

Detecting BrakTooth Attacks

Achyuth Nandikotkur¹^a, Issa Traore¹^b and Mohammad Mamun²^c

¹*ECE Department, University of Victoria, British Columbia, Canada*

²*National Research Council Canada, New Brunswick, Canada*

Keywords: Bluetooth Security, Intrusion Detection, Machine Learning, BrakTooth.


Abstract: More than 5.1 billion Bluetooth-enabled devices were shipped in the year 2022 and this trend is expected to exceed 7.1 billion by the year 2026. A large proportion of these devices are used in smart homes designed for older adults, to help them age in place. Monitoring vitals, climate control, illumination control, fall detection, incontinence detection, pill dispensing, and several other functions are successfully addressed by many of these Bluetooth-enabled devices. Therefore it becomes crucial to protect them from malicious attacks and ensure the safety and well-being of their users. Some of these devices have only Bluetooth connectivity which makes patching them challenging for older adults, as a result, most remain unpatched. The family of vulnerabilities recently found in the Bluetooth Classic (BT Classic) stack called BrakTooth, poses a genuine threat to such devices. In this study, we develop an experimental procedure to capture traffic at the Link Manager Protocol (LMP) layer of the BT Classic stack and use machine learning algorithms to detect BrakTooth-based attacks.


1 INTRODUCTION


According to the Bluetooth Special Interest Group (SIG, 2023), it is projected that over 7.1 billion Bluetooth-enabled devices will be shipped in 2026. This figure encompasses all Bluetooth technology devices, including BT Classic and newer versions such as Bluetooth Low Energy (BLE). Although the number of devices shipped with just BT Classic is expected to decline, 100% of the future Bluetooth-enabled devices are expected to support dual-mode (BT Classic + BLE). A substantial number of these devices are used by aging adults, such as smartwatches, hearing aids, fall detection devices, blood pressure monitoring devices, weighing scales, pillboxes, diapers, and more, to assist them in everyday activities (Wagner et al., 2012). Therefore, it becomes crucial to protect these devices against attacks that intend to compromise them.

In recent years, Garbelini et al. (2021) has disclosed a family of new security vulnerabilities in commercial Bluetooth stacks that can compromise the availability of BT Classic devices, called BrakTooth. The BrakTooth vulnerabilities exist especially in the link manager and baseband layers of the Bluetooth

stack. Several vendors producing BT Classic system-on-chips (SoCs) were notified of the disclosures and they have already started patching their implementations of the stack, or have already patched them. Usually, vendors provide security patches to SoCs through wireless over-the-air (OTA) firmware updates, or wired updates (i.e. via USB) (El Jaouhari and Bouvet, 2022), or replace the devices with patched SoCs; the last one requires recall and is rarely done. Typically, updating the firmware of Bluetooth-only devices over the air requires two separate devices: a Device Firmware Update (DFU) target and a DFU controller. The DFU controller, which is usually a mobile device running a vendor-specific application, is responsible for transferring the firmware image to the DFU target device where the update needs to be made. Nordic Semiconductors employs this technique to update their Bluetooth-based devices over BLE (NordicSemiconductor, 2018). Several devices used in aging in place have only Bluetooth connectivity, therefore, older adults without sufficient technical expertise find it challenging to update them. As a result, several such devices remain vulnerable and prone to compromise. In this study, we propose a novel method to detect BrakTooth attacks using machine learning and inexpensive hardware.

^a <https://orcid.org/0000-0001-7098-8374>

^b <https://orcid.org/0000-0003-2987-8047>

^c <https://orcid.org/0000-0002-4045-8687>

1.1 Background

Bluetooth is a wireless communication standard that can be used to exchange data between stationary and fixed devices within a range of up to 100 meters (Ferro and Potorti, 2005). Although Bluetooth occupies the ISM band of 2.4 GHz which is 83MHz wide, it does not use the Direct Sequence Spread Spectrum (DSSS) used by WiFi. Instead, it uses Frequency Hopping Spread Spectrum (FHSS) to hop between 79 different 1 MHz-wide channels in this band. Due to its use of FHSS, interference with other devices is reduced. However, WiFi uses a single channel that is 22MHz wide, and when both WiFi and Bluetooth networks are in the same range, the 22MHz channel of WiFi occupies 22 of the 79 Bluetooth channels leading to some interference. When a Bluetooth device experiences interference, it addresses this problem by hopping to the next channel and retrying. In contrast, WiFi issues an Automatic Repeat Request and slows down the data rate in an attempt to reduce the Bit Error rate. However, the number of channels and the channel bandwidth differ based on the type of Bluetooth device used. There exist two types of Bluetooth devices: Bluetooth Classic (BT Classic) and Bluetooth Low Energy (BLE). A few salient differences between these two devices are listed in Table 1.

1.2 Sniffing Bluetooth Traffic

Several devices and tools are readily available to monitor wireless WiFi traffic. However, monitoring Bluetooth traffic reliably is limited to the realm of professional equipment like the Ellisys Vanguard or the Frontline Sodera and they can be very expensive (Cominelli et al., 2020). In general, there exist two types of sniffing:

- Passive sniffing: It involves a device that sniffs over-the-air Bluetooth packets between two communicating devices in radio range.
- Active sniffing: It involves a device that connects to a remote BT device and captures the packets exchanged down to the lowest layer in the Bluetooth protocol stack.

Due to the fact that Bluetooth uses frequency-hopping spread spectrum technology with a vendor-specific hopping pattern, it is difficult to monitor its connections. Despite these challenges, a few inexpensive passive sniffers such as Ubetooth One (Os-smann, 2011) and nRF Sniffer (nRFSniffer, 2023) were developed, however, they can only capture packets of BLE reliably, and not BT Classic (Nijholt et al., 2020). To the best of our knowledge, there is still no

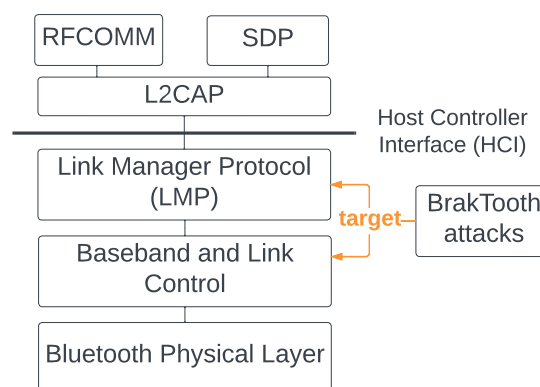


Figure 1: BrakTooth attacks targeting the LMP and baseband layers of the BT Classic stack.

non-commercial solution for monitoring BT Classic traffic. One approach to capturing the packets pertaining to BrakTooth attacks would be to set up an Android device as the active sniffing target and enable a debug functionality in its Bluetooth stack known as the HCI Snoop Log feature (Ditton et al., 2020). This feature captures all BT traffic above the HCI layer of the Bluetooth stack. However, as shown in Figure 1, since the BrakTooth attacks are aimed to exploit the vulnerabilities in the link manager protocol (LMP) and baseband layers of the Bluetooth stack, we need a method to monitor the lower-level traffic. Thus, we adopted a well-known framework called Internal-Blue (Mantz et al., 2019), which is capable of monitoring communications below the HCI layer. The researchers who developed this framework reverse-engineered widely used Broadcom BCM4339 Bluetooth Controller firmware and developed patches that give access to the LMP layer. Note that the Nexus 5 is one of the popular mobile devices that contain the BCM4339 controller (Schulz et al., 2015). In this study, we utilized the InternalBlue framework to enable LMP monitoring on a Nexus 5 device and use it as an active sniffer to capture LMP packets from both benign and malicious devices.

It should be noted that this approach to capturing LMP packets was not explored in prior works. As a consequence, detecting BrakTooth-based attacks using machine learning and inexpensive hardware was not feasible until now. Therefore, our study presents only our findings without a comparative analysis.

The remainder of the paper is organized as follows. Section 2 describes the related works. Section 3 presents the experimental setup and dataset collection. Section 4 presents results from applying various machine learning algorithms to the dataset. Finally, Section 5 presents our conclusions and future work.

Table 1: Salient differences between BT Classic and BLE.

| Feature | Bluetooth Classic | Bluetooth LE |
|----------------------|------------------------------------|--|
| Application | Audio streaming and data transfer. | Audio streaming, data transfer, location services and device networks. |
| Frequency band | 2.402 - 2.480 GHz (ISM) | 2.402 - 2.480 GHz (ISM) |
| Channels | 79 | 40 channels |
| Channel bandwidth | 1 MHz | 2 MHz |
| Spread spectrum | FHSS | FHSS |
| Power Consumption | 1W | ~0.001 W-0.5 W |
| Data rate | 1 Mb/s, 2 Mb/s, 3Mb/s | 125 Kb/s, 500Kb/s, 2 Mb/s |
| Device discovery | Inquiry or paging | Advertising |
| Encryption algorithm | E0/SAFER+ | AES-CCM |
| Network topology | Point-to-point | Point-to-point, Broadcast and Mesh |

2 RELATED WORKS

The following section discusses some works that use both commercial and non-commercial devices to detect attacks utilizing various methods such as thresholds in Bluetooth specifications, signatures, statistical distributions of data, and anomaly detection.

2.1 Detecting Bluetooth Attacks

Wu et al. (2020) used the Ubertooth One device to capture over-the-air packets in the BLE advertising channel and extracted features such as the advertising pattern, state transitions of the BLE device, advertising interval, the RF (Radio frequency) signal frequency offset, and the RF signal strength. They detect spoofing attacks by creating a statistical distribution of CFO (Carrier frequency offset) and RSSI (Received signal strength indication) values and use them to identify any value that falls out of a predetermined threshold. They first look for a connection request packet to identify the detection of a new device and then use the above statistical distributions to detect a spoofing condition. One drawback of using the Ubertooth One device for BT classic monitoring is that it is unable to capture all packets reliably (Nijholt et al., 2020; Antonioli et al., 2019).

OConnor and Reeves (2008) presented a Bluetooth-based network intrusion detection system that uses the Merlin LeCroy protocol analyzer to non-intrusively capture the Bluetooth traffic and detect malicious behavior using pattern matching. It decodes Bluetooth packets by synchronizing with the master device on a piconet and following the hopping sequence of that piconet. However, since the detection is signature-based, novel intrusions cannot be detected. The authors have used a commercial protocol analyzer and hence were able to observe and analyze up to the LMP layer of the Bluetooth stack.

Huang et al. (2018) used loopholes in the Bluetooth specification pertaining to low-power mode transitions, to perform Bluetooth DOS attacks. Particularly, they exploited loopholes in the sniff and park mode conversions to trigger DOS conditions, using slave devices in a piconet. They also suggested that DOS conditions can be triggered intentionally or unintentionally when multiple piconets pertaining to different devices come closer to each other. Because devices operating in different piconets use their own frequencies for communication and are unaware of each other's hopping sequences, they may end up causing interference which can lead to a DOS condition. To detect DOS conditions caused by interference, they compare quality characteristic data (i.e. bit error rate and invalid data rate) of all the channels with the normal thresholds in the Bluetooth specification. To detect DOS attacks by slave devices, they compared the proportion of data transmission time to the slave timeslots, under normal conditions and during the activity under observation.

Satam et al. (2018) proposed an intrusion detection system that runs on a Linux server machine and captures the Bluetooth data frames like SCO (Synchronous Connection Oriented) data frame, the HCI protocol frame, and the HCI (Host controller interface) data frame. Although the paper does not mention the hardware used to sniff the Bluetooth traffic, it appears that the authors used the machine's stock Bluetooth adapter because captured frames represent packets above the HCI layer. Once captured, the frames are converted into flows of size T seconds, which are then converted to n-grams. These n-grams are used to train various machine learning algorithms to establish normal behavior and thus classify a new observation flow as normal or abnormal. They performed Bluesnarfing and power-draining attacks and achieved precision and recall of 99.6%.

Although the aforementioned works employ various novel techniques to detect different types of Blue-

tooth attacks, to the best of our knowledge, there is currently no existing research that focuses on the detection of BrakTooth-based Bluetooth attacks using affordable hardware and machine learning techniques. Our work aims to fill this gap.

3 EXPERIMENTAL SETUP

The LMP protocol is used by the link managers in different Bluetooth devices to establish links and exchange information about supported features, power-saving options, and encryption techniques. Most existing works that use inexpensive, off-the-shelf Bluetooth adapters could only observe packets above the Host Controller Interface (HCI) layer, as software running on the lower layers is not accessible to the public (Nijholt et al., 2020). To address this, we developed the experimental procedure seen in Figures 2 and 3 to capture the LMP traffic of normal data pertaining to benign Bluetooth communication and attack data pertaining to the exploitation of BrakTooth-based vulnerabilities. For this purpose, we used two devices:

1. Attack device: This device was used to craft and send BrakTooth attack packets to the victim device.
2. Victim device: This device received packets from the attacking device as well as normal Bluetooth packets from communication with other benign devices.

More details about these two types of devices are provided in the following subsections.

3.1 Attack Device

To exploit BrakTooth-based vulnerabilities, a proof-of-concept (PoC) tool (BrakToothPoC, 2021) was released. As shown in Figure 2, this tool is installed on a Ubuntu 18.04 host machine and is used to flash custom firmware on an ESP32-WROVER-KIT to execute various exploits.

3.2 Victim Device

A rooted Nexus 5 mobile device containing the Broadcom BCM4339 Bluetooth controller is used as a victim while a Ubuntu 22.04 host machine is used to run the InternalBlue framework. The host machine connects to the victim device via Android Debug Bridge (ADB). When the victim’s Bluetooth stack is compiled with debugging enabled, two new ports are opened by the android device as follows:

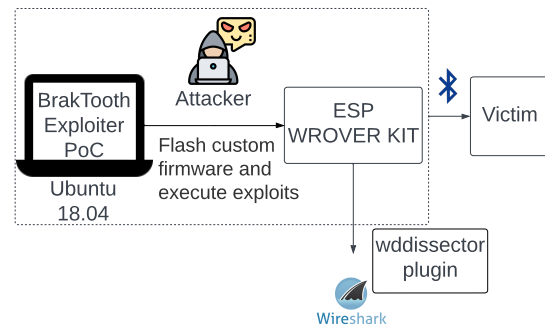


Figure 2: Setup of the attack device.

1. TCP 8872: This allows for an external agent to read the HCI packets exchanged between the BCM4339 controller and the Bluetooth stack.
2. TCP 8873: This can be used to send HCI commands to the BCM4339 controller through the Bluetooth stack.

As mentioned in Section 1.2, the InternalBlue framework helps in monitoring the LMP packets received by the victim device and it does so by modifying the BCM4339 firmware such that it relays the LMP packets it received to the host machine over TCP 8872. The framework also uses the Bluetooth H4 Broadcom Wireshark plugin to parse LMP and baseband packets from the incoming traffic and write to Wireshark. A setup of the victim device can be seen in Figure 3.

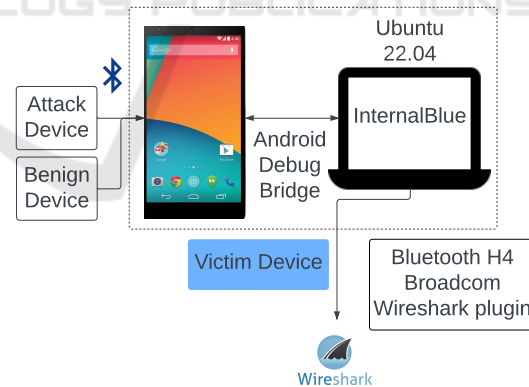


Figure 3: Setup of the victim device.

3.3 Dataset Collection

Both normal and attack data were extracted from the Wireshark capture files, generated by the InternalBlue framework running over the Nexus 5 device. To generate normal data, three mobile devices and a Windows laptop were paired with the Nexus 5 device, and they were allowed to exchange several sample documents, video, and audio files over the BT Classic pro-

tool. To generate attack data, we used the PoC tool running on an Ubuntu 18.04 host machine to execute various exploits listed by Garbelini et al. (2021), using the ESP-32-WROVER-KIT. The Nexus 5 device was kept as the target. Following is a sample command to execute exploits using the PoC tool.

```
sudo bin/bt_exploiter --host-port=/dev/
ttyUSB1 --target=<target bdaddress>
--exploit =<exploit name>
```

The captured Bluetooth traffic has the following features:

1. Protocol: This field refers to the protocol used in the Bluetooth packet such as L2CAP, OBEX, Service Discovery Protocol (SDP), RFCOMM, and others.
2. Info: This field provides additional information about each packet depending on the type of protocol used.
3. Length: This field indicates the length of the packet in bytes.
4. Delta: This field indicates the time difference between the current packet and the previous packet in the pcap file.
5. Type: This field is a target variable representing normal vs attack conditions and is labeled manually.

The `LabelEncoder` was imported from the `scikit-learn` library to encode the `Protocol` and `Length` features and a standard scaler was used to standardize all the features. The distribution of the attack and normal data is shown in Table 2. As the number of features is limited, in this paper, we focus on classifying only two categories, i.e, normal and attack. All entries pertaining to the vulnerabilities listed in Table 2 are mapped to just one class i.e, Attack. Both the attack and normal data were manually labeled, and the dataset has been made publicly available¹.

4 RESULTS AND DISCUSSION

For the experimental evaluation, we split the dataset into an 80-20 ratio to train three machine learning models: Random Forest, K-Nearest Neighbour, and an Artificial Neural Network. Gholamy et al. (2018) suggested that the best results are obtained if 20-30% of the data is used for testing and the remaining 80% for training. As for evaluation criteria, we chose the true positive rate (TPR), false positive rate (FPR), and

¹ISOT Laboratory, *BrakTooth Attack Dataset*, 2023, <https://onlineacademiccommunity.uvic.ca/isot/datasets/>

Table 2: Distribution of normal and attack data.

| Type of vulnerability | | Packets |
|------------------------|---------------------------------|---------|
| normal | | 6269 |
| attack | au_rand_flooding | 655 |
| | truncated_sco_link_request | 340 |
| | duplicated_iocap | 299 |
| | truncated_lmp_accepted | 274 |
| | invalid_feature_page_execution | 250 |
| | feature_response_flooding | 216 |
| | invalid_timing_accuracy | 211 |
| | lmp_overflow_dml | 159 |
| | lmp_auto_rate_overflow | 151 |
| | duplicated_encapsulated_payload | 111 |
| invalid_setup_complete | 67 | |

Table 3: Results from applying various classifiers.

| Classifier | TPR | FPR | Precision |
|---------------|--------|-------|-----------|
| Random Forest | 98.33% | 1.47% | 99.36% |
| ANN | 97.05% | 7.51% | 96.74% |
| k-NN | 97.77% | 1.28% | 99.43% |

precision as they collectively provide a comprehensive understanding of the model’s performance in attack detection. TPR captures the model’s ability to detect actual attacks, FPR assesses the rate of false alarms, and precision ensures the reliability of positive predictions. We obtained encouraging results, as shown in Table 3, with the Random Forest model performing the best, achieving a precision of 99.36% and a true positive rate (TPR) of 98.33%. As mentioned in Section 1.2, there are no prior works that deal with the problem of detecting BrakTooth-based attacks. Therefore, we will not provide a comparative analysis.

Furthermore, due to the lack of non-commercial Bluetooth sniffers, in this paper, we propose an experimental procedure that utilizes an active sniffer to capture attack traffic. However, it is worth noting that in a real-world application, this approach may not be entirely feasible as the attacker would need to specifically target the active sniffer for it to effectively detect any attacks. One strategy to enhance the likelihood of an attack is to periodically change the name of the active sniffer (i.e, victim) to match the names of common devices detected in the Bluetooth piconet being targeted.

5 CONCLUSION AND FUTURE WORK

In this research, we used a two-prong approach to detect BrakTooth-based attacks. First, we set up an attack device using ESP32-WROVER-KIT running a proof of concept (PoC) tool. Second, we set up

an active sniffer to capture normal and attack traffic, using the InternalBlue framework and a Broadcom BCM4339 Bluetooth Controller. Finally, we used the collected data to train various machine-learning models to classify attack data and achieved good performance with the Random Forest model. Our inexpensive and simple detection setup can also be extended to notify various stakeholders, such as caretakers and family members as soon as an attack is detected, thus ensuring the safety and welfare of the senior residents in a smart home. In the future, we intend to enhance our current low-cost setup to capture additional features such as RF signal strength, RF signal frequency offset, bit error rate, invalid data rate, and more to model and identify a broader range of Bluetooth attacks. Further, we also plan to integrate Ubertooth One, to extend the existing setup to detect SweynTooth-based (Garbelini et al., 2020) attacks on BLE devices.

ACKNOWLEDGEMENTS

This project was supported in part by collaborative research funding from the National Research Council of Canada's Aging in Place Program.

REFERENCES

- Antonioli, D., Tippenhauer, N. O., and Rasmussen, K. B. (2019). The knob is broken: Exploiting low entropy in the encryption key negotiation of bluetooth br/edr. In *USENIX Security Symposium*, pages 1047–1061.
- BrakToothPoC (2021). Braktooth proof of concept. https://github.com/Matheus-Garbelini/braktooth_esp32_bluetooth_classic_attacks.
- Cominelli, M., Gringoli, F., Patras, P., Lind, M., and Noubir, G. (2020). Even black cats cannot stay hidden in the dark: Full-band de-anonymization of bluetooth classic devices. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 534–548.
- Ditton, S., Tekeoglu, A., Bekiroglu, K., and Srinivasan, S. (2020). A proof of concept denial of service attack against bluetooth iot devices. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–6. IEEE.
- El Jaouhari, S. and Bouvet, E. (2022). Secure firmware over-the-air updates for iot: Survey, challenges, and discussions. *Internet of Things*, 18:100508.
- Ferro, E. and Potorti, F. (2005). Bluetooth and wi-fi wireless protocols: a survey and a comparison. *IEEE Wireless Communications*, 12(1):12–26.
- Garbelini, M. E., Chattopadhyay, S., Bedi, V., Sun, S., and Kurniawan, E. (2021). Braktooth: Causing havoc on bluetooth link manager.
- Garbelini, M. E., Wang, C., Chattopadhyay, S., Sun, S., and Kurniawan, E. (2020). Sweyntooth: Unleashing mayhem over bluetooth low energy. In *Proceedings of the 2020 USENIX Conference on Usenix Annual Technical Conference*, pages 911–925.
- Gholamy, A., Kreinovich, V., and Kosheleva, O. (2018). Why 70/30 or 80/20 relation between training and testing sets: A pedagogical explanation.
- Huang, Y., Hong, P., and Yu, B. (2018). Design of bluetooth dos attacks detection and defense mechanism. In *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, pages 1382–1387. IEEE.
- Mantz, D., Classen, J., Schulz, M., and Hollick, M. (2019). Internalblue-bluetooth binary patching and experimentation framework. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pages 79–90.
- Nijholt, T., Poll, E., and Vaandrager, F. (2020). Bluespec: Development of an lmp state machine and a stateful black-box br/edr lmp fuzzer.
- NordicSemiconductor (2018). Device firmware update process. https://infocenter.nordicsemi.com/topic/com.nordic-infocenter.sdk5.v15.0.0/lib_bootloader_dfu_process.html.
- nRFSniffer (2023). Nrf sniffer for bluetooth le. <https://www.nordicsemi.com/Products/Development-tools/nrf-sniffer-for-bluetooth-le>.
- OConnor, T. and Reeves, D. (2008). Bluetooth network-based misuse detection. In *2008 Annual Computer Security Applications Conference (ACSAC)*, pages 377–391. IEEE.
- Ossmann, M. (2011). Ubertooth one. <https://github.com/greatscottgadgets/ubertooth>.
- Satam, P., Satam, S., and Hariri, S. (2018). Bluetooth intrusion detection system (bids). In *2018 IEEE/ACIS 15th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–7.
- Schulz, M., Wegemer, D., and Hollick, M. (2015). Nexmon: A cookbook for firmware modifications on smartphones to enable monitor mode. *arXiv preprint arXiv:1601.07077*.
- SIG, B. (2023). Bluetooth market update. <https://www.bluetooth.com/2023-market-update/>.
- Wagner, F., Basran, J., and Dal Bello-Haas, V. (2012). A review of monitoring technology for use with older adults. *Journal of geriatric physical therapy*, 35(1):28–34.
- Wu, J., Nan, Y., Kumar, V., Payer, M., and Xu, D. (2020). Blueshield: Detecting spoofing attacks in bluetooth low energy networks. In *RAID*, pages 397–411.