your VMS server IP
address>/vms/nsdb/html/expsig_$SigID.html\n\n");
print(OUT "-----------------------------------------
-------\n");

}

close(OUT);

## Now call "blat" to send contents of the file in the
body of an email message.
## Blat is a freeware email program for WinNT/95, it
comes with VMS in the
## $BASE\CSCOpx\bin directory, make sure you install it
first by running:
##
## blat -install <SMTP server address> <source email
address>
##
```

```
## For more help on blat, just type "blat" at the
command prompt on your VMS system (make
## sure it's in your path (feel free to move the
executable to c:\winnt\system32 BEFORE
## you run the install, that'll make sure your system
can always find it).

system ("blat \"$TempIDSFile\" -t \"$EmailRcpt\" -s
\"Received IDS alert\"");
```

5.x传感器脚本

请使用此脚本版本5.x传感器。

5.x传感器

```
#!/usr/bin/perl
use Time::Local;

#*****************************************************
***************
#
#  FILE NAME   : emailalertv5.pl
#
#  DESCRIPTION : This file is a perl script that will be
executed as an
#                action when an IDS-MC Event Rule
triggers, and will send an
#                email to $EmailRcpt with additional
alert parameters (similar to
#                the functionality available with CSPM
notifications)
#
#                NOTE: this script only works with 5.x
sensors.
#
#  NOTES       : This script takes the ${Query} keyword
from the
#                triggered rule, extracts the set of
alarms that caused
#                the rule to trigger.  It then reads the
last alarm of
#                this set, parses the individual alarm
fields, and
#                calls the legacy script with the same
set of command
#                line arguments as CSPM.
#
#                The calling sequence of this script
must be of the form:
#
#                emailalert.pl "${Query}"
#
#                Where:
#
#                   "${Query}"  - this is the query
keyword dynamically
#                                 output by the rule
when it triggers.
#                                 It MUST be wrapped in
double quotes
#                                 when specifying it in
```

```
the Arguments
#                                   box on the Rule
Actions panel.
#
#
#*****************************************************
***************
##
## The following are the only two variables that need
changing.  $TempIDSFile can be any
## filename (doesn't have to exist), just make sure the
directory that you specify
## exists.  Make sure to use 2 backslashes for each
directory, the first backslash is
## so the Perl interpretor doesn't error on the
pathname.
##
## $EmailRcpt is the person that is going to receive the
email notifications.  Also
## make sure you escape the @ symbol by putting a
backslash in front of it, otherwise
## you'll get a Perl syntax error.
##

$TempIDSFile = "c:\\temp\\idsalert.txt";
$EmailRcpt = "gfullage\@cisco.com";

# subroutine to add leading 0's to any date variable
that's less than 10.
sub add_zero {
  my ($var) = @_;
  if ($var < 10) {
      $var = "0" .$var
  }
  return $var;
}

# subroutine to find one or more IP addresses within an
XML tag (we can have multiple
# victims and/or attackers in one alert now).
sub find_addresses {
  my ($var) = @_;
  my @addresses = ();
  if (m/$var/) {
      $raw = $&;
      while ($raw =~ m/(\d{1,3}\.){3}\d{1,3}/) {
          push @addresses,$&;
          $raw = $';
      }
      $var = join(', ',@addresses);
      return $var;
  }
}

# pull out command line arg

$whereClause = $ARGV[0];

# extract all the alarms matching search expression

$tmpFile = "alarms.out";

# Extract the XML alert/event  out of the database.
```

```
system("IdsAlarms -os -s\"$whereClause\" -
f\"$tmpFile\"");

# open matching alarm output

if (!open(ALARM_FILE, $tmpFile)) {
  print "Could not open $tmpFile\n";
  exit -1;
}

# read to last line

while (<ALARM_FILE>) {
   chomp $_;
   push @logfile,$_;
}

# clean up

close(ALARM_FILE);
unlink($tmpFile);

# Open temp file to write alert data into,

open(OUT,">$TempIDSFile");

# split XML output into fields

$oneline = join('',@logfile);
$oneline =~ s/\<\/sd\:events\>//g;
$oneline =~
s/\<\/sd\:evIdsAlert\>/\<\/sd\:evIdsAlert\>,/g;
@items = split(/,/,$oneline);

# If you want to see the actual database query result in
the email, un-comment out the
# line below (useful for troubleshooting):
# print(OUT "$oneline\n");

# Loop until there's no more alerts

foreach (@items) {
  unless ($_ =~ /\<\/env\:Body\>/) {

    if (m/\<sd\:hostId\>(.*)\<\/sd\:hostId\>/) {
      $hostid = $1;
    }

    if (m/severity="(.*?)"/) {
      $sev = $1;
    }

    if (m/Zone\=".*"\>(.*)\<\/sd\:time\>/) {
      $t = $1;
      if ($t =~ m/(.*)(\d{9})/) {

($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) =
localtime($1);

        # Year is reported from 1900 onwards (eg. 2003
is 103).
        $year = $year + 1900;

        # Months start at 0 (January = 0, February = 1,
```

```
etc), so add 1.
        $mon = $mon + 1;

        $mon = add_zero ($mon);
  $mday = add_zero ($mday);
  $hour = add_zero ($hour);
  $min = add_zero ($min);
  $sec = add_zero ($sec);
      }
    }

    if (m/description="(.*?)"/) {
      $SigName = $1;
    }

    if (m/\ id="(.*?)"/) {
      $SigID = $1;
    }

    if (m/\<cid\:subsigId\>(.*)\<\/cid\:subsigId\>/) {
      $SubSig = $1;
    }

    if
(m/\<cid\:riskRatingValue\>(.*)\<\/cid\:riskRatingValue\
>/) {
      $RR = $1;
    }

    if (m/\<cid\:interface\>(.*)\<\/cid\:interface\>/) {
      $Intf = $1;
    }

    $attackerstring =
"\<sd\:attacker.*\<\/sd\:attacker";
    if ($attackerstring = find_addresses
($attackerstring)) {
    }

    $victimstring = "\<sd\:target.*\<\/sd\:target";
    if ($victimstring = find_addresses ($victimstring))
{
    }

    if
(m/\<cid\:alertDetails\>(.*)\<\/cid\:alertDetails\>/) {
      $AlertDetails = $1;
    }

    @actions = ();
    if (m/\<sd\:actions\>(.*)\<\/sd\:actions\>/) {
      $rawaction = $1;
      while ($rawaction =~ m/\<\w*?:(\w*?)\>(.*?)\</) {
         $rawaction = $';

         if ($2 eq "true") {
            push @actions,$1;
         }
      }
      if (@actions) {
        $actiontaken = join(', ',@actions);
      }
    }
    else {
```

```
        $actiontaken = "None";
      }

## Now write your email notification message. You're
writing the following into
## the temporary file for the moment, but this will then
be emailed.
##
## Again, make sure you escape special characters with a
backslash (note the : between
## the SigID and the SubSig).
##
## Put your VMS servers IP address in the NSDB: line
below to get a direct link
## to the signature details within the email.

    print(OUT "\n$hostid reported a $sev severity alert
at $hour:$min:$sec on $mon/$mday/$year\n");
    print(OUT "Signature: $SigName
\($SigID\:$SubSig\)\n");
    print(OUT "Attacker: $attackerstring  --->  Victim:
$victimstring\n");
    print(OUT "Alert details: $AlertDetails \n");
    print(OUT "Risk Rating: $RR,  Interface: $Intf \n");
    print(OUT "Actions taken: $actiontaken \n");
    print(OUT "NSDB: https\://sec-
srv/vms/nsdb/html/expsig_$SigID.html\n\n");
    print(OUT "----------------------------------------
-----------\n");

   }
}

close(OUT);

## Now call "blat" to send contents of the file in the
body of an email message.
## Blat is a freeware email program for WinNT/95, it
comes with VMS in the
## $BASE\CSCOpx\bin directory, make sure you install it
first by running:
##
##     blat -install <SMTP server address> <source email
address>
##
## For more help on blat, just type "blat" at the
command prompt on your VMS system (make
## sure it's in your path (feel free to move the
executable to c:\winnt\system32 BEFORE
## you run the install, that'll make sure your system
can always find it).

system ("blat \"$TempIDSFile\" -t \"$EmailRcpt\" -s
\"Received IDS alert\"");
```

验证

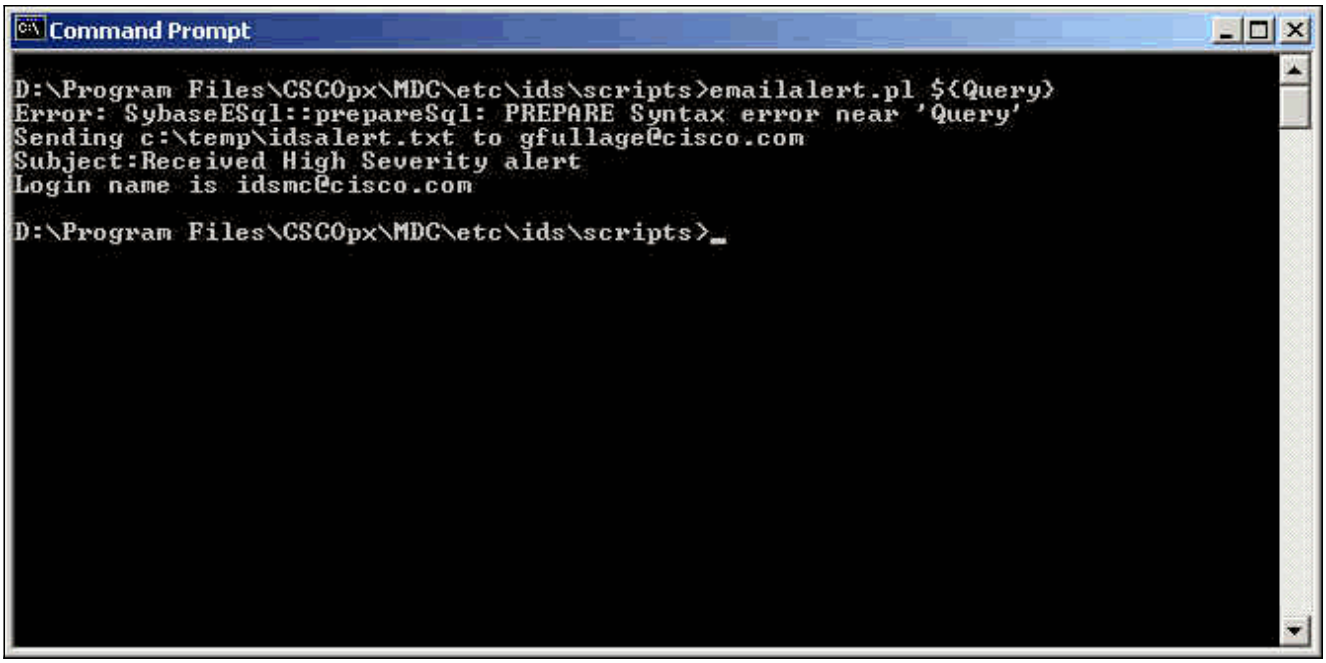
当前没有可用于此配置的验证过程。

故障排除

请按照以下说明排除配置故障。

1. 从prompt命令运行此命令为了检查适当地咩咩叫工作：

   blat <filename> -t <customer's email> -s "Test message" <filename>是完整路径到在VM系统的所有文本文件。如果电子邮件脚本处理的用户接收在电子邮件的正文的此文件，则您知道咩咩叫工作。
2. 如果电子邮件没有接收，在警报被触发后，请设法从命令提示符窗口运行Perl脚本。这选定所有Perl或路径类型问题。为了执行此，请打开prompt命令并且输入：**>cd Program Files/CSCOpx/MDC/etc/ids/scripts >emailalert.pl ${Query}** 您能潜在收到Sybase错误，类似于此示例。这归结于事实您通过的$ {}参数实际上不包含信息，不同于，当从安全监视器时通过。

   除看到此错误之外，脚本正确运行并且发送电子邮件。其中任一在电子邮件正文内的提醒的参数是空白的。如果收到任何Perl或路径错误，他们需要修复，在电子邮件被发送前。

相关信息

- Cisco Secure入侵防御支持页面
- 技术支持和文档 - Cisco Systems