Scientific Research Publishing

Journal of Information Security, 2024, 15, 433-447
https://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

Side-Channel Attacks & Data Exfiltration Using Wall Outlet USB Power Adapters

Andrew Masters, Vijay K. Madisetti

School of Cybersecurity and Privacy, Georgia Institute of Technology, Atlanta, United States
Email: amasters9@gatech.edu, vkm@gatech.edu

How to cite this paper: Masters, A. and Madisetti, V.K. (2024) Side-Channel Attacks & Data Exfiltration Using Wall Outlet USB Power Adapters. *Journal of Information Security*, **15**, 433-447.
https://doi.org/10.4236/jis.2024.154025

Received: July 16, 2024
Accepted: September 7, 2024
Published: September 10, 2024

Abstract

The number and creativity of side channel attacks have increased dramatically in recent years. Of particular interest are attacks leveraging power line communication to 1) gather information on power consumption from the victim and 2) exfiltrate data from compromised machines. Attack strategies of this nature on the greater power grid and building infrastructure levels have been shown to be a serious threat. This project further explores this concept of a novel attack vector by creating a new type of penetration testing tool: an USB power adapter capable of remote monitoring of device power consumption and communicating through powerline communications.

Keywords

Cybersecurity, Side Channel Attack, Power Line Communication, Penetration Testing, Hotplug Attack Tool

1. Introduction

The defensive cybersecurity industry has made significant strides in developing countermeasures and protection strategies in recent years. This technological advancement has only been matched by that of the attackers, who have demonstrated regularly that ingenuity is not itself a virtue. Of note are the creative uses of side channels to gather and exfiltrate information. While many interesting techniques exist, here we will consider the use of power line communication (PLC) for malicious gain.

Powerline communication is not a new concept. It is primarily found in the power distribution industry, where it is used to monitor and communicate with remote substations without the need for dedicated data lines [1]. It can also be found in Internet of Things (IoT) roles, where simple devices (appliances, security cameras, etc.,) can be connected to the greater network without requiring dedicated

data hardlines or wireless infrastructure [2]. PLC, like hardwired data lines, has an intrinsic security quality that wireless networks do not: exploitation requires direct access to the infrastructure. This can make powerline communication exceedingly useful in situations where confidentiality must be convincingly assured and physical security defenses have already been established [3].

Powerline communication is no different than any other form of wired communication - almost any attack strategy found in traditional communication lines translates into PLC. This can easily be seen at the power grid level: signals between substations can be intercepted for espionage purposes, grid controllers can be fooled with false data injections, and grids themselves can be destabilized using timed man-in-the-middle attacks [4] [5]. If the attacker considers the cyber-physical nature of the system, the attacks can become even more creative [6].

However, attacks at the power grid level are not the only concern. In 2018, researchers at the Ben-Gurion University of the Negev in Israel proved that powerline communication could be used to steal data from devices unconnected to any network or other machines (said to have an "air gap"). This malware, named Powerhammer, uses its control over the CPU load to adjust machine power consumption. These changes in power usage travel through the power lines; an adversary - with a non-invasive tap in the building's infrastructure—can then listen in on the power changes and transcribe that as data the malware sent [7].

In 2022, another iteration of this type of attack was made. Instead of using PLC to extract data via modulations in power consumption, the researchers instead listened to the subsonic frequencies the computer power supply made as power usage changed [8]. Because the attacker needs to be within 6 feet of the machine to "listen in", this attack was aptly named Covid-bit.

These examples are given to show that data over power transmission and power consumption have the potential to be used by adversaries for nefarious purposes.

Also of note is the *hotplug* attack tool—a common find in an attacker's arsenal. These seemingly innocent/normal devices, when unwittingly connected to a device by a victim, execute some pre-programmed attack devised by the attacker. Many such devices exist; one of the most famous of which today is likely the Rubber Ducky. This normal-appearing USB drive contains a sophisticated keystroke injection suite capable of many and varied attacks [9].

2. Proposed Approach

This paper proposes a combination of the ideas expressed above. Put simply, our desire was to create a new attack device: a standard-appearing USB-wall outlet adapter like those used for phone and laptop charging. This device appears innocuous to the victim and simply seems to charge their device. However, it also closely monitors the power usage (*i.e.*, voltage and current) passing through in real time and transmits that data wirelessly to a device held by the deployer. This data could be seen as the goal itself—usage and behavior patterns of the victim could be weaponized, or it could simply serve as the receiver for a larger PLC-based data

extraction operation. By the nature of it being closer to the victim device on the power network than any other currently existing solution in this space, such a device is not likely to be foiled by standard anti-PLC measures.

The adapter's small size, comparatively low cost, and standard appearance give it similar utility to that of other *hotplug* attack tools, and it could serve as a powerful and instructive tool in the penetration tester's arsenal. The idea of data over power line communication is often forgotten in security analysis. To be fair, it is easy to understand why this is the case. There are often more effective and efficient ways for adversaries to accomplish their goals than relying on PLC, and execution requires physical access to the victim's premises.

However, such an argument shows the potential utility of our approach. What better way to increase understanding of these increasingly creative attacks than with a simple, applicable tool penetration tester can use that opens the gateway for a more in-depth delve into the realm of side channel attacks and cyber-physical system security?

3. Implementation and Testing of Proposed Approach

The following section describes the construction and testing of our prototype, called the ThunderMole.

3.1. Implementation

To ensure ease in construction and certainty in subterfuge, ThunderMole is comprised of commercially available electronics housed in a generic casing. For the sake of stability, it uses a transformer taken from the 5W Apple USB adapter to convert the power from the wall outlet into direct current. Power is drawn from this transformer to feed the control/transmit board and the voltage/current sensor before being sent to the victim's connected device.

The control board responsible for reading and transmitting the data is the Arduino Nano ESP32 - a very small, capable board with multiple wireless transmission protocols in the Bluetooth and Wi-Fi realms at its disposal. The control board receives data from an INA219 digital power monitor. This sensor was chosen for this role as the INA219 is capable of measuring both voltage and current simultaneously from above the load, meaning that the device does not have to contend with a moving ground reference. This allows the ThunderMole to be powered by the same source it is measuring with minimal impact to the accuracy of the results.

All the components are housed in the casing of a repurposed LCGENS USB-C 20W wall charger block. A circuit diagram of ThunderMole can be seen in **Figure 1**. A picture of the completed prototype can be seen in **Figure 2**. Note the inclusion of an unmodified power adapter to showcase the similarities in outside appearance. A view of the internals is included in **Figure 3**.

The software controlling the ThunderMole prototype was created by modifying an existing ESP32 example sketch. The ThunderMole creates a Bluetooth Low Energy (BLE) server that the deployer can subscribe to in order to receive real-time

voltage and current values. The transient voltage and current are relayed as a single pair. A copy of the source code used on the ThunderMole can be found in **Appendix A**. Any device capable of subscribing to the server can read the data.

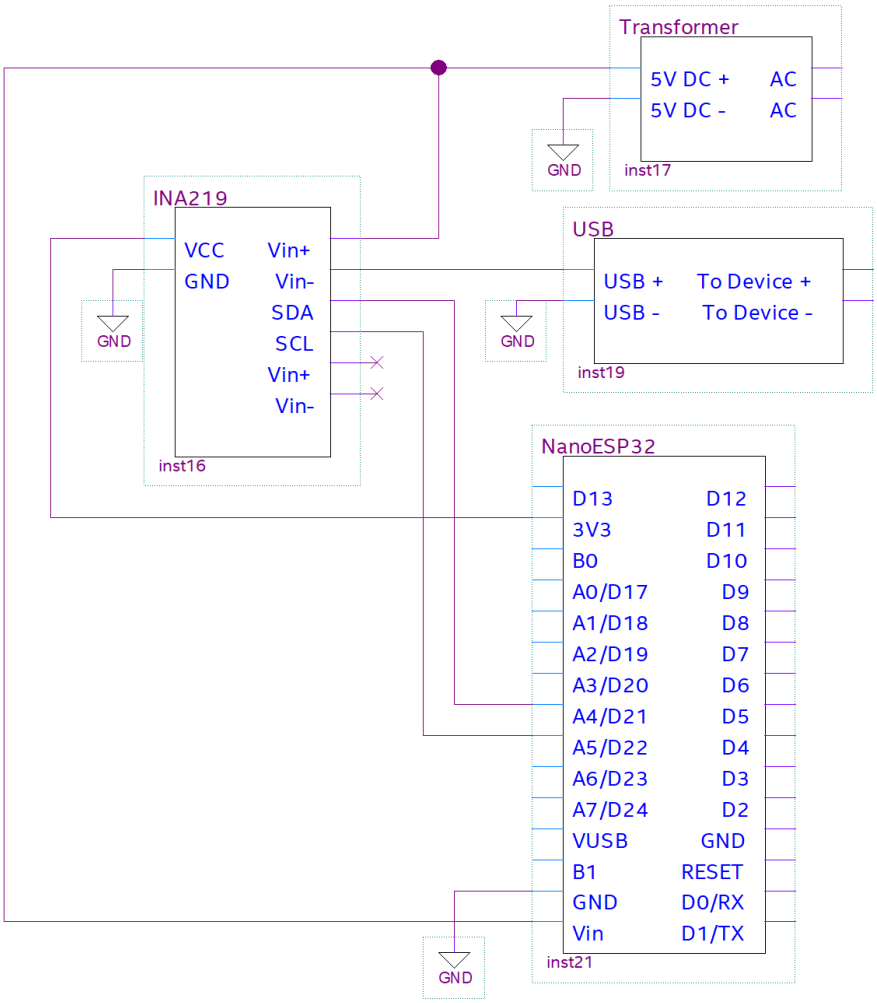


Figure 1. Circuit diagram of ThunderMole.

Figure 2. ThunderMole (Left), normal power adapter (Right).

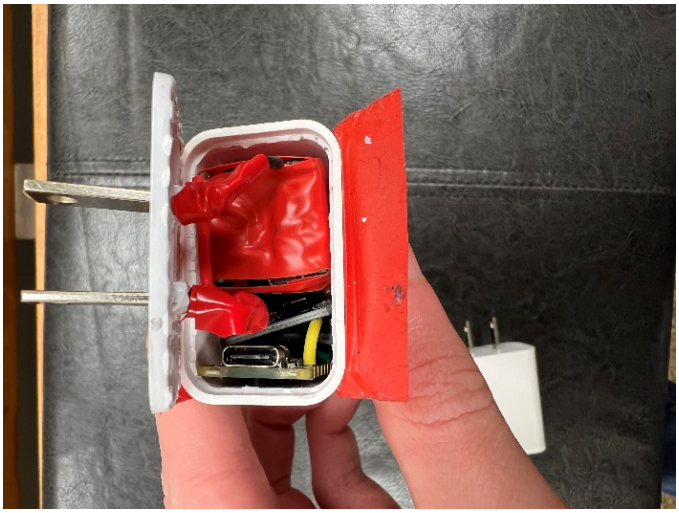


Figure 3. An internal view of ThunderMole.

3.2. Characteristic Testing

The first tests conducted on ThunderMole were designed to establish the physical characteristics of the device. This was done so that a baseline of its capabilities could be determined. Accuracy of the data received, speed of data acquisition, and range of transmission were all considered. Note that for all testing done on the ThunderMole, the receiver was an iPhone 13 Pro running the application Bluefruit LE Connect.

Data accuracy was found to be largely limited by current noise. By recording data received from the ThunderMole while no device was attached, a basic understanding of the current noise could be established. Over the course of approximately 5 minutes, the average current was found to be -0.089 mA, with a maximum deviation of -0.63 mA from the expected value of 0. Voltage noise was found to have a negligible effect on ThunderMole's performance under testing (<0.01 V on average). Multiple repetitions of these tests show similar results; a sample of which can be seen in Figure 4.

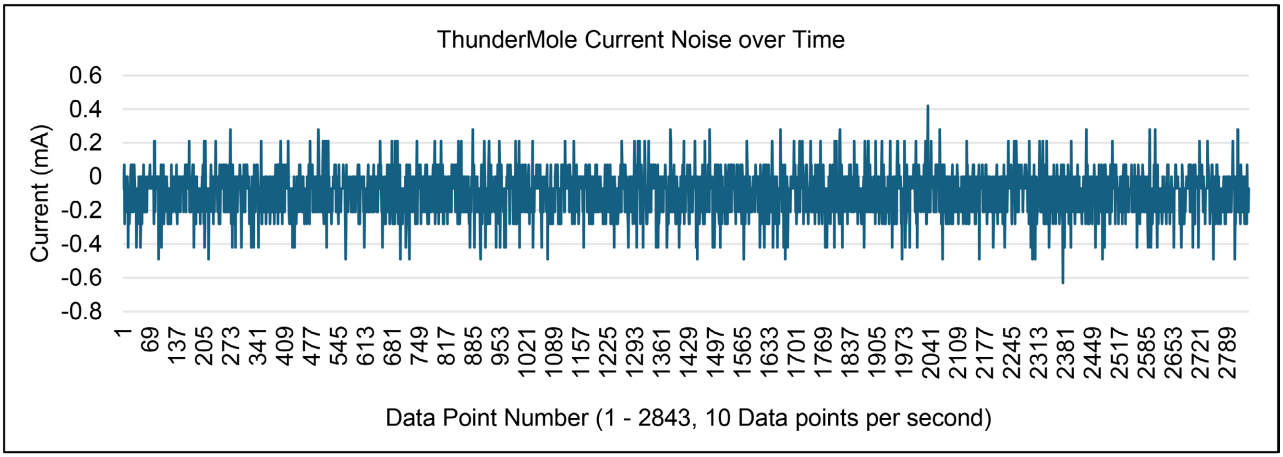


Figure 4. ThunderMole current noise over time.

The speed of data acquisition was determined to be primarily limited to the physical limitations of transmission. The BLE server created by the ThunderMole requires a certain amount of delay between transmissions to prevent collisions on the receiver end [10]. By repeatedly dropping the transmission delay until failure, an idea of the lowest required delay was established. Approximately 10 ms of delay was found to be the minimum achievable value under standard conditions (medium surrounding noise caused by proximity to other wireless devices) to transmit the data required. This results in an average read-and-transmit speed of ~100 symbols per second.

A deployer may wish to store the data on the Arduino instead of transmitting it to better avoid detection. If so, then the speed of data acquisition is improved; the limiting factor then becomes the physical capabilities of the INA219 sensor. The sensor requires roughly 5 ms of delay to conduct the data reading, suggesting a maximum read speed of ~200 symbols per second.

The range of transmission is ultimately determined by the transmission protocol and amount of interference. For the purposes of this testing, ThunderMole established a BLE server, and the receiver was moved to various positions around the device. The received signal strength was measured in dBm at each position and the distance noted. Under conditions with interference from multiple devices in a business environment (the likely deployment field of the device), ThunderMole was found to have a maximum consistent transmission range of 27 ft, with 25 ft being the comfortable limit. A graph correlating distance from the ThunderMole with signal strength can be found in Figure 5.

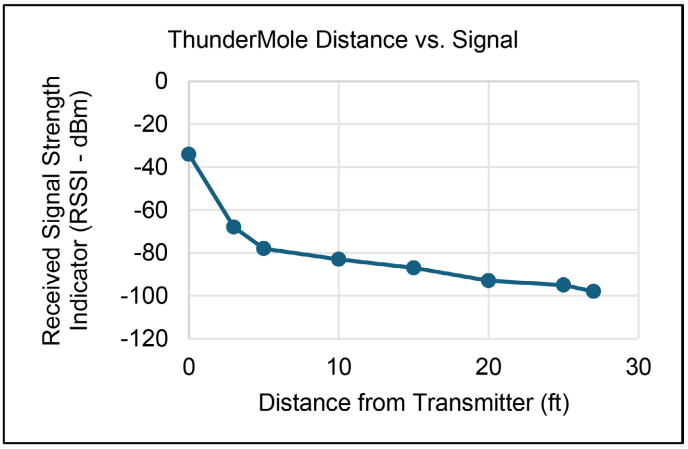


Figure 5. Range from ThunderMole vs. received signal.

3.3. Device Signaturing

With the physical attributes of ThunderMole established, we now introduce the concept of device power signatures. For many attackers, the power consumption data of the connected device is valuable in and of itself. Indeed, much can be gleaned from this information. For instance, devices often be identified by their power consumption profile alone. This is not a new concept, and ThunderMole

was found to be able to facilitate this sort of analysis [11]. A number of different devices were tested on their power consumption, and the results indicate clear classifications of device families in most cases. In fact, in some cases even seemingly identical devices were able to be differentiated by the miniscule variances in their power profiles. For a closer look at the results of these experiments, see **Appendix B**.

3.4. Unexpected Data over Power

The next test was conducted to ascertain ThunderMole's effectiveness at facilitating unexpected data over power attacks to exfiltrate data from devices. If malware capable of affecting power consumption could be implanted onto a device connected to a ThunderMole, then it is possible that such modulation in power could be interpreted as data and used for exfiltration.

An easily visible example of this was constructed as follows: An Arduino Nano functioned as the "infected device"—it used the Arduino's capability to temporarily shut down internal components (effectively a sleep mode) to modulate power consumption. The Arduino was connected to the ThunderMole for power, which read this visible modulation as data (**Figure 6**). The reader might recognize these graphs as a form of Morse code. Indeed, the fluctuation in milliamps pulled by the connected device spells out the phrase "Hello World".

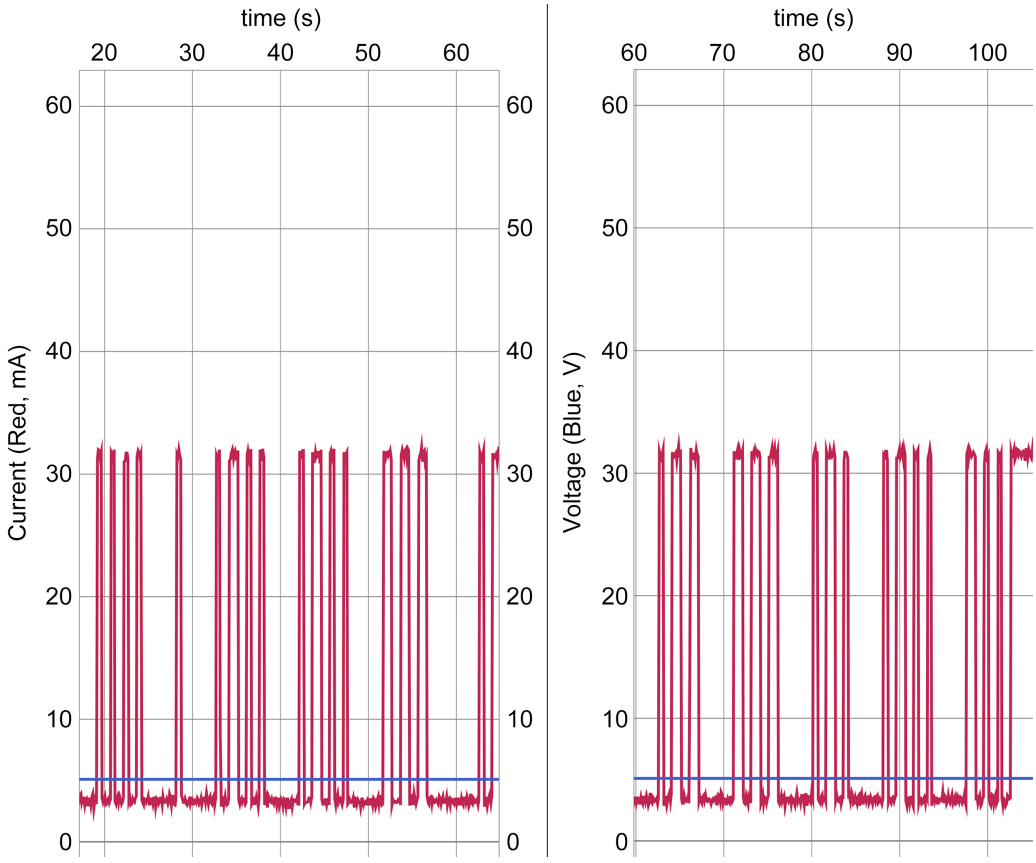


Figure 6. Current (Red, mA) and Voltage (Blue, V) graphed over time (s) to show signal modulation.

For a more interesting example, consider a case where an Arduino is used to hold AES session keys. If an adversary infects the Arduino and gains the ability see the keys and control the onboard LED, then the small difference in current draw the LED creates could be used to transmit data for the ThunderMole to read. Three AES keys received by the ThunderMole can be seen in Figure 7.

These examples demonstrate that power lines must be treated as a communication vector by security professionals.

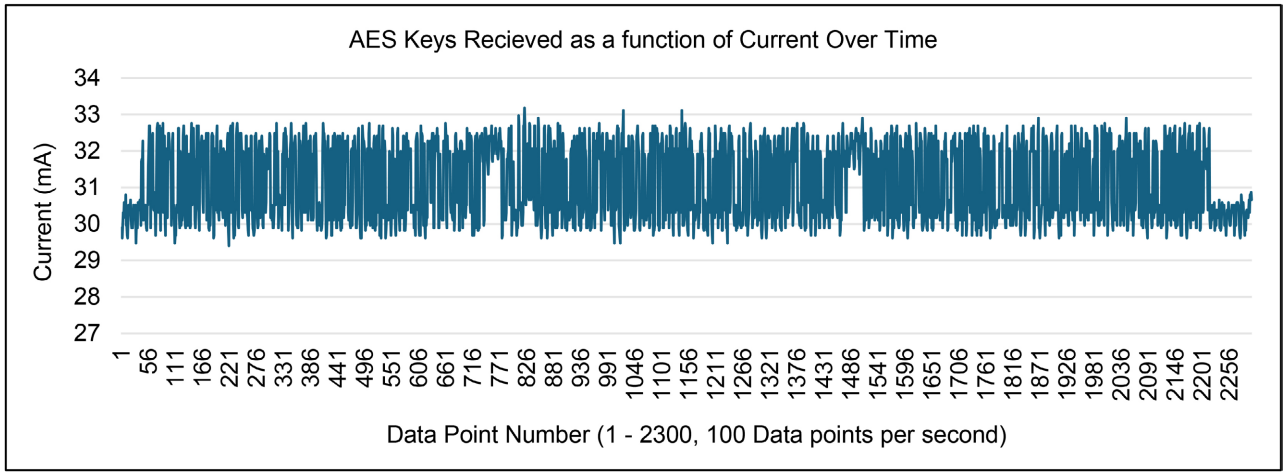


Figure 7. Three different AES keys received as binary.

4. Comparison with Prior Work

From a technical perspective, ThunderMole builds on several existing principles. Power consumption meters exist, and the weaponization of unexpected data over power attacks was shown to be possible in 2019 [7]. However, the form and threat vector of ThunderMole is novel. Most every existing solution (both offensive and defensive) focuses either on the building's infrastructure in its walls or on the main electrical grid, whereas the ThunderMole targets the gap between the victim's device and the wall (Figure 8). It requires no access to the building's power infrastructure or greater power grid to be deployed and is easily pocketed.

Figure 8. Prior work focuses primarily on grid and building infrastructure, but ThunderMole targets the end user's device.

Powerhammer, ThunderMole's closest neighbor in this space, has similar technological capabilities in many regards. In ideal, low-interference conditions, Powerhammer achieved a data read rate of approximately 1000 symbols a second using a sensor attached to the building's infrastructure [7].

The ThunderMole trades read/transmission rate for versatility and portability.

The highest achieved data acquisition rate without compromising on data transmission was only ~100 symbols per second due to wireless limitations. However, the minimum signal level required to be distinguished above noise was only ~1.4 mA (an approximately 3.5 dB noise margin level). This is largely due to ThunderMole's close position to the victim's device, reducing the chances for interference to affect results. In comparison, PowerHammer requires a noise margin level around 8 - 10 dB to ensure fidelity. Data resolution was also found to be comparable to Powerhammer.

Our results above indicate that while ThunderMole does make some concessions in data acquisition speed, it does not compromise its effectiveness when attempting to glean power consumption information from users in its chosen role.

5. Summary & Conclusions

In summary, ThunderMole serves as a natural continuation of existing side-channel, data-over-power attacks that focuses on a hereto unexplored attack space: the distance between the victim's device and the building's power grid. Despite its small size and deceptive form, it has been shown to be no less effective at its job than existing solutions.

It also serves as a springboard for future work in this area. Indeed, the same concepts that make ThunderMole effective can be applied to a wide range of devices: modified battery power banks, laptop chargers, power strips, uninterrupted power supplies, and so forth could all be weaponized in a similar fashion. There is no doubt that more reliable, less conspicuous, and more accurate devices of this nature could be created. As it stands now, ThunderMole is simply a proof-of-concept illustration of the following: it is not enough to secure the power grid, nor can defensive professionals rely wholly upon their building's defensive infrastructure. Our creativity must be brought to bear on the entire ecosystem we seek to protect, as there is no doubt attackers are doing the same.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Horowitz, S.H., Phadke, A.G. and Henville, C.F. (2022) Power System Relaying. John Wiley & Sons.

[2] Sagar, N. (2011) Powerline Communication Systems: Overview and Analysis. Ph.D. Thesis, Rutgers University.

[3] Dierks, A. (2023) The Cybersecurity Benefits of Data-over-Power Communication. Smart Industry.

[4] Yaacoub, J.P.A., Fernandez, J.H., Noura, H.N. and Chehab, A. (2021) Security of Power Line Communication Systems: Issues, Limitations and Existing Solutions. *Computer Science Review*, **39**, Article ID: 100331. https://doi.org/10.1016/j.cosrev.2020.100331

[5] Zhu, Y., Yan, J., Tang, Y., Sun, Y.L. and He, H. (2014) Coordinated Attacks against

Substations and Transmission Lines in Power Grids. 2014 *IEEE Global Communications Conference*, Austin, 8-12 December 2014, 655-661.
https://doi.org/10.1109/glocom.2014.7036882

[6] Chung, H., Li, W., Yuen, C., Chung, W., Zhang, Y. and Wen, C. (2019) Local Cyber-Physical Attack for Masking Line Outage and Topology Attack in Smart Grid. *IEEE Transactions on Smart Grid*, **10**, 4577-4588.
https://doi.org/10.1109/tsg.2018.2865316

[7] Guri, M., Zadov, B., Bykhovsky, D. and Elovici, Y. (2020) Powerhammer: Exfiltrating Data from Air-Gapped Computers through Power Lines. *IEEE Transactions on Information Forensics and Security*, **15**, 1879-1890.
https://doi.org/10.1109/tifs.2019.2952257

[8] Guri, M. (2022) Covid-Bit: Keep a Distance of (at Least) 2m from My Air-Gap Computer! arXiv: 2212.03520.

[9] (2024) USB Rubber Ducky. Hak5.
https://hak5.org/collections/hotplug-attack-tools/products/usb-rubber-ducky

[10] Woolley, M. (2024) The Bluetooth Low Energy Primer.
https://www.bluetooth.com/wp-content/uploads/2022/05/Bluetooth_LE_Primer_Paper.pdf

[11] Streubel, R. and Yang, B. (2012) Identification of Electrical Appliances via Analysis of Power Consumption. 2012 47*th International Universities Power Engineering Conference* (*UPEC*), Uxbridge, 4-7 September 2012, 1-6.
https://doi.org/10.1109/upec.2012.6398559

Appendix A

The source code used on the ThunderMole prototype for all testing is included below. It is largely a modified version of the BLE_uart code example provided by the ESP32 BLE Arduino library.

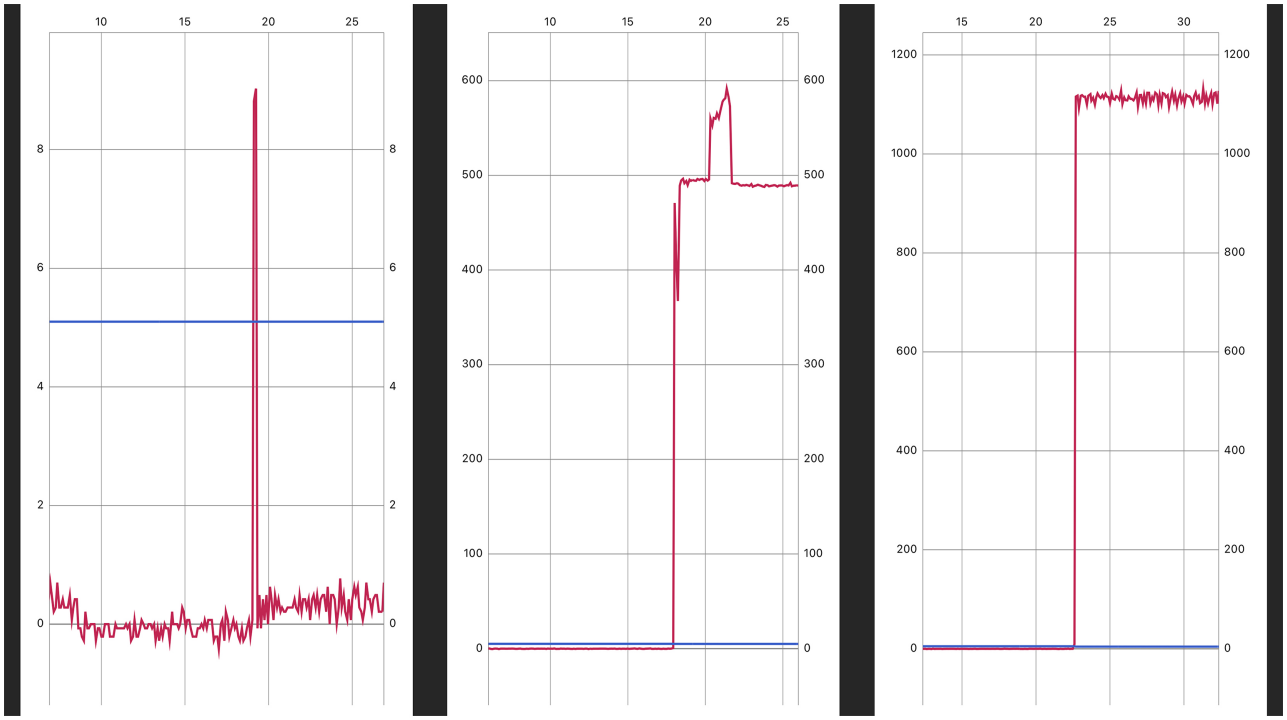
```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
#include <Wire.h>
#include <INA219.h>

INA219 monitor;
BLEServer *pServer = NULL;
BLECharacteristic * pTxCharacteristic;
bool deviceConnected = false;
bool oldDeviceConnected = false;
uint8_t txValue = 0;

#define SERVICE_UUID           "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID_RX "6E400002-B5A3-F393-E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID_TX "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"

class MyServerCallbacks: public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
      deviceConnected = true;
    };

    void onDisconnect(BLEServer* pServer) {
      deviceConnected = false;
    }
};

class MyCallbacks: public BLECharacteristicCallbacks {
    void onWrite(BLECharacteristic *pCharacteristic) {
      std::string rxValue = pCharacteristic->getValue();

      if (rxValue.length() > 0) {
        Serial.println("*********");
        Serial.print("Received Value: ");
        for (int i = 0; i < rxValue.length(); i++)
          Serial.print(rxValue[i]);

        Serial.println();
```

```
        Serial.println("*********");
      }
    }
};
void setup() {
  Serial.begin(115200);

  // Create the BLE Device
  BLEDevice::init("Power Meter");
  // Create the BLE Server
  pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());
  // Create the BLE Service
  BLEService *pService = pServer->createService(SERVICE_UUID);
  // Create a BLE Characteristic
  pTxCharacteristic = pService->createCharacteristic(CHARACTERISTIC_UUID_TX, BLECharac-
teristic::PROPERTY_NOTIFY);
  pTxCharacteristic->addDescriptor(new BLE2902());
  BLECharacteristic * pRxCharacteristic = pService->createCharacteris-
tic(CHARACTERISTIC_UUID_RX, BLECharacteristic::PROPERTY_WRITE);
  pRxCharacteristic->setCallbacks(new MyCallbacks());
  // Start the service
  pService->start();

  // Start advertising
  pServer->getAdvertising()->start();
  Serial.println("Waiting a client connection to notify...");
  monitor.begin();
}

void loop() {

    if (deviceConnected) {
        float current = monitor.shuntCurrent() * 1000;
        float volts = monitor.busVoltage();
        char tosend[50];
        char tosend2[20];
        dtostrf(current, 15, 2, tosend);
        dtostrf(volts, 15, 2, tosend2);
        strcat(tosend, ", ");
        strcat(tosend, tosend2);
        strcat(tosend, "\n");
        pTxCharacteristic->setValue(tosend);
```

```
        pTxCharacteristic->notify();

        delay(10);
       }
    // disconnecting
    if (!deviceConnected && oldDeviceConnected) {
        delay(500); // give the bluetooth stack the chance to get things ready
        pServer->startAdvertising(); // restart advertising
        Serial.println("start advertising");
        oldDeviceConnected = deviceConnected;
    }
    // connecting
    if (deviceConnected && !oldDeviceConnected) {
            // do stuff here on connecting
        oldDeviceConnected = deviceConnected;
    }
}
```

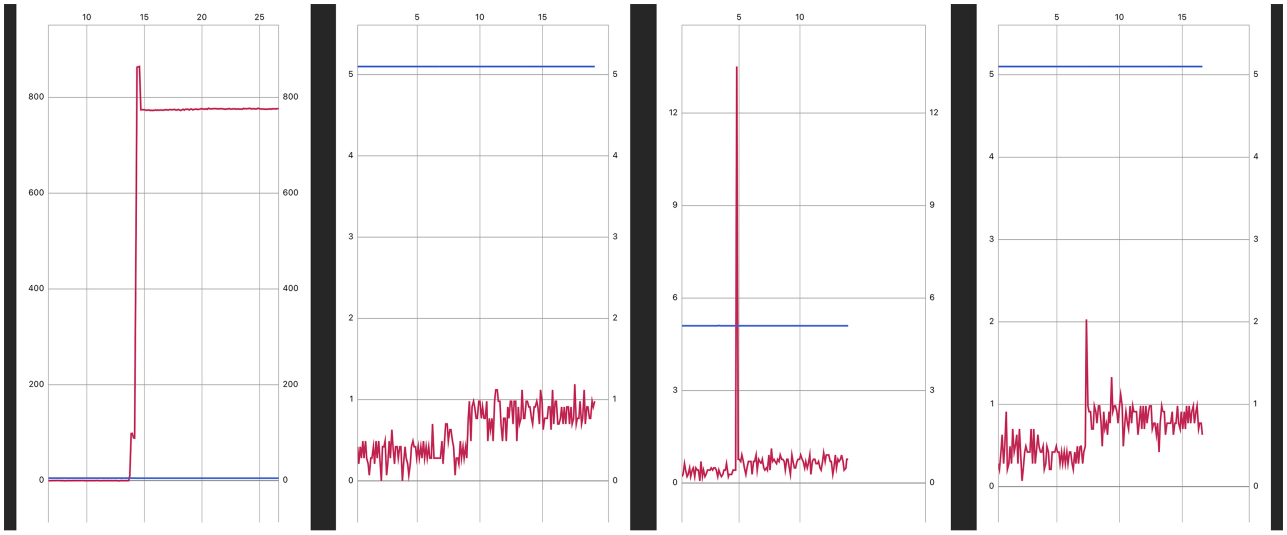
Appendix B

This appendix contains a closer look at the results of performing device classification via power signature with ThunderMole. For all graphs, red is current (mA) and blue is voltage (V), measured over time in seconds. First, let us consider some laptops tested:

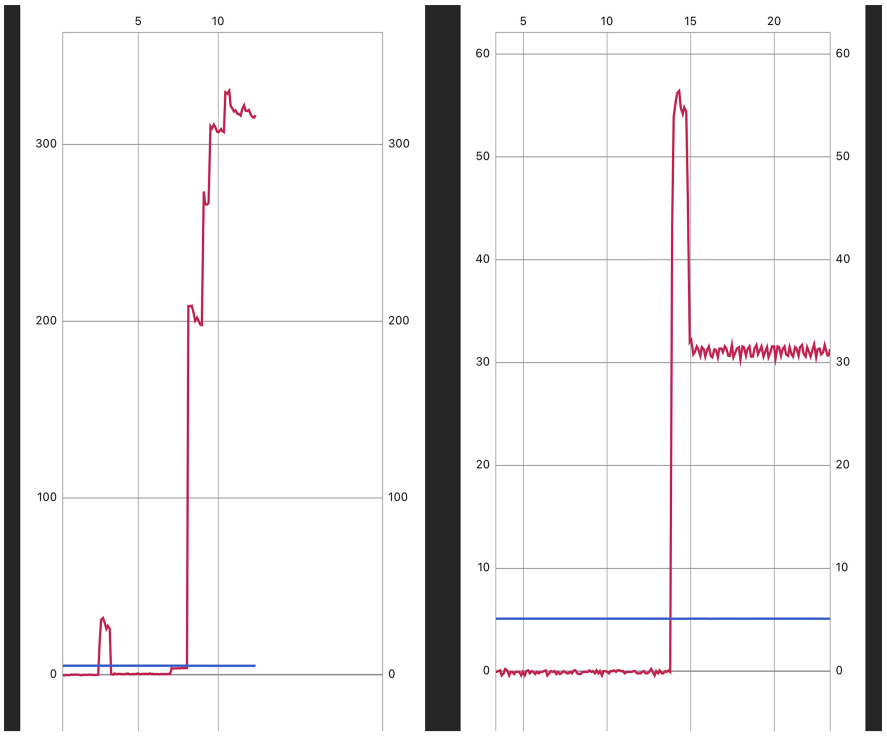


System76 Darter Pro (2022) Apple MacBook Pro (2019) Apple MacBook Air (M1)

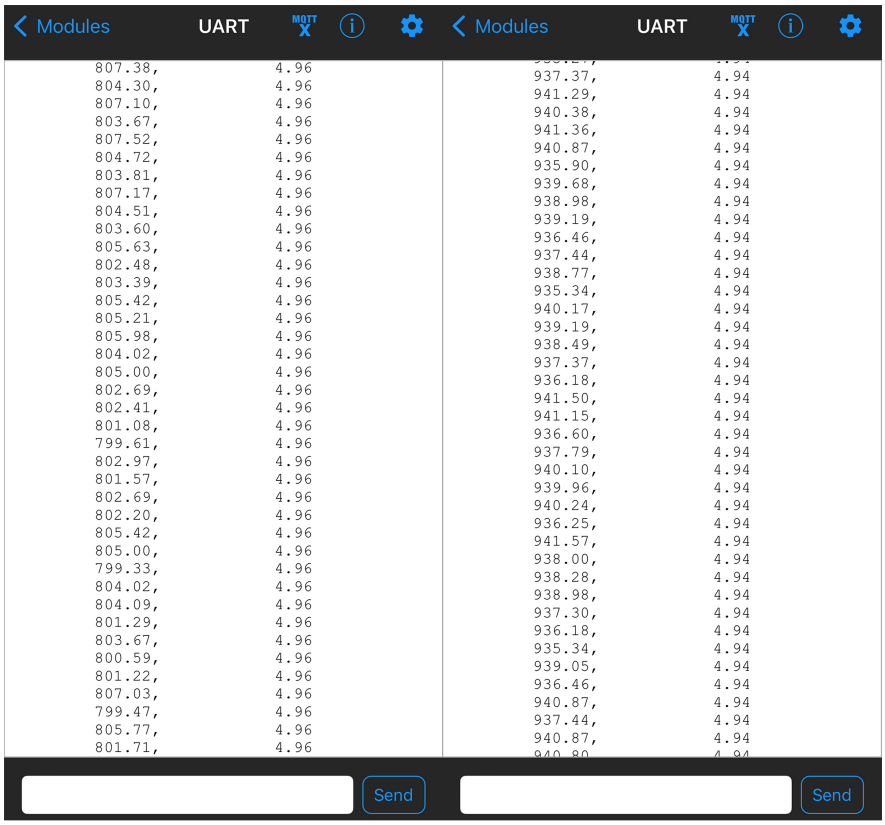
Note that while MacBooks exhibit a similar power signature, different models can be differentiated. This phenomenon is seen in phones as well:



As seen above, iPhone-class phones tend to consume power in a similar fashion, whereas others tested behave differently. This allows ThunderMole deployers to build a sort of identification taxonomy that can help identify connected devices entirely by their power usage characteristics. Furthermore, as expected, IoT devices exhibit their own power consumption profile:



The final inclusion is a comparison of raw data between two Miady portable power banks. Each line contains a current (mA) and voltage (V) data pair. Data was received at 10 data pairs per second. Despite being of the same model and year, there is enough difference in their power consumption to tell the two apart based purely on the received data:



Miady Power Bank A Miady Power Bank B