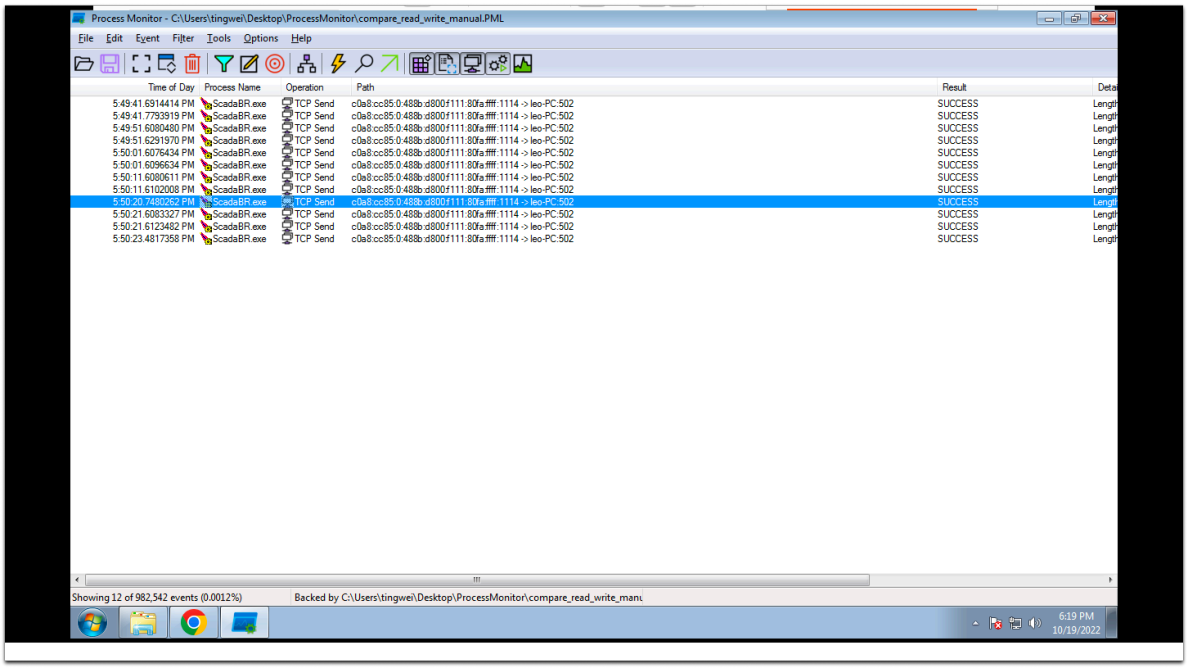
Identify a WRITE API call

Argument Analysis

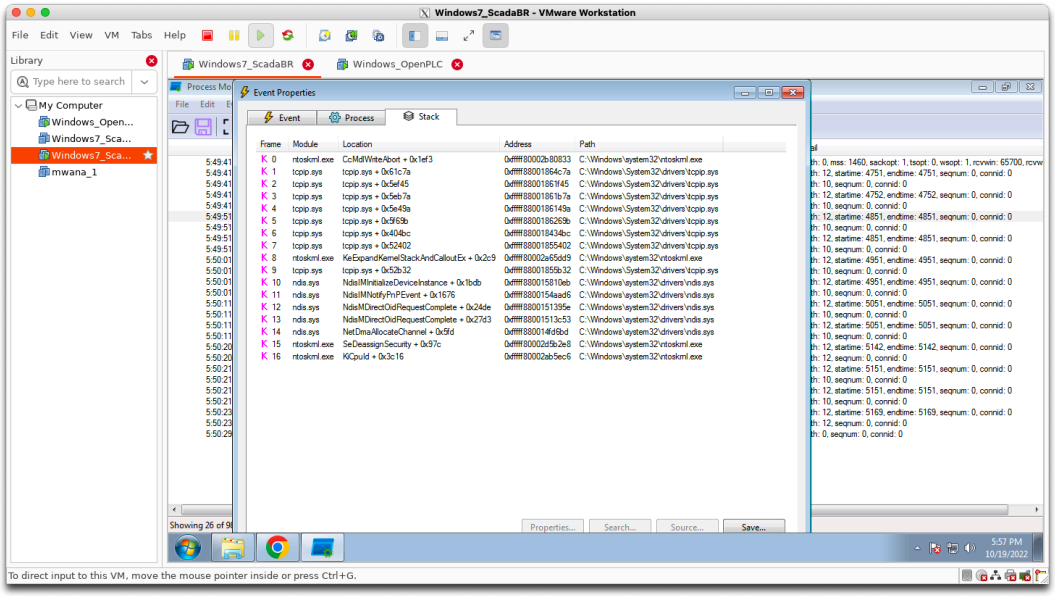
First, I used Process Monitor to see if there is any difference between READ API calls and WRITE API calls (compare_read_write_manual.pcapng, compare_read_write_manual.PML)

Here's all the TCP Send API calls.

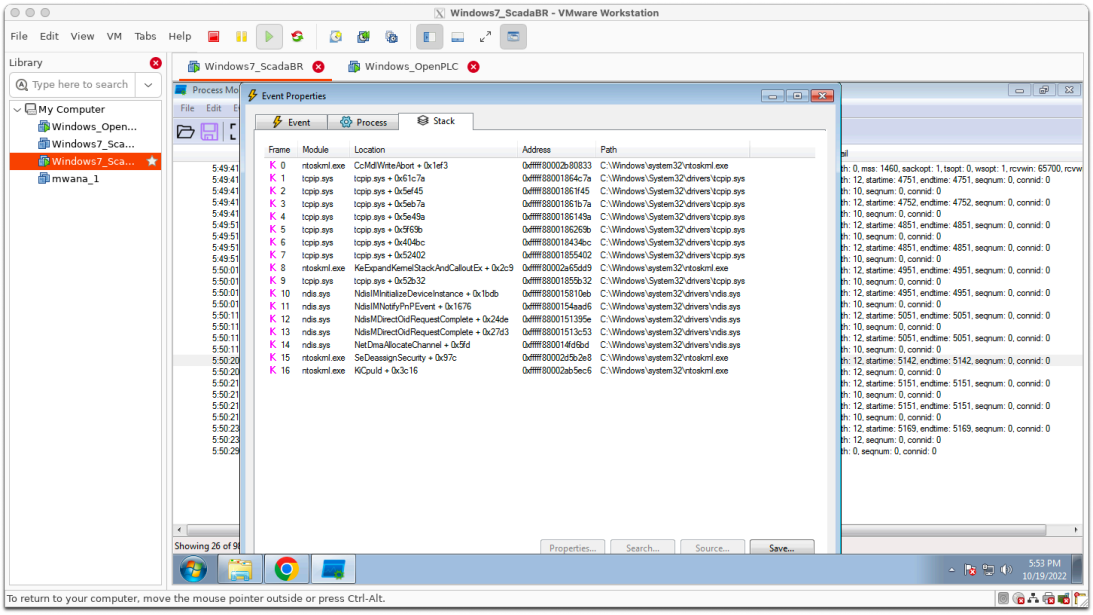


I compare
every column and stack. There is no information about WRITE and READ, and there's no arguments information. There is no clear identifier to differentiate the two, most of them are the same.

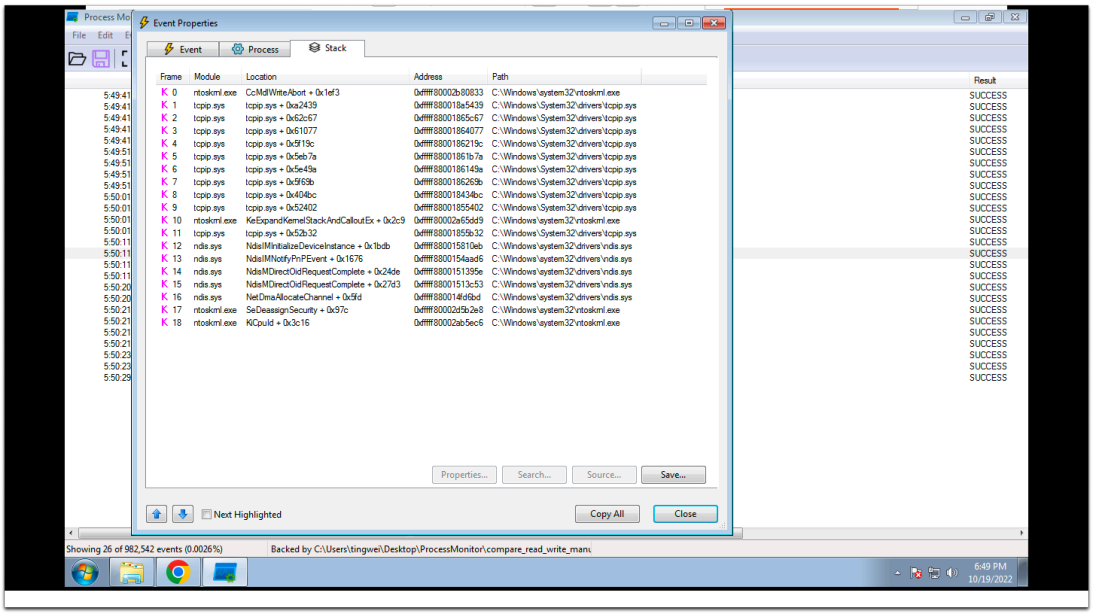
READ TCP Send



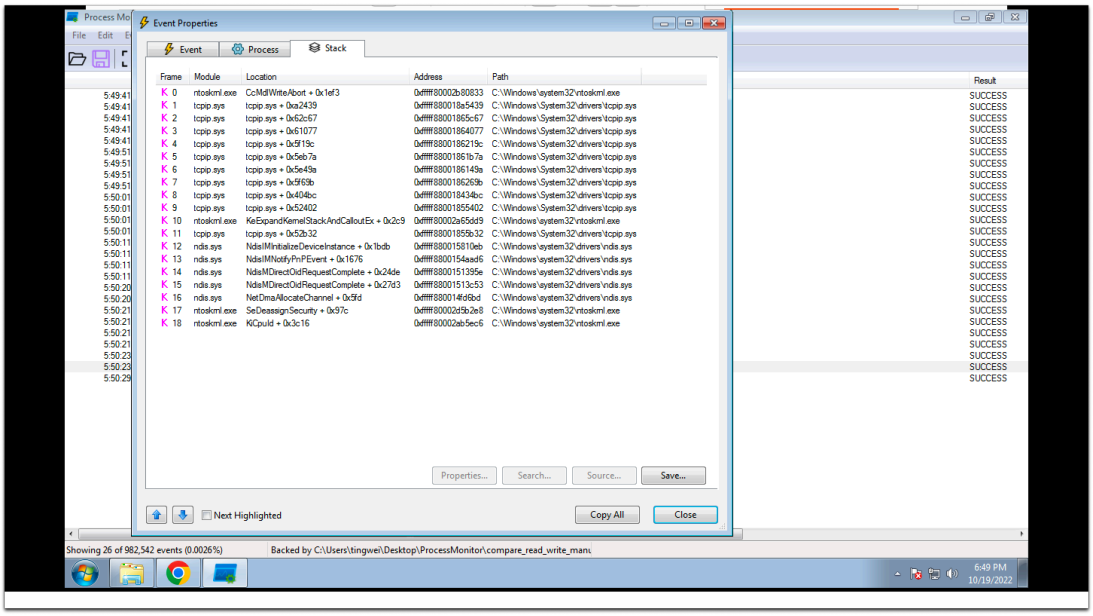
WRITE TCP Send



READ TCP Recv



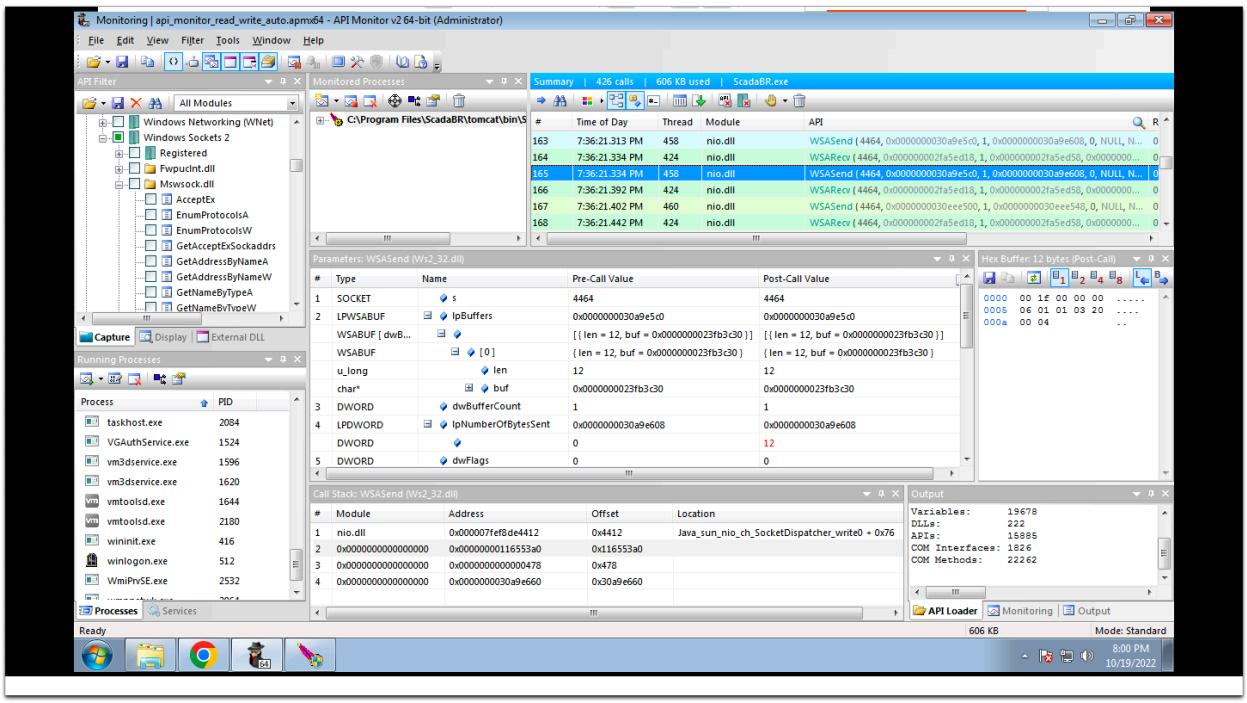
WRITE TCP Recv



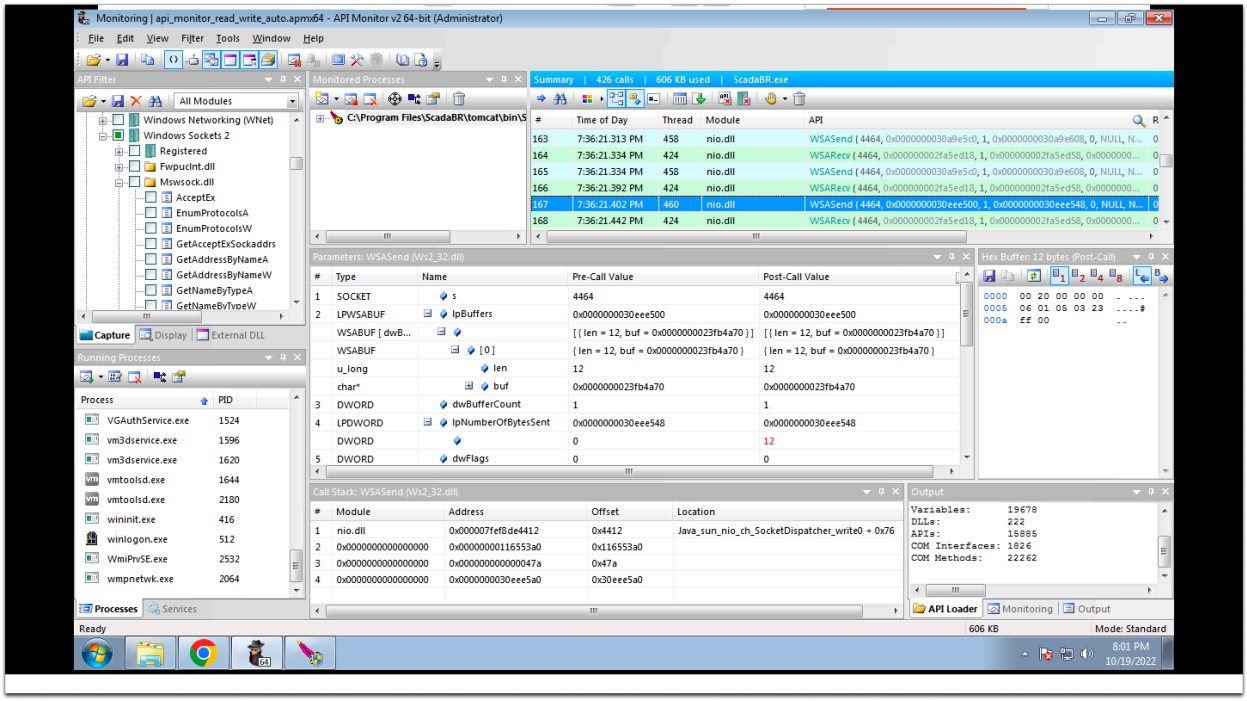
After that, I tried API Monitor (api_monitor_read_write_auto.apmx64), and it shows that the arguments for sending the network traffic is exactly the same. The only difference is the content in the buffer. The content in the buffer decides if this is a READ call or a WRITE call.

Here are some examples.

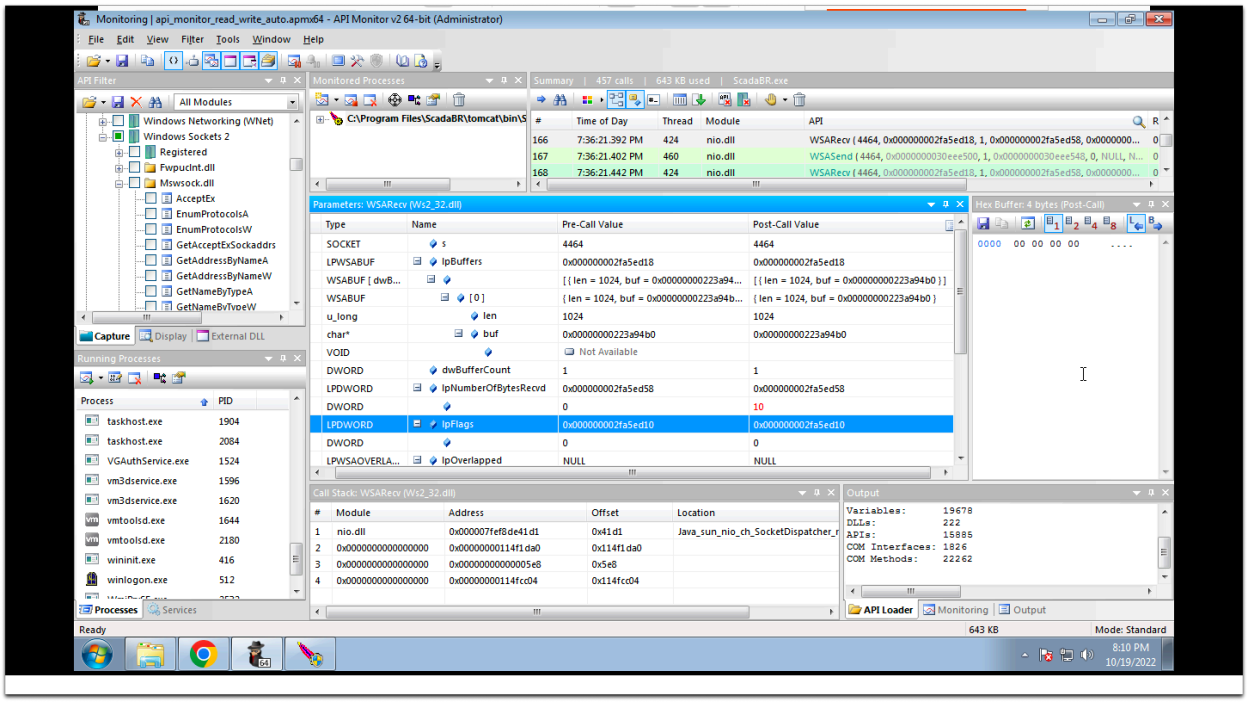
WSASend for READ



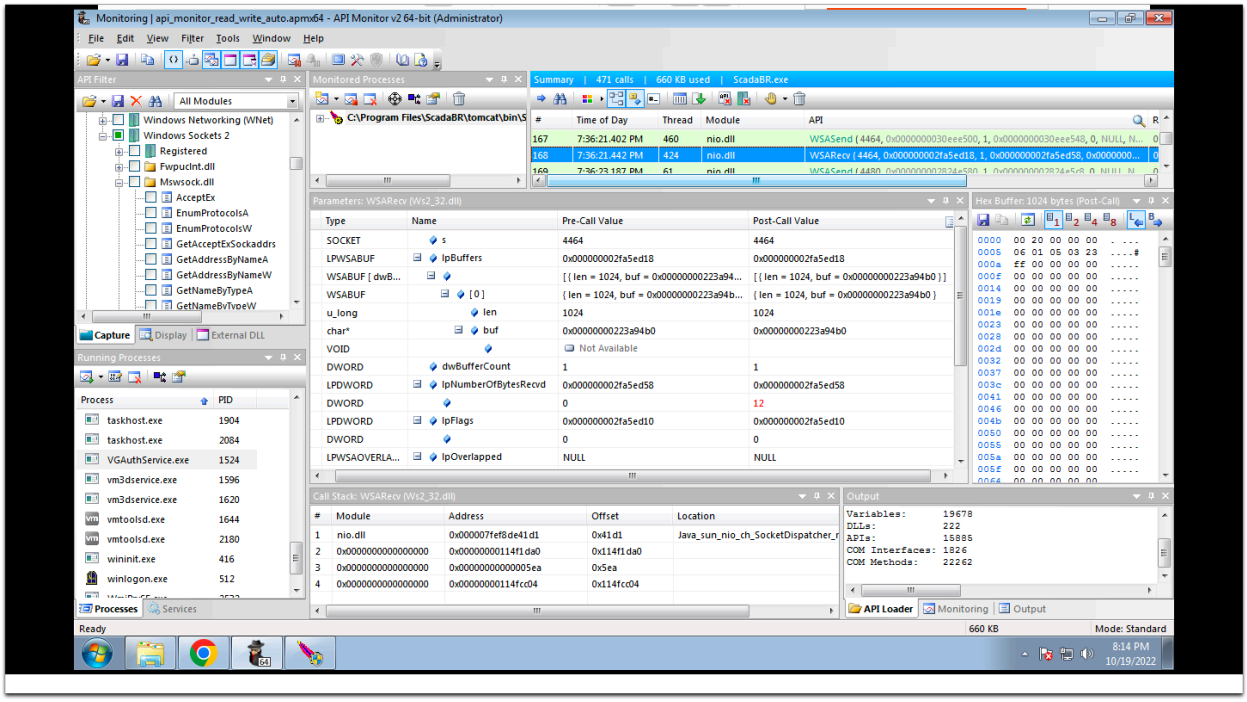
WSASend for WRITE



WSARecv for READ



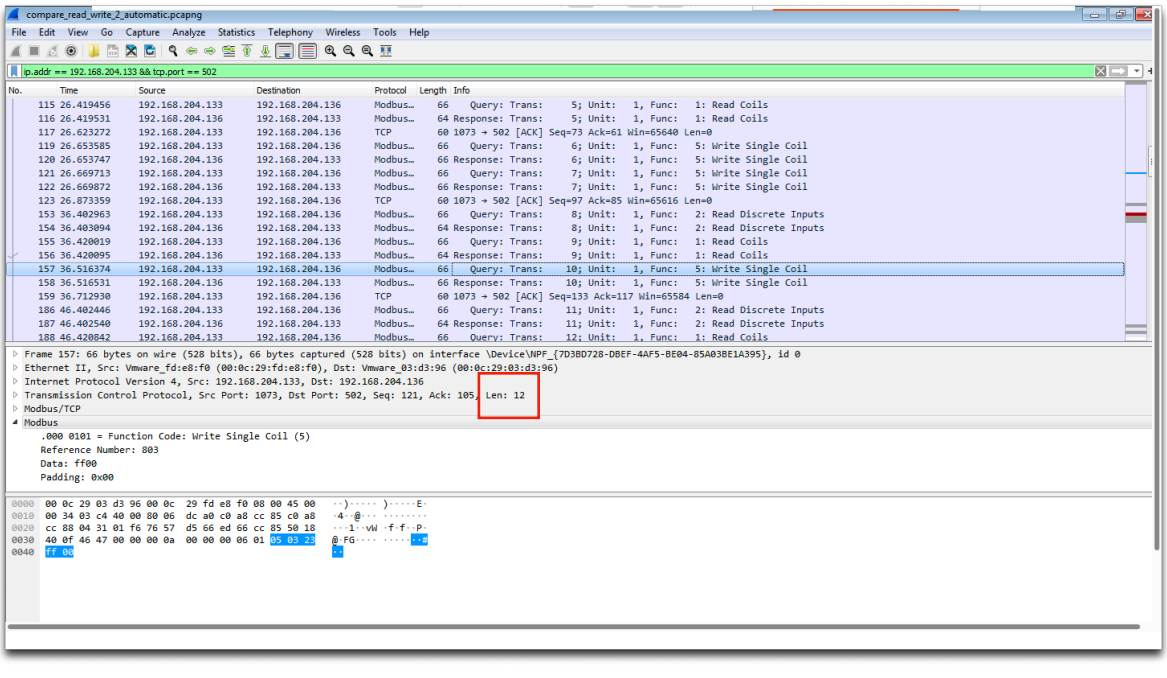
WSARecv for WRITE



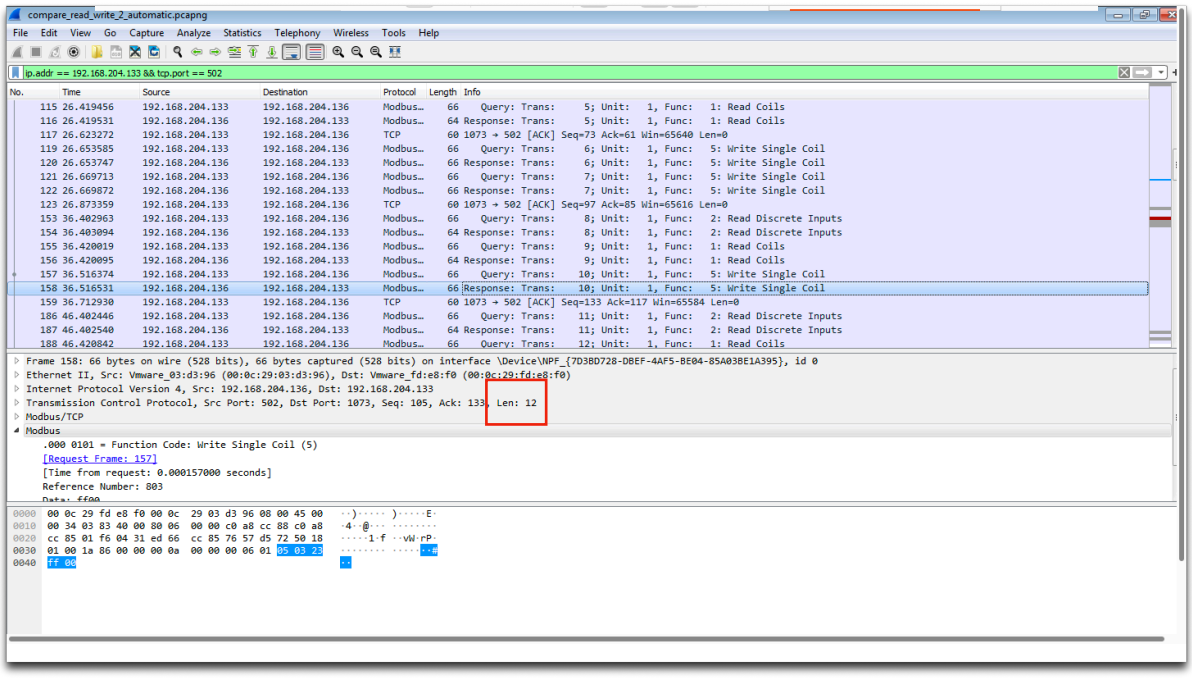
I also compared READ calls with READ calls, and still there's no single identifier that can be used to differentiate.

Length Analysis

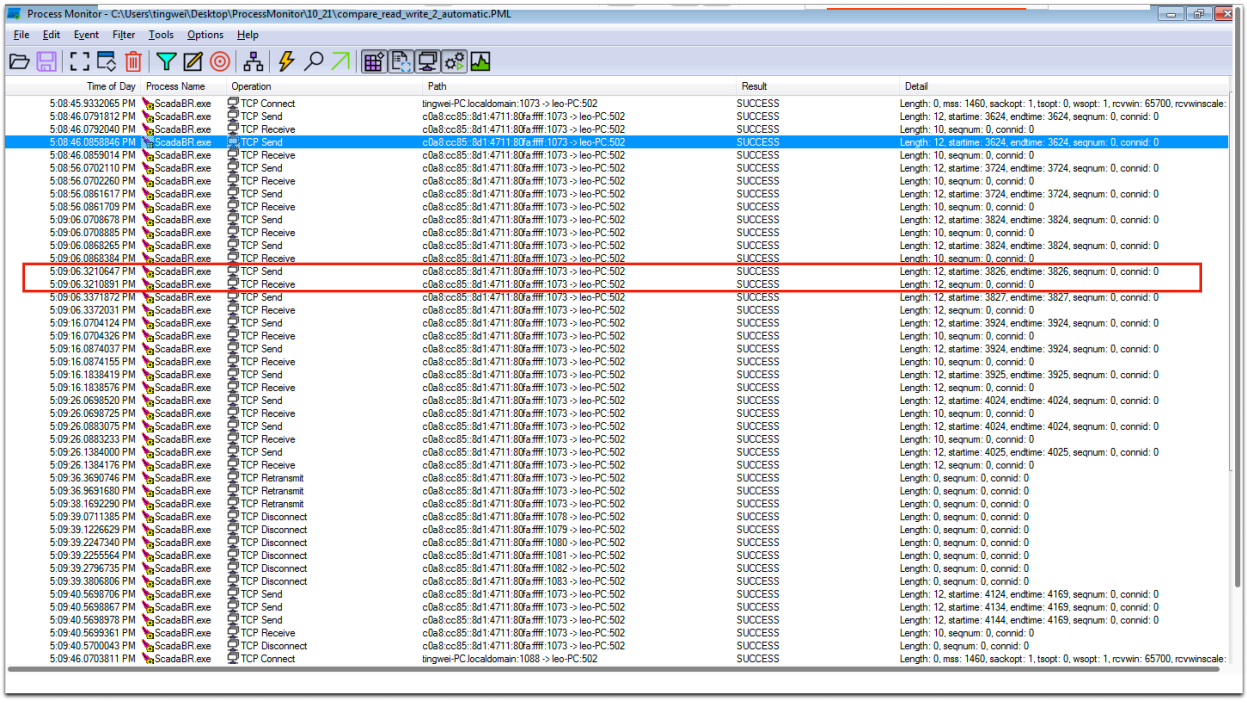
In Process Monitor, it shows the length of the packet in TCP Send and TCP Recv. And I found that when it's a WRITE call, the length of the outbound packet will always be the same as the length of the inbound packet.



The reason for that is OpenPLC simply replies the same data to ScadaBR when it receives a WRITE call. In my case, it's always 12.



In above screenshots, we see the data in write request and response is exactly the same.

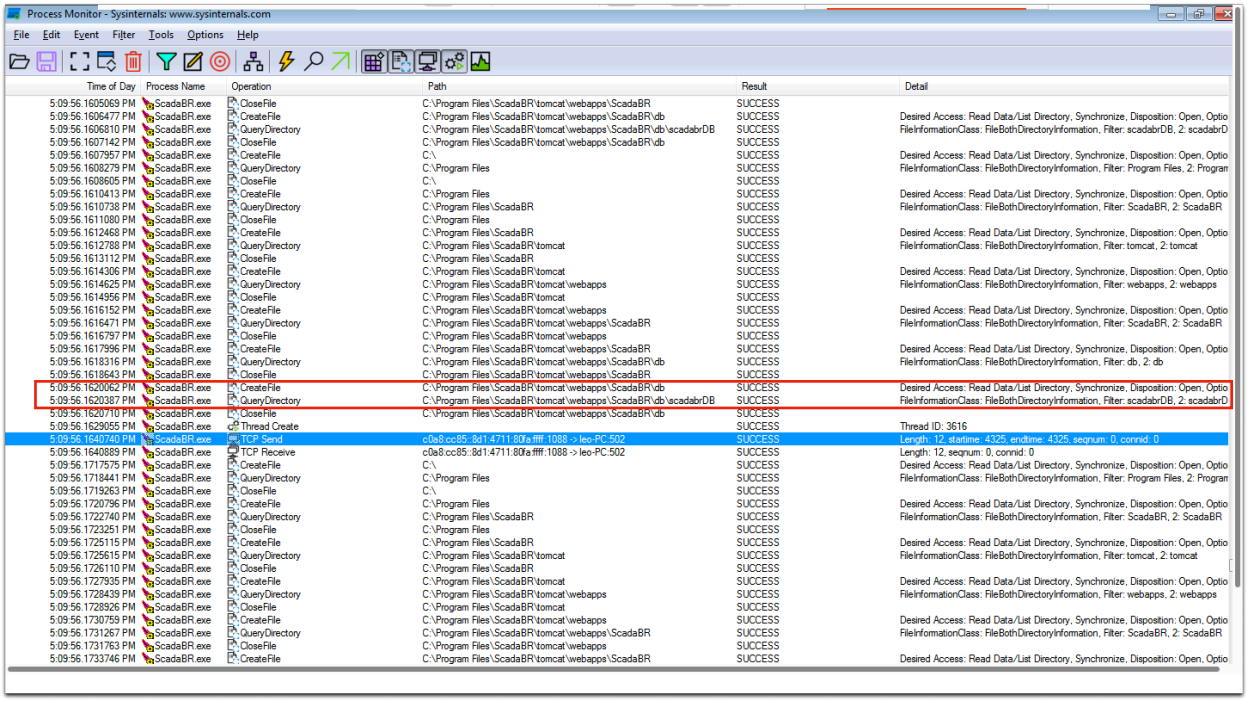


The red block above shows a pair of WRITE call request and response based on the same TCP length.

Behavior Analysis

From my observation, I found that before ScadaBR sends a WRITE call, it always check the database. It totally makes sense because data is stored in the database, and it checks the database to see if it needs to send a WRITE call.
On the other hand, READ calls are executed regularly. It does not check the database before sending the request.

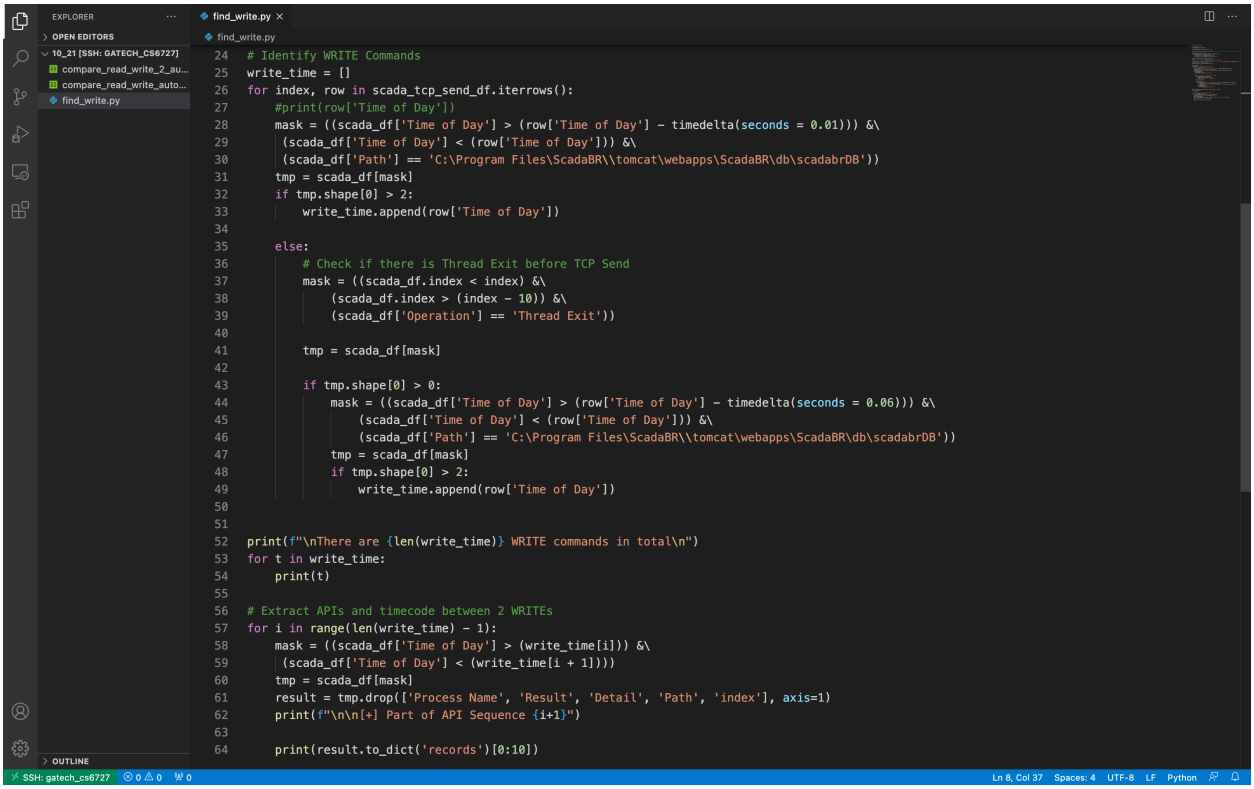
Here's how it looks before sending a WRITE call, we see it's checking scadabrDB



I collect data twice.
Network traffic and Process Monitor data is in the folder "Process Monitor/automatic_write/1st Try" and "Process Monitor/automatic_write/2nd Try"

I also created a script and use those data to look for WRITE API calls, and it successfully identified every WRITE calls.

Here's how my script looks.



```python
# Identify WRITE Commands
write_time = []
for index, row in scada_tcp_send_df.iterrows():
    #print(row['Time of Day'])
    mask = ((scada_df['Time of Day'] > (row['Time of Day'] - timedelta(seconds = 0.01))) &\
     (scada_df['Time of Day'] < (row['Time of Day'])) &\
     (scada_df['Path'] == 'C:\Program Files\ScadaBR\\tomcat\webapps\ScadaBR\db\scadabrDB'))
    tmp = scada_df[mask]
    if tmp.shape[0] > 2:
        write_time.append(row['Time of Day'])

    else:
        # Check if there is Thread Exit before TCP Send
        mask = ((scada_df.index < index) &\
            (scada_df.index > (index - 10)) &\
            (scada_df['Operation'] == 'Thread Exit'))

        tmp = scada_df[mask]

        if tmp.shape[0] > 0:
            mask = ((scada_df['Time of Day'] > (row['Time of Day'] - timedelta(seconds = 0.06))) &\
                (scada_df['Time of Day'] < (row['Time of Day'])) &\
                (scada_df['Path'] == 'C:\Program Files\ScadaBR\\tomcat\webapps\ScadaBR\db\scadabrDB'))
            tmp = scada_df[mask]
            if tmp.shape[0] > 2:
                write_time.append(row['Time of Day'])


print(f"\nThere are {len(write_time)} WRITE commands in total\n")
for t in write_time:
    print(t)

# Extract APIs and timecode between 2 WRITEs
for i in range(len(write_time) - 1):
    mask = ((scada_df['Time of Day'] > (write_time[i])) &\
     (scada_df['Time of Day'] < (write_time[i + 1])))
    tmp = scada_df[mask]
    result = tmp.drop(['Process Name', 'Result', 'Detail', 'Path', 'index'], axis=1)
    print(f"\n\n[+] Part of API Sequence {i+1}")

    print(result.to_dict('records')[0:10])
```

And the output. In the output, I only keep the timestamp and the API call.

```
twang626@bird:~/Desktop/10_21$ python3 find_write.py

There are 5 WRITE commands in total

2022-10-21 11:58:30.732544
2022-10-21 11:58:30.748723
2022-10-21 11:58:40.734190
2022-10-21 11:58:50.645582
2022-10-21 11:59:20.736315


[+] Part of API Sequence 1
[{'Time of Day': Timestamp('2022-10-21 11:58:30.732560'), 'Operation': 'TCP Receive'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.732597'), 'Operation': 'CreateFile'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.732631'), 'Operation': 'QueryDirectory'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.732666'), 'Operation': 'CloseFile'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.732790'), 'Operation': 'CreateFile'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.732823'), 'Operation': 'QueryDirectory'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.732857'), 'Operation': 'CloseFile'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.732979'), 'Operation': 'CreateFile'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.733012'), 'Operation': 'QueryDirectory'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.733046'), 'Operation': 'CloseFile'}]


[+] Part of API Sequence 2
[{'Time of Day': Timestamp('2022-10-21 11:58:30.748742'), 'Operation': 'TCP Receive'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.749707'), 'Operation': 'Thread Exit'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.750083'), 'Operation': 'WriteFile'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.750105'), 'Operation': 'WriteFile'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.750176'), 'Operation': 'WriteFile'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.750567'), 'Operation': 'CreateFile'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.750608'), 'Operation': 'QueryDirectory'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.750650'), 'Operation': 'CloseFile'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.750785'), 'Operation': 'CreateFile'}, {'Time of Day': Timestamp('2022-10-21 11:58:30.750819'), 'Operation': 'QueryDirectory'}]
```