Level 1.5 - Malware signature in FTP

Time to add another protocol to our repertoire of Deep-packet-inspection! FTP is a file transfer protocol - we should detect whenever it is used to download a malware from an FTP server.

Password

`y98jhvzrv3`

Instructions

1. We have a solution for tracking TCP streams, but so far we only apply it on streams where the source port is 80 - i.e. server responses.
2. However, FTP file downloads don't use a constant port - it is generated dynamically (out of a small range of options):
    1. Run the trigger to view the FTP session in wireshark
    2. The main FTP session control is to port 21 - this is the main interface``
    3. When a download is started, the server tells the client what port to connect to (termed a "**passive port**" in FTP lingo) for the file data
3. The problem is, while we know what the source port for the stream is (passive port), we don't know the destination port our client will use.

4. That means we need to...
    1. Analyze FTP (port 21) packets until we find such a directive (no need to use our Stream class for this, because we don't want to hold the entire port 21 stream in memory. we can just examine each packet at a time, until we find the passive port)
    2. Extract the passive port from the reply received from port 21
    3. Start tracking a TCP stream that has the given passive port as the source port
    4. Once the stream is completed, hash the stream and check it against known malware hashes
    5. Delete the stream from memory once it has been checked

Notes

GUIDING QUESTIONS Use the guiding questions below to identify the malicious trigger and structure your code's detection:

- How can we extract the port used for the FTP file download?
    - Which packet is the port information in?
    - How can we identify this packet? Are there any flags or keywords we can use?
    - How do we get the port number from the data in this packet?
- How can we tell our program to track the stream for the FTP file download?
    - What do we know about the FTP file download stream? Where is it coming from?
    - Do we have existing functions that already do the tracking and hashing?
    - How can we reuse those functions for this FTP file download stream?

To submit

Submit a ZIP `nids.zip` containing all `.py` files.