Traces

I ran Process Monitor from the start of ScadaBR service manager, and lasts more than 20 minutes, the update frequency of ScadaBR (read) is 5 seconds (twice every.5 seconds, Read Discrete and Read Coils) , and we got 2,540,406 API calls in total.

Filtered Trace of just SCADABR stuff (alone) unmodified

Stores in the file “ScadaBREXE.CSV”

```
twang626@bird:~/Desktop/10_25$ head ScadaBREXE.CSV
"Time of Day","Process Name","Operation","Path","Result","Detail"
"10:10:21.8032717 AM","ScadaBR.exe","Process Start","","SUCCESS","Parent PID: 492, Command line: ""C:\Program Files\Sca
;aBR\tomPath=C:\Program Files\Common Files\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem
;C:\Windwindows_tracing_logfile=C:\BVTBin\Tests\installpackage\csilogfile.log"l
"10:10:21.8032821 AM","ScadaBR.exe","Thread Create","","SUCCESS","Thread ID: 2044"
"10:10:21.8722590 AM","ScadaBR.exe","Load Image","C:\Program Files\ScadaBR\tomcat\bin\ScadaBR.exe","SUCCESS","Image Bas
e: 0x13f710000, Image Size: 0x24000"
"10:10:21.8723702 AM","ScadaBR.exe","Load Image","C:\Windows\System32\ntdll.dll","SUCCESS","Image Base: 0x76e90000, Ima
ge Size: 0x19f000"
"10:10:21.8725530 AM","ScadaBR.exe","CreateFile","C:\Windows\Prefetch\SCADABR.EXE-A99A12FB.pf","NAME NOT FOUND","Desire
d Access: Generic Read, Disposition: Open, Options: Synchronous IO Non-Alert, Attributes: n/a, ShareMode: None, Allocat
ionSize: n/a"
"10:10:21.8728262 AM","ScadaBR.exe","RegOpenKey","HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Executio
n Options","SUCCESS","Desired Access: Query Value, Enumerate Sub Keys"
"10:10:21.8729022 AM","ScadaBR.exe","RegQueryValue","HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execu
tion Options\DisableUserModeCallbackFilter","NAME NOT FOUND","Length: 1,024"
"10:10:21.8729369 AM","ScadaBR.exe","RegOpenKey","HKLM\System\CurrentControlSet\Control\Session Manager","REPARSE","Des
ired Access: Read"
"10:10:21.8729679 AM","ScadaBR.exe","RegOpenKey","HKLM\System\CurrentControlSet\Control\Session Manager","SUCCESS","Des
ired Access: Read"
```

Trace containing the other direct dependencies

Process Monitor cannot do “or” operation on its filter, so I created a Python script to filter the raw captured traces.

The raw traces of all API calls stores in file “All_API_Calls.CSV”
I used my script “get_scadabr_and_dependency_api_calls.py” to filter out unrelated API calls.
The filter I used is

(Process Name contains scadabr) ||
(Process Name is svchost.exe && Path contains scada) ||
(Process Name is services.exe && (Detail contains scada || Path contains scada))

The output file is “ScadaBR_Dependencies.csv”

```
twang626@bird:~/Desktop/10_25$ head ScadaBR_Dependencies.csv
Time of Day,Process Name,Operation,Path,Result,Detail
10:10:21.2064452 AM,ScadaBRw.exe,RegQueryKey,HKLM,SUCCESS,"Query: HandleTags, HandleTags: 0x0"
10:10:21.2064666 AM,ScadaBRw.exe,RegQueryKey,HKLM,SUCCESS,Query: Name
10:10:21.2064905 AM,ScadaBRw.exe,RegOpenKey,HKLM\SOFTWARE\Wow6432Node\Microsoft\CTF\KnownClasses,NAME NOT FOUND,Desired
 Access: Read
10:10:21.2069808 AM,ScadaBRw.exe,Thread Create,,SUCCESS,Thread ID: 4064
10:10:21.7281979 AM,services.exe,RegOpenKey,HKLM\System\CurrentControlSet\services\ScadaBR,SUCCESS,Desired Access: Read
10:10:21.7282119 AM,services.exe,RegQueryValue,HKLM\System\CurrentControlSet\services\ScadaBR\ObjectName,SUCCESS,"Type:
 REG_SZ, Length: 52, Data: NT Authority\LocalService"
10:10:21.7282254 AM,services.exe,RegCloseKey,HKLM\System\CurrentControlSet\services\ScadaBR,SUCCESS,
10:10:21.7283619 AM,services.exe,RegOpenKey,HKLM\System\CurrentControlSet\services\ScadaBR,SUCCESS,Desired Access: Read
10:10:21.7284931 AM,services.exe,RegQueryValue,HKLM\System\CurrentControlSet\services\ScadaBR\ImagePath,SUCCESS,"Type:
REG_EXPAND_SZ, Length: 128, Data: ""C:\Program Files\ScadaBR\tomcat\bin\ScadaBR.exe"" //RS//ScadaBR"
```

Parsed: Physical-Targeted Execution TRACE

Script name: find_write_api_sequence.py
Input file: ScadaBREXE.CSV
Output file 1: ‘WRITE_API_Sequences.txt'
Output file 2: ‘All_TCP_Send_and_Receive_Sequences.txt’

For the output file ‘WRITE_API_Sequences.txt,’ it contains API calls between every 2 Write Calls. In every list, the first value is the timestamp, and the second value is the API call.


```
[[1666606302934552, "TCP Send"], [1666606302934573, "TCP Receive"], [1666606302939423, "CreateFile"]
, [1666606302939478, "QueryDirectory"], [1666606302939533, "CloseFile"], [1666606302939720,
"CreateFile"], [1666606302939769, "QueryDirectory"], [1666606302939815, "CloseFile"],
[1666606302939984, "CreateFile"], [1666606302940031, "QueryDirectory"], [1666606302940078,
"CloseFile"], [1666606302940246, "CreateFile"], [1666606302940293, "QueryDirectory"],
[1666606302940339, "CloseFile"], [1666606302940508, "CreateFile"], [1666606302940555,
"QueryDirectory"], [1666606302940600, "CloseFile"], [1666606302940776, "CreateFile"],
[1666606302940827, "QueryDirectory"], [1666606302940871, "CloseFile"], [1666606302941039,
```


For the output file ‘All_TCP_Send_and_Receive_Sequences.txt,’ it contains all the TCP API Calls performed by ScadaBR.exe. I created this file because it contains the information when we are doing timing and frequency analysis. In every list, the first value is the timestamp, the second value is the API call, and the third value is the length of that TCP packet. It helps us to distinguish READ API calls from WRITE API calls.


```
[[1666606282848464, "TCP Send", "12"], [1666606282848491, "TCP Receive", "10"], [1666606282939798,
"TCP Send", "12"], [1666606282939821, "TCP Receive", "10"], [1666606287702644, "TCP Send", "12"],
[1666606287702668, "TCP Receive", "10"], [1666606287721377, "TCP Send", "12"], [1666606287721403,
"TCP Receive", "10"], [1666606292646266, "TCP Send", "12"], [1666606292646301, "TCP Receive", "10"],
[1666606292660656, "TCP Send", "12"], [1666606292660682, "TCP Receive", "10"], [1666606297736352,
"TCP Send", "12"], [1666606297736385, "TCP Receive", "10"], [1666606297776117, "TCP Send", "12"],
[1666606297776152, "TCP Receive", "10"], [1666606302637019, "TCP Send", "12"], [1666606302637043,
"TCP Receive", "10"], [1666606302647284, "TCP Send", "12"], [1666606302647300, "TCP Receive", "10"],
[1666606302934552, "TCP Send", "12"], [1666606302934573, "TCP Receive", "12"], [1666606308089717,
"TCP Send", "12"], [1666606308089740, "TCP Receive", "10"], [1666606308106427, "TCP Send", "12"],
```


It also calculates the correctness of Use_Time_DB_READ_Heuristic, compares with Use_TCP_RECIEVE _Heuristic.
With different argument values (how many lines of API calls to check, how many times of database checks performed in those lines, how many lines of “Thread Exit” check before TCP Send, how much time should be added if “Thread Exit” exists) provided, the correctness changes between 76.37% to 94.51%. And the false positive rates changes between 6.04% to 25.27%.

Classify TCP Send as a WRITE call if there are more than 8 database check in the previous 250 lines.

```
twang626@bird:~/Desktop/10_25$ python3 find_write_api_sequence.py
[+] Running USE_TIME_DB_READ_HEURISTIC Approach ... Finished!!
It takes 44.66 seconds
[+] Running USE_TCP_RECEIVE_HEURISTIC Approach ... Finished!!
It takes 1.01 seconds
USE_TIME_DB_READ_HEURISTIC Statistics
        Correctness Rate: 76.37%
        False Positive Rate: 6.04%

There are 182 WRITE commands in total


[+] Exporting Files ... Finished!!
It takes 2.43 seconds
```

Classify TCP Send as a WRITE call if there are more than 7 database check in the previous 300 lines.

```
twang626@bird:~/Desktop/10_25$ python3 find_write_api_sequence.py
[+] Running USE_TIME_DB_READ_HEURISTIC Approach ... Finished!!
It takes 41.47 seconds
[+] Running USE_TCP_RECEIVE_HEURISTIC Approach ... Finished!!
It takes 0.79 seconds
USE_TIME_DB_READ_HEURISTIC Statistics
        Correctness Rate: 94.51%
        False Positive Rate: 25.27%

There are 182 WRITE commands in total


[+] Exporting Files ... Finished!!
It takes 2.16 seconds
```

Timing and Frequency Analysis

Script name: timing_and_frequency_analysis.py
Input file: 'All_TCP_Send_and_Receive_Sequences.txt'

Output

```
twang626@bird:~/Desktop/10_25$ python3 timing_and_frequency_analysis.py
Write_to_Write_Timing:
        Mean: 6546.36 milliseconds
        Standard Deviation: 5718.26 milliseconds
Read_to_Write_Timing:
        Mean: 96.47 milliseconds
        Standard Deviation: 52.61 milliseconds
Read_to_Read_Timing:
        Mean: 1465.96 milliseconds
        Standard Deviation: 2284.02 milliseconds
0.27286 Write calls per TCP Send (WRITE / TCP Send)
0.0002 Write calls per API Call (WRITE / All API Calls)
```