NovaTooth

Project Requirements Document v1

Men-in-the-Middle:
Kevin Chan (Project Lead)
Trevor Morris (Scribe)
Robert Stosick
Henry Yu
Albert Chen

Revision History:

- 10/28/2016 Initial Draft

Intro:

(Problem,Innovation, Science, Core Tech. Advance)

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.

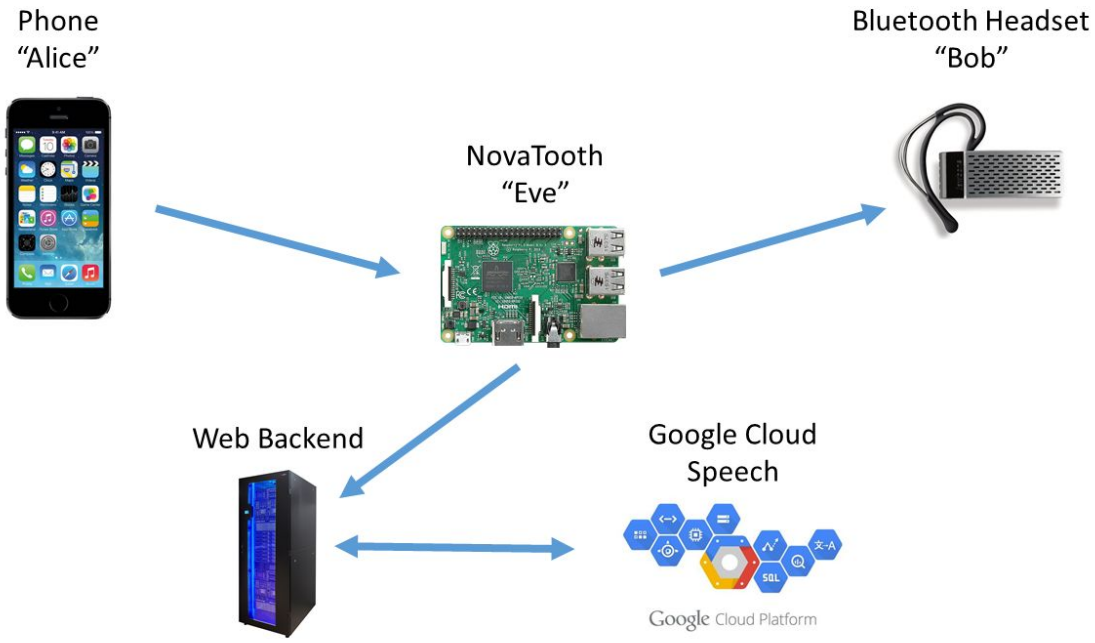
The core technological advance we seek to make is to develop the software necessary to allow a Linux-based device to impersonate a Bluetooth headset and forward packets/audio

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 computer
- UberTooth One: Open source hardware and software platform for Bluetooth experimentation. Offers features such as spectrum analysis and limited packet-sniffing capabilities. This will initially be used as a testing tool, and not as an integral component of the project.
- Jawbone One: bluetooth headset (range: 33ft 10m)
- Bluetooth Dongle: an external Bluetooth radio that connects to a Linux-powered computer (the “Eve” device) using USB. Using a dongle ensures that the same model of Bluetooth radio can be used on a wide variety of hardware.

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, while other devices will have their 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:

Alice

- A smartphone such as the iPhone 6s

Eve

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

Bob

- The Jawbone 1 bluetooth headset

Web Backend

- Python using Django
- Running on cloud hosting service
- SQLite3 Database

Device Interaction:

Alice -> 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 -> 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):
- Google Speech API
- Send audio, get back text
- Must work within the same framework
- Must be on web

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:

User stories:
- The attacker can identify nearby bluetooth devices and retrieve some basic information about the devices
- The attacker can open up a bluetooth connection point that other devices can connect to.
- Instead of the sender communicating directly to the receiver, the sender will communicate to the attacker, who will then communicate to the receiver.
- The bluetooth packets will be transcribed to audio, then transcribed to the text.
- The user can access the transcribed text in an organized database

Use Cases:
- We will use Kali Linux for the incorporation of many bluetooth attack tools such as hciconfig

- We will be able to start a bluetooth scanner which will retrieve the following information about the device: MAC address, type of device, name of device
- We can establish and maintain a connection with both the sender and the receiver by using bluetooth tools in Pybluez by opening bluetooth sockets.
- Once we receive the data packets, the data packets will be stored in a byte array. This byte array will be converted into an audio format such as raw or mp3.
- The raw or mp3 file will be streamed and uploaded to the Google Service cloud to be transcribed from audio to text.
- We will upload the data we get from the Google Service cloud to a website using DJango and other frontend information so we can view the data in an organized manner

Prototyping Code and Test Cases:

- https://github.com/rstosick/cs189a-repo1

System Models:

- N/A

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
    - Panasonic Bluetooth Jawbone One
    - CanaKit Raspberry Pi WiFi Wireless Adapter / Dongle (802.11 n/g/b 150 Mbps)
    - 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
    - Python 2.7.12
    - Google Cloud Speech API
    - PyBluez - Bluetooth Python Extension Module
    - Django
    - SQLite3