

Understand the First Responder Program

Contents

[Introduction](#)

[Prerequisites](#)

[Requirements](#)

[Components Used](#)

[Background Information](#)

[Automated E-mail](#)

[Script / Commands](#)

[Reason for This Email](#)

[Automated E-mail](#)

[Introduction Block](#)

[Data Request Block](#)

[Generated Command](#)

[Firepower.py Script](#)

[Automation](#)

[Interactive](#)

[Expected Output of the Script](#)

[Common Issues](#)

[E-mail Security / URL Re-write](#)

[Steps to Resolve](#)

[DNS Failure](#)

[Steps to Resolve](#)

[Failure to Open / Create Log File](#)

[Steps to Resolve](#)

[Failure to Open / Write Notify File](#)

[Steps to Resolve](#)

[Failure to Lock sf_troubleshoot.pid File](#)

[Steps to Resolve](#)

[Upload Issues](#)

[Steps to Resolve](#)

[Related Information](#)

Introduction

This document describes the use and implementation of the First Responder Program for Cisco Secure Firewall.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

This document is based on Cisco Secure Firewall products.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

The First Responder program was created by TAC to make it easier and faster to provide diagnostic data for the open cases. There are two main components that make up the program:

Automated E-mail

This e-mail is sent out at the start of the case with instructions on how to gather and upload diagnostic data for TAC analysis. There are multiple technologies that leverage this system, and each e-mail is mapped to the Technology and Sub-Technology that are chosen when the case is created.

Script / Commands

Each implementation of the First Responder program has its own unique way to handle data collection and delivery. The Secure Firewall implementation utilizes the TAC-authored `firepower.py` Python script to accomplish this. The automated e-mail process generates a one-line command, unique to this specific case, that can be copied and pasted into the CLI of Secure Firewall devices to run.

Reason for This Email

There are certain technologies that are enabled for the first responder program. This means that every time a case is opened against one of these enabled technologies, a first responder e-mail gets sent out. If you receive a first responder e-mail and do not believe the data request are relevant, please feel free to ignore the communication.

For the Secure Firewall use case, the first responder program is limited to the Firepower Threat Defense (FTD) software. If you run an Adaptive Security Appliance (ASA) code-base, please ignore this e-mail. Since these two products run on the same hardware, it is commonly observed that ASA cases are created in the Secure Firewall technology space, which generates the first responder e-mail.

Automated E-mail

Here is an example of the automated e-mail that is sent out as part of this program:

```
From: first-responder@cisco.com <first-responder@cisco.com>  
Sent: Thursday, September 1, 2022 12:11 PM  
To: John Doe <john.doe@cisco.com>  
Cc: attach@cisco.com  
Subject: SR 666666666 - First Responder Automated E-mail
```

Dear John,

In an effort to resolve your case faster it may be necessary to collect some diagnostic data from your device. Based on the problem statement you provided, below are a few pieces of data that would help speed the resolution of your case.

*** Troubleshoot File ***

```
* Connect to the device using SSH
* Issue the command expert, skip this step for FMC version 6.4.x and earlier
* Issue the command sudo su
* When prompted for the password, enter your password.
* For FMC 6.4 or FTD 6.7 and later issue the command
curl -s -S https://cxd.cisco.com/public/ctfr/firepower.py | python - -c 666666666 -t aBcDeFgHiJkLmNoP -p <CURRENT_CXD_IP1>

* For FMC 6.3 or FTD 6.6 and earlier issue the command
curl -k -s -S https://cxd.cisco.com/public/ctfr/firepower.py | python - -c 666666666 -t aBcDeFgHiJkLmNoP -p <CURRENT_CXD_IP2>
```

For more information on what this command does, or to understand why you are receiving this e-mail - please visit the following link: LINK_TO_THIS_ARTICLE

For 6.3 and earlier versions we recommend confirming cxd.cisco.com resolves to CURRENT_CXD_IP1 or CURRENT_CXD_IP2. Furthermore, we recommend validating the SHA checksum of the file by running `curl -s -k https://cxd.cisco.com/public/ctfr/firepower.py | shasum` which should output CURRENT_SHA.

If you are unable to upload troubleshooting files (or would prefer not to), please let us know what hardware and software version you are running if you have not already.

Sincerely, First Responder Team

The automated e-mails for the first responder program are split into two parts known as the introduction block and the data request block.

Introduction Block

The introduction block is a static string that is included in every first responder e-mail. This introductory sentence simply serves to provide context to the data request block(s). Here is an example of an introduction block:

Dear <NAME>,

In an effort to resolve your case faster it may be necessary to collect some diagnostic data from your device. Based on the problem statement you provided, below are a few pieces of data that would help speed the resolution of your case and the steps to collect them:

Data Request Block

The data request blocks are the heart of the first responder program. Each block is a pre-defined set of steps to gather data for a given technology. As mentioned in the **Background Information** section, each data request block is mapped to a specific technology. This is the same technology that is chosen to open a support case. Typically the automated e-mail contains a single data request block. However, if the chosen technology has more than one data request block mapped to it, then multiple data requests are included in the e-mail. Here is an example format of the Data Request block with multiple data requests:

*** <REQUEST NAME 1> ***

<REQUEST 1 STEPS>

*** <REQUEST NAME 2> ***

<REQUEST 2 STEPS>

For example, in the case of Secure Firewall, multiple data request blocks are often included when a request is raised for assistance with Remote Access VPN (RA-VPN) issues with Firepower Threat Defense (FTD) as the VPN technology also has a mapped data request block configured for assistance to gather the DART bundles.


Generated Command

For the Secure Firewall use case specifically, a unique one-line command is generated for each case as part of the automated e-mail. Here is a breakdown of the structure of the one-line command:

```
# curl -k -s -S https://cxd.cisco.com/public/ctfr/firepower.py | python -c 6666666666 -t aBcDeFgHiJkLmNoP -fr --auto-upload &
```

1 2 3 4 5 6 7 8 9 10 11 12

1. The `curl` command is used to download the latest version of the `firepower.py` script.
2. The `-k` flag is an option for `curl` to ignore certificate errors during connection.
3. The `-s` flag is an option for `curl` to run in silent mode. This is used to suppress the normal `curl` output as it is noisy.
4. The `-S` flag is an option for `curl` to show errors. This is used to force `curl` to still show output errors even with the silent option enabled.
5. The URL where the latest version of `firepower.py` script is hosted. This path instructs the `curl` command to pull the latest body of the script to be run.
6. This is a Linux pipe, which passes the output of the `curl` command (the contents of a python script) over to an execution statement in the next step.
7. At this step the `python` binary on the device is called with an additional `-`. This instructs `python` that the source is taken from `stdin` (since the contents of the script are piped over from `curl`).
8. The `-c` flag is an input argument for the `firepower.py` script, which indicates the case number to which the data has to be uploaded. The `6666666666` value after this option is the example case number.
9. The `-t` flag is an input argument for the `firepower.py` script, which indicates a unique token (password) that was generated for this particular case. The `aBcDeFgHiJkLmNoP` value after this option is the example token for this case.
10. The `-fr` flag was added to indicate that the name of the file must be changed to be prefixed with the string `SFFR-` to indicate that the file was generated by the first responder script.
11. The `--auto-upload` flag is a special argument for the `firepower.py` script, which indicates the script to run in automation mode. More information on this can be found in the script specific section.
12. The `&` instructs this entire command to run in the background, which allows the user to continue to interact with their shell while the script executes.

 **Note:** The `-k` flag is required for any FMC version prior to 6.4 and any FTD version prior to 6.7 since the root certificate used by CXD was not trusted by Firepower devices until FMC version 6.4 and FTD version 6.7, this causes certificate verification to fail.

Firepower.py Script

The main goal of the script is to generate and upload a diagnostic bundle from the Secure Firewall device referred to as a troubleshoot. To generate this troubleshoot file, the **firepower.py** script simply calls the **built-in sf_troubleshoot.pl** script that is responsible to build this bundle. This is the same script that is called when you generate a troubleshoot from the GUI. In addition to the troubleshoot file, the script also has the ability to collect other diagnostic data that is not included as part of the troubleshoot bundle. Currently, the only additional data that can be gathered are Core Files - but this can be expanded in the future if the need arises. The script can be run in either Automation or Interactive mode:

Automation

This mode is enabled and uses the **--auto-upload** option when you run the script. This option disables the interactive prompts, enables core file collection, and automatically uploads data to the case. The one-line command generated by the automated e-mail includes the **--auto-upload** option.

Interactive

This is the default run mode for the script. In this mode the user receives prompts to confirm whether or not to gather additional diagnostic data such as core files. Regardless of execution mode, meaningful output gets printed to screen and logged to a log file to indicate the progress of the scripts execution. The script itself is extensively documented via in-line code comments and can be downloaded/reviewed at <https://cxd.cisco.com/public/ctfr/firepower.py>.

Expected Output of the Script

Here is an example of a successful execution of the script:

```
root@ftd:/home/admin# curl -k -s -S https://cxd.cisco.com/public/ctfr/firepower.py | python - -c 666666
[1] 26422
root@ftd:/home/admin#
`/var/common/first_responder_notify` successfully uploaded to 666666666
Running sf_troubleshoot.pl command to create a troubleshoot file...
Troubleshoot file successfully generated at /ngfw/var/common/results-08-30-2022--135014.tar.gz
Attempting to upload troubleshoot to case...
#####
`/ngfw/var/common/results-08-30-2022--135014.tar.gz` successfully uploaded to 666666666
Found the following core files:
    (0 B) - /ngfw/var/common/core_FAKE1.gz
    (0 B) - /ngfw/var/common/core_FAKE2.gz
    (0 B) - /ngfw/var/common/core_FAKE3.gz
Successfully created /ngfw/var/common/cores_666666666-1661867858.tar.gz
Attempting core file upload...
#####
`/ngfw/var/common/cores_666666666-1661867858.tar.gz` successfully uploaded to 666666666
FINISHED!
```

Please note that this output example includes core file uploads. If there are no core files present on your device, the message "No core files found. Skipping core file processing" is presented instead.

Common Issues

Here are some common issues that you can experience (in order of process / execution):

E-mail Security / URL Re-write

Often times, it is observed that the end user has some level of E-mail Security that rewrites the URL. This alters the one-line command that is generated as part of the automated e-mail. This results in an execution failure since the URL to pull the script has been re-written and is invalid. Here is an example of the expected one-line command:

```
<#root>
```

```
curl -s -S https://cxd.cisco.com/public/ctfr/firepower.py | python - -c 666666666 -t aBcDeFgHiJkLmNoP -f
```

Steps to Resolve

If the URL in the command from the e-mail is anything other than <https://cxd.cisco.com/public/ctfr/firepower.py>, then the URL was likely rewritten in transit. To fix this issue, simply replace the URL before you run the command.

DNS Failure

This curl error is often seen when the device is unable to resolve the URL to download the script:

```
<#root>
```

```
curl: (6) Could not resolve host: cxd.cisco.com
```

Steps to Resolve

To fix this issue, please check the DNS settings on the device to ensure that it is able to resolve the URL properly to proceed.

Failure to Open / Create Log File

One of the first things that the script attempts to do is create (or open, if it already exists) a log file named **first-responder.log** in the current working directory. If this operation fails, then an error which indicates a simple permission issue is displayed:

```
<#root>
```

```
Permission denied while trying to create log file. Are you running this as root?
```

As part of this operation, all other errors are identified and printed to screen in this format:

```
<#root>
```

```
Something unexpected happened while trying to create the log file. Here is the error:
```

<EXCEPTION BODY>

Steps to Resolve

To fix this error, simply run the script as an administrative user such as admin or root.

Failure to Open / Write Notify File

As part of the script execution a 0-byte file named **first_responder_notify** gets created on the system. This file is then uploaded to the case as part of the automation for this program. This file is written to the "/var/common" directory. If the user who executes the script does not have sufficient permissions to write files to this directory, then the script displays the error:

<#root>

```
Failed to create file -> `/var/common/first_responder_notify`. Permission denied. Are you running as root?
```

Steps to Resolve

To fix this error, simply run the script as an administrative user such as admin or root.



Note: If a non-permissions related error is encountered, then a catch-all error is printed on screen "Unexpected error while trying to open file -> `/var/common/first_responder_notify`. Please check first-responder.log file for full error". The full exception body can be found in the **first-responder.log** .

Failure to Lock sf_troubleshoot.pid File


To ensure only one troubleshoot generation process is run at a time, the troubleshoot generation script tries to lock the /var/sf/run/sf_troubleshoot.pid file before proceeding. If the script fails to lock the file, an error is displayed:

<#root>

```
Failed to run the `sf_troubleshoot.pl` command - existing sf_troubleshoot process detected. Please wait
```

Steps to Resolve

Most of the time, this error means that a separate troubleshoot generation task is already in process. Sometimes this is a result of users who accidentally execute the one-line command twice in a row. To fix this issue, wait for the current troubleshoot generation job to finish and try again later.

 **Note:** If an error within the sf_troubleshoot.pl script itself occurs, then this error is displayed on screen "Unexpected PROCESS error while trying to run `sf_troubleshoot.pl` command. Please check first-responder.log file for full error". The full exception body can be found in the **first-responder.log** .

Upload Issues

There is a common upload function in the script that is responsible for all file uploads throughout the scripts execution. This function is simply a python wrapper to execute a curl upload command to send the files to the case. Because of this, any errors encountered during execution, returns as a curl error code. In the event of an upload failure, then this error is displayed on screen:

```
<#root>
```

```
[FAILURE] Failed to upload `/var/common/first_responder_notify` to 666666666. Please check the first-res
```

Check the **first-responder.log** file to see the full error. Typically, the **first-responder.log** file looks like:

```
<#root>
```

```
08/29/2022 06:51:57 PM - WARNING - Upload Failed with the following error:
```

```
-----
```

```
Command '['curl', '-k', '--progress-bar', 'https://666666666:aBcDeFgHiJkLmNoP@cxd.cisco.com/home/',  
'--upload-file', '/var/common/first_responder_notify']' returned non-zero exit status 6
```

```
-----
```

Steps to Resolve

In this case, curl returned an exit status of **6** which means **Could not resolve host**. This is a simple DNS failure while you try to resolve the hostname **cxd.cisco.com** . Please refer to the curl documentation to decode any unknown exit statuses.

Related Information

- [Cisco Technical Support & Downloads](#)