Synchronous Fire Data Collection Report

System Information

For testing the synchronous firing feature, data was collected on the following system:

OS: Arch Linux x86_64
Host: Latitude 7280
Kernel: 5.10.12-hardened1-1-hardened
CPU: Intel i5-7300U (2) @ 3.500GHz
GPU: Intel HD Graphics 620
Memory: 7826MiB
Disk: SK Hynix SC308 SATA 256GB

Test Information

- The test was ran using the same network model input file, for both synchronous fire and no synchronous fire.
- The test was ran using the same exploit parser input file, with the exception of the "time_advance" exploit, which included a flag to indicate synchronous fire.
- All data points were run in an environment progressing from time 0 to a time of 12 months.
- Time was progressed in 1 month increments.
- All services were to be performed prior to advancing time, with no restrictions on asset or service priority.
- Each progression of time incremented all asset qualities: ac_odometer by 10,000 miles, vacuum_odometer by 10,000 miles, brake_months by 1 month, exhaust_months by 1 month, and the time_flag by 1 month.

Data

The following table illustrates the results collected regarding the State Space, with units of "states":

	1 car, no sync	1 car, sync	2 car, no sync	2 car, sync	2 car state space reduction factor
1 Service	39	39	2145	301	7.13
2 Services	61	61	4697	433	10.85
3 Services	105	105	12705	961	13.22
4 Services	209	209	43681	3329	13.12

As shown by the table, the synchronous fire feature led to a state space reduction by a factor of over 7x to over 13x. In terms of the number of states, the synchronous fire eliminated 1844 states for the environment with one service implemented. These 1844 states were states that were realistically infeasible; there can exist no state in which one asset can progress through time at a different pace than another. When progressing through 12 time steps, the number of states eliminated continued to grow. However, the state space reduction factor did not grow from 3 services to 4 services. This is presumed to be due to the increased number of permutations. This will be discussed further in the "Future Testing" section.

The following table illustrates the results collected regarding the Runtime, with units of “seconds”:

	1 car, no sync	1 car, sync	2 car, no sync	2 car, sync	2 car time saving factor from sync fire
1 Service	0.00659399	0.008201	0.712052	0.107292	6.64
2 Services	0.022439	0.046315	1.630228	0.16007	10.18
3 Services	0.034439	0.041392	4.968957	0.356927	13.92
4 Services	0.088275	0.096118	20.43759	1.46965	13.91

The primary purpose of collecting runtime is to analyze the performance impact of the synchronous fire feature. From the previous states table, it is shown that the number of states is identical for 1 asset, regardless of synchronous firing. This is as expected, since there is nothing to synchronize with only 1 asset. However, by running the compliance graph tool with synchronous fire enabled for only 1 asset, this effectively creates a “control group” to be utilized for performance analysis.

When comparing the 1 car columns, the timing without the synchronous fire is lower than the timing with the synchronous fire for every instance. For 2 services, the synchronous fire feature took over twice as long. For systems where no synchronization is necessary, there will be performance impacts by including the feature; it is better to only use synchronous fire when there are multiple assets to be synchronized.

While there is a performance impact by including the feature, there is an increase in performance for every instance in the 2 car column. The time saving factor ranges from over 6x to over 13x. Even though there is an impact by including the feature, it is still better in terms of performance to include the synchronous firing when it is applicable.

Future Testing

From the data collected, the state space reduction factor did not increase from 3 services to 4 services. While more services means more states will be eliminated through the synchronous fire feature, there is also the additional state generation for the extra services. When the number of services increases, so do the permutations in which services can be run in regards to order - both of service type and asset. Future testing could reveal if there is a theoretical maximum, or a point in which the increase in permutation reaches a saturation point with respect to the state elimination.

Other testing could include the state space effect of more assets. By adding an additional 1, 2, or 3 assets, the state space reduction factor could be examined further. As mentioned, with more services, there is a greater number of permutations possible. It is likely the same will occur with more assets, where more assets will cause an increase in states due to the permutations. A saturation point could be examined from this perspective as well.

Conclusion

While the synchronous fire feature does have additional computation requirements, the feature led to a speedup in all instances where synchronization could be applied. By testing 1 and 2 assets, and 1 to 4 services on this small-scale and observing a speedup in all instances, it is probable that the speedup will continue to be observable on larger scale tests. In terms of the number of states, the reduction in state space is expected to ease future analysis, and is expected to lower the computation required for doing so.