Process Behavior Analysis with Focused Clustering: Milestone 3

Isaac Amann

4/21/2024

Contents

1 Progress

1.1 Backend Server

1.1.1 API Rate Limits

Implemented a token bucket API rate limiter using the bucket4j library. Should prevent bad actors from brute forcing passwords and API tokens. The limits were set to 15 per minute for login requests and 200 per minute for regular API calls

1.1.2 DBSCAN with Cosine Similarity

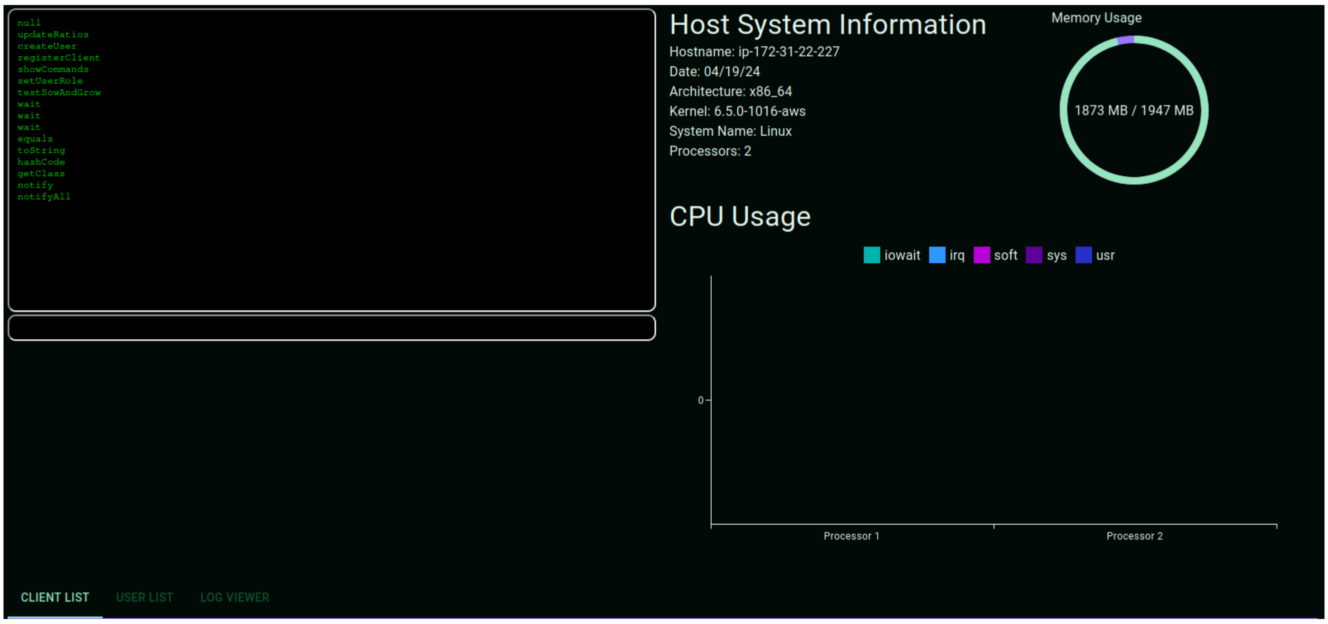
Created a python script to be run on a separate Java thread on the backend similar to how the SowAndGrow program was executed. Originally intended on finding a Java library for the clustering algorithm, but most libraries only had DBSCAN using euclidean distance. The sklearn Python library supports DBSCAN with cosine similarity and was simple to implement in under 50 lines of code.

1.1.3 Logging System

Implemented a logging system that stores log entries in memory that can be accessed through API calls. I wrote it to look similar to the default logs that come with the Spring framework with fields for log level, source, and message. This allows information about events on the system to be viewed from the admin dashboard on the frontend.

1.2 Frontend Server

1.2.1 Admin Dashboard



```
null
updateRatios
createUser
registerClient
showCommands
setUserRole
testSowAndGrow
wait
wait
wait
equals
toString
hashCode
getClass
notify
notifyAll
```

Host System Information

Hostname: ip-172-31-22-227
Date: 04/19/24
Architecture: x86_64
Kernel: 6.5.0-1016-aws
System Name: Linux
Processors: 2

Figure 1: Updated admin dashboard

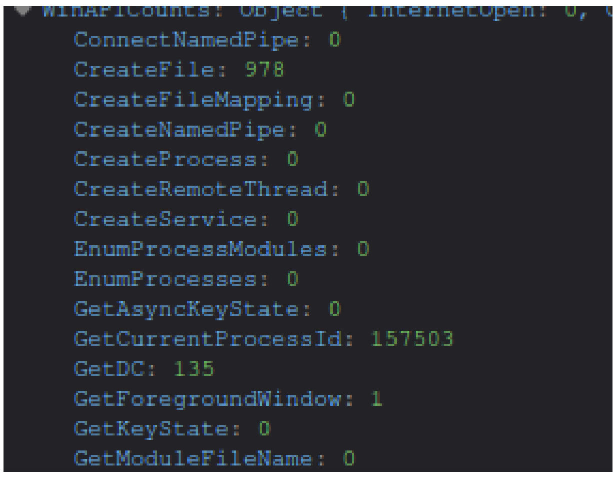
Added the following features to the admin dashboard:

- Information about the host system including hostname, system architecture, and kernel version.
- Memory usage gauge
- Tab menu containing tables for registered clients, users, and log entries

The styling of the command console was also changed to be more visually appealing. The text input was also changed so that it keeps focus on the element after entering a command without the user having to click on the text box again.

2 Challenges

2.1 API Monitor Concurrency Problem



```
▼ WinAPICounts: Object { InternetOpen: 0, 
    ConnectNamedPipe: 0
    CreateFile: 978
    CreateFileMapping: 0
    CreateNamedPipe: 0
    CreateProcess: 0
    CreateRemoteThread: 0
    CreateService: 0
    EnumProcessModules: 0
    EnumProcesses: 0
    GetAsyncKeyState: 0
    GetCurrentProcessId: 157503
    GetDC: 135
    GetForegroundWindow: 1
    GetKeyState: 0
    GetModuleFileName: 0
```

Figure 2: Datapoint showing a strange number of calls to GetCurrentProcessId
The API monitor may have a concurrency issue since multiple threads of a process may attempt to increment a counter value at the same time. A datapoint for runtimebroker.exe shows it made over 150,000 calls to GetCurrentProcessId. While it is difficult to verify the correct number of calls, this seems like an excessive number. This issue could be resolved by adding a mutex lock for each hook function that would have to be claimed before modifying the counter value. This fix could cause other issues since it could create deadlocks causing programs to hang or crash.

2.2 UWP Applications

Certain applications were crashing when the API monitor was attaching to them. These applications were all developed using Universal Windows Platform. I found through forum posts that others were experiencing the same issue when injecting DLL's. The work around to this was to just exclude UWP applications. To identify UWP applications, GetPackageFamilyName can be called on the target process to see if it fails. See code snippet below:

```
//Check that process is not a Windows store app
//They are sandboxed and crash upon DLL injection
//Using GetPackageFamilyName to see if it returns error, see following
    stack overflow post
//https://stackoverflow.com/questions/52207484/determine-if-c-application-
    is-running-as-a-uwp-app-with-legacy-support
UINT32 packageNameLength = 0;
LONG packageError = GetPackageFamilyName(process, &packageNameLength, NULL)
    ;
if (packageError == ERROR_INSUFFICIENT_BUFFER)
{
    std::cout << "Skipping UWP process\n";
}
```

3 Work for Next Milestone

3.1 Windows API Monitor

- More stability improvements
- Solve possible concurrency issue

3.2 Data Collection and Results

- Data collection with final version of API monitor
- Collect data from some malware samples
  - Will likely set up a secure virtual machine with limited network access
  - Will likely focus on ransomware since it is most likely to produce good results. Most ransomware should make a large number of file access Windows API calls
- Attempt to produce some results from clustering algorithms
  - Hopefully show that related malware samples cluster together

4 Work Log

4.1 3/29/2024

Backend Changes: Implemented unit tests for authentication methods

4.2 3/31/2024

Backend Changes: Created python script for DBSCAN with cosine similarity. Implemented logging service and API functions for accessing log entries. Frontend Changes: Updated styling of command console. Added ability to submit analysis jobs for DBSCAN with cosine similarity.

4.3 4/1/2024

Frontend Changes: Began work on system monitor component

4.4 4/2/2024

Frontend Changes: Setup memory usage gauge. Implemented CPU usage graph. Added system information to system monitor component.

4.5 4/3/2024

Backend Changes: Fixed parameters used for mpstat command for system info API call.

4.6 4/6/2024

Frontend Changes: Created log viewer table on admin dashboard. Added tab menu to admin dashboard. Created client datagrid to admin tab menu.

4.7 4/10/2024

Frontend Changes: Added documentation page with PDF downloads. Added user list to admin tab menu.

4.8 4/11/2024

Backend Changes: Fixed date formats on JSON objects returned by API calls

4.9 4/14/2024

Frontend Changes: Added notifications for login and job submission Backend Changes: Implemented API rate limits API Monitor Changes: Excluded UWP applications from monitoring. UWP (Windows Store) applications use some sort of sandboxing and crash upon DLL injection.

4.10 4/17/2024

Frontend Changes: Changed job viewer to allow for scrolling