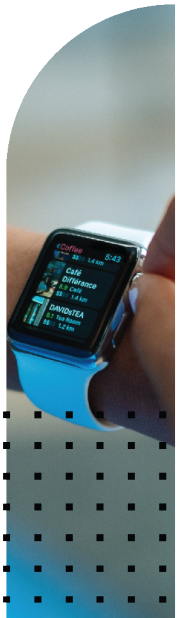
Performance Benchmarking

FortiSOAR 7.0.1



FORTINET DOCUMENT LIBRARY
https://docs.fortinet.com

FORTINET VIDEO GUIDE
https://video.fortinet.com

FORTINET BLOG
https://blog.fortinet.com

CUSTOMER SERVICE & SUPPORT
https://support.fortinet.com

FORTINET TRAINING & CERTIFICATION PROGRAM
https://www.fortinet.com/support-and-training/training.html

NSE INSTITUTE
https://training.fortinet.com

FORTIGUARD CENTER
https://www.fortiguard.com

END USER LICENSE AGREEMENT
https://www.fortinet.com/doc/legal/EULA.pdf

FEEDBACK
Email: techdoc@fortinet.com

July, 2021
FortiSOAR 7.0.1 Performance Benchmarking
00-400-000000-20210112

TABLE OF CONTENTS

Change Log

Date	Change Description
2021-07-21	Initial release of 7.0.1

FortiSOAR Performance Benchmarking for v7.0.1

This document details the performance benchmark tests conducted in Fortinet labs. The performance benchmarking tests were performed on FortiSOAR version 7.0.1 Build 628.

The objective of this performance test is to measure the time taken to create alerts in FortiSOAR, and complete the execution of corresponding playbooks on the created alerts in the following cases:

- Single-node FortiSOAR appliance
- Cluster setup of FortiSOAR

The data from this benchmark test can help you in determining your scaling requirements for a FortiSOAR instance to handle the expected workload in your environment.

Single Invocation Test for the single-node FortiSOAR appliance

Environment

FortiSOAR Virtual Appliance Specifications

Component	Specifications
CPU	8 CPUs
Memory	32 GB
Storage	250 GB virtual disk, with IOPS 2400, attached to an AWS Instance.

Operating System Specifications

Operating System	Kernel Version
CentOS 7	3.10.0-1160.6.1.el7.x86_64

External Tools Used

Tool Name	Version
Zabbix	4.2.3
Internal Script to gather data	

Pre-test conditions on both the standalone FortiSOAR machine and the FortiSOAR High Availability (HA) cluster

At the start of each test run -

- The test environment contained zero alerts.
- The test environment contained only the FortiSOAR built-in connectors such as IMAP, Utilities, etc.
- The system playbooks were deactivated such as, alert assignment notification, sla calculation,etc, and there were no running playbooks.
- The playbook execution logs were purged.
- Configured tunables as follows:
  - Changed celery workers to 16
  - Elastic heaps size to 8GB
  - Changes related to PostgresSQL:
    - max_connections = 1200
    - shared_buffers = 5GB
    - effective_cache_size = 15GB
    - maintenance_work_mem = 2GB
    - checkpoint_completion_target = 0.9
    - wal_buffers = 16MB
    - default_statistics_target = 500
    - random_page_cost = 4
    - effective_io_concurrency = 2
    - work_mem = 546kB
    - min_wal_size = 4GB
    - max_wal_size = 16GB
    - max_worker_processes = 8
    - max_parallel_workers_per_gather = 4
    - max_parallel_workers = 8
    - max_parallel_maintenance_workers = 4
  - Changes related to NGINX:
    - worker_processes auto;
      #or should be equal to the CPU core, you can use '`grep processor /proc/cpuinfo | wc -l`' to find; auto does it implicitly
    - worker_connections 1024;
      # default is 768; find optimum value for your server by 'ulimit -n'
    - access_log off;
      # to boost I/O on HDD we can disable access logs
      # this prevents nginx from logging every action in a log file named 'access.log'
    - keepalive_timeout 15;
      # default is 65;
      # server will close the connection after this time (in seconds)
    -

```
sudo mkdir -p /data/nginx/cache
```

```
proxy_cache_path /data/nginx/cache keys_zone=my_zone:10m inactive=1d;

server {

   ...

   location /api-endpoint/ {

       proxy_cache my_zone;

       proxy_cache_key "$host$request_uri$http_authorization";

       proxy_cache_valid 404, 302 1m;

       proxy_cache_valid 200 1d;

       add_header X-Cache-Status $upstream_cache_status;

   }

   ...

}
```

In a production environment the network bandwidth, especially for outbound connections, to applications such as VirusTotal might vary, which could affect the observations.

Test setup for the single-node FortiSOAR appliance

This test has been invoked on a standalone FortiSOAR machine with configurations mentioned in the Environment section.

Tests performed

Test 1: Perform Ingestion in FortiSOAR using the FortiSIEM Ingestion Playbook

This test is executed by manually triggering FortiSIEM Ingestion playbook that creates alerts in FortiSOAR.

Steps followed

1. Created the alerts using the FortiSIEM Ingestion playbook. You can download the JSON for the sample playbooks (PerfBenchmarking_Test01_PB_Collection_7_0_1.zip) so that you can run the same tests in your environment to see the performance in your version/hardware platforms. Or, if you want to do some additions that are specific to your environment, you can also tweak the existing playbooks.
2. Once the alerts are created, measured the total time taken to create all the alerts in FortiSOAR.

Observations

The data in the following table outlines the number of alerts ingested and the total time taken to ingest those alerts.

Single Invocation Test run on a single-node FortiSOAR appliance

Number of alerts created in FortiSOAR	Total time (in seconds) taken to create all alerts in FortiSOAR	Total number of playbooks executed in FortiSOAR
1	0.35	1
5	0.58	1
10	0.87	1
25	1.86	1
50	3.38	1
100	7.11	1

Note: Once this test is completed, refer to the pre-test conditions before starting a new test.

Test 2: Perform Ingestion in FortiSOAR using the FortiSIEM Ingestion Playbook after the alerts are created and after triggering "Extraction" playbooks

This test is executed by manually triggering FortiSIEM Ingestion playbook that creates alerts in FortiSOAR. Once the alerts are created in FortiSOAR, an "Extraction" playbook is triggered and the total time taken for all the extraction playbooks to complete their execution is calculated.

Steps followed

1. Created the alerts using the FortiSIEM Ingestion playbook.
2. Once the alerts are created, the "Extraction" playbooks are triggered. You can download the JSON for the sample playbooks (PerfBenchmarking_Test02_PB_Collection_7_0_1.zip) so that you can run the same tests in your environment to see the performance in your version/hardware platforms. Or, if you want to do some additions that are specific to your environment, you can also tweak the existing playbooks.
   The playbooks perform the following steps:
   a. Declares variables using the "Set Variable Step".
   b. Updates the existing indicator list using mapping.
   c. Retrieves indicators from the source data of the alert.
   d. Creates indicators in the "Indicators" Module.

e. Links alerts to the indicators.
f. Updates the state of the alerts.

Observations

The data in the following table outlines the number of alerts ingested, the total time taken to ingest those alerts, and the total time taken for all the triggered playbooks to complete their execution.

Single Invocation Test run on a single-node FortiSOAR appliance

Number of alerts created in FortiSOAR	Total time (in seconds) taken to create all alerts in FortiSOAR	Total number of play-books executed in FortiSOAR
1	1.86	2
5	2.33	6
10	3.42	11
25	8.15	26
50	14.87	51
100	26.85	101

Note: Once this test is completed, refer to the pre-test conditions before starting a new test.

Test 3: Perform Ingestion in FortiSOAR using the FortiSIEM Ingestion Playbook after the alerts are created and after triggering "Extraction" and "Enrichment" playbooks

The test was executed using an automated testbed that starts FortiSIEM ingestion which in turn created alerts in FortiSOAR. Once the alerts are created in FortiSOAR, "Extraction" and "Enrichment" playbooks are triggered and the total time taken for all the extraction and enrichment playbooks to complete their execution is calculated.

The setup for this test is exactly the same, however this test additionally requires the "VirusTotal" connector to be configured.

Steps followed

1. Created the alerts using the FortiSIEM Ingestion playbook.
2. Once the alerts are created, the "Extraction" playbooks are triggered. You can download the JSON for the sample playbooks (PerfBenchmarking_Test03_PB_Collection_7_0_1.zip) so that you can run the same tests in your environment to see the performance in your version/hardware platforms. Or, if you want to do some additions that are specific to your environment, you can also tweak the existing playbooks.
   The playbooks perform the following steps:
   a. Declares variables using the "Set Variable Step".
   b. Updates the existing indicator list using mapping.
   c. Retrieves indicators from the source data of the alert.
   d. Creates indicators in the "Indicators" Module.

e. Links alerts to the indicators.
f. Updates the state of the alerts.

3. Once the indicators are extracted, the "Enrichment" playbooks are triggered and they perform the following steps:
   a. Matches the IP in an internal subnet through the "Utilities" subnet.
   b. Validates whether the IP is Private or Public.
   c. Performs enrichment using the "Utilities" connector, if the IP is "Private".
   d. Performs enrichment using the "VirusTotal" connector, if the IP is "Public".
   e. Updates the indicator status based on the IP's vulnerability.
   f. Updates the state of the indicator.

Observations

The data in the following table outlines the number of alerts ingested, the total time taken to ingest those alerts, and the total time taken for all the triggered playbooks to complete their execution.

Single Invocation Test run on a single-node FortiSOAR appliance

Number of alerts created in FortiSOAR	Total time (in seconds) taken to create all alerts in FortiSOAR	Total number of playbooks executed in FortiSOAR
1	5.36	4
5	8.48	16
10	15.97	31
25	34.44	76
50	1 minute 7 seconds	151
100	2 minutes 10 seconds	301

Note: Once this test is completed, refer to the pre-test conditions before starting a new test. Also, note that enrichment of playbooks makes API calls over the Internet, and the times mentioned in this table to execute playbooks is inclusive of this time.

Sustained Invocation Test for the single-node FortiSOAR appliance

A sustenance test was also conducted with the configuration as defined in "Test 2", i.e., the test is executed by manually triggering the FortiSIEM Ingestion playbook, "FortiSIEM -> Ingest 100 Alerts", which creates alerts in FortiSOAR. Once the alerts are created in FortiSOAR, an "Extraction" playbook is triggered and the total time taken for all the extraction playbooks to complete their execution is calculated.

Number of alerts: **100/min**

Duration: **12 hours**

Playbooks configured: **As defined in Test 2 comprising of "Ingestion" and "Indicator Extraction" playbooks**.

Total number of playbooks executed: **72821**

Results

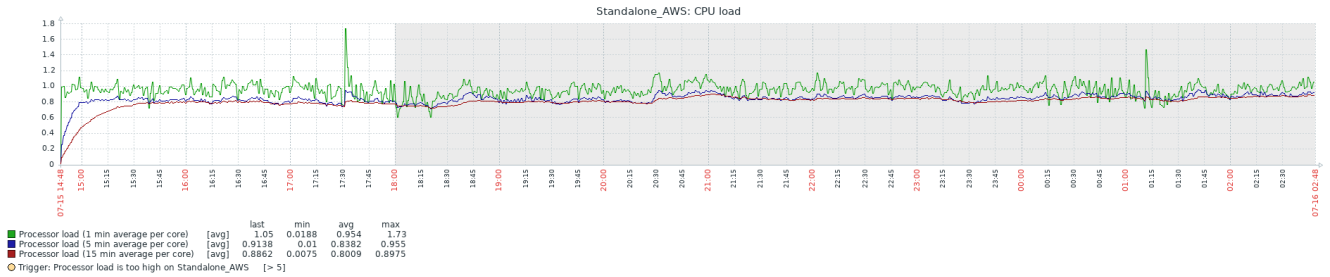
The system performed well under the sustained load. All 72100 alerts were successfully ingested and all the extraction playbooks were successfully completed without any queuing.

Graphs

The following graphs are plotted for the vital statistics for the system that was under test during the period of the test run.

CPU Load Average Utilization Graph

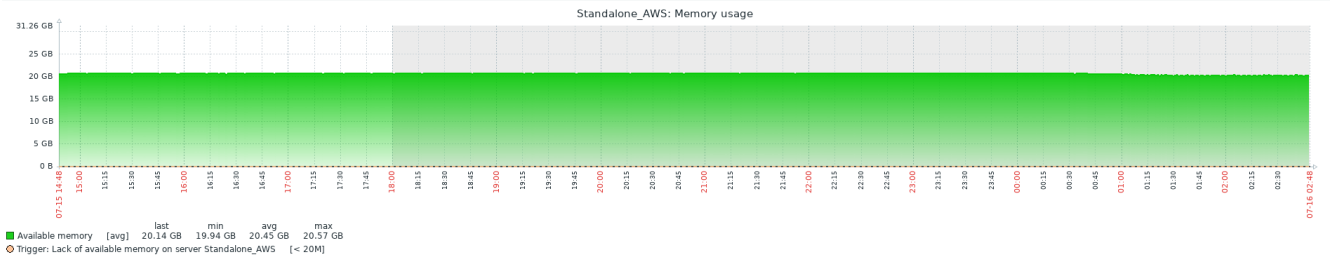
Analysis of CPU load average utilization when the test run was in progress on the appliance:



Using the system resources specified in the "Environment" and "Pre-Test Conditions" sections, it was observed that while the "Sustenance Test" was running the CPU utilization was normal and the performance of the system did not get impacted.

Memory Utilization Graph

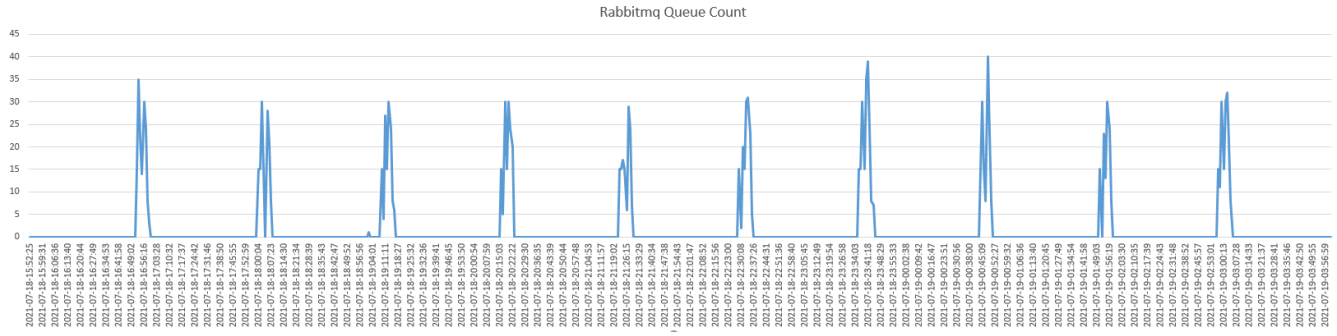
Analysis of memory utilization when the test run was in progress on the appliance:



Using the system resources specified in the "Environment" and "Pre-Test Conditions" sections, it was observed that while the "Sustenance Test" was running the Memory utilization was around 20GB.

RabbitMQ PB Queue Count Graph

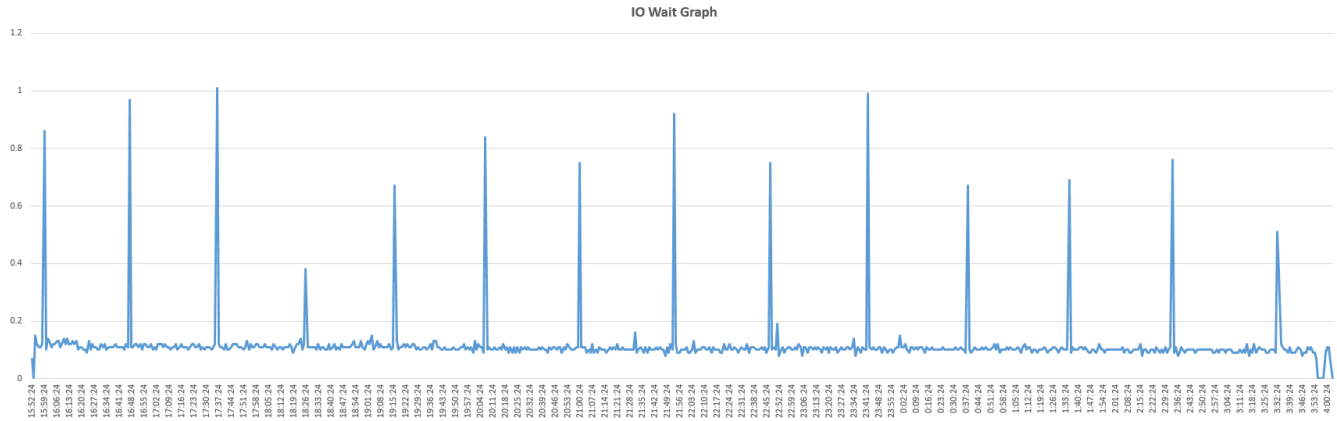
Analysis of RabbitMQ PB Queue Count when the test run was in progress on the appliance:



Using the system resources specified in the "Environment" and tunables configured as mentioned in the "Pre-Test Conditions" sections, it was observed that during the "Sustenance Test" the `rabbitmq_pb_queue` went to a maximum of 35 for some instances but eventually went back to 0. This means that no playbooks remained in queue and that all the required playbooks associated with the alerts were getting completed, i.e., alerts were created and their indicators were extracted and enriched in a minute.

IO Wait Graph

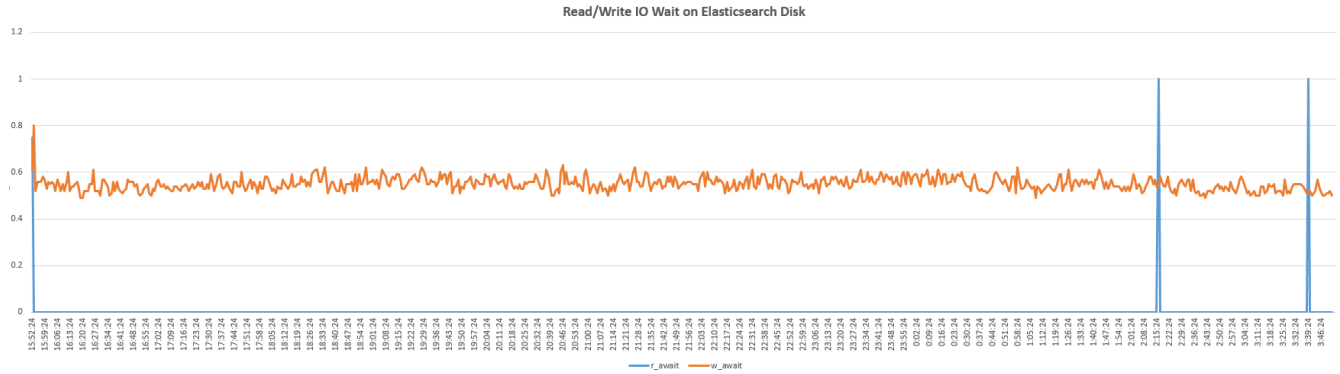
Analysis of IO Wait when the test run was in progress on the appliance:



Using the system resources specified in the "Environment" and "Pre-Test Conditions" sections, it was observed that while the "Sustenance Test" was running the IO Wait time was normal, with the average IO Wait being around 1% of the CPU idle time.

Read/Write IO Wait Graph for ElasticSearch

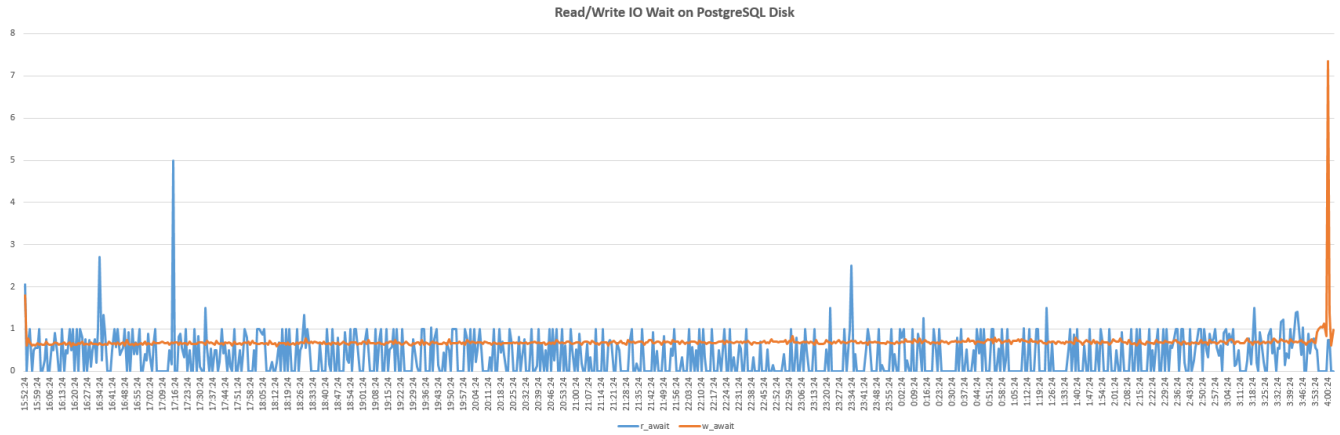
Analysis of Read/Write IO Wait for ElasticSearch when the test run was in progress on the appliance:



Using the system resources specified in the "Environment" and "Pre-Test Conditions" sections, it was observed that while the "Sustenance Test" was running the "Read" Wait for the ElasticSearch disk was almost 0 milliseconds. The "Write" Wait for the ElasticSearch disk averaged around 4 milliseconds, with the maximum wait of 8 milliseconds and the minimum wait of 4 millisecond.

Read/Write IO Wait Graph for PostgreSQL

Analysis of Read/Write IO Wait for PostgreSQL when the test run was in progress on the appliance:



Using the system resources specified in the "Environment" and "Pre-Test Conditions" sections, it was observed that while the "Sustenance Test" was running, the "Read" Wait for the PostgreSQL disk averaged around 1 millisecond, with the maximum wait of 5 milliseconds and the minimum wait of 0 milliseconds. The "Write" Wait for the PostgreSQL disk averaged around 1 millisecond, with the maximum wait of 7 milliseconds and the minimum wait of 1 millisecond.

Single Invocation Test for the High Availability (HA) active-active cluster of two FortiSOAR nodes

Test setup for the HA active-active cluster of two FortiSOAR nodes

This test has been invoked the following setup:

- Cluster of two FortiSOAR machines that are joined in the Active-Active state using the FortiSOAR HA feature.
- The machines that form the HA cluster must be in the same network subnet.

Tests performed

Test 1: Perform Ingestion in FortiSOAR using the FortiSIEM Ingestion Playbook

This test is executed by manually triggering FortiSIEM Ingestion playbook that creates alerts in FortiSOAR.

Steps followed

1. Created the alerts using the FortiSIEM Ingestion playbook. You can download the JSON for the sample playbooks (PerfBenchmarking_Test01_PB_Collection_7_0_1.zip) so that you can run the same tests in your environment to see the performance in your version/hardware platforms. Or, if you want to do some additions that are specific to your environment, you can also tweak the existing playbooks. .
2. Once the alerts are created, measured the total time taken to create all the alerts in FortiSOAR.

Observations

The data in the following table outlines the number of alerts ingested and the total time taken to ingest those alerts.

Single Invocation Test run on a two-node active-active FortiSOAR appliance

Number of alerts created in FortiSOAR	Total time (in seconds) taken to create all alerts in FortiSOAR	Total number of play-books executed in FortiSOAR
1	0.39	1
5	0.64	1
10	0.91	1
25	1.97	1
50	3.55	1
100	7.53	1

Note: Once this test is completed, refer to the pre-test conditions before starting a new test.

Test 2: Perform Ingestion in FortiSOAR using the FortiSIEM Ingestion Playbook after the alerts are created and after triggering "Extraction" playbooks

This test is executed by manually triggering FortiSIEM Ingestion playbook that creates alerts in FortiSOAR. Once the alerts are created in FortiSOAR, an "Extraction" playbook is triggered and the total time taken for all the extraction playbooks to complete their execution is calculated.

Steps followed

1. Created the alerts using the FortiSIEM Ingestion playbook.
2. Once the alerts are created, the "Extraction" playbooks are triggered. You can download the JSON for the sample playbooks (PerfBenchmarking_Test02_PB_Collection_7_0_1.zip) so that you can run the same tests in your environment to see the performance in your version/hardware platforms. Or, if you want to do some additions that are specific to your environment, you can also tweak the existing playbooks.
   The playbooks perform the following steps:
   a. Declares variables using the "Set Variable Step".
   b. Updates the existing indicator list using mapping.
   c. Retrieves indicators from the source data of the alert.
   d. Creates indicators in the "Indicators" Module.
   e. Links alerts to the indicators.
   f. Updates the state of the alerts.

Observations

The data in the following table outlines the number of alerts ingested, the total time taken to ingest those alerts, and the total time taken for all the triggered playbooks to complete their execution.

Single Invocation Test run on a two-node active-active FortiSOAR appliance

Number of alerts created in FortiSOAR	Total time (in seconds) taken to create all alerts in FortiSOAR	Total number of playbooks executed in FortiSOAR
1	1.87	2
5	2.41	6
10	3.68	11
25	6.21	26
50	11.24	51
100	20.27	101

Note: Once this test is completed, refer to the pre-test conditions before starting a new test.

Test 3: Perform Ingestion in FortiSOAR using the FortiSIEM Ingestion Playbook after the alerts are created and after triggering "Extraction" and "Enrichment" playbooks

The test was executed using an automated testbed that starts FortiSIEM ingestion which in turn created alerts in FortiSOAR. Once the alerts are created in FortiSOAR, "Extraction" and "Enrichment" playbooks are triggered and the total time taken for all the extraction and enrichment playbooks to complete their execution is calculated.

The setup for this test is exactly the same, however this test additionally requires the "VirusTotal" connector to be configured.

Steps followed

1. Created the alerts using the FortiSIEM Ingestion playbook.
2. Once the alerts are created, the "Extraction" playbooks are triggered. You can download the JSON for the sample playbooks (PerfBenchmarking_Test03_PB_Collection_7_0_1.zip) so that you can run the same tests in your environment to see the performance in your version/hardware platforms. Or, if you want to do some additions that are specific to your environment, you can also tweak the existing playbooks.
   The playbooks perform the following steps:
   a. Declares variables using the "Set Variable Step".
   b. Updates the existing indicator list using mapping.
   c. Retrieves indicators from the source data of the alert.
   d. Creates indicators in the "Indicators" Module.
   e. Links alerts to the indicators.
   f. Updates the state of the alerts.
3. Once the indicators are extracted, the "Enrichment" playbooks are triggered and they perform the following steps:
   a. Matches the IP in an internal subnet through the "Utilities" subnet.
   b. Validates whether the IP is Private or Public.
   c. Performs enrichment using the "Utilities" connector, if the IP is "Private".
   d. Performs enrichment using the "VirusTotal" connector, if the IP is "Public".
   e. Updates the indicator status based on the IP's vulnerability.
   f. Updates the state of the indicator.

Observations

The data in the following table outlines the number of alerts ingested, the total time taken to ingest those alerts, and the total time taken for all the triggered playbooks to complete their execution.

Single Invocation Test run on a two-node active-active FortiSOAR appliance

Number of alerts created in FortiSOAR	Total time (in seconds) taken to create all alerts in FortiSOAR	Total number of playbooks executed in FortiSOAR
1	5.7	4
5	7.43	16
10	10.36	31
25	24.98	76
50	48.77	151
100	1 minute 29 seconds	301

Note: Once this test is completed, refer to the pre-test conditions before starting a new test. Also, note that enrichment of playbooks makes API calls over the Internet, and the times mentioned in this table to execute playbooks is inclusive of this time.

Sustained Invocation Test for the HA active-active cluster of two FortiSOAR appliance

A sustenance test was also conducted with the configuration as defined in "Test 2", i.e., the test is executed by manually triggering the FortiSIEM Ingestion playbook, “FortiSIEM -> Ingest 100 Alerts”, which creates alerts in FortiSOAR. Once the alerts are created in FortiSOAR, an "Extraction" playbook is triggered and the total time taken for all the extraction playbooks to complete their execution is calculated.

Number of alerts: **100/min**

Duration: **12 hours**

Playbooks configured: **As defined in Test 2 comprising of "Ingestion" and "Indicator Extraction" playbooks**.

Total number of playbooks executed: **72821**

Results

The system performed well under the sustained load. All 72100 alerts were successfully ingested and all the extraction playbooks were successfully completed without any queuing.

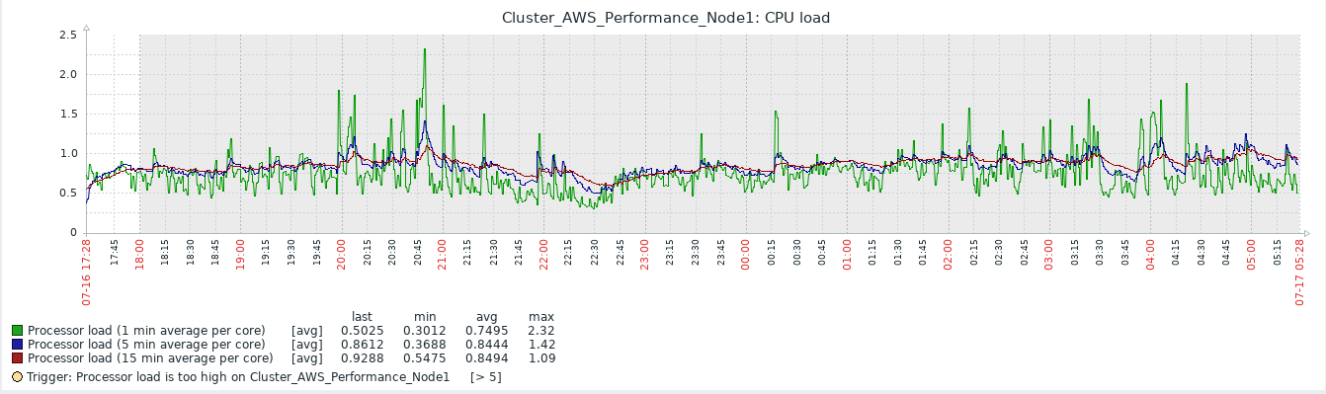
Graphs

The following graphs are plotted for the vital statistics for the HA cluster that was under test during the period of the test run.

All the graphs included in this section are from the Primary/Active Node.

CPU Load Average Utilization Graph

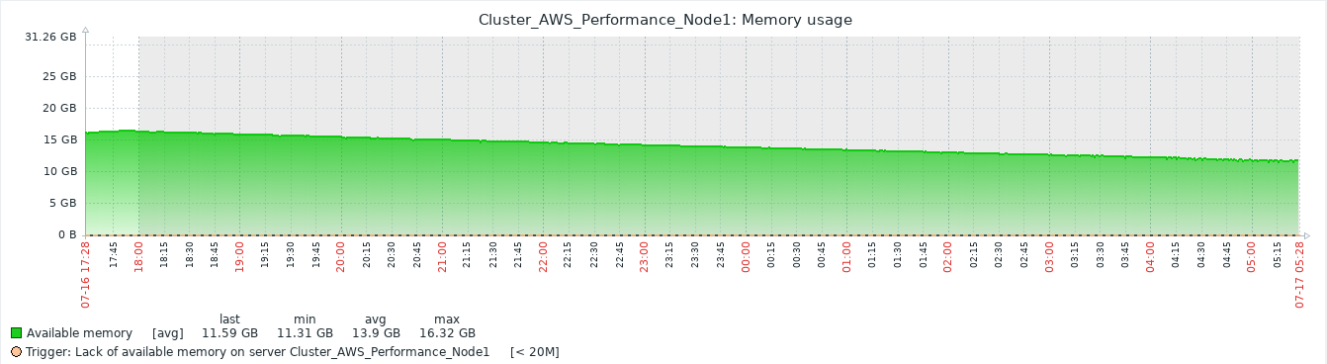
Analysis of CPU load average utilization when the test run was in progress on the appliance:



Using the system resources specified in the "Environment" and "Pre-Test Conditions" sections, it was observed that while the "Sustenance Test" was running the CPU utilization was normal and the performance of the system did not get impacted.

Memory Utilization Graph

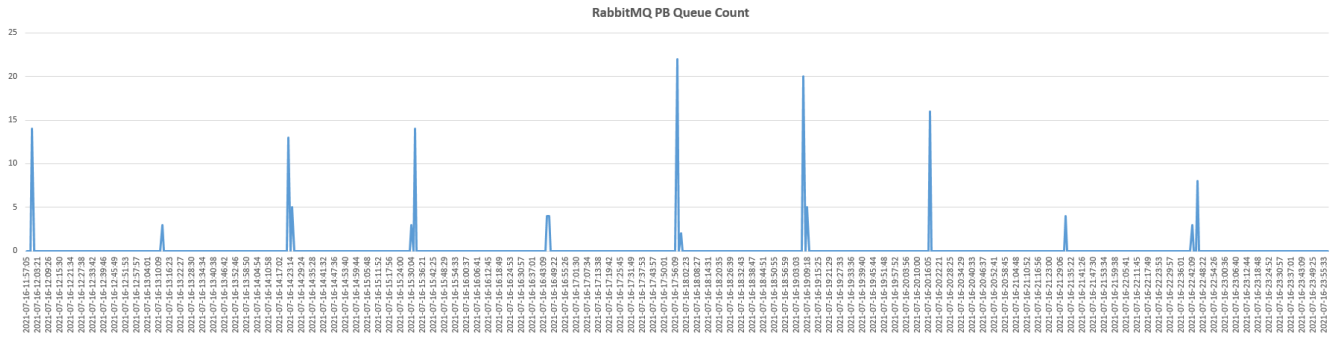
Analysis of memory utilization when the test run was in progress on the appliance:



Using the system resources specified in the "Environment" and "Pre-Test Conditions" sections, it was observed that while the "Sustenance Test" was running the Memory utilization was around 15GB.

RabbitMQ PB Queue Count Graph

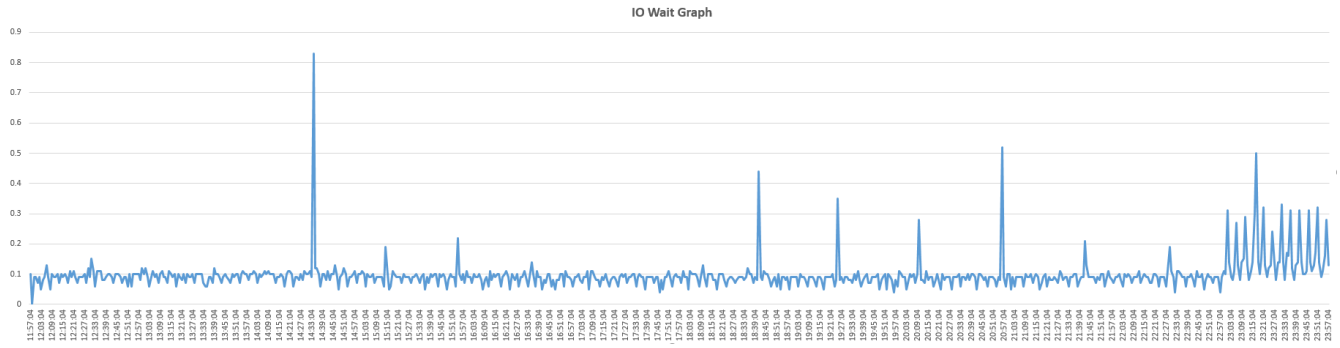
Analysis of RabbitMQ PB Queue Count when the test run was in progress on the appliance:



Using the system resources specified in the "Environment" and tunables configured as mentioned in the "Pre-Test Conditions" sections, it was observed that while the "Sustenance Test" the `rabbitmq_pb_queue` went to a maximum of 25 for some instances but eventually went back to 0. This means that no playbooks remained in queue and that all the required playbooks associated with the alerts were getting completed, i.e., alerts were created and their indicators were extracted and enriched in a minute.

IO Wait Graph

Analysis of IO Wait when the test run was in progress on the appliance:



Using the system resources specified in the "Environment" and "Pre-Test Conditions" sections, it was observed that while the "Sustenance Test" was running the IO Wait time was normal, with the average IO Wait being around 1% of the CPU idle time.

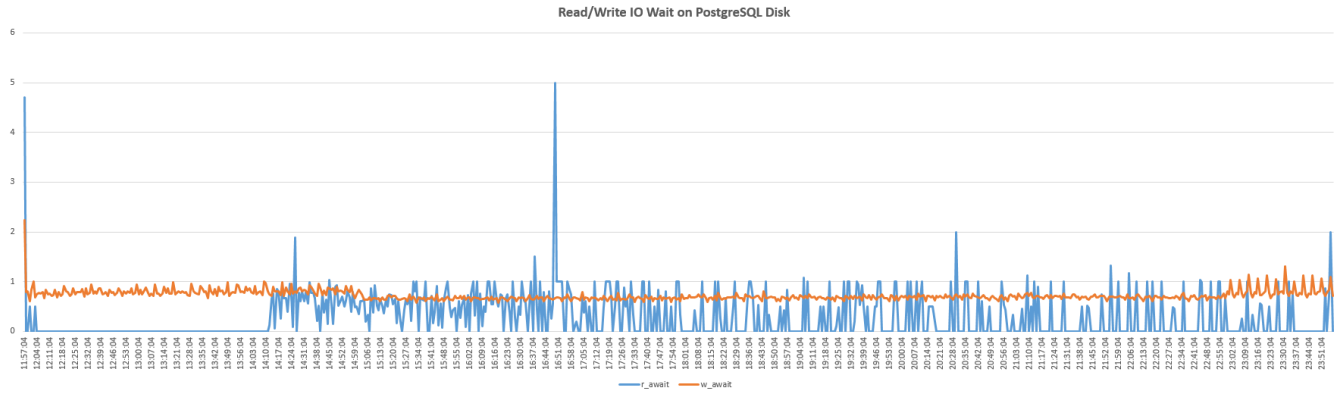
Read/Write IO Wait Graph for ElasticSearch

Analysis of Read/Write IO Wait for ElasticSearch when the test run was in progress on the appliance:

Using the system resources specified in the "Environment" and "Pre-Test Conditions" sections, it was observed that while the "Sustenance Test" was running the "Read" Wait for the ElasticSearch disk was almost 0 milliseconds. The "Write" Wait for the ElasticSearch disk averaged around 0.5 milliseconds, with the maximum wait of 35 milliseconds and the minimum wait of 0.5 millisecond.

Read/Write IO Wait Graph for PostgreSQL

Analysis of Read/Write IO Wait for PostgreSQL when the test run was in progress on the appliance:



Using the system resources specified in the "Environment" and "Pre-Test Conditions" sections,it was observed that while the "Sustenance Test" was running, the "Read" Wait for the PostgreSQL disk averaged around 1 millisecond. The "Write" Wait for the PostgreSQL disk averaged around 1 millisecond, with the maximum wait of 2 milliseconds and the minimum wait of 1 millisecond.

www.fortinet.com