FTP Server Inspector

FTP Server Inspector Overview

Type	Inspector (service)
Usage	Inspect
Instance Type	Multiton
Other Inspectors Required	`ftp_client`, `stream_tcp`
Enabled	`true`

File Transfer Protocol (FTP) is a network protocol used to transfer files between clients and servers over TCP/IP. Once a client and server establish a connection, the client issues commands to the server to upload files to or download files from the server, and interprets responses from the server.

The `ftp_server` inspector examines and normalizes commands on the FTP command channel.

Given an FTP command channel buffer, the `ftp_server` inspector identifies the FTP commands and parameters, and enforces correctness of the parameters. The `ftp_server` determines when an FTP command connection is encrypted and when an FTP data channel is opened.

FTP Server Inspector Parameters

FTP Server port configuration

The `binder` inspector defines the FTP Server configuration. For more information, see the Binder Inspector Overview.

Example:

```
[
    {
        "when": {
            "role":"any",
            "service":"ftp",
            "ports": ""
        },
        "use": {
            "type":"ftp_server"
        }
    }
]
```

chk_str_fmt

Specifies a list of FTP commands to check for string format attacks. You can enable rule 125:5 to generate an alert, and in an inline deployment, drop offending packets when the inspector detects this condition. Separate multiple commands with a space character.

Type: string

Valid values: A list of valid FTP commands.

Default value: None

data_chan_cmds

Specifies a list of FTP commands to check for correct formatting. Separate multiple commands with a space character.

Type: string

Valid values: A list of one or more of the following commands: `PORT PASV LPRT LPSV EPRT EPSV`.

Default value: None

data_xfer_cmds

Specifies a list of data transfer commands. Check for correct formatting of the commands. Separate multiple commands with a space character.

Type: string

Valid values: A list of one or more of the following commands: `RETR STOR STOU APPE LIST NLST`.

Default value: None

file_put_cmds

Specifies a list of `PUT` commands. Check for correct formatting of the commands. Separate multiple commands with a space character.

Type: string

Valid values: A list of one or more of the following commands: `STOR STOU APPE`.

Default value: None

Caution Do not change the `file_put_cmds` parameter unless directed to do so by Support.

file_get_cmds

Specifies a list of `GET` commands. Check for correct formatting of the commands. Separate multiple commands with a space character.

Type: string

Valid values: A list of `GET` command, such as `RETR`.

Default value: None

Caution Do not change the `file_get_cmds` parameter unless directed to do so by Support.

encr_cmds

Specifies a list of commands related to secure connections. Check for correct formatting of the commands. Separate multiple commands with a space character.

Type: string

Valid values: A list of commands related to secure connections, for example: `AUTH`.

Default value: None

login_cmds

Specifies a list of commands related to the login process. Check for correct formatting of the commands. Separate multiple commands with a space character.

Type: string

Valid values: Specify a list of one or more commands: `USER`, `PASS`.

Default value: None

check_encrypted

Specifies whether to check an encrypted session for a command to end encryption. Use with the `encrypted_traffic` parameter.

You can enable rule 125:7 to generate events and, in an inline deployment, drop offending packets for this parameter.

Type: boolean

Valid values: `true`, `false`

Default value: `false`

cmd_validity[]

An array of FTP commands and the criteria the inspector uses to validate them. These validity checks override the default checks performed by the `ftp_server` inspector (RFC 959).

You can enable rules 125:2 and 125:4 to generate events, and, in an line deployment, drop offending packets for this parameter.

Type: array (object)

Example:

```
{
   "cmd_validity": [
       {
           "command": "CWD",
           "format": "abc",
           "length": 250
       }
   ]
}
```

cmd_validity[].command

Specifies the name of an FTP command to validate.

Type: string

Valid values: A valid FTP command enclosed in double quotation marks.

Default value: None

cmd_validity[].format

Describes the valid format for `cmd_validity[].command`

Type: string

Valid values: One of the following formats:

- `int` – The parameter must be an integer
- `number` – The parameter much be an integer between 1 and 255
- `char` *`chars`* – The parameter must be a single character from *`chars`*, a list of one or more characters with no separators between them.
- `date` *`datefmt`* – The parameter follows the format specified, where *`datefmt`* is constructed using the following elements:
    - `#` = Number
    - `C` = Char
    - `[]` = Optional format enclosed
    - `|` = OR
    - `{}` = Choice of formats enclosed
    - `.+-` literal characters
- `string` – The parameter is an unlimited string.
- `host_port` – The parameter must be a host port specifier, per RFC 959.
- `long_host_port` – The parameter must be a long host port specifier per RFC 1639.
- `extended_host_port` – The parameter must be an extended host port specifier per RFC 2428.
- `{}`,`|` – The parameter must be one of the choices enclosed within the braces, separated by |.

- `{}`, `[]` – The parameter must be one of the choices enclosed within the braces. Optional values are enclosed within the brackets.

Default value: None

cmd_validity[].length

Specifies the maximum length in bytes for the `cmd_validity[].command` parameter, overriding the default value defined in `def_max_param_len`. If the parameter for the FTP command exceeds the `cmd_validity[].length`, and rule 125:3 is enabled, Snort generates an alert. Use `cmd_validity[].length` to restrict specific commands to small parameter values.

Specify `0` to indicate unlimited length.

Type: integer

Valid range: `0` to `4,294,967,295 (max32)`

Default value: `0`

def_max_param_len

Specifies the default maximum length in bytes that the inspector permits for all FTP commands handled by the server. Use `def_max_param_len` for basic buffer overflow detection. (This can be overridden for individual commands using `cmd_validity[].length`.) You can enable rule 125:3 to generate events and, in an inline deployment, drop offending packets for this parameter.

Specify `0` to indicate unlimited length.

Type: integer

Valid range: `0` to `4,294,967,295 (max32)`

Default value: `100`

encrypted_traffic

Specifies whether to check for encrypted FTP traffic. Use with the `check_encrypted` parameter. You can enable rule 125:7 to generate events and, in an inline deployment, drop offending packets for this parameter.

Type: boolean

Valid values: `true`, `false`

Default value: `false`

ftp_cmds

A list of FTP commands the server supports beyond those described in RFC 959. (If your installation uses the "X" commands specified in RFC 775, for instance, you can add them to the inspector using this parameter.)

Type: string

Valid values: Space-separated list of valid FTP commands, enclosed in double quotation marks.

Default value: None

ignore_data_chan

Specifies whether to ignore the FTP data channels.

Type: boolean

Valid values: `true`, `false`

Default value: `false`

ignore_telnet_erase_cmds

Specifies whether to ignore the telnet escape sequences for the erase character (TNC EAC) and the erase line character (TNC EAL) when normalizing the FTP command channel. Set `ignore_telnet_erase_cmds` to match how your FTP server handles telnet erase commands. Newer FTP clients typically ignore these telnet escape sequences, while legacy clients typically process them.

If telnet erase commands are not ignored, and rule 125:1 is enabled, Snort generates an event, and, in an inline deployment, drops offending packets.

Type: boolean

Valid values: `true`, `false`

Default value: `false`

print_cmds

Specifies whether to print the configuration for each FTP command for this server on initialization.

Type: boolean

Valid values: `true`, `false`

Default value: `false`

telnet_cmds

Specifies whether to check for telnet commands on the FTP command channel. The presence of such commands could indicate an evasion attempt on the FTP command channel.

Type: boolean

Valid values: `true`, `false`

Default value: `false`

FTP Server Inspector Rules

Enable the `ftp_server` inspector rules to generate events and, in an inline deployment, drop offending packets.

Table 1: FTP Server Inspector Rules

GID:SID	Rule Message
125:1	TELNET command on FTP command channel
125:2	invalid FTP command
125:3	FTP command parameters were too long
125:4	FTP command parameters were malformed

GID:SID	Rule Message
125:5	FTP command parameters contained potential string format
125:7	FTP traffic encrypted
125:9	evasive (incomplete) TELNET cmd on FTP command channel

FTP Server Inspector Intrusion Rule Options

The `ftp_server` inspector does not have any intrusion rule options.