**Security Design Project**

**Version** 1.0

22 April 2015

***The Veni VA Checkin System***

Advanced Software Engineering Project

SE 6324 – Spring 2015



*Anant Kambli*

*Brian MacKay*

*Michelle Mulkey*

*Raleigh Murráy*

*Shahed Shuman*

*Kathryn Whitmire*

Contents

[1. Overview 3](#_Toc417213561)

[2. Secure Design Principals 3](#_Toc417213562)

[2.1 Security Policy 4](#_Toc417213563)

[2.2 Network Architecture 4](#_Toc417213564)

[2.2.1 The Smartphone to Veni Link 5](#_Toc417213565)

[2.2.2 Security in the AWS Cloud 5](#_Toc417213566)

[2.2.3 Additional VistA Security 6](#_Toc417213567)

[2.3 Security Mechanisms 6](#_Toc417213568)

[3. Security Tools 7](#_Toc417213569)

[4. Test 7](#_Toc417213570)

[Bibliography 8](#_Toc417213571)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Description** | **Authors** |
| 1.0 | 21-Mar-2015 | Completed initial draft | Group |
| 2.0 | 11-Apr-2015 | Update of section 2 content | M. Mulkey |
| 3.0 | 22-Apr-2015 | Minor formatting corrections | K. Whitmire |

# 1. Overview

The Department of Veterans Affairs (the “VA”) manages the largest integrated health network in the country. On average the VA provides care to more than 6 million Veterans in more than 150 hospitals, 800 clinics and 135 skilled nursing home facilities. The VA’s Electronic Health Record (EHR), called VistA (Veterans Health Information Systems and Technology Architecture) is the heart, soul, and integral electronic information service essential to the quality of care the VA delivers.

A large part of the VA’s health services delivery system relies on its outpatient facilities. Veterans making use of these services are required to go through a multi-step process:

* Make an appointment via telephone or the VA website
* Show up at the facility and “check-in” at a central location
* Go to the location of their appointment and wait to be called

This system addressed the second step of that process. At the moment, veterans can check in two different ways. All facilities provide a manual system; one in which the veteran interacts with a clerk at a computer terminal. Some facilities also provide checkin kiosks – much like the automated checkin devices that airlines operate. These systems have high acquisition and operating costs. They also require regular mechanical maintenance

This system will provide a third means for a veteran to check in; one based on a Smartphone application that communicates with the VA’s VistA system via a cloud-resident middle-tier system.

Much of the VA’s clientele comes from an older generation; veterans of the Viet Nam war or earlier conflicts. However, after more than 10 years of war, the country’s veteran population has skewed much younger. This younger generation of veterans is tech-savvy and impatient with manual system.

The goal of the Veni Checkin System isn’t to supplant either of the two existing systems. Instead, it means to provide a third check in mechanism; one that is both lower cost and capable of improving veteran satisfaction with the VA.

The system will include two components: a user interface in the form of a smartphone app and a cloud-resident server interface to handle communication between the phone and the VA VistA EHR system. In addition, as part of this prototype work, the prototype includes a test-only VistA system containing sanitized data – data with no personal information or protected health information.

The system has been constructed with a mindset to reduce cost, minimize support and implementation.

# 2. Secure Design Principals

The prototype includes three components: a user interface in the form of a smartphone app, a cloud server (referred to as the ‘middle-tier’) to handle communication between the phone and the test-only VA VistA EHR system. The security policy for the VA VistaA database application may be found in the VA Handbook 6500 Risk Management for VA Information Systems, Appendix F VA System Security Controls. The security policy for the Veni application draws from many topics covered in the Information Security class.

## 2.1 Security Policy

The system is designed with the knowledge that it will maintain both “personally identifiable information” (PII) and “protected health information” (PHI). Both are targets for hackers and information thieves.

All aspects of the Veni system are designed around the need to protect that information. This includes minimizing and obfuscating the data stored by the middle tier and transferred between the Smartphone app and the middle tier. For example, once a session is established, all “key” data transferred between the phone and the server is based on large random numbers unrelated to any user data or to other keys (so as to prevent a malicious user from guessing another user’s key). In addition, the system will not store clear-text passwords, instead, it stores salted password hashes. Eventually, the system will include selected encryption (by database column) of important information in the middle tier database.

## 2.2 Network Architecture

The diagram below shows the rough network topology of the Veni system and its interconnections. The VistA server and its accompanying database are currently resident in the Amazon Web Service (AWS) cloud during the development and early test phases of the project. However, when this system goes into production, the VistA system(s) will be housed within a Veterans Administration datacenter.



The VistA system runs on Linux. Currently both the Veni Application Server and its accompanying database server run Windows Server – mostly for development expediency.

The primary data-path runs from the Veteran’s Smartphone to the middle-tier Veni Application Server to the VistA Server. The Veni database holds only enough data to make the system functional. The bulk of the business data resides in the VistA system.

### 2.2.1 The Smartphone to Veni Link

The Smartphone app is being built in an OS-independent development environment. During development, it communicates with the Veni system using HTTP. However, once it goes into production, HTTPS/TLS will be used. At the application level, the communication uses JSON and a simple stateless REST-based protocol.

Veterans – the primary Veni system users – authenticate to the system using a simple username/password combination. The middle-tier API is flexible enough to substitute other credentials (perhaps provided by the phone OS) if available. Initially, the only server authentication in the system will be that provided by TLS. However, this may be upgraded to provide the veteran with visible authentication of the server using a system similar to Bank of America’s SiteKey® picture-based system.

The system currently doesn’t authenticate user devices. However, our long term plans included leaving an encrypted and hashed token in persistent storage on the phone. If the veteran connects to Veni using a new device (one that doesn’t have this token), the user will be asked to answer his/her “password reset questions” to prove that the device truly belongs to him/her.

### 2.2.2 Security in the AWS Cloud

The Amazon Web Service cloud provides server security using “Virtual Private Clouds” described using “Security Groups”. Security Groups are conceptually similar to firewalled subnets, but are more flexible and less resource intensive. Servers within a security group are protected by a list of Port Number – Protocol – IP Address (or address range) tuples. Other systems can only interact with servers in the security group if their interaction matches one of these tuples.

The current system includes three security groups:

1. The Veni Application Server Security Group:  
   Servers in this group (currently a single server) can only be accessed by:
   1. Any system using HTTP on port 80 (this will be HTTPS/TLS on port 443 in production)
   2. A short list of IP addresses using the Windows Remote Desktop Protocol (RDP) on port 3389. This allows the administration of the system
2. The Veni Database Server Security Group:  
   Servers in this group (currently a single server) can only be accessed using SQL Server’s TDS protocol on the standard SQL Server port (1433). There are two entries in the Security Group access list:
   1. Members of the Veni Application Server Security Group
   2. A short list of IP address – this allows administrative access to the database server
3. The VistA System Security Group  
   The VistA System Security Group only allows TLS access over port 443. It does not restrict this by IP address. The VistA system’s normal authentication and authorization mechanisms provide an additional layer.

### 2.2.3 Additional VistA Security

The VistA system isn’t designed to be protected by the AWS Security Group mechanisms. The Veni system will communicate with it using a REST-based protocol implemented by an EWD.js software server. That system protects itself beyond using simple SSL/TLS. In general, only authorize principals can access VistA. This protection is based on SSH-like key-pairs that are integrated into the EWD authentication protocol.

When a user or system authenticates to VistA via EWD, a multi-round-trip handshake results in the exchange of a session key using the provided SSH public/private key-pair. Once the session key is established, all communication is encrypted using that symmetric key (before being transferred using TLS).

## 2.3 Security Mechanisms

Only a veteran with an account can access his or her user data. The system includes session time-outs on both the client and the server; should a veteran pause too long within the app (or abandon his/her session), a re-authentication will be required.

In the eventual system, Transport Layer Security (TLS) protocol is used to authenticate servers (*Veni* middle-tier and to encrypt messages between the authenticated parties. Encryption is used to protect information transmitted over the datalink and stored on the mobile device. The data being transmitted contains personal information, which left unprotected, could lead to “inference” of a patient’s health.

The smartphone app utilizes several design principals to yield assurance the security policy and mechanisms are sound.

* **Separation of Duties** – Only one user is authorized, per mobile device, to use the smartphone app at a time. The authorized user only has his or her data.
* **Least Privilege** – There is only one level or type of user. The smartphone app does not escalate user privilege and there is no system administrator role (except via direct-connect mechanisms).
* **Unsuccessful Logon Attempts** – Both the smartphone app and the server will provide an “anti-hammering” mechanism to prevent repeated logon attempts – artificial delays are introduced after each unsuccessful logon. A future design consideration is the addition of a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) test.
* **System Use Notification** – Users will be shown the *Veni* Privacy and Security Policy Notice on each logon
* **Previous Logon (Access) Notification** – The smartphone app notifies the user, upon successful logon (access) to the system, of the date and time of the last logon (access). This can help identify system breaches
* **Concurrent Session Control** – The smartphone app does NOT allow concurrent sessions. In addition, a user can only be accessing the system from a single device at one time.
* **Session Termination** – Both the smartphone app and the middle-tier server will timeout after a period of inactivity. Attempts to access the system beyond the timeout will require re-authentication.
* **Access Control for Mobile Devices** – Other than the veteran’s username, no user information is stored locally on the veteran’s smartphone.
* **Non-Repudiation** – The main purpose of this system is to allow a veteran to checkin at a VA health facility. The GPS (and the phone’s location services) in the veteran’s phone are used to verify that the veteran is, in fact, at the facility at the time of the checkin operation. A record of the time and location of the checkin transaction is recorded in the middle tier system to prove that the veteran was at the facility and did, in fact, check in.

# 3. Security Tools

The smartphone app leverages many open source tools and technologies as well as the corresponding vendor security policies.

* TSL - This ensures the logon details and contents of messages remain encrypted and provides authentication of the server to the client.
* Only salted password cryptographic hashes will be kept by the system (using the PBKDF2/RFC-2898 hashing mechanisms)
* Opaque tokens

# 4. Test

The smartphone app leverages many open source tools and technologies as well as their corresponding vendor security policies. As such, there is minimal testing for security vulnerabilities.

* **Malware** – The smartphone app runs in the security sandbox provided by the Smartphone OS. The application itself does not provide for defense against malware. The recovery mechanism is to uninstall/reinstall the smartphone app.
* **Activity Log** - The middle-tier contains an activity logging; every event in the middle-tier is logged. Currently, this is a debugging feature. A future design consideration is to leverage both an Intrusion Detection System (IDS) and an Intrusion Prevention Systems (IPS). The logs will also be obfuscated to prevent PII and PHI leakage.
* **DBMS** – The middle-tier uses is a relational database application uses the security provided by the database system vendor
* **Web Service** – The middle-tier is an Amazon cloud based application and is dependent upon the Amazon Security Policy.
* **Buffer Overflow** - the smartphone app uses JavaScript and the middle-tier uses C#, both of which make buffer overflow, integer overflow and similar vulnerabilities unlikely.
* **Penetration and Fuzz Testing** – The goal is to test the middle tier against possible attacks using malicious inputs to the system.

Bibliography

* B. Lampson. Protection. ACM Operating System Reviews, 8, 1974
* M. Kantarcioglu.CS6324.M01 Information Security Class Notes, Spring 2015
* ["VA Publications." VA Publications. Web. 29 Mar. 2015.](http://www.easybib.com/cite/clipboard/id/1427665762_55187362d617a1.23664215/style/mla7)
* "Amazon Web Services: Overview of Security Processes." Amazon Web Services. Web. June 2014
* "Security Guidelines." MySQL. Oracle, 2015. Web. 29 Mar. 2015.