Introduction Information Security

Spring 2015

Security Design Project

Anant Kambli

Brian MacKay

Michelle Mulkey

Raleigh Murray

Shahed Shuman

Kathryn Whitmire

Table of Contents

[Overview 3](#_Toc415415066)

[Secure Design Principals 3](#_Toc415415067)

[Security Policy 3](#_Toc415415068)

[Security Mechanisms 4](#_Toc415415069)

[Access Control 5](#_Toc415415070)

[Other Access Control Methods 6](#_Toc415415071)

[Security Tools 6](#_Toc415415072)

[Test 6](#_Toc415415073)

[Bibliography 7](#_Toc415415074)

# Overview

Veterans Affairs (VA) is the largest managed integrated health network in the country. On average VA provides care to more than 6 million Veterans from 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. How can we leverage VistA integrated capabilities? How do we empower Veterans? Let us start with the “VENI App Check-in System.” The VENI App Check-in System is a cloud based system that allows veterans to seamlessly check in to all clinics from his or her smartphone.

With the country winding down from more than 10 years of war, the veteran population has become much younger and much more technically adept. Not every veteran will want to use a smartphone application to interact with VA medical facilities, but with a clientele that numbers in the millions, the number of potential users is very large. The current check-in process at the VA requires veterans to wait in line to check-in with either an administrator or a kiosk. Throughout the VA medical system, veterans and their families can spend hours waiting in line to check-in for their appointments. Once checked-in at the reception, they have another wait after arriving at the appropriate doctor’s office or clinic. Mistakes as a result of incorrectly transcribed appointment information only compound the problem. The *Veni* system will allow veterans to check-in upon arrival without this ridiculous wait-time, plus have the capability to download appointment times and related information and give directions to the appointment facility and office location.

The system will include three components: a user interface in the form of a smartphone app, a cloud server interface to handle communication between the phone and the VA VistA database, and a cloud virtual machine to simulate the VistA database (for initial prototype usage). For the VA, the system offers a simple lightweight solution that will free administrative staff from the check-in process. With a simplified, veteran-focused, check-in experience, the reception area can be rededicated to offer services other than simple registration.

The system has been constructed with a mindset to reduce cost, minimize support and implementation. Open source tools and technologies were leveraged to reduce the cost and expenditure, so the VA can spend monies on our returning home veterans. From one veteran to the next, *Veni* got your “six”!

# Secure Design Principals

The software architecture 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 VA VistA database. The security policy for the VA VistA 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.

## Security Policy

The class notes lists the top three types of breached information as:

* Real names
* Birth dates
* Government ID numbers (Social Security)

The smartphone app requires access to user data which includes real names and VA appointment data. This information must be protected against mobile device malware and unauthorized access during data transmission.

The smartphone app security policy provides for Confidentiality, Integrity, and Availability of this user data:

* Confidentiality - user data is NOT disclosed to arbitrary users of the mobile device.
* Integrity – user data is NOT modified by arbitrary users of the mobile device.
* Availability – user data is exchanged with VistA when the user checks in.

## 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.

### 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, we may decide 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.

### 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

### Additional VistA Security

The VistA system isn’t designed to be protected by the AWS Security Group mechanisms. It 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 authentication protocol.

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

## Security Mechanisms

Only an authorized user shall access their user data. The smartphone app uses password hash and encryption to enforce the security policy. User authentication, via password, is used to verify the identity of users.

The Transport Layer Security (TLS) protocol is used to authenticate servers (*Veni* middle-tier) and clients (*Veni* smartphone app), 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 access to the same user’s data. The middle-tier design includes the concept of a session. The sessions are uniquely identified and monitored for session expiration. When a session expires, the user is automatically logged off.
* **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.
* **Unsuccessful Logon Attempts** – the smartphone app enforces a limit of consecutive invalid logon attempts by a user during a specified time period and automatically takes action when the maximum number of unsuccessful attempts is exceeded. 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** – the smartphone app displays to user a system use notification message or banner before granting access to the system that provides privacy and security notices. Currently, the *Veni* system displays a *lorem ipsum* banner. A future design consideration is the addition of the *Veni* Privacy and Security Policy Notice.
* **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).
* **Concurrent Session Control** – The smartphone app does NOT allow concurrent sessions. A future design consideration is to limit the number of concurrent sessions for each account to a maximum number of sessions.
* **Session Lock** – the middle-tier is stateless and does NOT provide for session locking/unlocking. After a period of inactivity, the session is terminated. The user must reestablish access using username and password.
* **Session Termination** – The smartphone app automatically terminates a user session after defined conditions or trigger events requiring session disconnect. This termination trigger is no activity between phone and middleware for five (5) minutes.
* **Remote Access** – Today, the middle-tier may only be accessed through a smartphone. A future design consideration is to provide for website access.
* **Access Control for Mobile Devices** – the smartphone app does not store/transmit VA data. There is no VA data stored on the Mobile Device. A future design consideration is to provide for device recognition, security questions, and a security image for middle-tier assurance.
* **Non-Repudiation** – the smartphone app does NOT create user appointments. The middle-tier downloads new appointments and checks the status (changed or cancelled) of existing appoints for the user each time the user logs in. The Veni “Check-in” feature invokes the mobile device to request GPS location which is then sent to the middle-tier. If the user is within the appointment’s spatial and temporal boundaries (e.g. proximity to the facility and 30 minutes before scheduled time), the smartphone app displays instructions on how to proceed. In the event of a technical failure or malfunction, the user may be instructed to find a real person for further assistance.

## Access Control

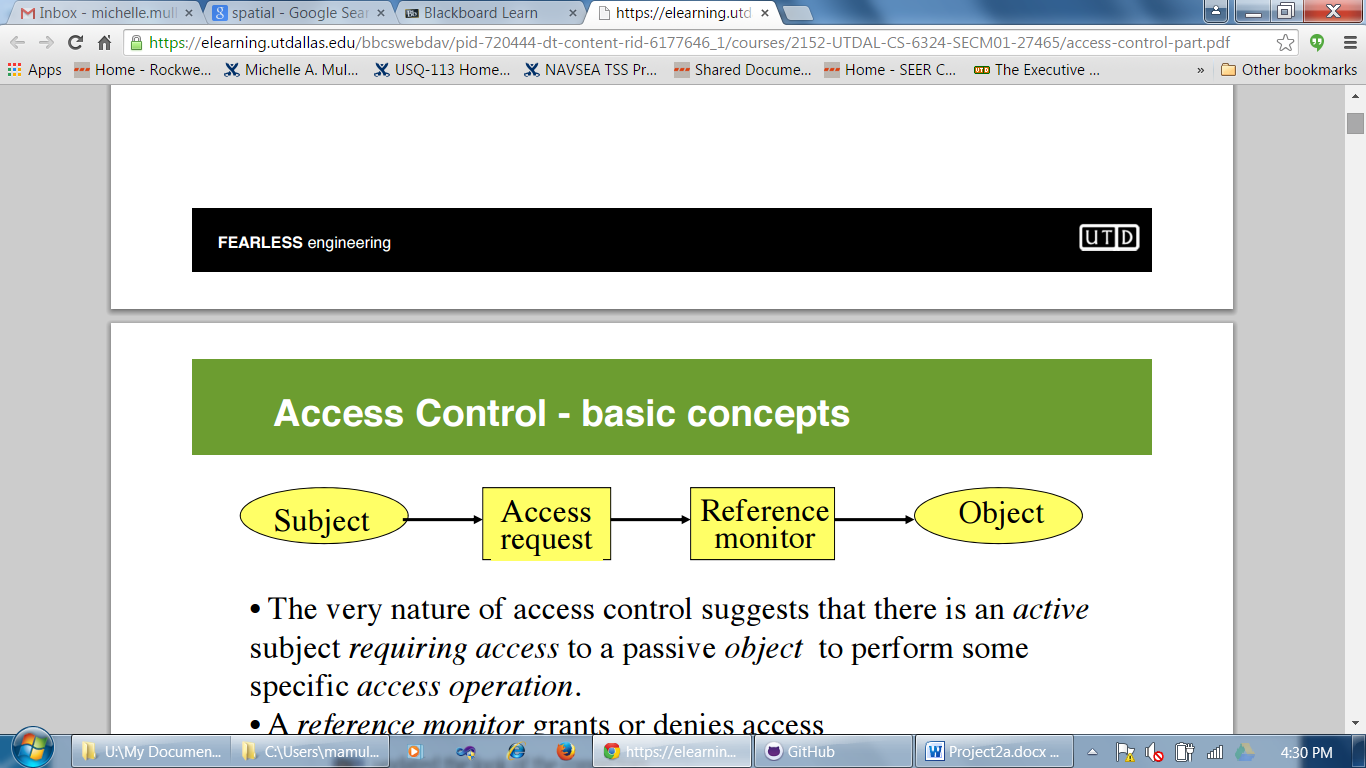


Figure 1 Access Control Model

Lampson’s Access Control Model “suggests that there is an active subject requiring access to a passive object to perform some specific access operation. A reference monitor grants or denies access.”

For the *Veni* smartphone app, the active **subject** is the user or smartphone app component. The user has a username and password. The passive **object** is the VistA appointments, updates, reminders, and confirmations. The **reference monitor** is the middle-tier component. The middle-tier stores the HMAC password and generates GUID like session identifiers to authenticate access requests. The transport protocol is TLS which authenticates the subject, reference monitor, and the object. TLS also prevents “man –in-the-middle attacks”.

### Other Access Control Methods

The smartphone app executes using discretionary access which is defined by VistA. The middle-tier has a database to store user information (list of appointments, list of facilities, user name, and a cache.) Since this middle-tier database contains both Personal Health Information (PHI) and Personally Identifiable Information (PII), a future design consideration is to encrypt this database. The smartphone app uses a username and password for user authentication. A future design consideration may employ fingerprint and/or retina scans.

# 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. This renders intercepted traffic useless to a hacker.
* HMAC algorithm to encode passwords
* Opaque tokens

# 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 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 develop both an Intrusion Detection System (IDS) and an Intrusion Prevention Systems (IPS).
* **DBMS** – The middle-tier uses is a MySQL database application and is dependent upon the MySQL Security Guidelines (e.g. do not store plain-text passwords in the database)
* **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 which eliminates buffer overflow.
* **Fuzz Testing** – the smartphone app input parameters are tested.

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.