Conduct an Application Security Review

Scenario Document

Customer Information Management System (CIMS)

Summary

In order to facilitate better communication and support for customers, we have built a customer information management system (CIMS). This system features a database containing two tables for customer-related data. This data can be updated by either customers directly through our portal or by customer relationship managers using our internal web interface. In order to better serve our customers, our data scientists are also able to query the database to gather data.

Requirements

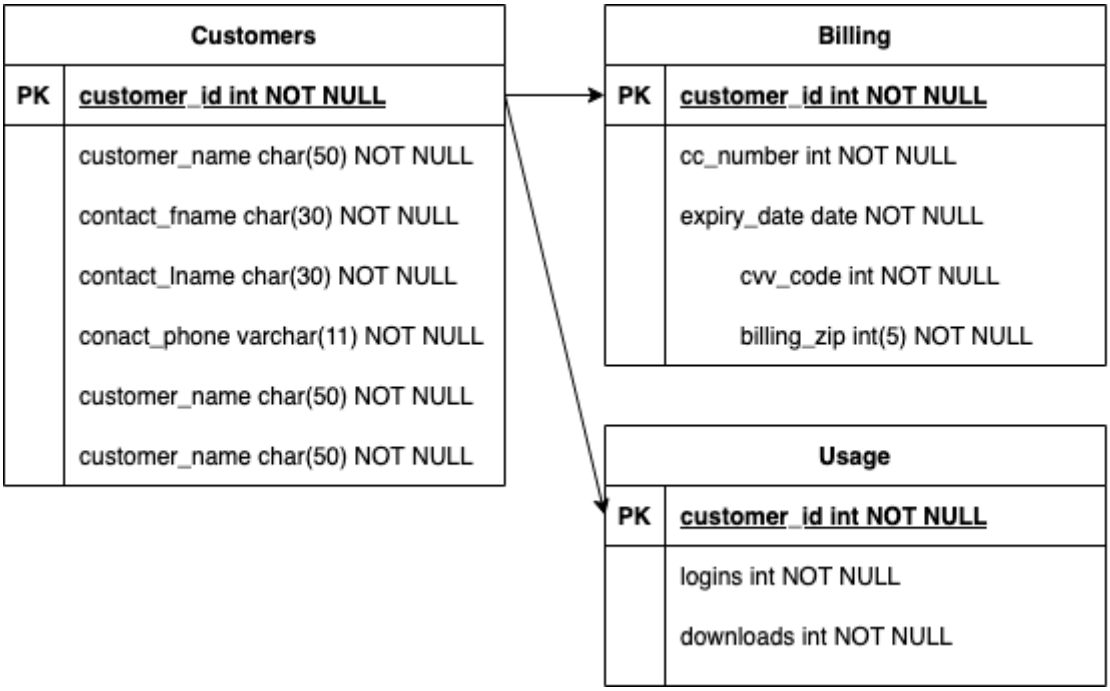
- Customers must be able to update their own information through the product portal.
- Customer Relationship Managers must be able to update customer information through the internal web interface.
- Data scientists must be able to query the database.
- Applications must be compliant with SOX and PCI/DSS.

Architecture

PostgreSQL Database

The architecture is centered around a PostgreSQL database located in our datacenter on premises. The database was deployed by the customer support team following an online tutorial and so has not been integrated with any of the information technology group's existing infrastructure. The database has been left in its default configuration without audit logging, SSL certificates, or database encryption.

In the database, there are three tables. One holds the customer point-of-contact's name, email, and phone number, using the customer's organization ID as the primary key. This table also contains the number of users in the organization, their contract value, and renewal date. This key lets us link to the second table, where all processed credit card information is held so it can be used in recurring billing. Customers can then use the same interface to update their billing information. A third table, again using the organization ID as a primary key, holds customer product usage information.



Customer Updates

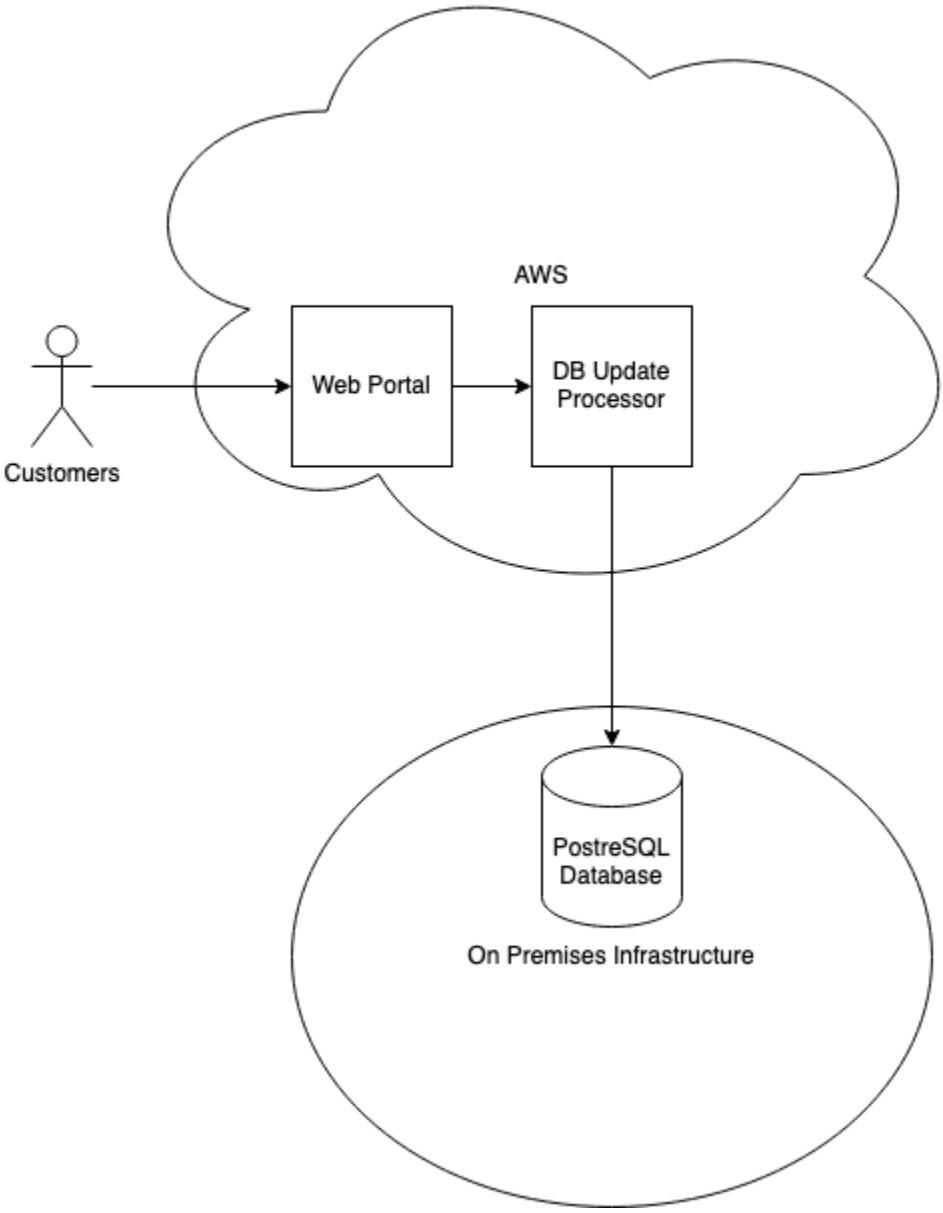
The web team implemented an update feature for customers. Customers will log in through the normal portal using the Information Security-approved login methods. The integration is a new page that allows customers to enter updates to names and contact information. The updated information is wrapped into a SQL query, put in the body of a POST request, and sent over HTTPS from the portal server in AWS to another virtual machine in AWS, on the same virtual private cloud (VPC), that processes the database updates. This POST request must contain an authentication header.

```
POST /query HTTP/1.1
Host: database.local
Connection: keep-alive
Content-Length: 0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90 Safari/537.36
Content-Type: text/plain;charset=UTF-8
Accept: */*
Origin: webmanager.local
Referer: webmanager.local
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;
Cookie:UID=212=Lcl8QvHtoF9MhbPNrdXZqukjfw_vZrL0LVzD-CKHRai-ySeaWAkc5Am27x8iMYjrzoACVCrNaOjCvEpld8EIGSYXafS8hXGjov7NVhuzUqEYptUTKpFK8YpHH58hhL1fSbaR_bd5YYaKMdu4A7vOWmYDs_Z5y5BNMAW5FsF

{query: INSERT INTO {{table}} VALUES ({{vals}});}

```

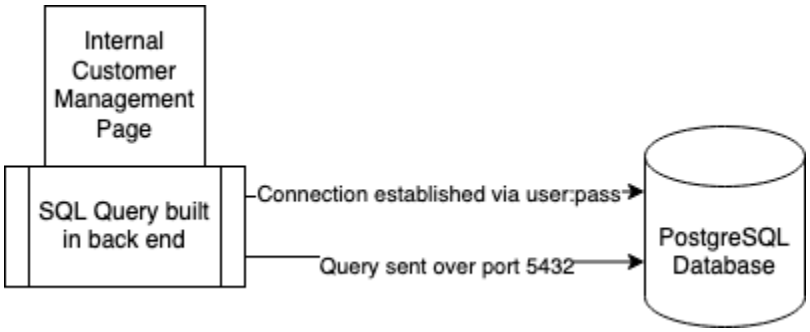
When the database update processor receives a POST request, it uses SQLAlchemy to create a session with the on-premises database using the issued username and password. Then the script relays the supplied SQL query to the database. Additionally, a firewall exception request has been granted for port 5432 from the public IP address of the DB Update Processor.



Customer Relationship Manager Access

In order to facilitate access for customer relationship managers, a username and password has been built into an internal web form. Similarly to the database update processor in AWS, this form lets customer relationship managers search, update, add, and remove records. This is part of the internal management page, and being part of our on-premises infrastructure, the traffic can be routed directly to the PostgreSQL database from the management portal's machine.

A similar SQLAlchemy backend as in the DB update processor is built into the internal tooling, using a different username and password than the account in the DB update processor. Rather than sending the command via POST request, the query is built on the back-end of the web application and the connection to the database is established directly and temporarily.



Data Scientist Access

Since data scientists need a different sort of database access and are generally more fluent with SQL, each data scientist is issued a read-only account to the SQL database to pull the data back to their local machines. Data scientists are allowed to query in whatever way they prefer (SQLAlchemy, Psycopg2, etc.) but can only query the database if they are connected to the corporate network.

Once data scientists have done their work, the intent is to wrap the data into machine learning models that are deployed to the cloud via Kubernetes containers so that they can help predict things like whether a customer is at risk for non-renewal, could be upsellable, and so on. The output of these models will be served to customer relationship managers on the internal customer management page.