NovaTooth
Project Requirements Document v2

Men-in-the-Middle:

Kevin Chan (Project Lead)
- kevchan216@gmail.com

Trevor Morris (Scribe)
- joeyotrevor@gmail.com

Robert Stosick
- rstosick@gmail.com

Henry Yu
- henryyu.yu@gmail.com

Albert Chen
- treblanehc95@gmail.com

Revision History:

- 10/28/2016 Initial Draft
- 11/12/2016 Updated Appendix & Glossary of Terms
- 11/18/2016 Added System Models
- 11/27/2016 Added Tests -> GitHub Commits/Issues

Introduction:

Bluetooth is a widely-used wireless standard that operates on the unlicensed 2.4 GHz band. Bluetooth is ubiquitous in both the corporate environment as well as in consumer products, and it can be found in everything from automobiles to phones, medical devices, keyboards, and hundreds of other applications. Over 25,000 companies are a member of the Bluetooth Special Interest Group, the organization responsible for the maintenance of the Bluetooth standard.

Although Bluetooth's universal deployment is an asset, it also creates liabilities. One problem that companies who employ Bluetooth devices face is how to audit the security of their Bluetooth devices, and how to handle the security implications of their Bluetooth usage. This problem is compounded by the lack of commercially-available tools for Bluetooth security analysis and penetration testing. Our team's research discovered that there are no such devices available on the open market, and furthermore there is no code available to facilitate such analysis.

The problem our team seeks to solve is to develop a discrete, portable Man-in-the-Middle (MITM) device that will be able to attack Bluetooth connections, and thus eavesdrop on connections between Bluetooth headsets and Bluetooth-enabled phones. We plan to wait for unsuspecting users to connect to our Bluetooth MITM device, at which time will either reroute the connection or eavesdrop on data transmitted between devices. Our MITM device will feature the ability to record intercepted speech to an audio file, and furthermore, we will use machine learning techniques by integrating with Google Speech or another third party API that utilizes neural networks in order to transcribe intercepted audio and to facilitate efficient mass data analysis.

The core technological advance we seek to make is to develop the software necessary to allow a Linux-based device to impersonate a Bluetooth headset and forward packets/audio data as necessary to conceal from the user that their Bluetooth connections is being eavesdropped on. A further core technical advance we seek to make is to develop the software necessary to take streaming audio and convert it to transcribed text, utilizing APIs for existing services such as the Google Cloud Speech platform.

Furthermore, all of this software must be efficient and reliable, so that it can be run on a compact device, possibly one which is battery-powered (to be determined in a later phase of the project). This means the software must consume minimal resources, use bandwidth efficiently, and be able to run with minimal or no operator interaction.

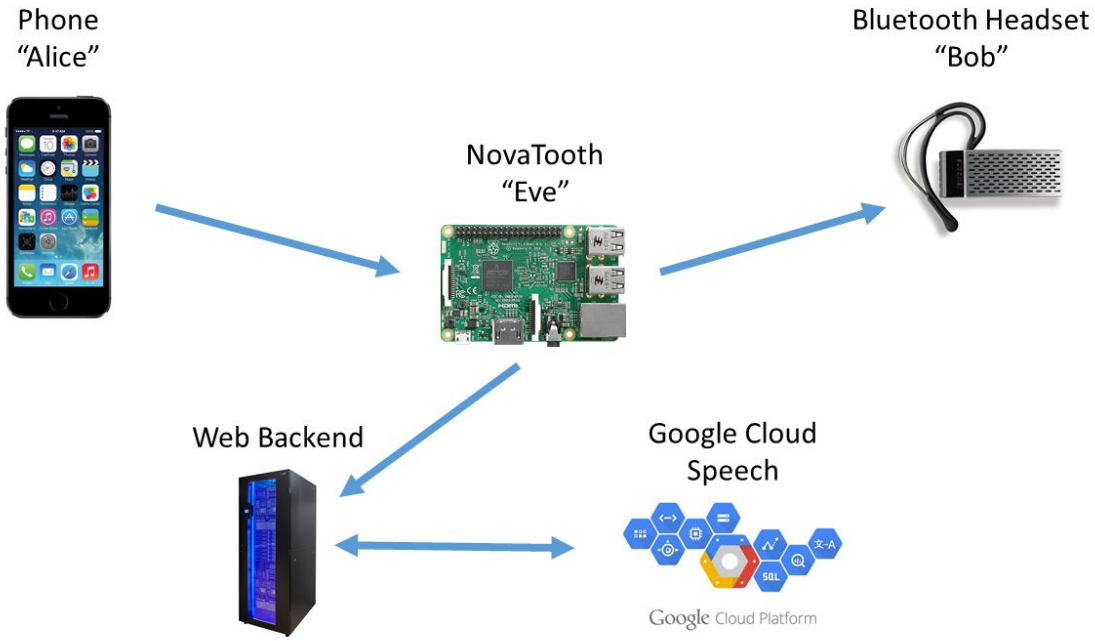
Glossary of Terms:

- Raspberry Pi 3: single board ARM computer. Future device for our software to run on. Small, inexpensive, and low power consumption.
- Intel NUC: single-board x86-64 computer. We are currently using x86-64 compatible laptops for testing.
- Kali Linux: the operating system we have chosen as the base platform for the project. It is well-suited for our needs because it has numerous commands built-in, can be easily "hacked" by users of our product to add additional functionality, and is open-source.
- Plantronics M50: Bluetooth Headset (range: 33ft 10m): Streams audio from smartphone and makes every call worry-free with an extended battery life, whispered talk-time alerts and noise reduction features
- Bluetooth: a wireless technology for exchanging data over short distances.
- Bluetooth Profile:
    - Profiles are definitions of possible applications and specify general behaviors that Bluetooth enabled devices use to communicate with other Bluetooth devices
    - A2DP: Advanced Audio Distribution Profile

        - allows for the wireless transmission of stereo audio from an A2DP source (typically a phone or computer) to an A2DP receiver (a set of Bluetooth headphones or stereo system)
        - multimedia audio streaming
    - HSP: Headset Profile
        - defines a simple audio connection between a Bluetooth audio gateway, like a cell phone, and a headset
        - defines how the two devices can transmit monophonic audio between each other, one channel at a time.
    - HFP: Hands Free Profile
        - much more robust than the headset profile.
        - Lets your device control the phone (place calls and hang up calls)
        - Can switch between calls on your phone's call waiting systems, dial numbers, access your phone's address book and handle three-way calls.
        - Hands-free devices can even read off caller ID numbers, control your phone's echo cancellation and noise reduction functions, and use voice control.
- Bluetooth Dongle
    - an external Bluetooth radio that connects to a Linux-powered computer (the “Eve” device) using USB.
    - ensures that the same model of Bluetooth radio can be used on a wide variety of hardware, and that commodity hardware can support multiple Bluetooth radios.
- WiFi Dongle:
    - a portable device that can be plugged into a computer's Ethernet port, providing mobile access to a finite amount of Internet data.
    - can give users 3G or 4G connectivity speed, and they are offered by numerous companies and service providers at varying prices and service plans
- PulseAudio:
    - a sound system for POSIX OSes, meaning that it is a proxy for sound applications
    - allows you to do advanced operations on your sound data as it passes between your application and hardware
    - things like transferring audio to a different machine, changing the sample format or channel count and mixing several sounds into one are easily achieved using a sound server
    - Pavucontrol
        - PulseAudio Volume Control
    - designed for Linux Systems
- oFono:

    - a mobile telephony (GSM/UMTS) application development framework that includes consistent, minimal, and easy to use complete APIs.
    - Includes high-level D-Bus API for use by telephony applications
    - Includes a low-level plug-in API for integrating with open source as well as third party telephony stacks, cellular modems, and storage back end.
    - Contains over 150,000 lines of C code
- Bluez:
    - official Linux Bluetooth protocol stack
    - provides support for core Bluetooth layers and protocols

System Architecture Overview:



The man-in-the-middle attack system involves 3 components, nicknamed Alice, Bob, and Eve. A target person will use their phone (Alice) and try to pair to a bluetooth headset (Bob). Our NovaTooth device (Eve) will disguise itself to look like Bob so that Alice pairs to it instead. Once the attack is established, Eve will upload intercepted audio to the Web Backend which will use Google Cloud Speech APIs to transcribe it to text.

User Interaction:

For a user of the NovaTooth device, there is minimal required action. From a black box perspective, the user will only have to power up the device and place it at a location of their

choice. If the device chosen to implement the NovaTooth device is a Raspberry Pi, then it will be powered with an external battery. Alternatively, if the device used is a Intel NUC or other x86 compatible single board computer, it will have its own power source. Once powered up, the device will connect to devices and then intercept audio, which is then translated to text that is stored in the cloud. What the user does with this audio data is up to their discretion, but the NovaTooth team aims to implement an organized web database which will log all audio-to-text files of specific targets.

Device Specifications:

NovaTooth

- For the first stage of development, the NovaTooth will be a regular laptop computer with bluetooth capabilities
- Operating System: Kali Linux
- Processor: Intel x64 or compatible
- Needs to be able to process and forward audio
- Must be connected to internet

Web Backend

- Python 3.4 using Django 1.10
- Hosted with AWS ElasticBeanstalk
- SQLite3 Database
- Secure connection with SSL certificate

Device Interaction:

Alice to Eve (Rob):

- Python/Ruby thread which broadcasts loudly and quickly pretending to be Bob
- Accepts connections, handshakes etc
- Accepts packets from Alice and forwards to Eve
- Accepts packets forwarded to Eve from Bob and passes them to Alice

Eve to Bob (Henry):

- Handshake/pairing
- Connects to Bob and forwards packets from Eve
- Receives packets from Bob and forward back to Alice

Speech-to-Text Transcription (Trevor):

- Server has an API for the device to upload a file with an HTTP POST request

- Server saves all uploaded audio files and sends them to Google Cloud API for speech-to-text transcription
- Server stores all of the text transcriptions in a database

Integration (Kevin):

- Coordination, make sure everyone coordinates together to deliver data
- Autostart scripts
- Unit testing
- Set up threading between these layers
- Must be on top of getting people to thread

Data Handling (Albert):

- Transmission of payload to google transcription layer
- Take google transcription results and store them into text
- Store bluetooth pair and the results
- Encode to mp3
- Take bytestream from bluetooth(or higher level bluetooth API to get audio stream)

Requirements:

Use Cases:

1. Detect and identify nearby bluetooth devices
    a. https://github.com/rstosick/cs189a-repo1/commit/719626249cfe8d81010c4cbf23fb827bc17b409f
    b. https://github.com/rstosick/cs189a-repo1/blob/master/rob-folder-testcode/rob_script.py
2. Retrieve some basic information about the devices
    a. Manufacturer, MAC address, and RSSI (received signal strength indicator)
    b. https://github.com/rstosick/cs189a-repo1/blob/master/get_rssi.py
    c. https://github.com/rstosick/cs189a-repo1/commit/37f3906b5aff59dbaa8aeb0dd6c018773382f21f
3. Broadcast a bluetooth connection point that other devices can connect to
    a. https://github.com/rstosick/cs189a-repo1/commit/de88a2e79970b81fb9ae8edd9e9d2d30a577f7d1
4. Connect to desired bluetooth devices, using provided MAC addresses
    a. https://github.com/rstosick/cs189a-repo1/commit/7beaf1e45ce73171f7f3001a133e51716f3bd9f4
5. Detect and report to the user failed pairings and connections
    a. https://github.com/rstosick/cs189a-repo1/commit/4fd3690d0d9f8c9dcd81a540402a5d11bf66c36a

6. Open up a channel for streaming audio from both connected phone and bluetooth device to Novatooth
    a. https://github.com/rstosick/cs189a-repo1/blob/master/rob-folder-testcode/11-11-2016_terminal_output_2 (manually tested)
    b. https://github.com/rstosick/cs189a-repo1/commit/4fd3690d0d9f8c9dcd81a540402a5d11bf66c36a
7. Use a mobile telephony application that allows Novatooth to capture phone calls from the phone and bluetooth headset
    a. https://github.com/rstosick/cs189a-repo1/blob/master/rob-folder-testcode/11-11-2016_terminal_output_6_CALL_AUDIO_WORKING (manually tested)
    b. https://github.com/rstosick/cs189a-repo1/commit/b350eae7cdea0e1906353bdf4117c9d7d3b9fed7
8. Redirect the bluetooth headset's output voice audio and phone's incoming voice audio to Novatooth
    a. https://github.com/rstosick/cs189a-repo1/blob/master/rob-folder-testcode/11-11-2016_terminal_output_6_CALL_AUDIO_WORKING (manually tested)
    b. https://github.com/rstosick/cs189a-repo1/commit/b350eae7cdea0e1906353bdf4117c9d7d3b9fed7
9. From Novatooth, send bluetooth headset's output audio to output from phone
10. From Novatooth, send phone's incoming voice audio to bluetooth headset to listen to
    a. https://github.com/rstosick/cs189a-repo1/blob/master/rob-folder-testcode/11-11-2016_terminal_output_6_CALL_AUDIO_WORKING (manually tested)
11. Record the input audio to an mp3
    a. https://github.com/rstosick/cs189a-repo1/blob/master/startup_script/start_recording.sh
12. Send mp3 input audio to back-end web server
    a. https://github.com/rstosick/cs189a-repo1/blob/master/SpeechToTextServer/SpeechToTextServer/app/tests.py
13. Transcribe the mp3 audio to text from back-end web server to Google Speech
    a. https://github.com/rstosick/cs189a-repo1/blob/master/SpeechToTextServer/SpeechToTextServer/app/tests.py
14. Transcriptions visible via web application
    a. https://github.com/rstosick/cs189a-repo1/blob/master/SpeechToTextServer/SpeechToTextServer/app/tests.py
15. Applications can use our API to retrieve transcription records and analyze them
    a. https://github.com/rstosick/cs189a-repo1/blob/master/SpeechToTextServer/SpeechToTextServer/app/tests.py

User Stories: (Yellow = have test for)

We will be able to set up Eve wherever there are potential bluetooth devices around, regardless of Internet access.

- We will be able use Kali Linux for the incorporation of many bluetooth attack tools such as hciconfig.
- We will be able to boot up Eve automatically and start running processes (script).
    - bool testNovaToothStart()
        - https://github.com/rstosick/cs189a-repo1/commit/f17597cf751398e5fcd30786e43bc22e2a538f21
- We will be able to start a bluetooth scanner which will retrieve the following information about surrounding devices.
    - bool testScannerStart()
        - https://github.com/rstosick/cs189a-repo1/commit/f17597cf751398e5fcd30786e43bc22e2a538f21
- We will be able to retrieve devices' information: MAC address, type of device, name of device.
    - bool testScannerRetrieve()
        - https://github.com/rstosick/cs189a-repo1/commit/f17597cf751398e5fcd30786e43bc22e2a538f21
- We will be able to establish pairings with two devices simultaneously (sender and receiver (Alice and Bob)).
    - bool testPairTwo()
        - https://github.com/rstosick/cs189a-repo1/commit/f17597cf751398e5fcd30786e43bc22e2a538f21
- We will be able to trust two devices simultaneously (sender and receiver (Alice and Bob)).
    - bool testTrustTwo()
        - https://github.com/rstosick/cs189a-repo1/commit/f17597cf751398e5fcd30786e43bc22e2a538f21
- We will able to connect to two devices simultaneously (sender and receiver (Alice and Bob)).
    - bool testConnectTwo()
        - https://github.com/rstosick/cs189a-repo1/commit/f17597cf751398e5fcd30786e43bc22e2a538f21
- We (as well as the phone) will be able to receive audio from the bluetooth headset
    - bool testReceiveAudio()
        - https://github.com/rstosick/cs189a-repo1/commit/f17597cf751398e5fcd30786e43bc22e2a538f21
- We will be able to use PulseAudio and the command parec to redirect received input audio to an output mp3 file

    - bool testMp3NotEmpty()
        - https://github.com/rstosick/cs189a-repo1/commit/f17597cf751398e5fcd30786e43bc22e2a538f21
- We will be able to securely upload the audio file to our web backend.
    - bool testFileUpload()
        - https://github.com/rstosick/cs189a-repo1/commit/f17597cf751398e5fcd30786e43bc22e2a538f21
- The web backend will send the audio to Google Cloud Speech and receive a text transcription.
    - bool testAudioTranscription()
        - https://github.com/rstosick/cs189a-repo1/commit/f17597cf751398e5fcd30786e43bc22e2a538f21
- The web backend will store the audio, transcription, and any other information about the attack in a database.
    - bool testDatabase()
        - https://github.com/rstosick/cs189a-repo1/commit/f17597cf751398e5fcd30786e43bc22e2a538f21
- The web backend will also provide a frontend interface.
    - bool testLogin()
        - https://github.com/rstosick/cs189a-repo1/commit/f17597cf751398e5fcd30786e43bc22e2a538f21
- We will be able to view the data in an organized and aesthetic manner and interface.
- We will be able to exit Novatooth at any given time.
    - bool testNovaToothExit()
        - https://github.com/rstosick/cs189a-repo1/commit/f17597cf751398e5fcd30786e43bc22e2a538f21

Prototyping Code and Test Cases:

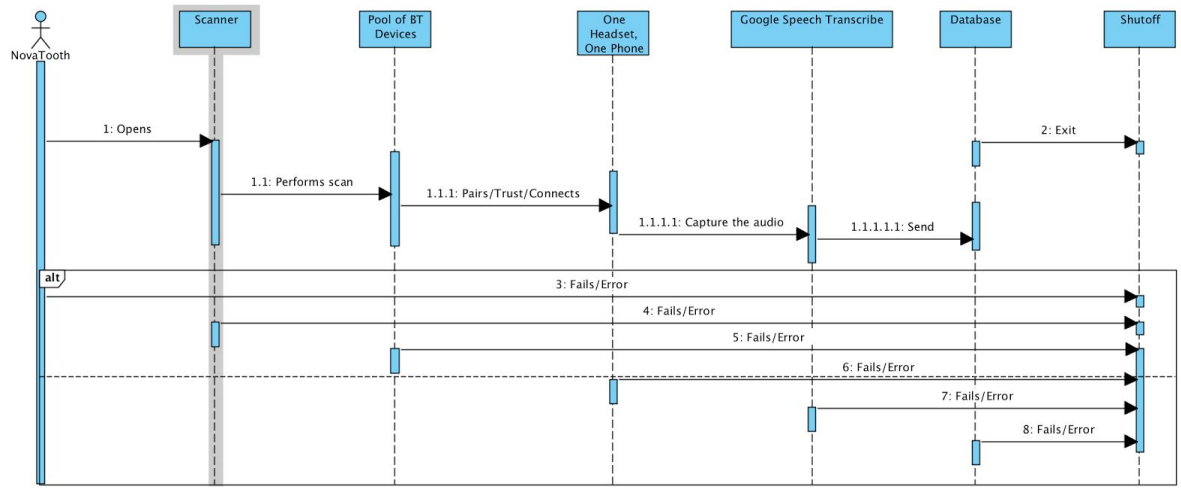
- https://github.com/rstosick/cs189a-repo1/blob/master/TestCases.xlsx

1. bool testNovaToothStart()
    a. Verifies that NovaTooth will be started successfully
2. bool testScannerStart()
    a. Verifies that scanner will scan for surrounding devices
3. bool testScannerRetrieve()
    a. Verifies that scanner will retrieve MAC address, type of device, name of device
4. bool testPairTwo()
    a. Verifies that two way pairing will be successful

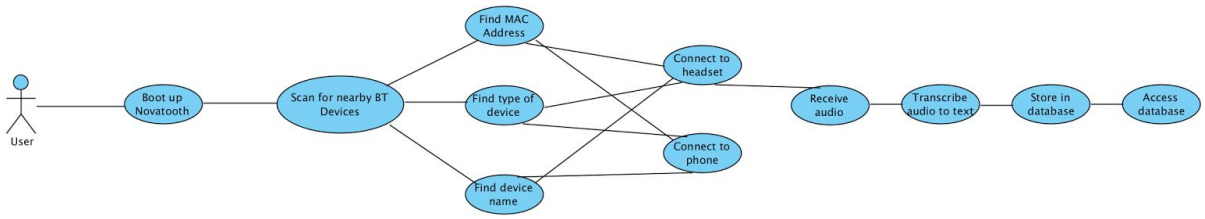
5. bool testTrustTwo()
    a. Verifies that two way trusting will be successful
6. bool testConnectTwo()
    a. Verifies that two way connecting will be successful
7. bool testReceiveAudio()
    a. Verifies that we are receiving audio from the Bluetooth Headset
8. bool testMp3NotEmpty()
    a. Verifies that the output mp3 is not empty
9. bool testFileUpload()
    a. Verifies that the audio file was uploaded successfully to the database
10. bool testAudioTranscription()
    a. Verifies that the audio file was successfully transcribed by Google Speech
11. bool testDatabase()
    a. Verifies that the database contains all required information
12. bool testLogin()
    a. Verifies that we can login to the frontend successfully
13. bool testNovaToothExit()
    a. Verifies that NovaTooth will be exited successfully

System Models:

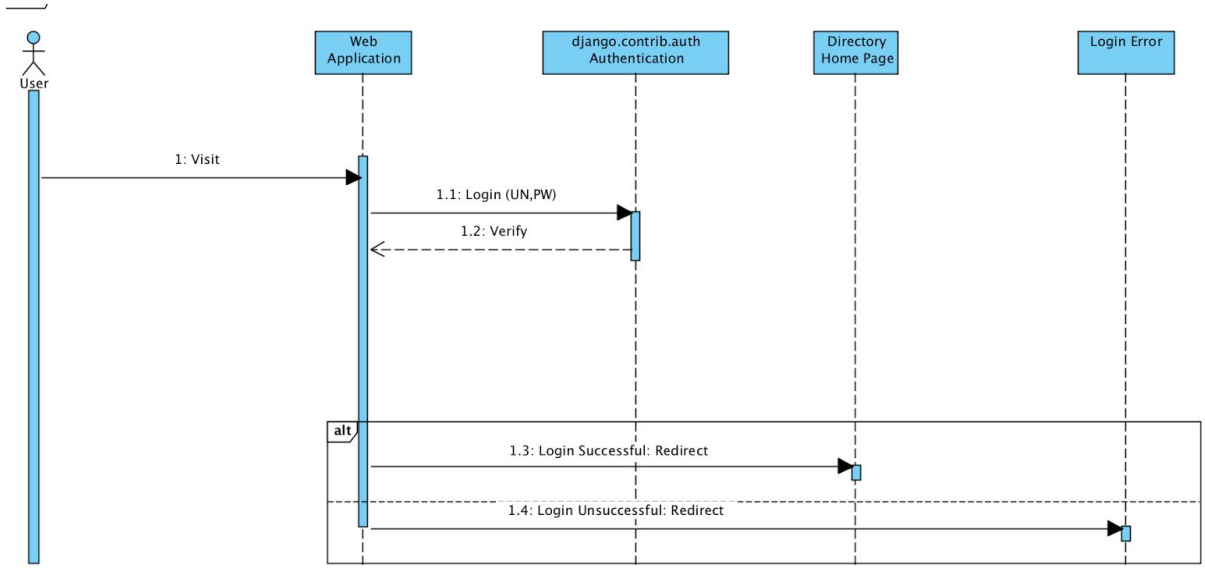
Sequence Diagram for NovaTooth:



Use Cases:



Sequence Diagram for Web Application:



Appendices:

- What we are not doing:
  - Creating this device with malicious intentions
  - Promoting Bluetooth Man-in-the-Middle attacks
- Hardware used:
  - iPhone (or equivalent mobile device)
  - Intel® NUC / Raspberry Pi 3
  - Plantronics M50 Bluetooth Headset
  - CanaKit Raspberry Pi WiFi Wireless Adapter / Dongle (802.11 n/g/b 150Mbps)
  - Panda Bluetooth 4.0 USB Nano Adapter
- Software used:
  - Kali Linux - Operating System
  - GitHub - Git Repository Hosting Service
  - Trello - Project Management Application
  - Slack - Team Collaboration Tool
  - Google Groups / Docs - Documentation Medium
- Programming Languages / Libraries / APIs used:
  - Bluetooth Protocols
  - Bluez - Official Linux Bluetooth Protocol Stack
  - PulseAudio - Sound system for Linux
  - Python 2.7.12

- Google Cloud Speech API
- PyBluez - Bluetooth Python Extension Module
- HTML/CSS
- JavaScript/JQuery
- Django
- SQLite3
- Ofono: D-Bus mobile telephony API. Used to support Hands-Free Profile implementation on Kali Linux