# Configurar notificações de E-mail com scripts para alertas IDS usando o Centro de monitoramento CiscoWorks para segurança

## Índice



## Introdução

O monitor da Segurança tem a capacidade para enviar notificações de E-mail quando uma regra do evento é provocada. Os variáveis integrada que podem ser usados dentro da notificação de E-mail para cada evento não incluem coisas tais como o ID de assinatura, a fonte e o destino do alerta, e assim por diante. Este documento fornece instruções que você pode se usar para configurar o monitor da Segurança para incluir estas variáveis (e muito mais) dentro da mensagem da notificação de E-mail.

## Pré-requisitos

### Requisitos

Não existem requisitos específicos para este documento.

### Componentes Utilizados

Este documento não se restringe a versões de software e hardware específicas. Contudo, seja certo usar o script apropriado Perl baseado no que as versões do sensor executam em seu ambiente.

## Convenções

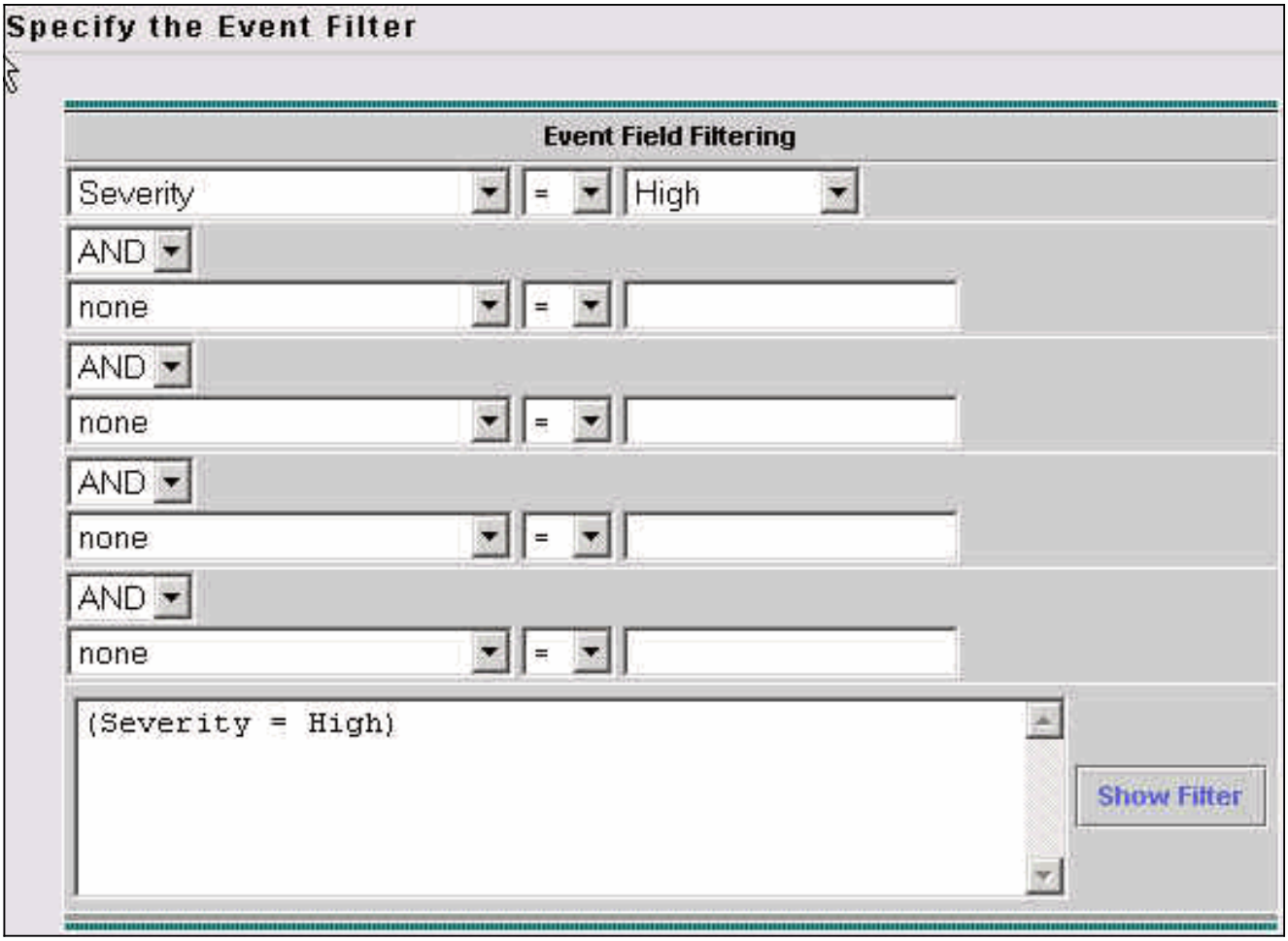
Consulte as Convenções de Dicas Técnicas da Cisco para obter mais informações sobre convenções de documentos.

# Procedimento de configuração da notificação de E-mail

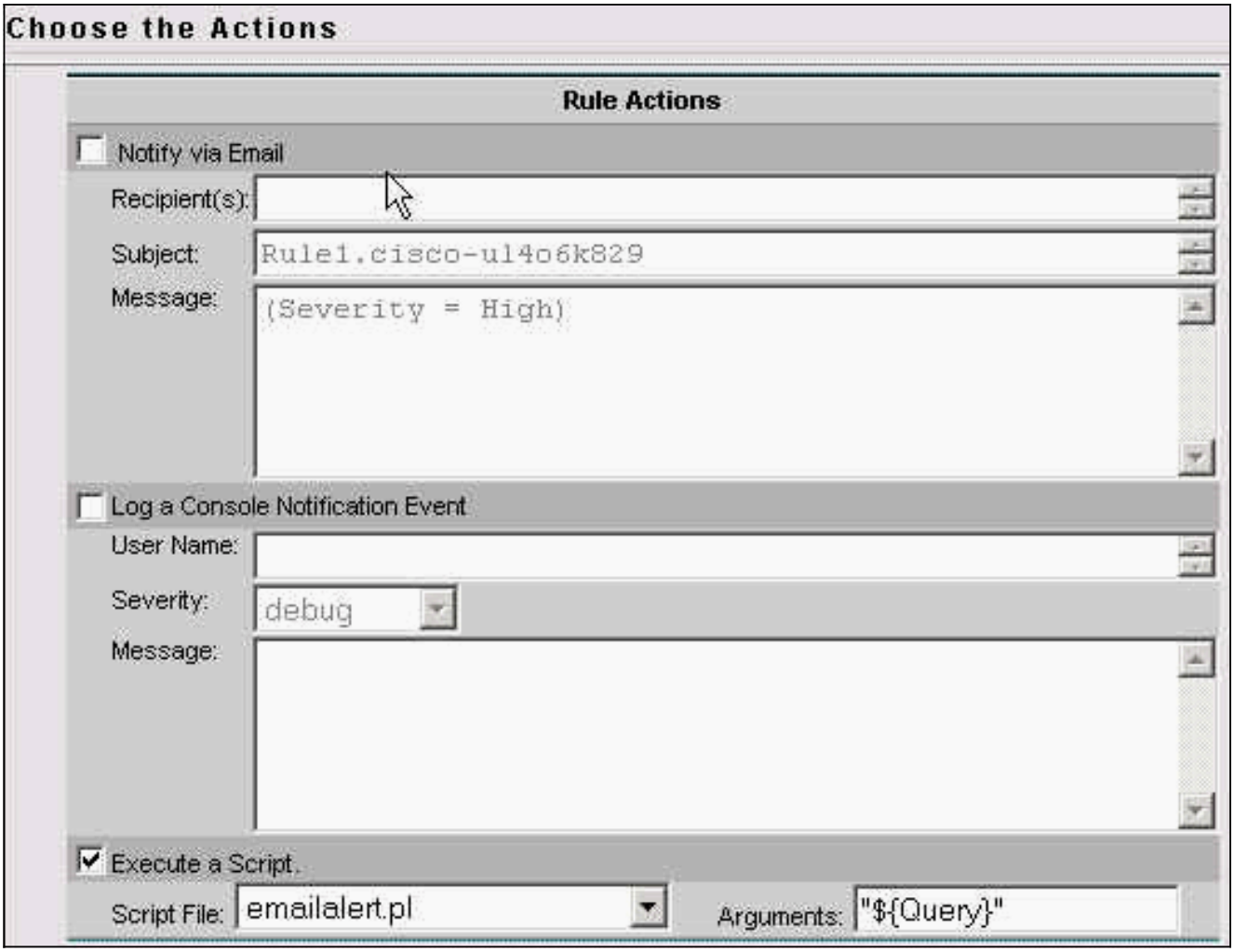
Use este procedimento para configurar notificações de E-mail.

**Nota: A fim enviar o email ao endereço email correto, seja certo mudar o endereço email no script.**

1. Copie um destes scripts no $BASE \ CSCOpx \ CDM \ etc. \ ids \ diretório dos scripts no server do VPN/Security Management Solution (VMS). Isto permite que você selecione-o mais tarde no processo quando você define uma regra do evento. Salvar o script como **emailalert.pl.Nota:** Se você usa um nome diferente, assegure-lhe a referência que nomeia na regra do evento definida nestas etapas.Para sensores da versão 3.x, use o script dos sensores 3.xPara sensores da versão 4.x, use o script dos sensores 4.xPara sensores da versão 5.x, use o script dos sensores 5.xSe você tem uma combinação de versões do sensor, Cisco recomenda-o elevação de modo que todos estejam a mesmo nível da versão. Isto é porque somente um destes passa pelo processo de script pode ser sido executado a qualquer altura.
2. O script contém os comentários que explicam cada parcela e toda a entrada exigida. Em particular, altere a variável `$EmailRcpt` (perto da parte superior do arquivo) para ser o endereço email da pessoa que deve receber os alertas.
3. Defina uma regra do evento dentro do monitor da Segurança para chamar um script novo Perl. Da página de monitor de segurança principal, escolha **Admin > regras do evento** e adicionar um evento novo.
4. Na especificação o indicador do filtro do evento, adiciona os filtros que você quer provocar o alerta do email (na amostra aqui, um email é enviado para todo o alerta da severidade elevada).




5. Na escolha o indicador da ação, verifica a caixa para executar um script e para selecionar o nome de script da caixa suspensa.
6. Nos argumentos secione, incorpore **"$ {pergunta}"** como mostrado aqui.**Nota:** Isto deve ser entrado exatamente como está aqui, incluindo as aspas duplas. É igualmente diferenciando maiúsculas e
minúsculas.




7. Quando um alerta, como definido em seus filtros do evento (neste exemplo, em um alerta da severidade elevada) é recebido, o script chamado emailalert.pl está chamado com um argumento de $ {`pergunta}. Isto` contém a informação adicional sobre o alerta. O script analisa gramaticalmente para fora todos os campos separados e usa um programa chamado “blat” para enviar um email ao utilizador final.
8. Blat é um programa de e-mail do freeware usado em sistemas Windows para enviar email dos arquivos de lote ou dos scripts Perl. É incluído como parte da instalação de VMS no $BASE \ CSCOpx \ diretório bin. A fim verificar seus ajustes do trajeto, abra uma janela de prompt de comando no servidor VMS e o tipo **blat**.Se você recebe o erro `não encontrado do arquivo`, uma ou outra cópia o arquivo blat.exe no diretório winnt\system32, ou o encontra e o abre do diretório em que está ficado situado. A fim instalar isto, seja executado: **`blat -install <SMTP server address> <source email address>`** Uma vez que este programa é instalado, você está feito.

## Scripts

Estes são os scripts referidos em etapa 1 do procedimento de configuração:

- script do sensor 3.x
- script do sensor 4.x
- script do sensor 5.x

### script do sensor 3.x

Use este script para sensores da versão 3.x.

**sensores 3.x**

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

## script do sensor 4.x

Use este script para sensores da versão 4.x.

**sensores 4.x**

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

## script do sensor 5.x

Use este script para sensores da versão 5.x.

**sensores 5.x**

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

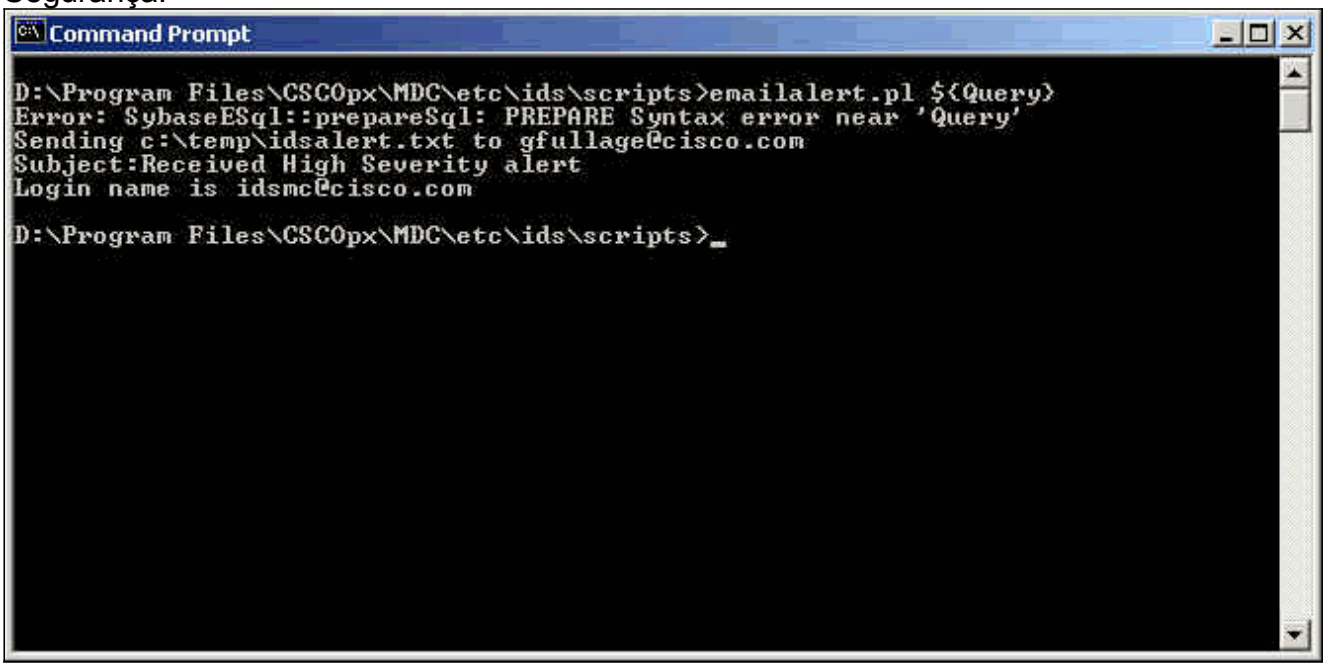
## Verificar

No momento, não há procedimento de verificação disponível para esta configuração.

## Troubleshooting

Siga estas instruções para resolver problemas da sua configuração.

1. Execute este comando de um comando prompt a fim verificar que blat trabalhos corretamente:
   **`blat <filename> -t <customer's email> -s "Test message"`** `o <filename>` é o caminho cheio a todo o arquivo de texto no sistema de VMS. Se o usuário a quem o script do email está dirigido recebe este arquivo no corpo de um email, a seguir você sabe que blat trabalhos.
2. Se nenhum email é recebido depois que um alerta está provocado, tente executar o script Perl de uma janela de prompt de comando.Isto destaca qualquer tipo edições Perl ou de trajeto. A fim fazer isto, abra um comando prompt e entre-o:`>`**`cd Program Files/CSCOpx/MDC/etc/ids/scripts >emailalert.pl ${Query}`** Você pode potencialmente receber um Erro de Sybase, similar a este exemplo. Isto é devido ao fato de que o parâmetro que `$ {pergunta}` você passa não contém realmente a informação, ao contrário de quando passa do monitor da
   Segurança.




   A não ser a vista deste erro, o script é executado corretamente e envia um email. Todos os parâmetros alertas dentro do corpo de e-mail estão vazios. Se você recebe quaisquer erros Perl ou de trajeto, precisam de ser fixados antes que um email esteja enviado.

## Informações Relacionadas

- Página de suporte segura da prevenção de intrusão de Cisco
- Suporte Técnico e Documentação - Cisco Systems