Konfigurieren von E-Mail-Benachrichtigungen mit Skripts für IDS-Warnungen mithilfe von CiscoWorks Monitoring Center for Security

Inhalt

Einführung

Security Monitor kann E-Mail-Benachrichtigungen senden, wenn eine Ereignisregel ausgelöst wird. Die integrierten Variablen, die in der E-Mail-Benachrichtigung für jedes Ereignis verwendet werden können, enthalten keine Elemente wie Signature-ID, Quelle und Ziel der Warnung usw. Dieses Dokument enthält Anweisungen, mit denen Sie Security Monitor so konfigurieren können, dass diese Variablen (und viele mehr) in die E-Mail-Benachrichtigungsmeldung aufgenommen werden.

Voraussetzungen

Anforderungen

Für dieses Dokument bestehen keine speziellen Anforderungen.

Verwendete Komponenten

Dieses Dokument ist nicht auf bestimmte Software- und Hardwareversionen beschränkt. Stellen Sie jedoch sicher, dass Sie das passende Perl-Skript verwenden, das auf den in Ihrer Umgebung ausgeführten Sensorversionen basiert.

Konventionen

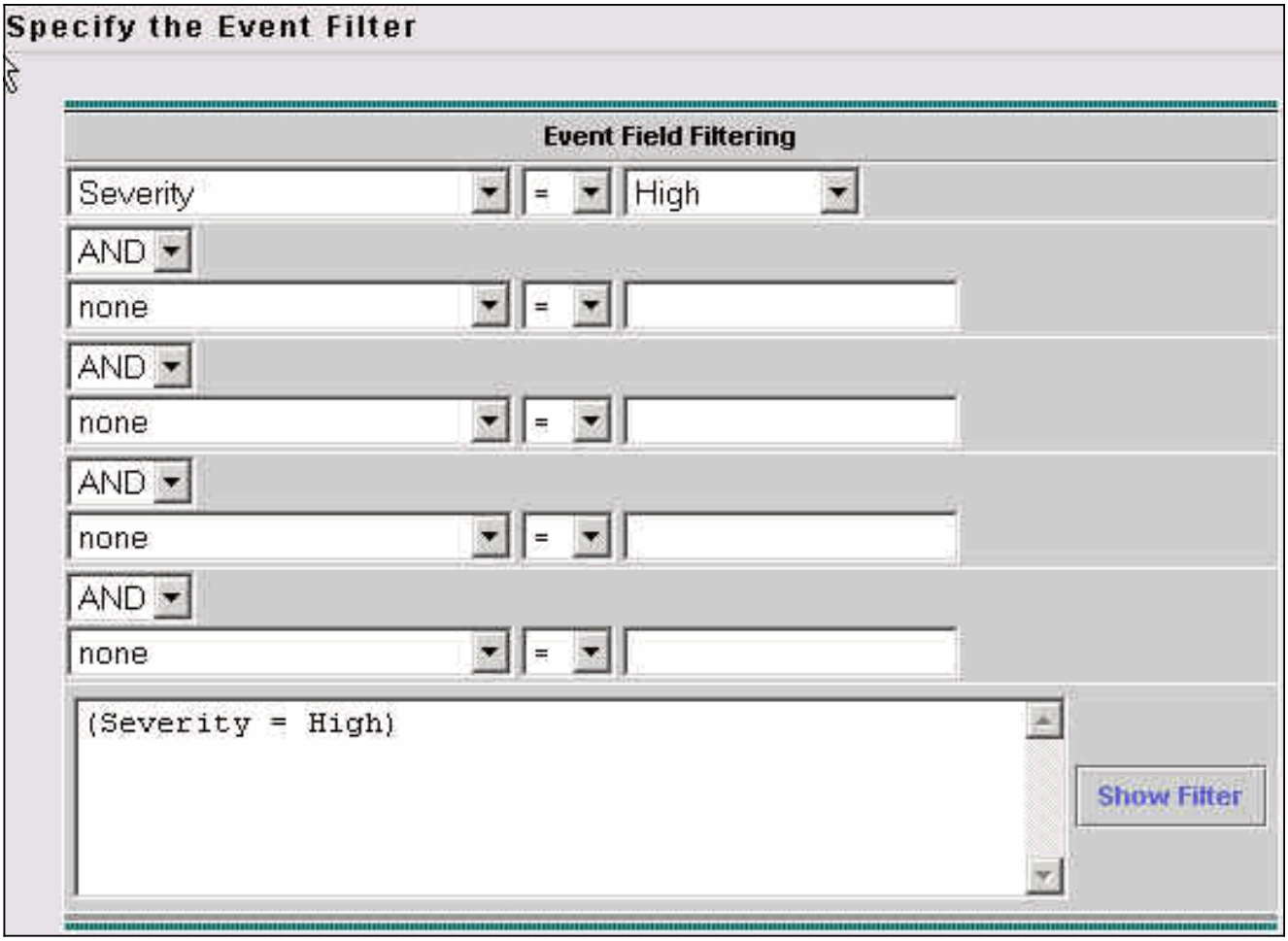
Weitere Informationen zu Dokumentkonventionen finden Sie in den Cisco Technical Tips Conventions (Technische Tipps zu Konventionen von Cisco).

Konfigurationsverfahren für E-Mail-Benachrichtigungen

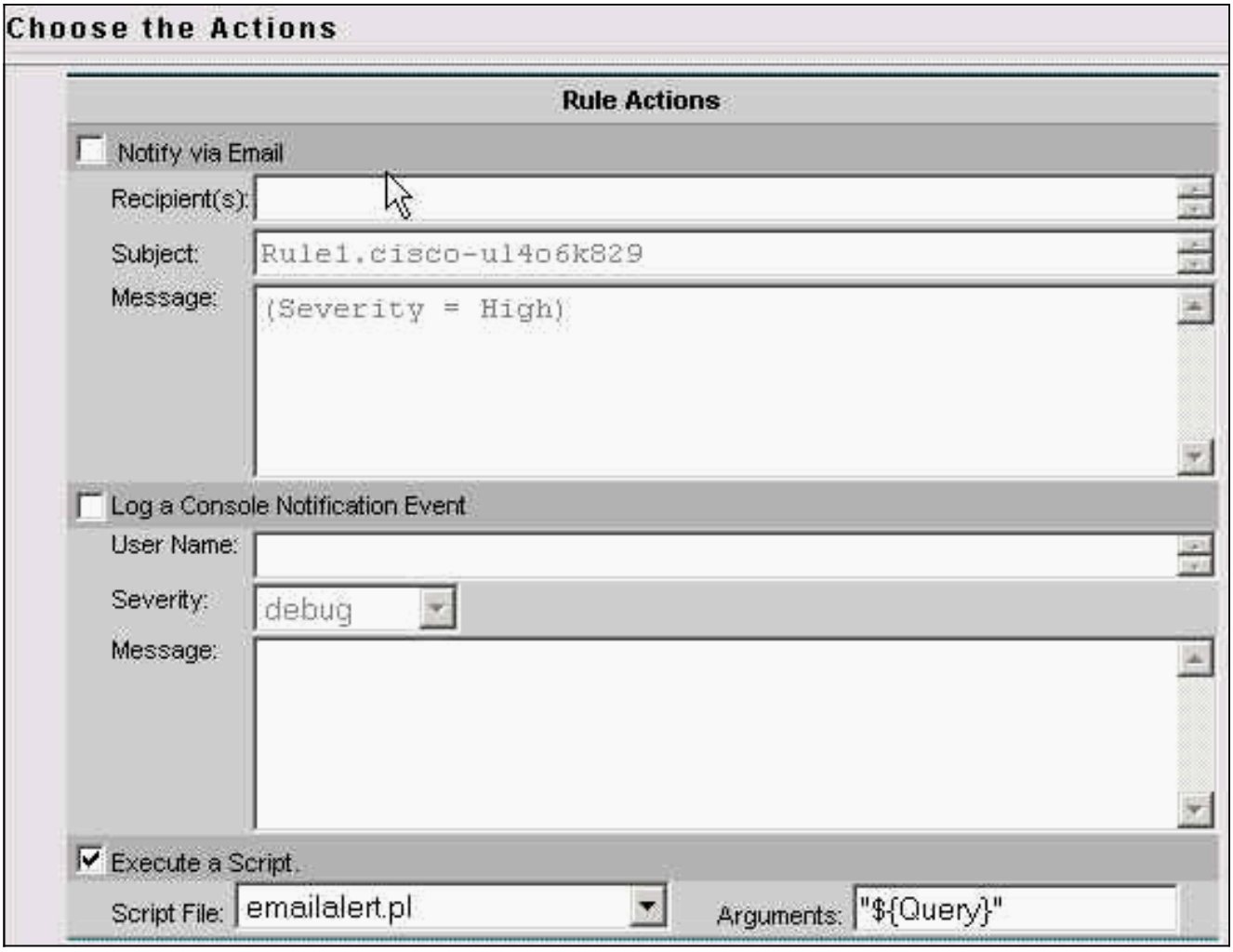
Mit diesem Verfahren konfigurieren Sie E-Mail-Benachrichtigungen.

Hinweis: Um E-Mails an die richtige E-Mail-Adresse zu senden, müssen Sie die E-Mail-Adresse im Skript ändern.

1. Kopieren Sie eines dieser Skripts in den $BASE\CSCOpx\MDC\etc\ids\scripts directory on the VPN/Security Management Solution (VMS)-Server. Dadurch können Sie es später im Prozess auswählen, wenn Sie eine Ereignisregel definieren. Speichern Sie das Skript als **emailalert.pl**.**Hinweis:** Wenn Sie einen anderen Namen verwenden, stellen Sie sicher, dass Sie in der in diesen Schritten definierten Ereignisregel auf diesen Namen verweisen.Verwenden Sie für Sensoren der Version 3.x das 3.x-Sensorskript.Verwenden Sie für Sensoren der Version 4.x das Skript für 4.x-Sensoren.Verwenden Sie für Sensoren der Version 5.x das Skript für 5.x-Sensoren.Wenn Sie über eine Kombination von Sensor-Versionen verfügen, empfiehlt Cisco ein Upgrade, damit alle auf derselben Versionsebene verfügbar sind. Dies liegt daran, dass nur eines dieser Skripts gleichzeitig ausgeführt werden kann.
2. Das Skript enthält Kommentare, in denen jeder Abschnitt und alle erforderlichen Eingaben erläutert werden. Ändern Sie insbesondere die Variable `$EmailRcpt` (in der Nähe des Dateianhangs) so, dass sie die E-Mail-Adresse der Person ist, die die Warnmeldungen empfangen soll.
3. Definieren einer Ereignisregel in Security Monitor, um ein neues Perl-Skript aufzurufen. Wählen Sie auf der Hauptseite Sicherheitsmonitor die Optionen **Admin > Event Rules (Ereignisregeln) aus,** und fügen Sie ein neues Ereignis hinzu.
4. Fügen Sie im Fenster Ereignisfilter angeben die Filter hinzu, die zum Auslösen der E-Mail-Warnung dienen sollen (im Beispiel wird hier eine E-Mail für eine Warnung mit hohem Schweregrad
gesendet).



5. Aktivieren Sie im Fenster Aktion auswählen das Kontrollkästchen, um ein Skript auszuführen, und wählen Sie im Dropdown-Feld den Skriptnamen aus.
6. Geben Sie im Abschnitt Argumente **"${Query}"** wie hier gezeigt ein.**Hinweis:** Sie müssen die doppelten Anführungszeichen wie hier eingegeben werden. Groß- und Kleinschreibung ist ebenfalls zu
beachten.



7. Wenn eine Warnung, wie in Ihren Ereignisfiltern definiert, (in diesem Beispiel eine Warnung mit hohem Schweregrad) empfangen wird, wird das Skript mit dem Namen emailalert.pl mit einem Argument von `${Query} aufgerufen.` Diese enthält zusätzliche Informationen zur Warnung. Das Skript analysiert alle separaten Felder und verwendet ein Programm namens "blat", um dem Endbenutzer eine E-Mail zu senden.
8. Blat ist ein Freeware-E-Mail-Programm, das auf Windows-Systemen zum Senden von E-Mails aus Batch-Dateien oder Perl-Skripts verwendet wird. Sie ist als Teil der VMS-Installation im $ BASE\CSCOpx\bin directory enthalten. Um die Pfadeinstellungen zu überprüfen, öffnen Sie ein Eingabeaufforderungsfenster auf dem VMS-Server, und geben Sie **blat ein**.Wenn Sie den Fehler `File not found (Datei nicht gefunden)` erhalten, kopieren Sie entweder die Datei blat.exe in das Verzeichnis winnt\system32, oder suchen Sie sie und öffnen Sie sie aus dem Verzeichnis, in dem sie sich befindet. Führen Sie zur Installation Folgendes aus:

```
blat -install
```

Sobald dieses Programm installiert ist, sind Sie fertig.

Skripte

Dies sind die Skripte, auf die in Schritt 1 des Konfigurationsverfahrens verwiesen wird:

- 3.x-Sensorskript
- 4.x-Sensorskript
- 5.x-Sensorskript

3.x-Sensorskript

Verwenden Sie dieses Skript für Sensoren der Version 3.x.

3,x Sensoren

```
#!/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-Sensorskript

Verwenden Sie dieses Skript für Sensoren der Version 4.x.

4,x Sensoren

```
#!/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-Sensorskript

Verwenden Sie dieses Skript für Sensoren der Version 5.x.

5,x Sensoren

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

Überprüfen

Für diese Konfiguration ist derzeit kein Überprüfungsverfahren verfügbar.

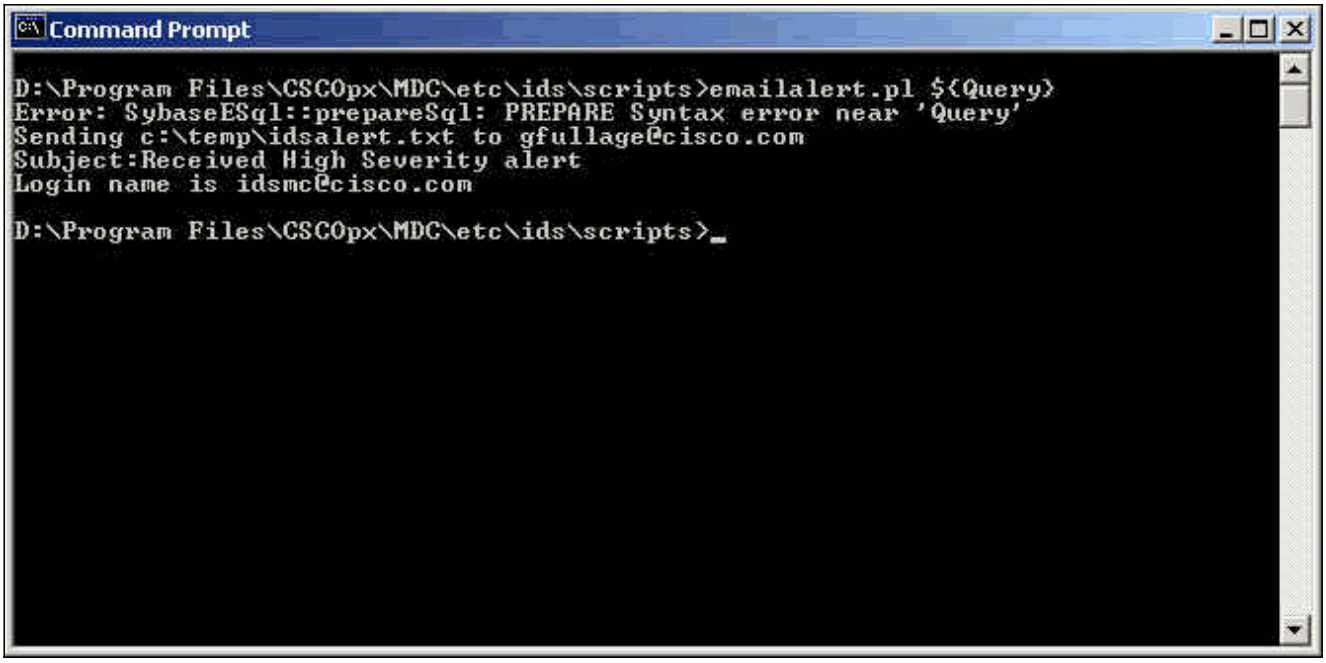
Fehlerbehebung

Befolgen Sie diese Anweisungen, um eine Fehlerbehebung für Ihre Konfiguration durchzuführen.

1. Führen Sie diesen Befehl über eine Eingabeaufforderung aus, um zu überprüfen, ob die Karte ordnungsgemäß funktioniert:

   blat

   `<Dateiname>` ist der vollständige Pfad zu einer beliebigen Textdatei im VMS-System. Wenn der Benutzer, an den das E-Mail-Skript weitergeleitet wird, diese Datei im Text einer E-Mail empfängt, dann wissen Sie, dass die blat funktioniert.
2. Wenn nach der Auslösung einer Warnung keine E-Mail empfangen wird, versuchen Sie, das Perl-Skript in einem Eingabeaufforderungsfenster auszuführen.Dieser Befehl hebt alle Probleme mit Perl- oder Pfadtypen hervor. Öffnen Sie dazu eine Eingabeaufforderung, und geben Sie Folgendes ein:
   >cd Program Files/CSCOpx/MDC/etc/ids/scripts
   >emailalert.pl ${Query}
   Möglicherweise erhalten Sie einen Sybase-Fehler, ähnlich dem Beispiel in diesem Beispiel. Dies liegt daran, dass der übergebene `${Query}`-Parameter keine Informationen enthält, anders als bei der Übertragung aus dem Sicherheitsmonitor.



Abgesehen von der Anzeige dieses Fehlers wird das Skript korrekt ausgeführt und sendet eine E-Mail. Alle Alarmparameter innerhalb des E-Mail-Textkörpers sind leer. Wenn Sie Perl- oder Pfadfehler erhalten, müssen diese behoben werden, bevor eine E-Mail gesendet wird.

Zugehörige Informationen

- Support-Seite für Cisco Secure Intrusion Prevention
- Technischer Support und Dokumentation - Cisco Systems