

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

## 目录

[简介](#)

[先决条件](#)

[要求](#)

[使用的组件](#)

[规则](#)

[电子邮件通知配置过程](#)

[脚本](#)

[3.x传感器脚本](#)

[4.x传感器脚本](#)

[5.x传感器脚本](#)

[验证](#)

[故障排除](#)

[相关信息](#)

## 简介

安全监控器能够在触发事件规则时发送电子邮件通知。在电子邮件通知中可用于每个事件的内置变量不包括签名ID、警报的源和目标等。本文档提供了配置安全监控器以在电邮通知消息中包含这些变量（以及更多变量）的说明。

## 先决条件

### 要求

本文档没有任何特定的要求。

### 使用的组件

本文档不限于特定的软件和硬件版本。但是，请务必根据您环境中运行的传感器版本使用相应的Perl脚本。

### 规则

有关文档约定的更多信息，请参考 [Cisco 技术提示约定](#)。

## 电子邮件通知配置过程

使用此过程配置电子邮件通知。

**注意：** 要将电子邮件发送到正确的电子邮件地址，请务必更改脚本中的电子邮件地址。

1. 将其中一个脚本复制到\$BASE\CSCOpX\MDC\etc\ids\scripts directory on the VPN/安全管理解决方案(VMS)服务器。这允许您稍后在定义事件规则时在流程中选择它。将脚本另存为 **emailalert.pl**。**注意：** 如果使用其他名称，请确保在这些步骤中定义的事件规则中引用该名称。对于3.x版传感器，请使用[3.x传感器脚本](#)对于4.x版传感器，请使用[4.x传感器脚本](#)对于5.x版传感器，请使用[5.x传感器脚本](#)如果您有传感器版本组合，思科建议您升级，以便它们都处于同一版本级别。这是因为，每次只能运行其中一个脚本。
2. 脚本包含解释每个部分和任何必需输入的注释。特别是，将\$EmailRcpt变量（靠近文件顶部）修改为接收警报的人员的电子邮件地址。
3. 在安全监控器中定义事件规则以调用新的Perl脚本。从Security Monitor主页中，选择**Admin > Event Rules**并添加新事件。
4. 在“指定事件过滤器”窗口中，添加要触发邮件警报的过滤器（在此示例中，会为任何严重性级别较高的警报发送邮件）。

Specify the Event Filter

Event Field Filtering

Severity = High

AND none =

AND none =

AND none =

AND none =

(Severity = High)

Show Filter

5. 在“选择操作”窗口中，选中该框以执行脚本，然后从下拉框中选择脚本名称。
6. 在“参数”部分，输入“**{Query}**”，如下所示。**注意：** 必须按原样输入，包括双引号。它还区分大小写。

**Choose the Actions**

**Rule Actions**

Notify via Email

Recipient(s):

Subject: Rule1.cisco-ul4o6k829

Message: (Severity = High)

Log a Console Notification Event

User Name:

Severity: debug

Message:

Execute a Script.

Script File: emailalert.pl Arguments: "\${Query}"

7. 当收到事件过滤器中定义的警报（在本例中为高严重性警报）时，调用名为emailalert.pl的脚本时，其参数为\${Query}这包含有关警报的其他信息。脚本解析所有单独的字段，并使用名为“blat”的程序向最终用户发送电子邮件。
8. Blat是Windows系统上用于从批处理文件或Perl脚本发送电子邮件的免费软件电子邮件程序。此VMS安装包包含在\$BASE\CSCOpX\bin directory中。要验证路径设置，请在VMS服务器上打开命令提示符窗口并键入blat。如果收到File not found错误，请将blat.exe文件复制到winnt\system32目录，或者找到该文件并从其所在的目录将其打开。要安装此软件，请运行：

```
blat -install
```

安装此程序后，即可完成。

## 脚本

以下是配置过程[步骤1](#)中引用的脚本：

- [3.x传感器脚本](#)
- [4.x传感器脚本](#)
- [5.x传感器脚本](#)

## 3.x传感器脚本

对3.x版传感器使用此脚本。

### 3.x传感器

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

```

##
## The following are the only two variables that need
## changing. $TempIDSFile can be any
## filename (doesn't have to exist), just make sure the
## directory that you specify
## exists. Make sure to use 2 backslashes for each
## directory, the first backslash is
## so the Perl interpreter 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];
$routersAddr = @fields[19];
$contextString = @fields[20];

## Open temp file to write alert data into,

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

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

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

## then call "blat" to send contents of that file in the
body of an email message.
## Blat is a freeware email program for WinNT/95, it
comes with VMS in the
## $BASE\CSCOpX\bin directory, make sure you install it
first by running:
##
## blat -install <SMTP server address> <source email
address>
##
## For more help on blat, just type "blat" at the

```

```

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

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

```

## 4.x传感器脚本

对4.x版传感器使用此脚本。

### 4.x传感器

```

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

```

```

## exists. Make sure to use 2 backslashes for each
directory, the first backslash is
## so the Perl interpreter 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,$yday,$isdst) =
localtime($1);

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

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

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

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

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

```

```

}

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

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

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

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

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

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

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

}

close(OUT);

```

```

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

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

```

## 5.x传感器脚本

对5.x版传感器使用此脚本。

### 5.x传感器

```

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

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

```

```

#
#           emailalert.pl "${Query}"
#
#           Where:
#
#           "${Query}" - this is the query
keyword dynamically
#                       output by the rule
when it triggers.
#                       It MUST be wrapped in
double quotes
#                       when specifying it in
the Arguments
#                       box on the Rule
Actions panel.
#
#
#*****
#*****
##
## The following are the only two variables that need
changing. $TempIDSFile can be any
## filename (doesn't have to exist), just make sure the
directory that you specify
## exists. Make sure to use 2 backslashes for each
directory, the first backslash is
## so the Perl interpreter 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,$yday,$isdst) =
localtime($1);

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

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

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

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

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

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

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

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

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

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

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

    @actions = ();
    if (m/\<sd\:actions\>(.)\<</sd\:actions\>/) {
        $rawaction = $1;
    }

```

```

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

    if ($2 eq "true") {
        push @actions,$1;
    }
}
if (@actions) {
    $actiontaken = join(', ',@actions);
}
else {
    $actiontaken = "None";
}

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

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

close(OUT);

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

```

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

## 验证

当前没有可用于此配置的验证过程。

## 故障排除

请按照以下说明排除配置故障。

1. 在命令提示符下运行以下命令，以检查blat是否正常工作：

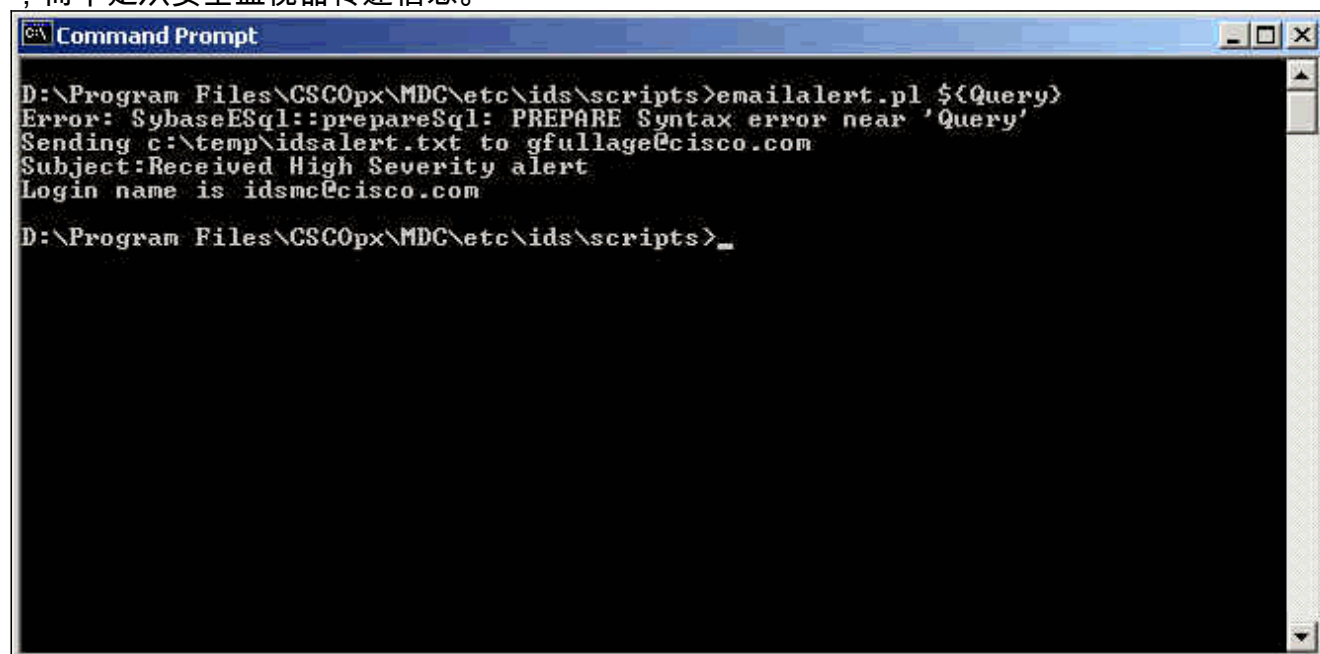
```
blat
```

<filename>是VMS系统上任何文本文件的完整路径。如果邮件脚本指向的用户在邮件正文中收到此文件，则您知道此文件有效。

2. 如果在触发警报后未收到电子邮件，请尝试从命令提示符窗口运行Perl脚本。这会突出显示所有Perl或路径类型问题。为此，请打开命令提示符并输入：

```
>cd Program Files\CSCOpX\MDC\etc\ids\scripts  
>emailalert.pl ${Query}
```

您可能会收到Sybase错误，与本示例类似。这是因为您传递的\${Query}参数实际上不包含信息，而不是从安全监视器传递信息。



```
Command Prompt  
D:\Program Files\CSCOpX\MDC\etc\ids\scripts>emailalert.pl ${Query}  
Error: SybaseESql::prepareSql: PREPARE Syntax error near 'Query'  
Sending c:\temp\idsalert.txt to gfullage@cisico.com  
Subject:Received High Severity alert  
Login name is idsmc@cisico.com  
D:\Program Files\CSCOpX\MDC\etc\ids\scripts>_
```

除了看到此错误外，脚本还能正确运行并发送电子邮件。邮件正文中的所有警报参数都为空。如果收到任何Perl或路径错误，则在发送电子邮件之前需要修复这些错误。

## 相关信息



- [思科安全入侵防御支持页](#)
- [技术支持和文档 - Cisco Systems](#)