|  |
| --- |
| Photo displaying partial image of two pie charts on a canvas-textured page |
| Security Audit Report  EMS 2 |
| |  |  |  | | --- | --- | --- | | Abacus | 4/23/19 | INFO2030-19W-Sec1-Software Quality 2 | |

Contents

[Introduction 1](#_Toc6852040)

[Verify that data at rest is encrypted 1](#_Toc6852041)

[Verify that front-end UI is robust against attacks 1](#_Toc6852042)

[Verify that any operating system/platform used within the solution is appropriately hardened against attack. 2](#_Toc6852043)

[One Vulnerability 2](#_Toc6852044)

[Vulnerability Tests 2](#_Toc6852045)

[Conclusion 2](#_Toc6852046)

[Appendix 2](#_Toc6852047)

## Introduction

Over the course of the past 14 weeks, the Abacus team has worked tirelessly to develop a fully operational EMS (Electronic Medical System) that is ready to be deployed as of April 23, 2019. This project has been developed with utmost security in mind. Each point within the system has been developed with security measures in place to ensure no data compromising, while still maintaining a fluid and enjoyable user experience. The Desktop application, the Azure hosted database and the mobile application implement various security features to keep the user and patient data safeguarded. Transparent data encryption, limiting mobile requests and creating logins are just a few of the multiple ways that Abacus has developed to keep the EMS secure.

## Verify that data at rest is encrypted

As a team, Abacus wanted to ensure that the hosted database would be secure from any compromises to the Microsoft Azure host as well as the Microsoft platform. To ensure that our data remained safe if an attack, leak, or other circumstances were to happen, we used some of the many, well deployed Azure resources. Within the Transparent Data Encryption resource, there is an option to encrypt your data. Once enabled it encrypts your databases backups and logs at rest without any changes to your application. This ensures that all of our data within the Microsoft Azure hosted database remains safe and vulnerability free. (See Appendix Figure 3)

## Verify that front-end UI is robust against attacks

To ensure that both our front end User Interfaces contained on our desktop and mobile applications remained secure we implemented various security measures at the critical points within our program. To start, to ensure the protection of the desktop application we require a sign in upon initiation of our application. This sign in requires the desktop user to know a login username and password. Encrypting this information maintains its integrity while keeping it safe from attackers. Additionally, items are available on a user to user basis, ensuring that only the top professionals have access to the most critical of information.

In addition, our mobile deployment requires the user to know both the first name and health card of the user they are trying to login to. This information should only be known by the individual belonging to that health card. To ensure complete safety, a password login could be deployed in a future update. The mobile app, although it does not require a username and password login, still maintains it’s integrity as a safe application as it’s features limit the amount available to any app user. The application only allows you to book one appointment at a time, and this appointment shows up as a notification when inputted by a desktop user. The application also only allows you to check in once, leaving not much information for hackers to access. For these reasons, it is our belief that the mobile application is secure. Employing these security measures and limiting access ensure that all patients data remains safe and the vulnerabilities to the overall system remain low.

## Verify that any operating system/platform used within the solution is appropriately hardened against attack.

All database queries are run through an SQL Connection with column encryption settings enabled using SQL Server Always Encrypted. This means that login attempts, patient details, or any other potentially sensitive information cannot be obtained in the event of an attack. Always Encrypted ensures this security by automatically encrypting and decrypting all data before sending it to the server, and by rewriting queries so that the true intent of any server interaction is obfuscated.

## One Vulnerability

As can be seen from figure 1, there are some vulnerabilities within the Azure database. While Azure lists 6 vulnerabilities as high, there is some good reason for us using the database in this state. For example, the firewall has been opened up for access from any location which is needed for development. Having the firewall open allows the development team to access the database from any location. In a real installation this would of course be locked down to only the required IP addresses. Azure also suggests that auditing should be enabled, this has deliberately not been enabled because of the cost the extra services. Since this is just the basic, student Azure account auditing is not included. So to sum it up, the database does have security issues that in a real world installation would be corrected by using the proper Azure account services and by limiting access to only those that require it.

## Vulnerability Tests

Because the database is hosted on Azure running our own scans on the database host using Kali was not preformed for legal reasons. However, Azure does provide a vulnerability assessment that can be run through the Azure portal. The results are displayed in figures 1 and 2 of appendix A. Kali was used however to test a windows machine used for running the EMS program. The tools used included Sparta which is tool that runs a suite of other tools such as nmap for example.

## Conclusion

Adopting the Azure platform for our database came with many advantages, especially with regards to security. Its built-in features are effective and activating them was relatively simple compared to implementing other security libraries for encryption and connection security. Furthermore, its security analysis tools provide insight to where there may be vulnerabilities, which are helpful for identifying areas for improvement over the course of the database’s lifetime. Though not robust, our authorization procedures for application log-in and mobile appointment registry are sufficient for the purposes of the EMS. In particular, the consistency of encryption which we adhered to in designing the EMS is appropriate and downright necessary for a system which could hold confidential data about potentially thousands of people, both demographic and financial.

## Appendix A

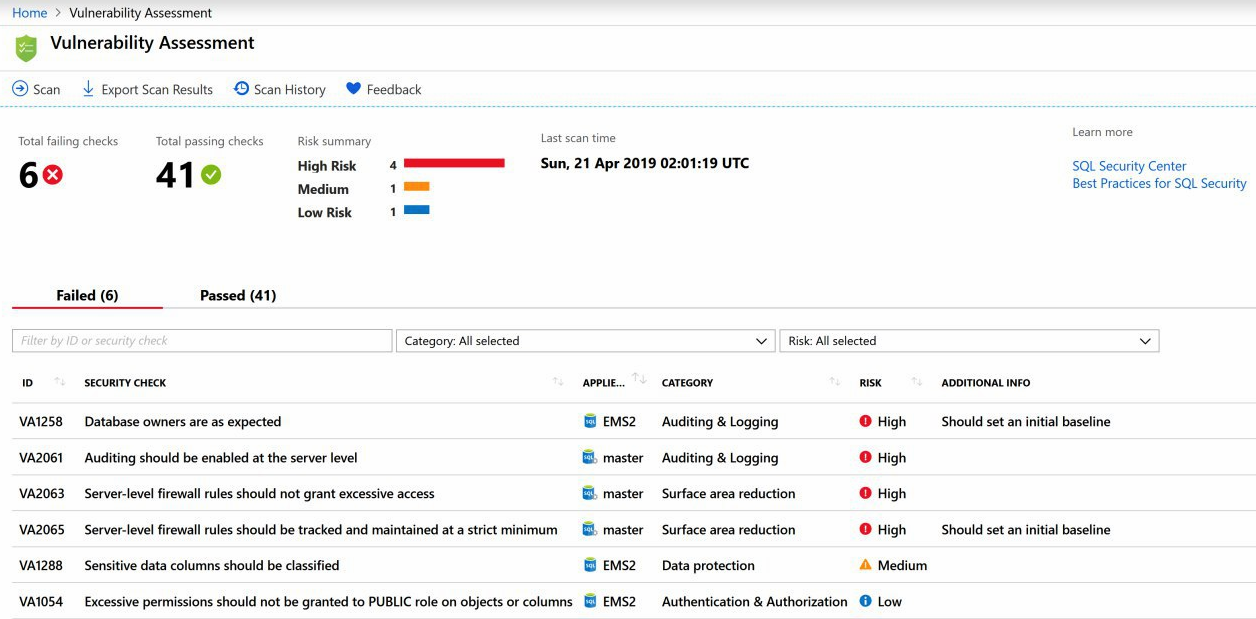


Figure 1 - Azure vulnerability Assessment

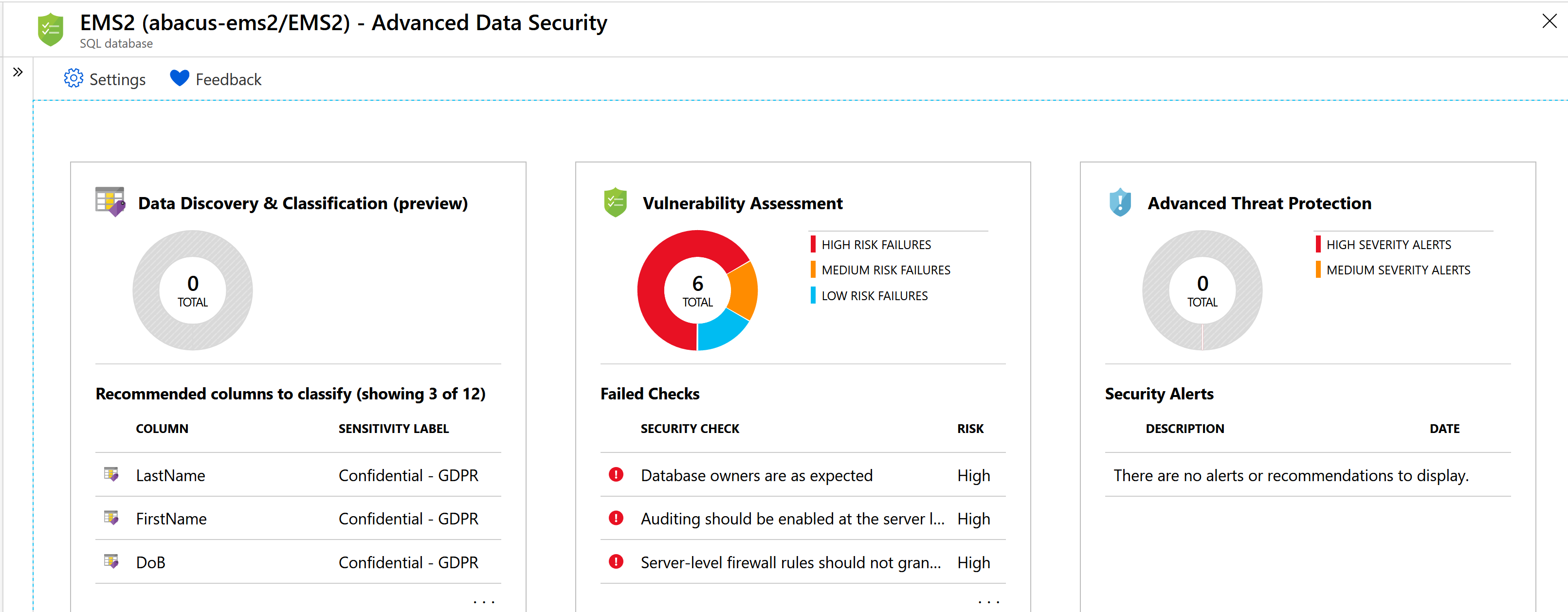


Figure 2 - Security Overview

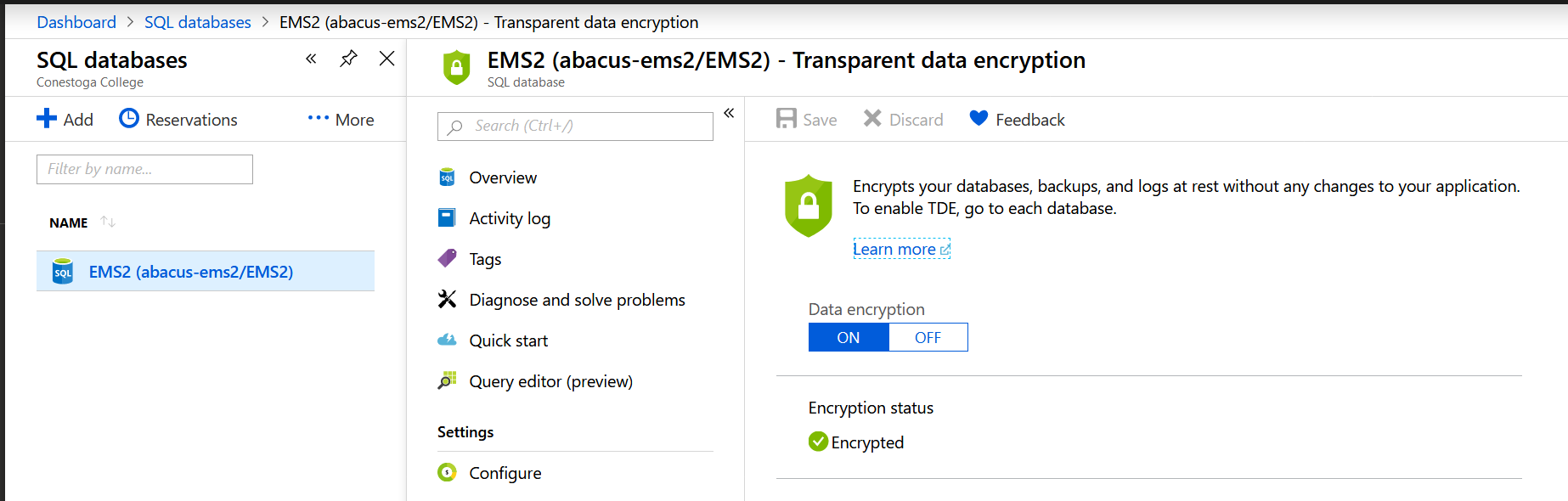


Figure 3 - Data Encryption