Securing your Vertica database with data at rest encryption.

Version 1.0

Contents

Introduction

Time to value and concerns about performance are common reasons companies do not protect their most sensitive data with encryption. Now that security breaches are becoming more common, organizations should consider encryption as an important element in their security architectures to reduce the impact of a hacker stealing their most sensitive data. It is important for organizations to understand what kinds of encryption are available today and what risks are mitigated by implementing the various types of encryption. Listed below are the different types of encryption available to protect your data.

- Application or tokenization level
- File level
- Disk level

The focus of this blog will be on file level (transparent) since it provides the most protection with least amount of complexity and risk. For instance, disk encryption only mitigates the risk of someone stealing your hard drive or laptop, once the system is up and running your files are still totally exposed with disk level encryption.

In addition to the database files there are many other files that should be candidates for encryption such as:

- Backups
- Backup config files that may contain clear text of passwords
- Source files to be ingested into the database that make contain sensitive data.
- Scripts, property files or system log files that might have userid and passwords in them.

Although some databases vendors provide their own encryption there are some good reasons to use an external product like Vormetric to encrypt your data such as:

1. Policy management and auditing capabilities.
2. Single solution for all data at rest.
3. Shielding the DBA from sensitive data with column level encryption or application encryption.
4. Reduce risk by implementing separation of duties by having a separate repository for encryption keys and encryption key management.

For more information on Vormetric see the following link.

Assumptions

- The focus of this document will be transparent encryption or encrypting (VTE) of data at rest. Future blogs will demonstrate how Vormetric can also provide the ability to encrypt specific Vertica database columns with industry standard PKCS11 apis or via REST calls using tokenization.
- Vertica 8.0.1 was used for testing.
- Vormetric 6.0.2 was used for testing.
- No optimization were done to the database. ***% Differences can be used as guidelines and starting points for your own testing.***
- Vertica was running on a VM with 4GB RAM and 2CPU with 250GB of total disk. All disk I/0 was on same disk in VM which was the bottleneck.
- No other activity on the database or machine at time of tests.
- Some knowledge of Vertica or another database will be helpful.

Overview

As indicate above a couple of reasons organizations do not encrypt their data are:

Time to value - *Encryption should not be disruptive to implement*. This blog demonstrates how quickly encryption can be implemented without having to write any code or make any modifications to the database or applications accessing data from the database.

Performance impact -. This blog will show some examples of different workloads, which allows organization to plan for proper implementation when considering encryption of a database from a performance and applications perspective.

A big benefit to using an external encryption product is because a single solution can encrypt any database so for testing purposes the choice of a database does not really matter because the process is the same. Other than being extremely fast, Vertica database was chosen for many reasons. Vertica has many system tables that can capture the state of the machine when queries or loads are running. These metrics were captured so comparisons for each query and load can be done. Although not tested, a Vertica cluster can very easily be setup to better understand how encryption may also be implemented in more complex multi node environments. Vertica also has the ability to implement its own key value pair store called “Routable queries” so that kind of workload can be tested as well.

Workloads

When testing the impact encryption has on databases, it is important to test with various workloads since each will have different performance profiles. The tests performed for this blog were meant to cover some of the most common workloads such as:

VMARTQUERYTXN – These are typical ***transactional type*** queries that return only one or two rows and no database functions. Typical response time is in milliseconds. A total of 9 queries that contained fact tables with joins to dimensions.

VMARTQUERY – These are typical ***analytical*** queries that have analytical functions like sum, max, subselects etc. Typical response time is in seconds. A total of 9 queries that all contained fact tables with joins to dimensions. These are the OOB queries provided by the Vertica samples.

VMARTLOAD – These are typical ***batch load scripts*** to load the OOB VMart tables. Load time is directly related to number of rows ingested. 15 tables were loaded using the standard COPY statement. No inserts command were issued only copy commands.

All of the above tables and sql statements are based on the out of the box standard Vertica examples called VMart.

Note: The intent of this blog is NOT to set performance records but to give customers an idea on what kind of percentage difference can be expect given different work loads and to better understand how a database can be encrypted without impacting the users or applications using the database.

How "Transparent" (Data at Rest) Encryption Works

The most successful implementations of encryption incorporate three main principles, confidentiality, integrity and availability. For instance, there are many opensource encryption libraries that provide confidentiality and integrity but when making the encrypted data available in clear text at the right time to the right person or process on the fly is very difficult to do.

Vormetric Transparent encryption does NOT require any code to implement which helps reduce the complexity and risk and is a major factor to achieving time to value. An example of how transparent encryption works might be the best way to see the value. It is estimated that 58 % of internal beaches are done by persons with privileged user access. Most hackers attempt to obtain root access to the system, which is why it is important to block and audit those situations when they are attempting to access sensitive data. An organization typically does not need the root administrator to actually see clear text data to do his job. Here is an example showing how the root user on the Vertica VM image is blocked to viewing clear text of the source data to be loaded into Vertica.

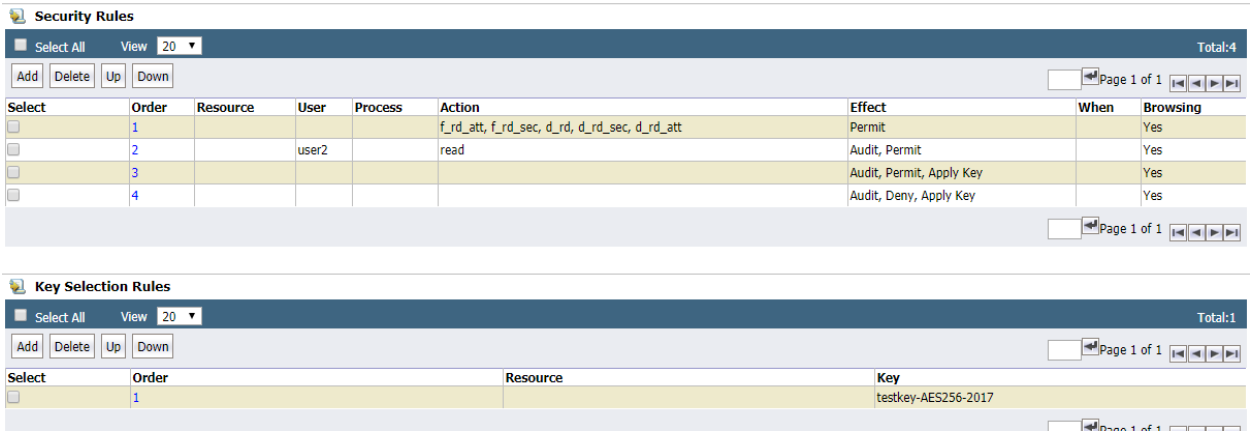
```
[root@vertica801 loadfiles]# head Call_Center_Dimension.tbl
¦0b¦pP5¦N¦¦    ¦<¦rx?V¦¦jz¦j¦¦¦z¦De¦?$¦¦V¦%}¦I?¦C¦¦¦;¦b¦@5¦R?

JA¦¦¦¦;¦¦¦m¦¦¦¦¦¦¦?¦69>¦%?¦cw¦¦L¦¦¦CE:d#V¦¦¦2s¦¦}0¦¦¦¦H¦>¦)`1¦
/¦¦"c:?¦¦KpL+¦¦j@¦:¦T^¦?¦Q¦¦O¦e¦]\F^C¦Z?¦F¦"¦¦K¦?¦">q¦¦6¦W¦¦¦781w¦¦¦~@L¦¦
¦¦¦c  ¦X¦IF¦¦?¦¦*¦!Xb6>¦0¦q]?d^¦¦u6"6¦b)¦¦d¦¦¦P]FO¦z*¦-¦>¦¦¦,,¦¦4¦]~qd@3y?¦¦SQ
¦:¦,¦¦jJøE¦?C¦54%¦¦(¦L¦¦¦¦J¦¦¦¦
```

What is needed with the file above is to allow the Vertica dbadmin user to run a job and have access to the file above in clear text to be loaded into Vertica. It is important to log any

unauthorized attempt to sensitive data to potentially identify a security breach. Here is an example of what an audit log of the above attempt to view the file above.

CGP2601I: [SecFS, 0] PID[110055] [AUDIT] Policy[test-linux-operational-vertica] User[root,uid=0,gid=0\root\] Process[/usr/bin/head] Action[read_file] Res[/data/loadfiles/Call_Center_Dimension.tbl] Key[testkey-AES256-2017] Effect[PERMIT Code (1A,2M)]

The following screenshot is an example of how to set up a policy in Vormetric allowing user2 to read the data but NOT see clear text. When user2 attempts to access the data he will see cipher text as shown above.



Most security products allow for these kinds of messages to be sent to a (SIEM) reporting tool like Splunk so security administrators can take appropriate action on these kinds of activities. Another great benefit to a security product like Vormetric is to block the root user to having access to switch users and assume their permissions. Vormetric can detect this kind of activity and log this activity as a "fake" user activity and deny them the ability to see clear text. As you can see that is why this kind of encryption is called transparent since implementing it is "**Transparent**" to the people and processes that need access to the data.

Summary of Results

Listed below is a summary of the tests performed. These rates should only be used as a guideline on what to expect based on various workloads. Obviously, there are many factors that impact performance. This blog attempts to provide as much detail as possible so expectations on the overall impact of encryption on your environment is understood. Your workloads may or may not have the same characteristics as what was tested but these results should provide a good starting point when running your own tests.

Use			*%*
Case	**Work load Type VMARTLOAD (Batch Load)**	**Avg. Seconds**	*Difference*
	Base Line	451.4	
2	Encrtypted database files(DB)	483.2	*7.04%*
1	Encrypted source files	485.8	*7.62%*
3	Encrypted both DB and source files	505.8	*12.05%*
			%
	Work load Type VMARTQUERY (Analytical Query)		*Difference*
	Base Line	16.8	
2	Encrtypted database files(DB)	18.6	*10.71%*
			%
	Work load Type VMARTQUERYTXN (Transaction)		*Difference*
	Base Line	4.2	
2	Encrtypted database files(DB)	4.4	*4.76%*
			%
	Work load Type Vertica Backup		*Difference*
	Base Line	109	
4	Encrtypted backup files	134	*22.94%*

Multiple runs were executed and the numbers above represent the average times after removing highs and lows. As you can see from the results the impact of encryption for half of the use cases in in the single digits. The appendix of this document shows some metrics on how I/O bound my VM image was which was the obvious bottleneck for my tests. Most likely, your environment will have a better-tuned system and as a result have better performance.

Having some performance degradation should be acceptable to most organizations considering the benefits encryption provides. I hope this blog demonstrated how using transparent encryption is not disruptive to implement and gives you a better understanding on how it performs under different workloads enabling you to protect your sensitive data achieving value quickly without complexity. I also hope organizations will adjust their cost benefit models to look at encryption more seriously and to make sure the right type of encryption is implemented. Doing this will ensure sensitive data is protected from thieves and protect **us** from the possibility of identity theft when this data gets into the wrong hands.

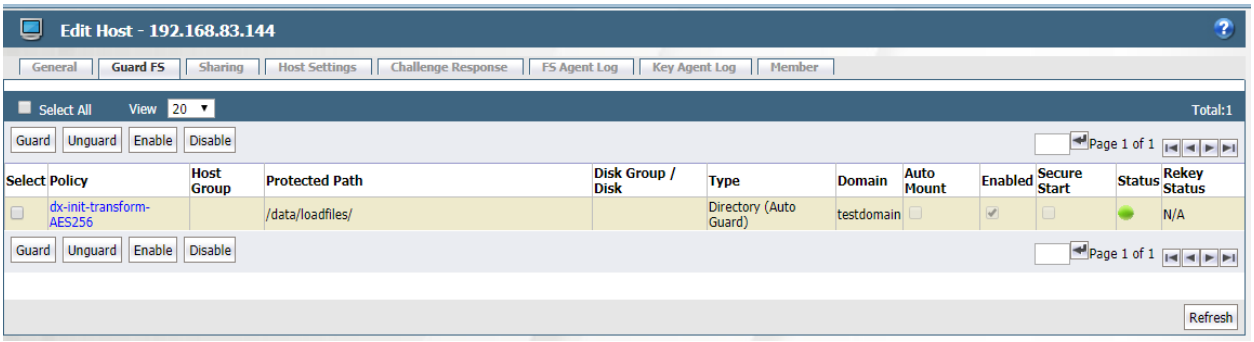
For more details on the how the tests were done and the above results, please see following sections.

Details on Use cases covered.

The tests performed for this blog were meant to cover some of the most common use cases where sensitive data resides when working with databases. Having different workloads provides insight on the potential impact of implementing encryption from a performance perspective.

Use Case 1 – Encrypt the source files to be loaded into the database.

This is for scenarios where you have sensitive data in source files to be loaded into the database. Also this use case might be for customers who have Oracle or SQL Server and using their TDE encryption on the database but are not encrypting the source files to be loaded. Times were captured for the encryption of the source files as well as times captured to load these encrypted files into Vertica. A Vormetric policy was written that allowed the Vertica dbadmin user to decrypt the files as the Vertica copy command was run. Here is an example in Vormetric on how to protect a directory that contains source files to be loaded.

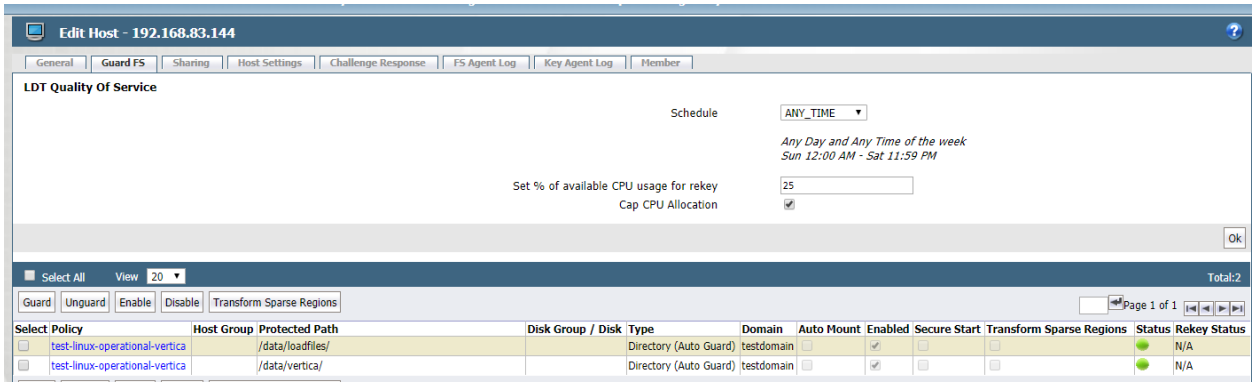


Use Case 2 – Encrypt the Vertica database files.

This is for scenarios where you already have sensitive data already in the database. Times were captured for the initial encryption of the Vertica database files as well as times captured to load data into Vertica. A Vormetric policy was written that allowed the Vertica dbadmin user to decrypt the database files.

Use Case 3 – Encrypt both the source files to be loaded into Vertica and the Vertica database files.

This shows how encryption can be implemented for both source files to be loaded into the database as well as the actual database files. Here is a screenshot showing the Vormetric Guard points for this use case.



Use Case 4 – Encrypt the Vertica backup.

This shows how encryption can be implemented for the database backups. This use case covers the situation where you want both the backup to be encrypted as well as the actual database files.

The encryption testing process

The process used for these tests was as follows:

1.)Baseline. Establish baseline encryption and metrics of loads and queries with NO encryption.

- Create a baseline on times by running out of the box queries and load scripts provided by Vertica multiple times on the sample VMart schema provided by Vertica with NO encryption.
- Workloads: ***transactional, analytical query, and batch loads.***

The intent of the baseline set of tests is to determine how long it takes to run the out of the box standard queries and loads provided by Vertica and use this as a comparison to all the other tests.

2.) Encryption and database workload testing.

The intent of these tests is to determine the integrity of results, availably of data and compare times of **Vertica queries and loads of encrypted data in Vertica**. This is done by running Vertica load and query jobs with different types of workloads with different use cases.

Detailed steps.

1. Download Vertica Community Edition and install VMart sample database.
2. Run baseline metrics with **no** encryption using Vertica VMart database for loads and query.
3. Install the Vormetric Data Security Manager.
4. Install the Vormetric File Agent on the Vertica databasevm image.
5. Configure the Vormetric DSM to encrypt Vertica VMart source load files.
6. Confirm data from previous step was encrypted and run Vertica load jobs to confirm data integrity.

7. Run Vormetric encryption on data in Vertica to encrypt all fact tables.
8. Confirm data from previous step was encrypted and run Vertica load and query jobs to confirm data integrity.

Baseline Results

Performance Results

Work load Type	Seconds
VMARTLOAD	445
VMARTLOAD	446
VMARTLOAD	445
VMARTLOAD	463
VMARTLOAD	458
Average	*451.40*
Work load Type	**Seconds**
VMARTQUERY	16
VMARTQUERY	16
VMARTQUERY	17
VMARTQUERY	17
VMARTQUERY	18
Average	*16.80*
Work load Type	**Seconds**
VMARTQUERYTXN	4
VMARTQUERYTXN	4
VMARTQUERYTXN	5
VMARTQUERYTXN	4
VMARTQUERYTXN	4
Average	*4.2*

Like all other test performed, the highs and lows were dropped from the test results and averages were calculated. The results here will be used to compare all other tests.

Use Case 1 Results– Encrypt the source files to be loaded into the database.

This section covers both steps 5 & 6 above.

For my example, I decided I wanted to encrypt the source data first to get a baseline on how long it takes to encrypt just regular text files. I ran the Vertica examples data generator vmart_gen passing in 50,000,000 for each of the 4 fact tables. This generated a total of 13.3GB of raw data. Listed below are the files generated.

Table	rows	size
call_center_dimension	200	20K
online_page_dimension	1,000	64K
online_sales_fact	50,000,000	3.6G
customer_dimension	50,000	6.2M
date_dimension	1,826	220K
employee_dimension	10,000	1.1M
inventory_fact	50,000,000	890M
product_dimension	60,000	6.7M
promotion_dimension	1,000	120K
shipping_dimension	100	4.0K
vendor_dimension	50	4.0K
warehouse_dimension	100	8.0K
store_dimension	250	28K
store_orders_fact	50,000,000	5.0G
store_sales_fact	50,000,000	3.6G
Total Rows	**200,124,526**	
Total GB		**13GB**

To encrypt this data with Vormetric the following command was issued:

```
[root@vertica801 data]#dataxform --rekey --gp /data/loadfiles --preserve_modified_time
Checking if /data/loadfiles is a guard point with a rekey policy applied
/data/loadfiles is a guard point with a rekey policy applied
About to perform the requested data transform operation
-- Be sure to back up your data
-- Please do not attempt to terminate the application
Do you wish to continue (y/n)?y
Scan found 15 files (13298 MB) in 1 directories for guard point /data/loadfiles
Processed 14 files (8202 MB) of 15 files (13298 MB) in /data/loadfiles, in
  progress 5096 MB, completed 4764 MB
Transformed 15 files (13298 MB) of 15 files (13298 MB) for guard point
  /data/loadfiles
The data transform operation took 0 hours, 4 minutes and 5 seconds
The data transform program ran from Thu Nov  2 10:39:29 2017 until Thu Nov  2
  10:44:40 2017
Data transform for guard point /data/loadfiles finished
```

Multiple Vormetric dataxform tests were run to get a baseline to compare the times of encrypting 15 source text files to the data after it has been loaded into Vertica. The average time was about 282 seconds for 13.3GB of data which is an average of 170GB per hour. ***This rate should be considered the worst case scenario since this test was done on a VM Image with only 4GB of***

RAM and 2 Cores with only one physical disk. All I/O was going over one channel and this is not best practice from a performance perspective.

Encryption is I/O intensive and determining how long it takes to encrypt data really depends on factors such as:

1. How optimized your disk configuration is: block size, number of spindles, format, etc.
2. How many files you have and the size of the files.
3. How much I/O activity is already going on.

Conceptually,optimizing Vertica performance is very similar to how you would want to optimize your system for encryption. Bottom line is, if you have optimized your Vertica database and you spend time on the other factors above then having some performance degradation should be acceptable considering the benefits that encryption provides. For example, customers who have optimized their disk layouts can get encryption rates of over 600GB per hour with only one instance of the Vormetric dataxform running at a time. Running multiple dataxform commands at once is another way to reduce overall time to encrypt data. Another section of this document will highlight the difference from a performance perspective that encryption can have on your database.

The intent of this testing is NOT to break any performance records but to give customers an idea on what kind of percentage difference they can expect given different work loads, transactional, analytical query, and batch loads.

Once the Vormetric dataxform job to encrypt the source files finished a Vormetric policy was written to ensure the dbadmin user can read the source data in clear text. This protects your data at rest so if someone tried to access this original data without the right rules they will either be denied or only see encrypted data like we should for the root user above. These rules can even further lock down the access so only dbadmin user running a certain process can have proper access to the file. Vormetric's management console makes is very easy to implement these kinds of policies to conform to an organizations security procedures to ensure that the data is confidential has integrity and is available.

Once the Vormetric policies are created to allow the dbadmin to have clear text access to the files the Vertica copy command can be issued. Here is an example:

```
/opt/vertica/bin/vsql -U dbadmin -w dbadmin -p 5433   -c "COPY store.Store_Sales_Fact
FROM '/data/loadfiles/Store_Sales_Fact*' on v_vmart_node0002  DELIMITER '|' NULL ''
DIRECT STREAM NAME 'VMART_STORE_SALES_FACT';" >>tpch_load_dataout.txt
```

This use case is also appropriate for Oracle or MS SQL Transparent Data Encryption (TDE) customers who might be already using TDE for the database encryption but need to also encrypt the source files that could have sensitive data in them.

Performance Results

Source file encryption times

These are the result from running the Vormetric dataxform command on the source files. Like all other test performed, the highs and lows were dropped from the test results and averages were calculated. As the table shows below the average GB/Hour to transform the raw text files to encrypted data is around 170GB/Hour.

Test	Seconds	
1	311	
2	245	
3	286	
4	278	
5	287	
6	285	
7	282	
Average	282	
Rates		
Total GB Backed up=		13.3
GB/Sec		0.04716312
GB/Min		2.82978723
GB/Hour		169.787234

Vertica database load times with encrypted source files only.

These are the results of running the Vertica batch loads jobs on a Vertica database directory that is protected by a Vormetric policy that encrypts the data. Like all other test performed, the highs and lows were dropped from the test results and averages were calculated.

Work load Type	**Seconds**
VMARTLOAD	482
VMARTLOAD	474
VMARTLOAD	503
VMARTLOAD	495
VMARTLOAD	475
Average	*485.80*

In summary comparing this number to the base line = (485.80-451.40)/451.40= 7.62 % difference in time compared to the baseline. The database loads with encrypted source data will take about 7.04% longer than the baseline to run.

Use Case 2 Results– Encrypt the Vertica database files.

This section covers both steps 7 & 8 above.

Instead of just encrypting all of the data in Vertica I decided to add another level of complexity to ensure that joins can be done in Vertica with unencrypted data in the dimension tables and encrypted data for the fact tables. Vertica makes this very easy to implement by creating a new data location and moving the data to this location. For example:

```
CREATE LOCATION '/data/vertica/' ALL NODES USAGE 'DATA' label 'FACTS';
```

And to move the data:

```
select set_object_storage_policy('online_sales.online_sales_fact','FACTS', true);
```

The above was done for all fact tables.

So once the above commands are issued all the Vertica database fact tables reside in /data/vertica.

I then decided to run the sample Vertica load scripts with the same un-encrypted source data files from previous runs. When Vertica loads data into the database it compresses the data so for my example you can see that the 13GB of raw text data was compressed down to 4.5GB Vertica database files. I did NOT create any Vertica database projections since my goal was NOT necessarily increase query performance but just to capture the percentage time difference of queries and loads on encrypted and unencrypted data.

Here is the Vormetric encryption command that was run.

```
[root@vertica801 ~]#dataxform --rekey --gp /data/vertica/  --preserve_modified_time
Checking if /data/vertica/ is a guard point with a rekey policy applied
/data/vertica/ is a guard point with a rekey policy applied
About to perform the requested data transform operation
-- Be sure to back up your data
-- Please do not attempt to terminate the application
Do you wish to continue (y/n)?y
.....................................................................................
Scan found 61 files (4507 MB) in 465 directories for guard point /data/vertica/
Transformed 61 files (4507 MB) of 61 files (4507 MB) for guard point
  /data/vertica/
The data transform operation took 0 hours, 1 minutes and 34 seconds
The data transform program ran from Fri Nov  3 14:08:03 2017 until Fri Nov  3
  14:09:37 2017
Data transform for guard point /data/vertica/ finished
```

As you can see it is the same command from the previous encryption on the source files just pointing to a different directory which happens to contain all the data for the Vertica database fact tables.l Notice how many more directories and files compared to the original 15 source files.

Database file encryption times

Multiple Vormetric dataxform tests were run to get an average of times of encrypting the Vertica 465 directories and 61 files. The average time was about 134 seconds for 4.5GB of data which is an average of 121GB per hour**.** *As you can see this is slower than encrypting the source data. In general, Vormetric normally is more efficient when encrypting fewer number of larger files then larger number of smaller files.* ***From a performance perspective, the same comments from the previous section apply regarding these numbers as absolutes.***

Performance Results for Database Queries and Loads

These are the results of running on a Vertica database directory that is protected by a Vormetric policy that encrypts the data for various workloads.

1. Vertica batch loads jobs (VMARTLOAD)
2. Analytical Queries (VMARTQUERY)
3. Transactional Queries (VMARTQUERYTXN)

As noted from before the highs and lows were dropped from the test results and averages were calculated.

Work load Type		% Difference
VMARTLOAD	481	
VMARTLOAD	470	
VMARTLOAD	501	
VMARTLOAD	486	
VMARTLOAD	478	
	483.2	7.04%
Work load Type		
VMARTQUERY	18	
VMARTQUERY	18	
VMARTQUERY	19	
VMARTQUERY	19	
VMARTQUERY	19	
	18.6	10.71%
Work load Type		
VMARTQUERYTXN	6	
VMARTQUERYTXN	3	
VMARTQUERYTXN	5	
VMARTQUERYTXN	3	
VMARTQUERYTXN	5	
	4.4	4.76%

In summary the loads are about 7.04% slower with encryption. The vmartquery workload is about 10.71 % slower and the vmartquerytxn workload is about 4.76 % slower with encryption. Queries that

normally take milliseconds to run will not have a big % difference compared to analytical queries which normally take longer than transactional queries.

Use Case 3 Results – Encrypt both the source files to be loaded into Vertica and the Vertica database files.

Performance Results for Database Loads

In this case the only test that needs to be done is the load since the query times will be no different than the previous use case.

Work load Type		% Difference
VMARTLOAD	498	
VMARTLOAD	503	
VMARTLOAD	506	
VMARTLOAD	509	
VMARTLOAD	513	
	505.8	12.05%

In summary the database loads are about 12.05 % slower with encryption on both the source files to load and the Vertica database files.

Use Case 4 Results– Encrypt the Vertica backup.

For testing purposes, a full backup of the entire database was done. This was about 4.8GB of data in Vertica.

Performance Results for Vertica Backups

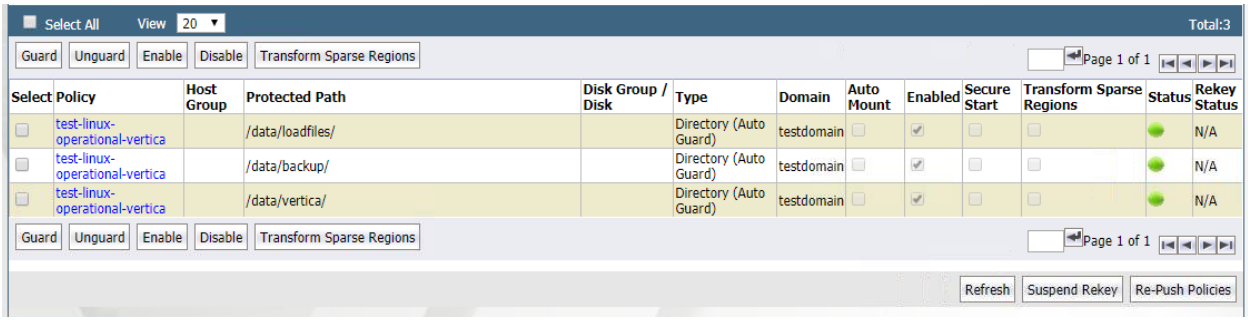
Like all other test performed, the highs and lows were dropped from the test results and averages were calculated.

Baseline Backups Not Encrypted	Seconds
1	115
2	111
3	108
4	106
5	105
Avg	109
Encrypted Backups	
1	137
2	127
3	134

4	135
5	137
Avg	134
Percent Difference	22.94%

In summary the database backups are about 22.94 % slower with encryption on both the backup files and the Vertica database files.

Here is the Vormetric guard points showing the locations of the encrypted files.



Here is a sample Vertica backup command

[dbadmin@vertica801 ~]$ vbr --task init --config-file backup_snapshot.ini
Initializing backup locations.
Backup locations initialized.
vbr --task backup --config-file backup_snapshot.ini
Starting backup of database vmart.
Participating nodes: v_vmart_node0002.
Snapshotting database.
Snapshot complete.
Approximate bytes to copy: 4954726885 of 4954726885 total.
[==]
100%
Copying backup metadata.
Finalizing backup.
Backup complete!

Here is the average time to backup an encrypted Vertica database and the number of GB per hour. As mentioned earlier rates will vary depending on a few factors such as number of files and size of files. In this case Vertica has many smaller files since it is a column store which is why the rate is slower than the 170GB per hour rate of backing up fewer larger files which was the scenario for encrypting the ingest files.

Avg Seconds	134	
Total GB Backed up=		4.9
GB/Sec		0.04
GB/Min		2.19
GB/Hour		131.64

Appendix

Listed below are the timings for the various tests.

I/O

Vertica provides a utility that captures metrics on disk performance. Here are the results of running this tool. dbadmin=> SELECT MEASURE_LOCATION_PERFORMANCE('/data/vertica','v_vmart_node0002');

Throughput : 154 MB/sec. Latency : 108 seeks/sec

Here are the results of running an iostat on the vmimage. You can see that iowait is at 84 % which shows how I/O bound this image is.

%user %nice %system %iowait %steal %idle
0.20 0.00 8.23 84.30 0.00 7.26

system resource usage

Below is the overall average system resource usage during the different workloads.

SystemResourceUsage

Run Type	Run Desc	Avg. Average Cpu Usage Percent	Avg. Average Memory Usage Percent	Avg. Io Read Kbytes Per Second	Avg. Io Written Kbytes Per Second	Avg. Total Runtime Results
VMARTLOAD	Baseline	61	41	70,925	31,336	452
	EncryptedDataBaseFiles	65	42	68,627	31,360	484
	EncryptedLoadFiles	62	47	66,515	32,463	478

Information above describes the impact encryption has on various workloads from a system resource perspective.

SystemResourceUsage

Run Type	Run Desc	Avg. Average Cpu Usage Percent	Avg. Average Memory Usage Percent	Avg. Io Read Kbytes Per Second	Avg. Io Written Kbytes Per Second	Avg. Total Runtime Results
VMARTQUERYTXN	Baseline	40	34	52,784	582	4
	EncryptedDataBaseFiles	45	34	39,070	795	5

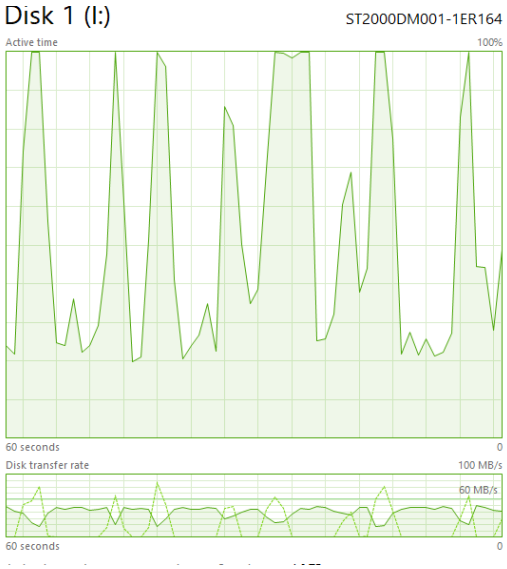
Information above describes the impact encryption has on various workloads from a system resource perspective.

SystemResourceUsage

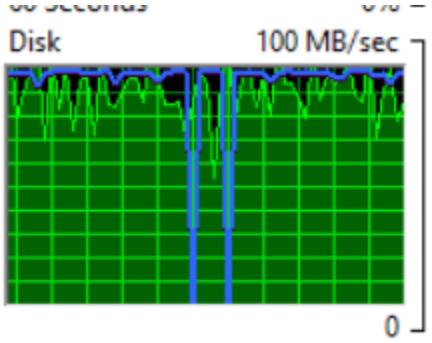
Run Type	Run Desc	Avg. Average Cpu Usage Percent	Avg. Average Memory Usage Percent	Avg. Io Read Kbytes Per Second	Avg. Io Written Kbytes Per Second	Avg. Total Runtime Results
VMARTQUERY	Baseline	36	32	55,763	1,251	17
	EncryptedDataBaseFiles	53	36	40,018	4,188	19

Information above describes the impact encryption has on various workloads from a system resource perspective.

The I/O for a normal Vertica load job will look like this



An example of disk usage when running a Vormetric dataxform job:



Note Vormetric live data transform (LDT) was NOT used to test. LDT allows for Quality of Service settings which can control how much system resources to use while it is running.

Database Encryption References.

http://go.thalesesecurity.com/rs/480-LWA-970/images/Database-Encryption-Defending-Databases-Against-Internal-And-External-Threats-SB.pdf

https://securosis.com/assets/library/reports/Securosis_Understanding_DBEncryption.V_.1_.pdf

https://public.tableau.com/views/VormetricVerticaEncryption/VerticaWithVormetricEncryption?:embed=y&:display_count=yes