The development of the Bluetooth Low Energy (BLE) standard has heralded a new classification of wireless devices and has introduced novel challenges in identifying devices for security, tracking, and other purposes. Bluetooth Low Energy is a protocol used by low-power devices such as tire pressure monitors, smart watches, headphones, and more to transmit data and negotiate Bluetooth pairing. Devices that are open for pairing or exchanging data will be broadcasting signals that can be intercepted. This paper explores innovative methods to fingerprint BLE-enabled devices by using intercepted data. The packets are intercepted using an external BLE network interface connected via USB. Our research focus is the development of a specialized Python script tailored to discriminate, classify, and compare the BLE packets transmitted by various devices obtained through Wireshark. This script aims to streamline the analysis process of large packet captures, enabling the differentiation of devices based on their unique communication patterns and behavior. By leveraging Wireshark and the Nordic dongle in conjunction with a purpose-built Python script, the study advances the current understanding of BLE device fingerprinting methodologies.

Our group was provided with a previous National Security Agency (NSA) project in which the goal was to create a pattern of life through identifying and tracking BLE devices. A problematic aspect of this project is that BLE devices have MAC Addresses that mutate in short intervals. Utilizing a network protocol and packet analyzing tool like Wireshark, we are able to collect all of a packet's attributes and header information. Using the Python library PyShark, we are able to parse the packet capture.In order to capture and analyze only BLE devices, we implemented the Nordic RF52840 Dongle (which can receive and send BLE signals), allowing only packets from BLE devices rather than all Bluetooth specifications.

We performed various scans ranging from five minutes to multiple hours in diverse locations such as a heavily trafficked academic center at Marist College and the middle of the woods in Upstate NY. Using these data sets, prior research, and our own research, we selected several attributes of interest. The time a packet was captured can be compared to behavioral patterns to follow a device despite a changing MAC address. RSSI level, or signal strength, can also be used to our advantage. Knowing the distance from a statically located device to our capture point provides underlying consistency despite other obscurities. Company ID (CID) is also advertised by BLE devices and tells us what manufacturer the device originates from.

We experienced and overcame several key challenges during our research process. When taking data captures in high-traffic areas such as a college campus, millions of packets were captured in under an hour. Isolation became pivotal in developing an initial analysis of device behavior. When this isolation was achieved (by hiking into the woods), the cold weather, the 5-volt power usage of our dongle, and the required processing power quickly exhausted the battery of our collection laptop. When a busy capture is to be analyzed, it requires significant resources to process the data. Since Python is a slower language that handles garbage collection on its own (frees up memory from data we are no longer using), we ran into limiting factors such as the amount of RAM on our processing machine. To overcome this we revised our code, changed how we interacted with the data, and implemented storage using Apache Avro. In simple terms, Avro can store our data in a uniform format while using less space. The Avro data can be loaded in sections, rather than the entire set. Using Avro and specifying explicit data types reduced our storage size and memory requirements by nearly three-fourths. Once data has been processed, we store it using Avro so that metrics can easily be regenerated from the same data.

Our findings are that BLE devices can be fingerprinted and tracked effectively using packet attributes such as time, RSSI, Power Level, and CID. Our Python script is able to visualize a change in MAC address on a graph as well as identify unique devices in an environment/detect when a device changes its address. Data can be collected on these endpoints and a device profile can be created that includes past aliases (MAC addresses) that it has used.

Our goal is to ensure we are identifying devices with high precision through a packet capture and create a pattern of life from that data for each device within scanning range. Our next step is to develop an intrusion detection system (IDS) that will be able to monitor BLE devices and warn us of attacks, anomalies, or unauthorized devices. Implications of our research include privacy concerns and endpoint security as devices can be identified. If an individual uses an insulin pump that communicates to a controller using BLE for example, this person could be tracked if additional security recommendations are not made. If an IDS can be developed, it will be able to protect organizations against unauthorized tracking devices similar to Apple Airtags and abnormal device behavior due to cyber attacks including bluesnarfing, bluejacking, and BrakTooth exploits.

Keywords: Bluetooth Low Energy, Device Fingerprinting, Mac Randomization, IoT, IDS.