Настройка уведомлений по электронной почте со скриптами для предупреждений IDS, с использованием CiscoWorks Monitoring Center for Security

Содержание

Введение

Когда Правило События инициировано, монитор безопасности имеет способность передать уведомления по электронной почте. Встроенные переменные, которые могут использоваться в рамках уведомления по электронной почте для каждого события, не включают вещи, такие как Идентификатор подписи, источник и назначение предупреждения, и т.д. Этот документ предоставляет инструкции, которые можно использовать для настройки Монитора Безопасности для включения этих переменных (и еще много) в рамках сообщения уведомления по электронной почте.

Предварительные условия

Требования

Для этого документа отсутствуют особые требования.

Используемые компоненты

Настоящий документ не имеет жесткой привязки к каким-либо конкретным версиям

программного обеспечения и оборудования. Однако обязательно используйте соответствующий сценарий Perl на основе какой Версии датчика, выполненные в вашей среде.

Условные обозначения

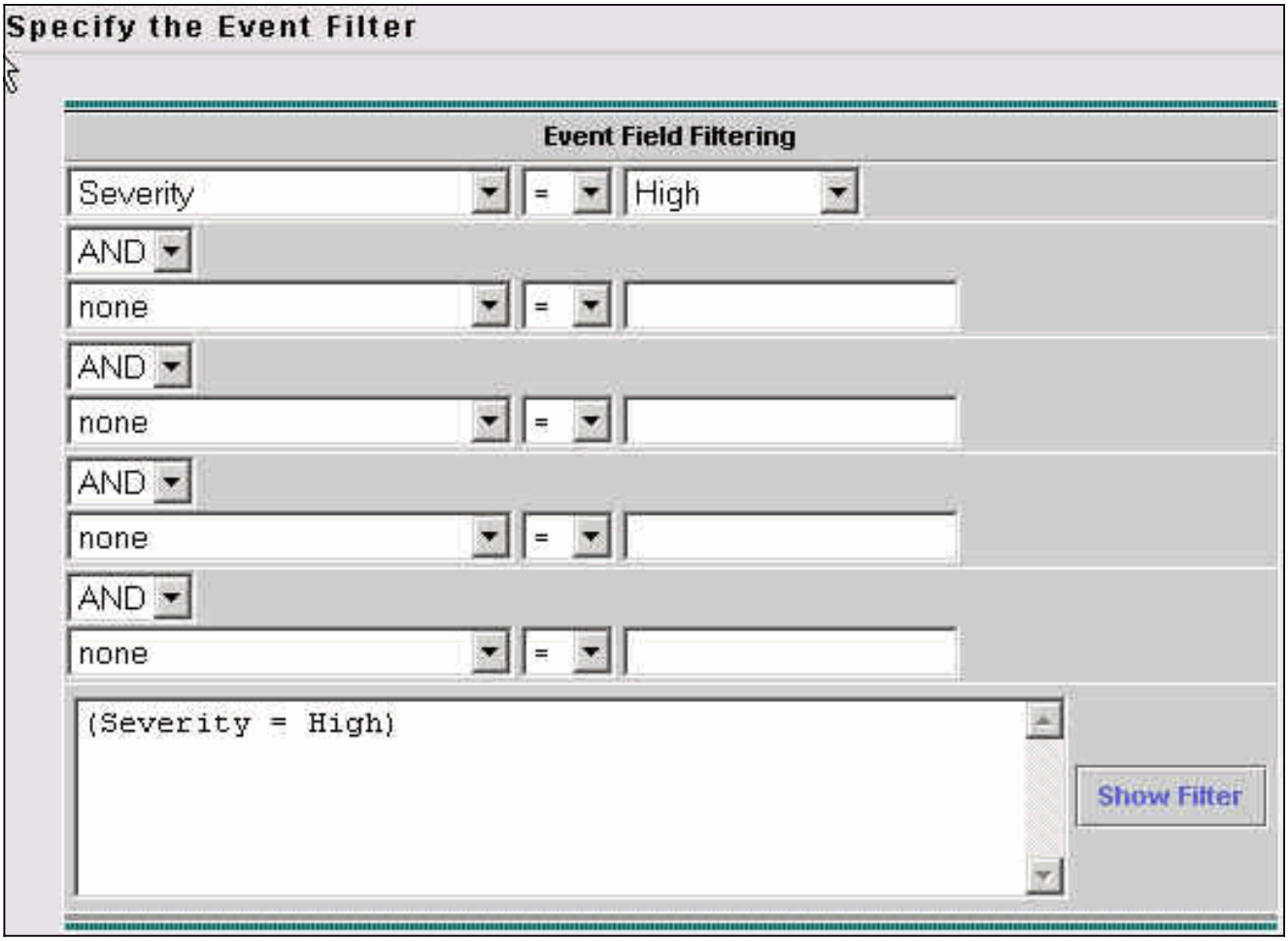
Более подробную информацию о применяемых в документе обозначениях см. в описании условных обозначений, используемых в технической документации Cisco.

Процедура настройки уведомления по электронной почте

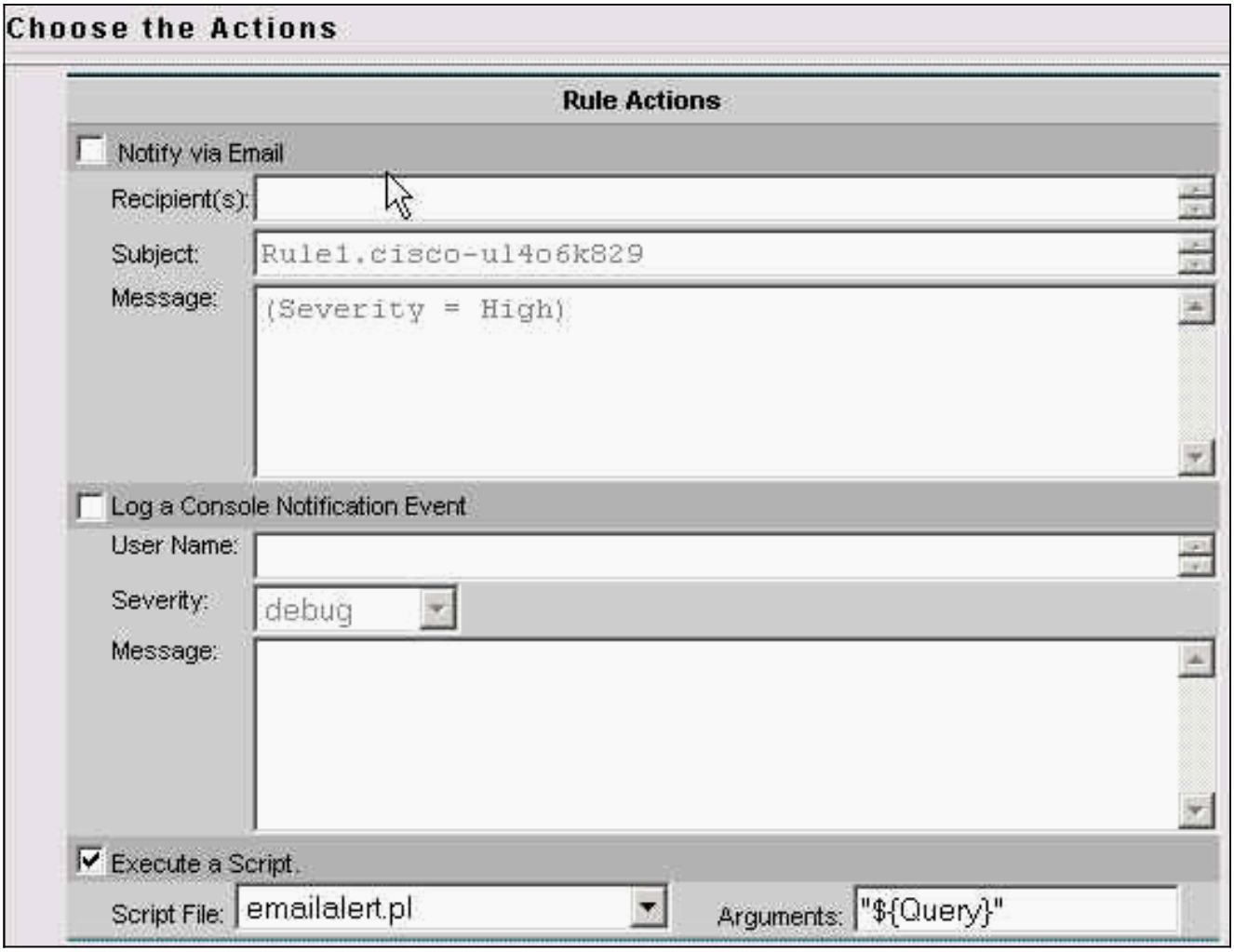
Используйте эту процедуру для настройки уведомлений по электронной почте.

Примечание: Для передачи электронного письма корректному адресу электронной почты, убедиться изменить адрес электронной почты в сценарии.

1. Скопируйте один из этих сценариев в каталог $BASE\CSCOpx\MDC\etc\ids\scripts на сервере VPN/Security Management Solution (VMS). Это позволяет вам выбрать его позже в процессе при определении правила события. Сохраните сценарий как **emailalert.pl**.**Примечание:** При использовании другое имя, гарантируете вам ссылку, которые называют в конечном счете Правило определенным в этих шагах.Для Датчиков версии 3.x используйте 3.x Сценарий ДатчиковДля Датчиков версии 4.x используйте 4.x Сценарий ДатчиковДля Датчиков версии 5.x используйте 5.x Сценарий ДатчиковЕсли у вас есть комбинация Версий датчика, Cisco рекомендует обновить так, чтобы они все были на том же уровне версии. Это вызвано тем, что только один из этих сценариев может быть выполнен в любой момент.
2. Сценарий содержит комментарии, которые объясняют каждую часть и любой требуемый ввод. В частности модифицируйте переменную `$EmailRcpt` (около вершины файла), чтобы быть адресом электронной почты человека, который должен получить предупреждения.
3. Определите Правило События в Безопасности, Прослушивают телефонный разговор новый сценарий Perl. От основной Страницы контроля безопасности выберите **Admin> Event Rules** и добавьте новое событие.
4. На окне Specify the Event Filter добавьте фильтры, что вы хотите инициировать уведомление по электронной почте (в выборке здесь, электронное письмо послано для любого предупреждения Высокого уровня важности).



5. На окне Choose the Action установите флажок, чтобы выполнить сценарий и выбрать имя сценария от раскрывающегося окна.
6. В разделе Аргументов введите **"$ {Запрос}"** как показано здесь.**Примечание:** Это должно быть введено точно, как это здесь, включая двойные кавычки. Это также чувствительно к
регистру.



7. Когда предупреждение, как определено в ваших фильтрах события (в данном примере, предупреждении высокого уровня важности) получено, сценарий, названный emailalert.pl, вызывают с аргументом `${Query}`., Это содержит дополнительные сведения о предупреждении. Сценарий анализирует все отдельные поля и использует вызванную программу, "блеют" для передачи электронного письма конечному пользователю.
8. Блейте бесплатная программа электронной почты, используемая на Системах Windows для передачи электронных писем из сценариев Perl или пакетных файлов. Это включено как часть установки VMS в каталоге $BASE\CSCOpx\bin. Для проверки параметров настройки пути откройте окно командной строки на сервере VMS, и тип **блеют**.Если вы получаете ошибку `File not found`, или копируете blat.exe файл в winnt\system32 каталог или находите его, и открывают это из каталога, в который это расположено. Для установки этого работайте:
   `blat -install <SMTP server address> <source email address>` Как только эта программа установлена, вы сделаны.

Сценарии

Это сценарии, упомянутые в шаге 1 процедуры настройки:

- 3.x сценарий датчика
- 4.x сценарий датчика
- 5.x сценарий датчика

3.x сценарий датчика

Используйте этот сценарий для Датчиков версии 3.x.

3.x Датчики

```
#!/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 сценарий датчика

Используйте этот сценарий для Датчиков версии 4.x.

4.x Датчики

```
#!/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 сценарий датчика

Используйте этот сценарий для Датчиков версии 5.x.

5.x Датчики

```
#!/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\"");
```

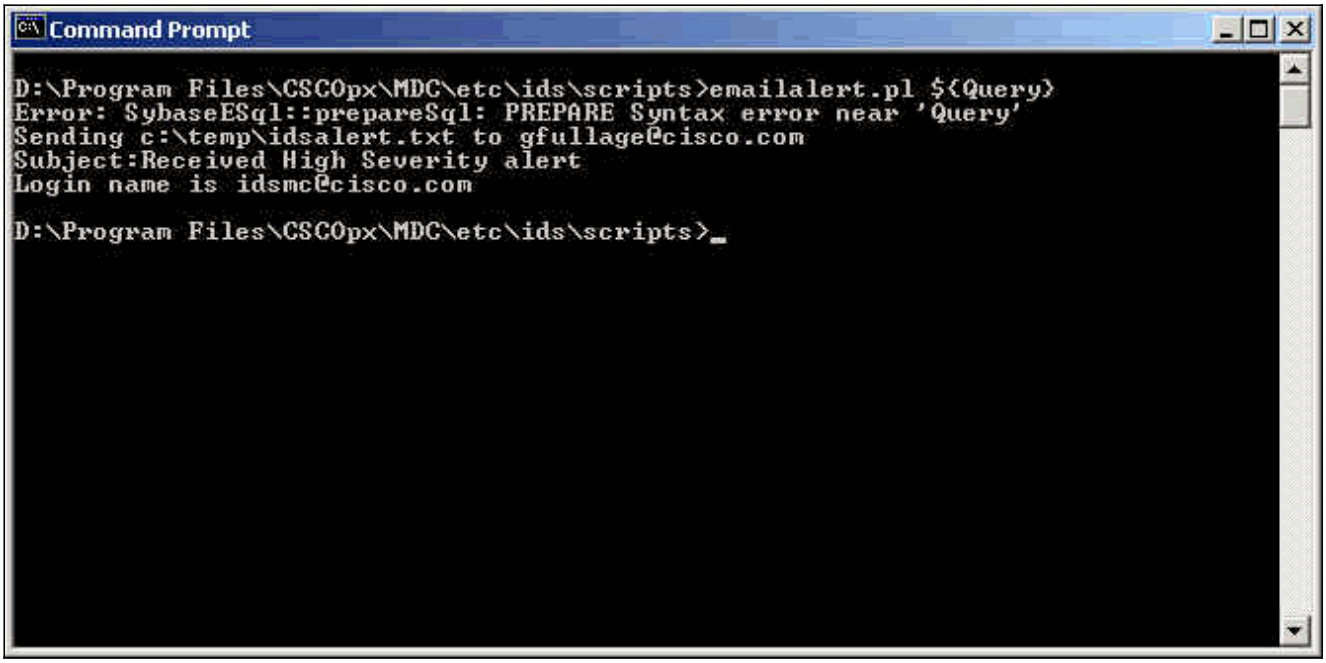
Проверка

В настоящее время для этой конфигурации нет процедуры проверки.

Устранение неполадок

Для устранения неполадок конфигурации выполните следующие действия.

1. Выполните эту команду от командной строки, чтобы проверить, что блеют, работает должным образом:
   blat <filename> -t <customer's email> -s "Test message" <filename> является полным путем к любому текстовому файлу на системе VMS. Если пользователь, к которому направлен почтовый сценарий, получает этот файл в теле электронной почты, то вы знаете, что блеют, работает.
2. Если никакое электронное письмо не получено после того, как предупреждение инициировано, попытайтесь выполнить сценарий Perl от окна командной строки.Это выделяет любой Perl или проблемы типа тракта. Чтобы сделать это, откройте командную строку и войдите:>**cd Program Files/CSCOpx/MDC/etc/ids/scripts >emailalert.pl ${Query}** Можно потенциально получить ошибку Sybase, подобную данному примеру. Это - то, вследствие того, что параметр ${Query}, который вы передаете, фактически не содержит информацию, в отличие от этого когда это проходит от Монитора Безопасности.



   Кроме наблюдения этой ошибки, сценарий выполняется правильно и посылает электронное письмо. Любые аварийные параметры в теле сообщения электронной почты являются пробелом. Если вы получаете какой-либо Perl или ошибки пути, они должны быть исправлены, прежде чем электронное письмо послано.

Дополнительные сведения

- Страница технической поддержки предотвращения вторжений Cisco Secure
- Cisco Systems – техническая поддержка и документация