COMS 4180 - FTP Guide

Development Guide

Christina Floristean (cf2469)
Justin Pugliese (jp3571)

March, 2018

Contents

1 Introduction

Ensuring files are transferred and stored safely is paramount to a success file transfer protocol (FTP). This document outlines a simple FTP application with a Intruder Detection System (IDS) which ensures contents of a file are not altered in transmission. Additional usage information and files are described.

2 Installation

Since the FTP application is written in Python 3, there are some language specific dependencies. The easiest way to install the required packages are as follows:

1. Create a new directory for the application to run from:
   `$ mkdir <ftp-(client|server)>`

2. Navigate to the newly created directory:
   `$ cd <ftp-(client|server)>`

3. Install Python3:
   `Linux: $ sudo apt-get install python3`

4. Use Python3 to execute scripts:
   `$ python3 <example-script.py>`

2.1 Virtual Machine (VM) configuration (Optional)

Alternatively Python's virtual environment tool can be used to create an isolated environment. On any new VM which will be running the application run the following commands from the parent directory of where the application will run:

1. Install Python's package manager (pip):
   `$ sudo apt-get install python-pip`

2. Install Python's virtual environment tool:
   `$ pip install --upgrade virtualenv`

3. Create a new Python 3 project:
   `$ virtualenv -p python3 <ftp-(client|server)>`

4. Navigate into the newly created directory:
   `$ cd <ftp-(client|server)>`

5. Activate virtual environment:
   `$ source bin/activate`

2.2 Client Specific directions

On a VM which is intended to act as the client, `client.py` must be copied into the `/ftp-client` directory which was created in 2.1. Additionally, a `/files` subdirectory should also be created for the client application to use as its persistent file storage.

2.3 Server Specific directions

On a VM which is intended to act as the server, `server_wrapper.py`, `server.py`, `ids.py`, and `pattern_manager.py` must be copied into the `/ftp-server` directory which was created in 2.1. Additionally, a `/files` subdirectory should also be created for the server application to use as its persistent file storage.

2.4 Persistent file storage

Both client and server applications manage their files directly from the `/files` subdirectory which is nested within `/<ftp-(client|server)>.`

3 IDS Description

The IDS is comprised of a set of named patterns stored in the `pattern-config` file within the `/ftp-server` directory. Up to 50 rules can be added and they are completely configurable by the end-user.

The contents of the `pattern-config` file must be valid `json` format, otherwise the `IDS` application will be unable to use the rules.

3.1 Automated rule maintenance

The best way to maintain the pattern library is via `pattern_manager.py` utility, which enables basic CRUD (Create, Read, Update, Delete) operations on the pattern library. Executing `python pattern_manager.py` from `/ftp-server` will start the tool and displays instructions to the user. The tool supports the following commands:

- `add <pattern_id> <pattern>` - Add a pattern and its `id` to the pattern library
- `delete <pattern_id>` - Remove a pattern and its `id`
- `print` - Print the currently defined patterns
- `exit` - Exit the running application

If a pattern is added with non-hex characters it will be treated as ascii text, otherwise, hex-based patterns will be managed as hex strings.

3.2 Manual rule maintenance

Rules can also be managed by manually opening `pattern-config` file in a text editor and altering the text. However, the resulting content must be valid `json` format otherwise, the `ids.py` file will be unable to read the updated rules. Specifically, the form should be: {id1:pattern1, id2:pattern2, id3:pattern3}.

4 Code Execution

4.1 Server

Since the server is comprised of multiple components, server and the IDS, the `server_wrapper.py` script orchestates application start up and component communication. To begin, from `/ftp-server` run:

```
$ python3 server_wrapper.py <port>
```

The only variable accepted by server during start up is the port number the IDS should open a socket and listen for incoming connections. Note: files on the server must also exist

with their associated hash in the form filename.hash in order for the client's 'get' command to work. Thus, if a file exists on the server without the hash the client will receive a 'File not found' error.

4.2 Client

The client application takes multiple input variables. The first is the IP address of the server's VM and the second is the port number the server is listening on. The command is as follows:

```
$ python3 client.py <ip_address> <port>
```

5 Application Testing

The application currently supports a variety of end to end tests. A few of them are listed below including their setup/teardown steps. The subsequent test cases assume both the server and client running and the server is listening for a new commande from the client.

5.1 Successful PUT request

First add a file, `test-file.txt`, to the `/ftp-client/files` directory, then execute:

```
$ cmd: put test-file.txt
```

The outcome will be `test-file.txt`, saved to `/ftp-server/files`.

5.2 Unsuccessful PUT request (without filename specified)

Attempt to transfer a file by executing:

```
$ cmd: put
```

The outcome will be an error being presented through the client terminal and the application waiting for a new command specified.

5.3 Successful GET request

First add a file, `test-file.txt`, to the `/ftp-server/files` directory, then execute:

```
$ cmd: get text-file.txt
```

The outcome will be `test-file.txt,` saved to `/ftp-client/files.`

5.4 Unsuccessful PUT request (file does not exist)

First remove the test file from `/ftp-server/files` , then attempt to transfer a file by executing:

```
$ cmd: get test-file.txt
```

The outcome will be an error being presented through the client terminal and the application waiting for a new command specified.

5.5 Dropped packet

First add a rule to the IDS pattern library which corresponds to the first few characters of the file in `/ftp-client/files`. Execute:

```
$ cmd: put text-file.txt
```

The result will be a line item added to the `ids-log` file documenting the pattern match, offending IP address, and time.