Design Document

Vinz

Team May14-32
Michael Davis
Maxwell Peterson
Jacob Hummel
Zach Heilman
Eric Feldman
Ario Xiao Qin

Advisor
Dr. Mitra

Client
Dave Tucker, WebFilings

Background

Summary

SSH (Secure Shell) is a network protocol for secure data transfer, remote server login and execution. They are widely used to manage shared resources in Linux Server when working on Cloud Computing environment. SSH uses a public/private key pair system in order to authenticate the user to the server.

The problem is that when companies or organizations have hundreds of Linux servers running, it becomes very hard to manage access to all of the servers. Current solutions consist of a number of options, including writing custom scripts to attempt to handle managing keys across an entire organization or attempting to use an LDAP system to handle authentication across an entire network. Custom scripts quickly fail when scaled to a large number of servers and LDAP and other authentication systems are very difficult to construct and maintain.

We aim to solve this problem by creating a system for managing SSH keys that will itself run in a cloud environment. The system will provide a secure way for administrators to manage SSH keys, apply access rules and group servers. Users can easily install and setup the system, and are able to manage access to logical group of servers. We believe that our system will solve this real issue for many companies that make use of cloud environments, and we will keep updating and enhancing the system beyond the duration of this course.

Target Operating Systems

Vinz itself will be installed on a Linux operating system. This machine can either be a physical server within an organization or a cloud server that they operate.

Clients will access Vinz with a web browser. Any operating system or web browser will work.

Functional Requirements

Users

Users are able to do the following operations after connecting to Vinz.

- Sign in using two-factor authentication
- Generate and upload new SSH keys
- Delete SSH keys

Administrators:

Administrators are able to do the following operations after connecting to Vinz.

- Create new users
- Add new servers
- Arrange servers in logical groups
- Manage the groups of users and servers, control what server/ group of servers can be accessed by which users
- View full access reports and existing key usage.

Optional Requirements

Administrators:

- Able to set up systems to automatically add new servers

Non-functional Requirements

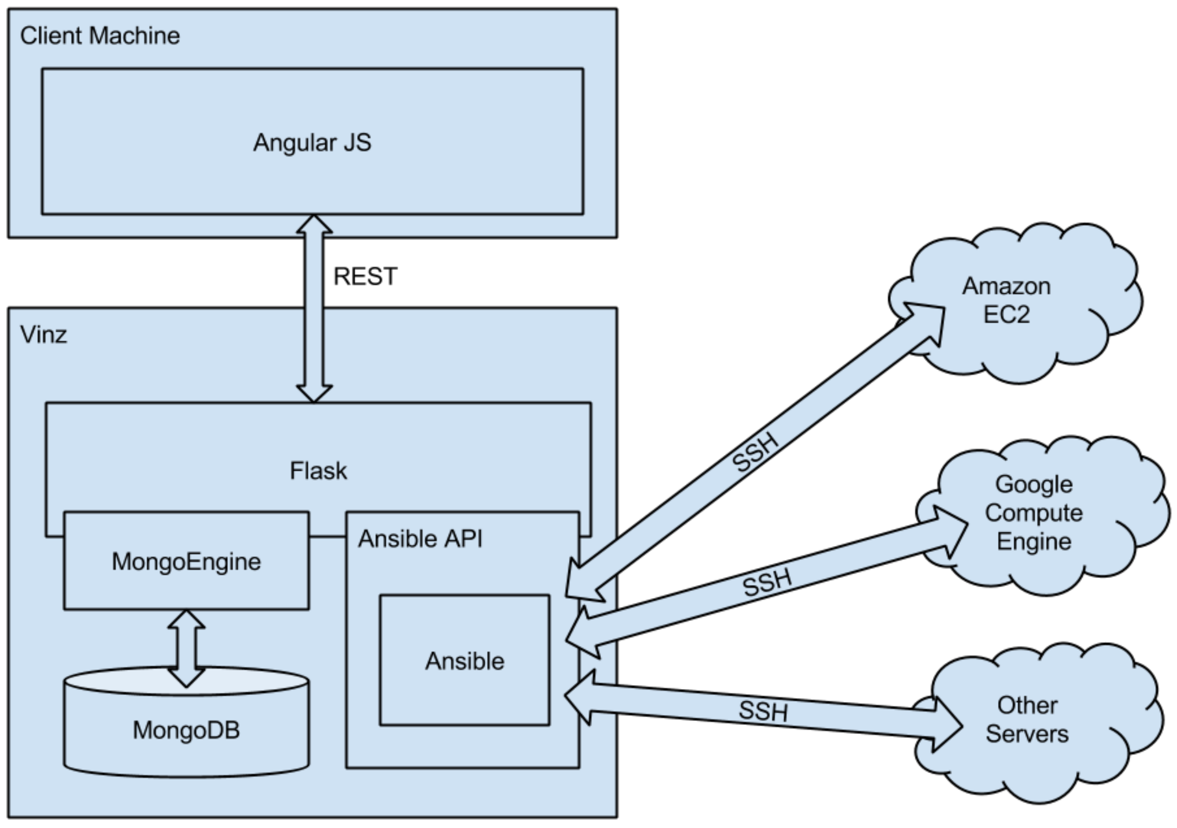
- Easy to install and use
- Account deletion and removal of access should be as quick as possible for the case of a user being fired
- Access restrictions must be enforced by scanning servers periodically
- Adding new APIs should be modular and easy to add

System Design

System Requirements

The system will consist of one application running on a linux-based server. The application will be responsible for serving up the frontend web application (described below) as well as responding to a number of different REST endpoints in its role as a REST API on the backend. It will store data in a MongoDB-driven database also hosted on the linux-based server. Additionally, the server will be connected to the same network as the servers it will manage as a means of communicating with them. In most cases this network will be the Internet.

Architecture Diagram



Functional Decomposition

Frontend

Our frontend will expose a graphical interface for users and administrators to be able to use Vinz. The interface allows users to upload their SSH public keys to the system so that Vinz can push them out to all of the servers that the user has access to.

Admins will also use the frontend for a number of different tasks, including managing users on the system, managing servers that Vinz controls, add servers and users to different groups to allow for easier user and server management, and viewing a number of different audit reports.

Backend

The backend server will expose a REST interface that allows access to the system and it's features. This will be the main interface for allowing access to the database and controlling all of the servers connected to Vinz.

Our frontend will leverage this backend in order to handle all of the tasks needed, but the backend will also be exposed so that other services can take advantage of it programmatically without having to add development time of exposing those services.

The backend will use the Ansible API in order to interact with servers that it controls. Every interaction that the backend has will go through the API.

The backend also contains a scheduler for the scanner. When the scanner reports an issue with one of the servers, the backend handles taking the proper action, whether that is to instantly remove the SSH key by using the Ansible API or simply alerting a administrator.

Ansible API

The Ansible API will leverage Ansible in order to provide a easy, sustainable, and extendable method for Vinz to access servers that have been added to the system. The API will expose a number of different services to the backend to allow for managing SSH public keys on systems.

Scanner

The scanner will be responsible for accessing all of the systems that Vinz controls in order to verify that the SSH keys on each server represent what should actually be on the server.

Database

Vinz will use MongoDB as a database to store a variety of data, including user info, ssh keys, and server info.

Detailed Design

Component Interfaces

Frontend<-->Backend (REST)

The AngularJS-based frontend will use AJAX calls to query the backend's REST API.

Backend<-->Servers (Ansible API)

The backend Python application will make calls to the Ansible API to add, remove, modify, and manage servers.

Backend<-->Database (Mongoengine)

The backend will use MongoDB's ORM (Mongoengine) to perform queries against the database.

Scanner<-->Backend

The Python-based scanner will make calls to the backend to ensure that managed servers are compliant with the assigned policies.

Database

Schema

The database is Non-Relational (NoSQL) and implemented with MongoDB. Each of the following underlinings represent one collection stored in the DB (logically similar to a SQL table).

Server

- id
- name
- hostname
- key_list
- creator
- create_date
- modifying_user
- modified_date

ServerGroup

- id
- name
- server_list
- creator
- create_date
- modifying_user
- modified_date

PublicKey

- id
- owner → Reference to User
- value
- create_date
- expire_date

User

- id
- first_name
- last_name
- email
- key_list

UserGroup

- id
- name

- user_list
- creator
- create_date
- modifying_user
- modified_date

ActivityLog
- id
- actor → Reference to User
- timestamp
- object → Reference to Server, ServerGroup
- action

Frontend UI

Target Audience

Vinz is targeted at any organization that uses SSH keys. The idea came to be because there are no easy to use solutions to the SSH key administration problem. Anyone who wants their SSH key distribution to be auditable, visual, and simple should be interested in this application. Vinz will be downloadable online, completely free of charge. The source code is available on GitHub for contributions and maintenance.

Implementation

The user interface will be implemented using HTML, CSS, and JavaScript. To create code that is as clean and testable as possible, we will make use of the Superheroic MVC JavaScript Framework by Google, AngularJS. To make things look nice (and more responsive) we will use Twitter Bootstrap 3.0.

Page Descriptions

Install setup

The download will come with a setup script to be run when you are ready to set up the application. This will require you to fill out some information, and then run it in the command line.

The following information will need to be filled out in the script:

- Admin username and password credentials
- Specified database information

Login

When any user enters the url of the hosted instance of Vinz, they will be directed to this login screen. The screen will be standard, and asks that you enter a valid username and password. If a valid credentials (username and password) are input, then the user will be directed to the *My Servers and Groups* homepage. If invalid credentials are given, then the user is shown the appropriate error message.

My Servers & Groups (Homepage)

The My Servers & Groups page will be the landing page for the application. The center of this page will focus on showing, in a table view, the currently enrolled server and user groups. The table will show metadata about each group, and allow you to explore the group details further by selecting it.

Settings Page

Show and edit user settings for a user account.

- Settings to be used:
    - notifications/alerts
    - profile pic/title/bio
    - change password
    - upload/manage public keys
    - 2-factor auth settings

User Management
Scanner Settings

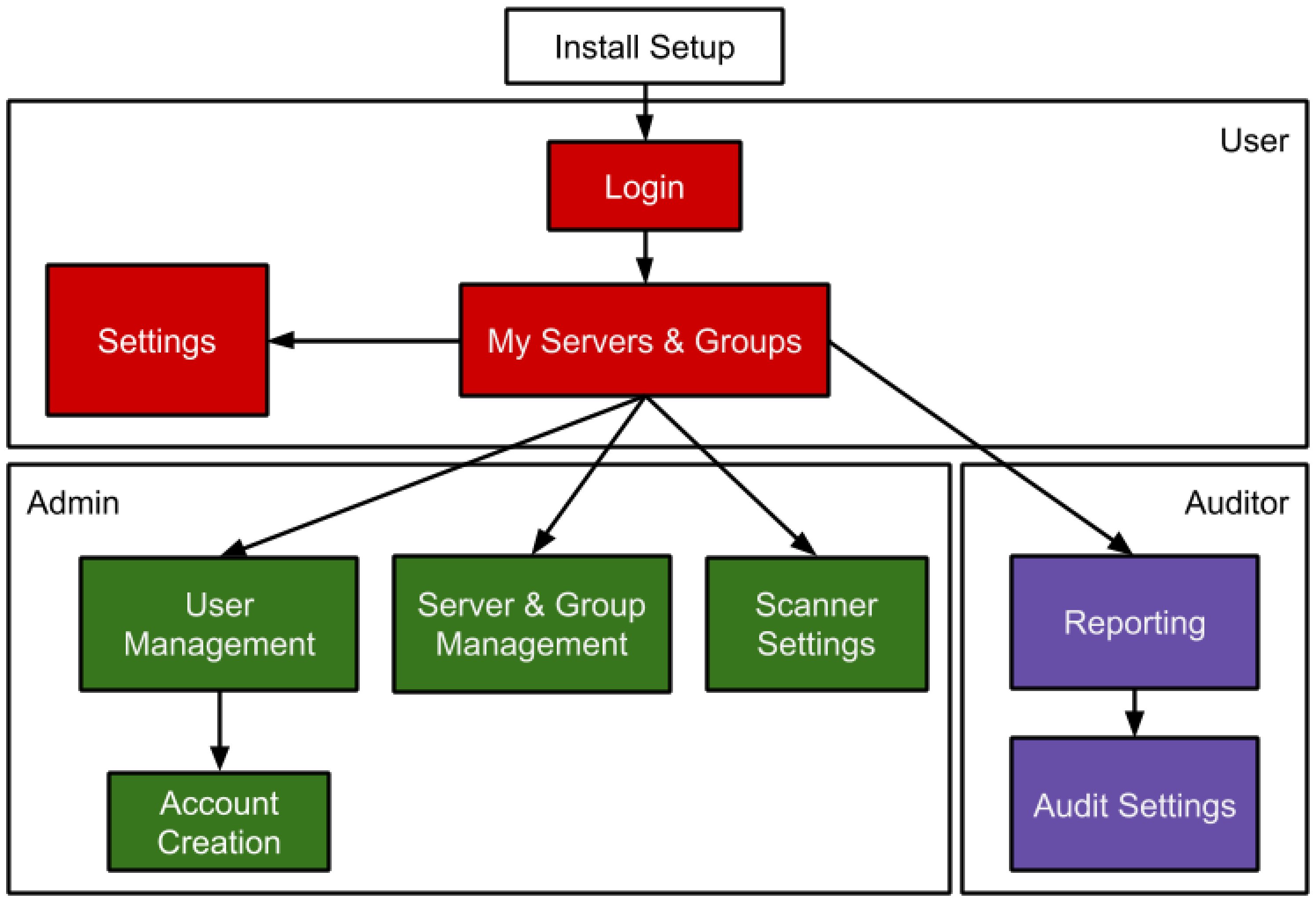
- Frequency
- What to do in event of anomaly

Account Creation
Reports page

- Every machine/who has had/has permission
- Every network group/who has had/has permission

Screenflow diagram

Vinz



*__Auditor__ is also a **User**
*__Admin__ is both an **Auditor** and a **User**