Configuration des notifications de messagerie avec les scripts pour les alertes IDS avec le CiscoWorks Monitoring Center for Security

Contenu

Introduction

Le contrôleur de sécurité a la capacité d'envoyer des notifications par courrier électronique quand une règle d'événement est déclenchée. Les variables incorporées qui peuvent être utilisées dans la notification par courrier électronique pour chaque événement n'incluent pas des choses telles que l'ID de signature, la source et la destination d'alerte, et ainsi de suite. Ce document fournit des instructions que vous pouvez employer pour configurer le contrôleur de sécurité pour inclure ces variables (et beaucoup plus) dans le message de notification par courrier électronique.

Conditions préalables

Conditions requises

Aucune spécification déterminée n'est requise pour ce document.

Composants utilisés

Ce document n'est pas limité à des versions de matériel et de logiciel spécifiques. Cependant, soyez sûr d'utiliser le script Perl approprié basé sur quelles versions de capteur fonctionnent dans votre environnement.

Conventions

Pour plus d'informations sur les conventions utilisées dans ce document, reportez-vous à Conventions relatives aux conseils techniques Cisco.

Procédure de configuration de notification par courrier électronique

Employez cette procédure pour configurer des notifications par courrier électronique.

Remarque: Afin d'envoyer le courrier électronique à l'adresse électronique correcte, soyez sûr de changer l'adresse électronique dans le script.

1. Copiez un de ces scripts dans le $BASE \ CSCOpx \ MDC \ etc. \ id \ répertoire de scripts sur le serveur de la solution de Gestion VPN/Security (VMS). Ceci te permet pour le sélectionner plus tard dans le processus quand vous définissez une règle d'événement. Sauvegardez le script comme **emailalert.pl.Remarque:** Si vous utilisez un nom différent, assurez-vous la référence qui nomment en cas la règle définie dans ces étapes.Pour des capteurs de version 3.x, utilisez le script des capteurs 3.xPour des capteurs de version 4.x, utilisez le script des capteurs 4.xPour des capteurs de version 5.x, utilisez le script des capteurs 5.xSi vous avez une combinaison des versions de capteur, Cisco vous recommande mise à jour de sorte qu'ils tous soient au même niveau de version. C'est parce que seulement un de ces scripts peut être exécuté en même temps.
2. Le script contient les commentaires qui expliquent chaque partie et n'importe quelle entrée exigée. En particulier, modifiez la variable `$EmailRcpt` (près du dessus du fichier) pour être l'adresse électronique de la personne qui doit recevoir les alertes.
3. Définissez une règle d'événement dans le contrôleur de sécurité d'appeler un nouveau script Perl. De la page principale de contrôleur de sécurité, choisissez l'**admin** > les **règles d'événement** et ajoutez un nouvel événement.
4. Sur le spécifier la fenêtre de filtre d'événement, ajoutent les filtres que vous voulez pour déclencher l'alerte par courrier électronique (dans l'échantillon ici, un courrier électronique est envoyé pour n'importe quelle alerte de à sévérité élevée).
5. Sur le choisir la fenêtre d'action, cochent la case pour exécuter un script et pour sélectionner le nom de script de la liste déroulante.
6. Dans les arguments sectionnez, écrivez **« $ {requête} »** comme affiché ici.**Remarque:** Ceci doit être entré exactement pendant qu'il est ici, y compris les double-devis. Il distingue les majuscules et minuscules également.
7. Quand une alerte, comme définie dans des vos filtres d'événement (dans cet exemple, une alerte de à sévérité élevée) est reçue, le script appelé l'emailalert.pl s'appelle avec un argument de $ {`requête}. Ceci` contient les informations complémentaires au sujet de l'alerte. Le script analyse tous les champs distincts et utilise un programme appelé « blat » pour envoyer un courrier électronique à l'utilisateur final.
8. Blat est un programme d'email de logiciel gratuit utilisé sur des systèmes Windows pour envoyer des courriers électroniques des fichiers batch ou des scripts Perl. Il est inclus en tant qu'élément de l'installation VMS dans le $BASE \ CSCOpx \ répertoire de coffre. Afin de vérifier vos configurations de chemin, ouvrir une fenêtre d'invite de commande sur le serveur et le type VMS **blat**.Si vous recevez l'erreur `non trouvée de fichier`, copiez le fichier blat.exe

dans le répertoire winnt\system32, ou trouvez-le et ouvrez-le à partir du répertoire dans lequel il se trouve. Afin d'installer ceci, exécutez-vous :

`blat -install <SMTP server address> <source email address>` Une fois que ce programme est installé, vous êtes fait.

Scripts

Ce sont les scripts visés à l'étape 1 de la procédure de configuration :

- script du capteur 3.x
- script du capteur 4.x
- script du capteur 5.x

script du capteur 3.x

Utilisez ce script pour des capteurs de version 3.x.

capteurs 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 du capteur 4.x

Utilisez ce script pour des capteurs de version 4.x.

capteurs 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 du capteur 5.x

Utilisez ce script pour des capteurs de version 5.x.

capteurs 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\"");
```

Vérifiez

Aucune procédure de vérification n'est disponible pour cette configuration.

Dépannez

Suivez ces instructions pour dépanner votre configuration.

1. Exécutez cette commande d'une invite de commande afin de vérifier qui blat des travaux correctement :
   blat <filename> -t <customer's email> -s "Test message" le <filename> est le chemin d'accès complet à n'importe quel fichier texte sur le système VMS. Si l'utilisateur vers qui le script de courrier électronique est dirigé reçoit ce fichier dans le corps d'un courrier électronique, alors vous savez qui blat des travaux.
2. Si aucun courrier électronique n'est reçu après qu'une alerte soit déclenchée, essayez d'exécuter le script Perl d'une fenêtre d'invite de commande.Ceci met en valeur toutes les questions de type Perl ou de chemin. Afin de faire ceci, ouvrez une invite de commande et entrez :>**cd Program Files/CSCOpx/MDC/etc/ids/scripts** >**emailalert.pl ${Query}** Vous pouvez potentiellement recevoir une erreur de Sybase, semblable à cet exemple. C'est dû au fait que le paramètre $ {requête} que vous passez ne contient pas réellement les informations, à la différence de quand il passe du contrôleur de sécurité.Autre que voir cette erreur, le script fonctionne correctement et envoie un courrier électronique. Tous les paramètres vigilants dans le corps de courrier électronique sont vides. Si vous recevez n'importe quelles erreurs Perl ou de chemin, elles doivent être réparées avant qu'un courrier électronique soit envoyé.

Informations connexes

- Page de support Cisco Secure de prévention des intrusions
- Support et documentation techniques - Cisco Systems