**Bad Karma BleedingTooth Exploit Report**

**Prepared For:**  
 

**Prepared By:**

**Team Evil Smurfs**

Kelly-Ann Downer

Kaitlyn Malone

Josh Minick

Vi Nguyen

Jack Raymond



Report Issued: December 2021

**TABLE OF CONTENTS**

[TARGET ENVIRONMENT SETUP 3](#_Toc311575097)

[Virtual Machine Setup 4](#_Toc1518582275)

[Ubuntu 20.04.1 LTS Setup 4](#_Toc1203620345)

[Bluetooth Setup 5](#_Toc998173141)

[ATTACKING ENVIRONMENT SETUP 5](#_Toc1359297465)

[Bad Karma Setup 6](#_Toc177736098)

[EXPLOITATION GUIDE 6](#_Toc545084160)

[Steps for Exploitation 7](#_Toc1783834234)

[RESULT 8](#_Toc717400085)

[DEMONSTRATION 10](#_Toc608201420)

[SOURCE CODE 11](#_Toc389703598)

# 

# TARGET ENVIRONMENT SETUP

The Bad Karma Bleedingtooth Vulnerability affects versions 4.8 through 5.9 of the Linux Kernel’s BlueZ Bluetooth stack. To protect the safety of our personal devices we opted to implement our target environment on a virtual machine. Our chosen target operating system is the same as the one used in the original proof of concept by Andy Nguyen: Ubuntu 20.04.1 LTS running kernel version 5.4.0-48-generic.

## Virtual Machine Setup

Our target virtual machine was created using [Oracle VM VirtualBox](https://www.virtualbox.org/), a cross-platform virtualization application. After creating a virtual disk, we booted the [Ubuntu 20.04.1 LTS image](https://old-releases.ubuntu.com/releases/20.04.1/ubuntu-20.04.1-desktop-amd64.iso) and followed the installation instructions from the live environment. This process is outside the scope of the report and detailed instructions can be found [here](https://www.virtualbox.org/manual/ch01.html#gui-createvm). It is additionally required to download and setup the [Oracle VM VirtualBox Extension Pack](https://www.virtualbox.org/wiki/Downloads) that is compatible with your version of version of VirtualBox in order to enable USB passthrough to the virtual machine. We used USB passthrough to connect a generic Bluetooth dongle to our virtual machine; thus providing the required Bluetooth connectivity. Instructions for installing the Extension Pack can be found [here](https://www.virtualbox.org/manual/ch01.html#intro-installing). While it is not required, it is highly recommended to install guest additions on the Ubuntu virtual machine for several quality of life improvements. Instructions for installing Guest Additions can be found [here](https://www.virtualbox.org/manual/ch04.html#guestadd-install).

## Ubuntu 20.04.1 LTS Setup

The default Linux kernel version used in Ubuntu 20.04.1 LTS is 5.11.0-38-generic which is not susceptible to the Bad Karma Bleedingtooth Vulnerability. The following steps must be performed to use the vulnerable 5.4.0-48-generic kernel version:

1. **Installation:** Install 5.4.0-48-generic by entering ‘sudo apt install linux-image-5.4.0-48-generic' into the terminal.
2. **Holding:** Use ‘sudo apt-mark hold linux-image-5.4.0-48-generic' to ensure the kernel is not removed in future updates
3. **Bluetooth Modules:** To obtain the Bluetooth capabilities not included in the base kernel, install the extra modules using ‘sudo apt install linux-modules-extra-5.4.0-48-generic'
4. **Reboot:** Restart the virtual machine and hold shift to open the Grub menu. From the advanced options select to boot from kernel version 5.4.0-48-generic.

## Bluetooth Setup

A generic Bluetooth dongle is required to pass Bluetooth capabilities to the virtual machine. It is possible to pass Bluetooth capabilities from your device’s integrated chipset, but this process is difficult and not guaranteed to work. After plugging in the dongle, verify that USB passthrough is enabled and a filter has been set for the dongle within the VM’s Virtual Box settings. Additionally verify that you are using the correct Kernel version via “uname -r” and that you have installed the extra modules. To check whether Bluetooth is working, attempt to turn it on using your preferred Bluetooth Front-end application. If you are attempting to replicate our setup using the same Ubuntu image, it is unlikely that this will work on the first attempt. To troubleshoot, use the bluetoothctl utility from the bluez-utils package which will often provide useful error messages. Common fixes include loading the btusb kernel module via “sudo modprobe btusb”, unblocking the hci device using “sudo rfkill unblock all”, and starting/enabling the Bluetooth service via ‘sudo systemctl start bluetooth.service’ / ‘sudo systemctl enable bluetooth.service’. More detailed configuration and troubleshooting information may be found [here](https://wiki.archlinux.org/title/Bluetooth) (Note that Ubuntu is Debian based so some instructions from the Arch wiki may not apply). After Bluetooth is verified to be working, make sure that the Bluetooth adapter is on and discoverable before proceeding with the exploitation.

# ATTACKING ENVIRONMENT SETUP

The attacking device can be any device with Bluetooth connectivity that can compile and run programs using the C programming language. We chose a raspberry pi due to the cheapness and the portability of the device. For the operating system, we flashed the pi with Raspberry pi OS 32-bit for maximum hardware compatibility. If using pi OS, additional packages may need to be installed for Bluetooth capabilities such as bluetooth, pi-bluetooth, bluez, blueman, and libbluetooth-dev. Verify that Bluetooth is on and working on the attacking machine before beginning the exploitation.

## Bad Karma Setup

On the Attacking machine, create a new C text file and paste in the contents of poc.c from [here](https://github.com/google/security-research/security/advisories/GHSA-h637-c88j-47wq). Compile this file using ‘gcc -o poc poc.c -lbluetooth’ where poc.c is the name of your C file.

# 

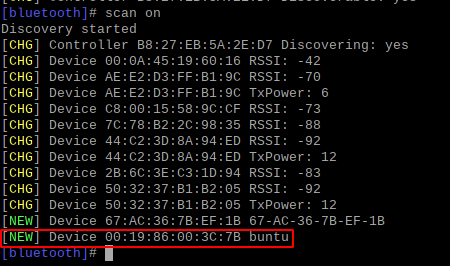
# EXPLOITATION GUIDE

To exploit the Bad Karma Bleedingtooth Vulnerability, the remote attacker needs to be within range and know the Bluetooth Device Address (BD\_ADDR) of their victim’s Bluetooth Device. Additionally, the victim device needs to be using a Linux kernel version between 4.8 and 5.9 and have their Bluetooth adapter on and discoverable. The provided [Proof of Concept Code](https://github.com/google/security-research/security/advisories/GHSA-h637-c88j-47wq) will send a malicious l2cap packet to the victim’s device which invokes a heap based type confusion. This causes the victim’s kernel to panic, thereby crashing the device’s Bluetooth services. Any new or existing Bluetooth connections with the device will become unresponsive, resulting in a denial of service.

## Steps for Exploitation

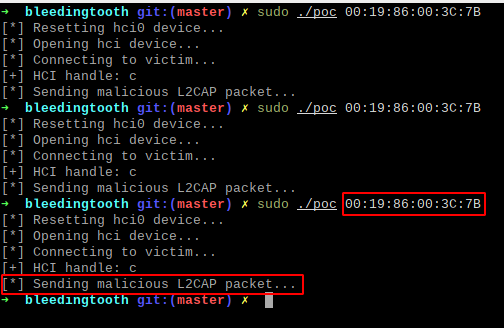
**Perform the following steps on the Attacking machine:**

1. **Scan for the Bluetooth Device Address (BD\_ADDR) of the victim's Bluetooth device using ‘bluetoothctl’. Then, turn on Bluetooth scanning by entering ‘scan on’.**



For this demonstration, the hostname of the victim’s Bluetooth adapter is ‘buntu’ (short for Ubuntu). This hostname often contains the name of the victim or the device type, making it easy to discover a target. In the screenshot above, bluetoothctl displays the Bluetooth Device Address (BD\_ADDR) of buntu: ‘00:19:86:00:3C:7B’.

1. **Execute the Bad Karma proof of concept code**



Executing the code requires superuser privileges and the victim’s BD\_ADDR as an argument. Performing a successful denial of service may require multiple executions.

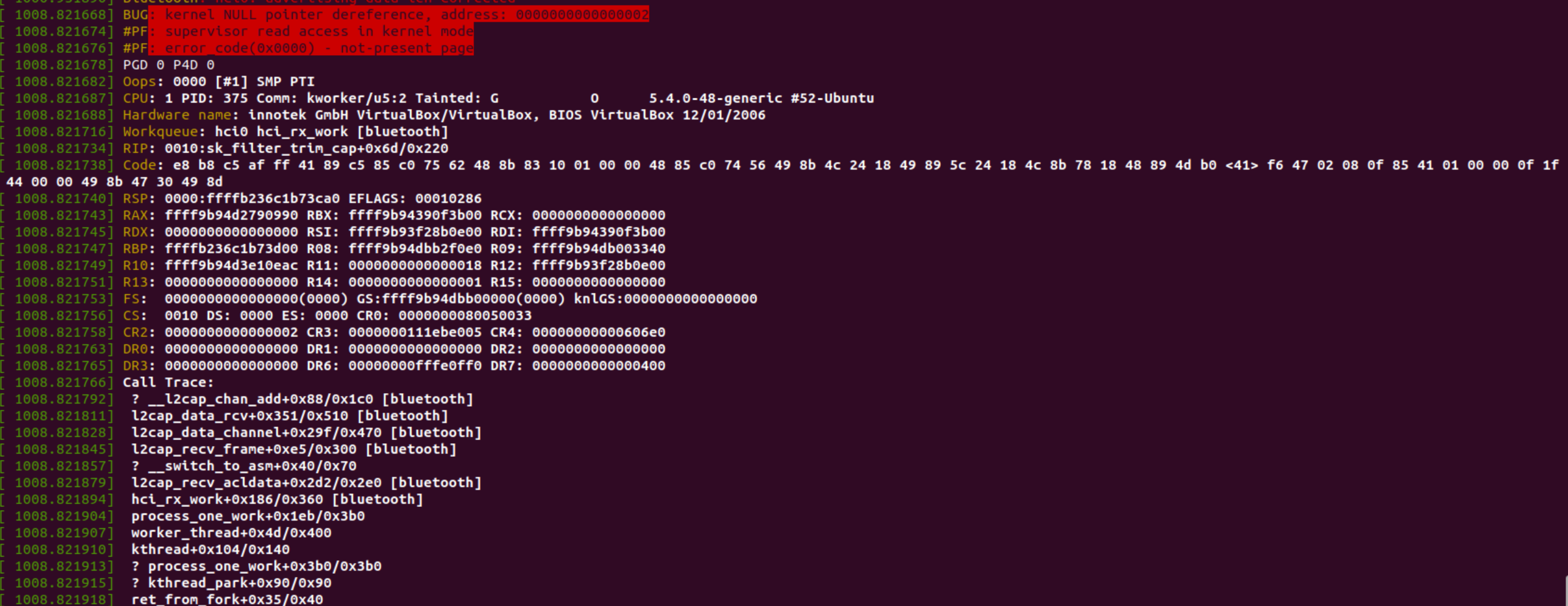
# 

# RESULT

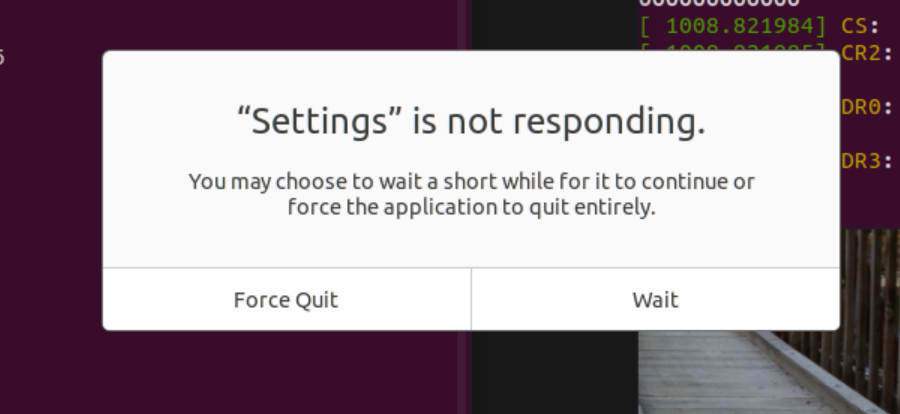
A successful execution of the proof of concept code will cause the victim’s kernel to panic, thereby crashing the device’s Bluetooth services. Any new or existing Bluetooth connections with the device will become unresponsive, resulting in a denial of service.

**The following screenshots display after-effects of the attack:**

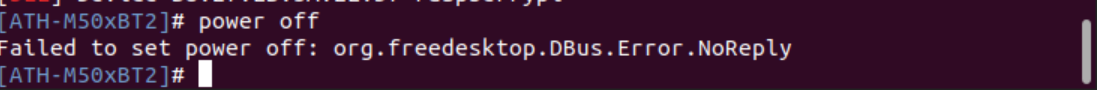
1. **Using the dmesg log, we can observe the following kernel panic:**



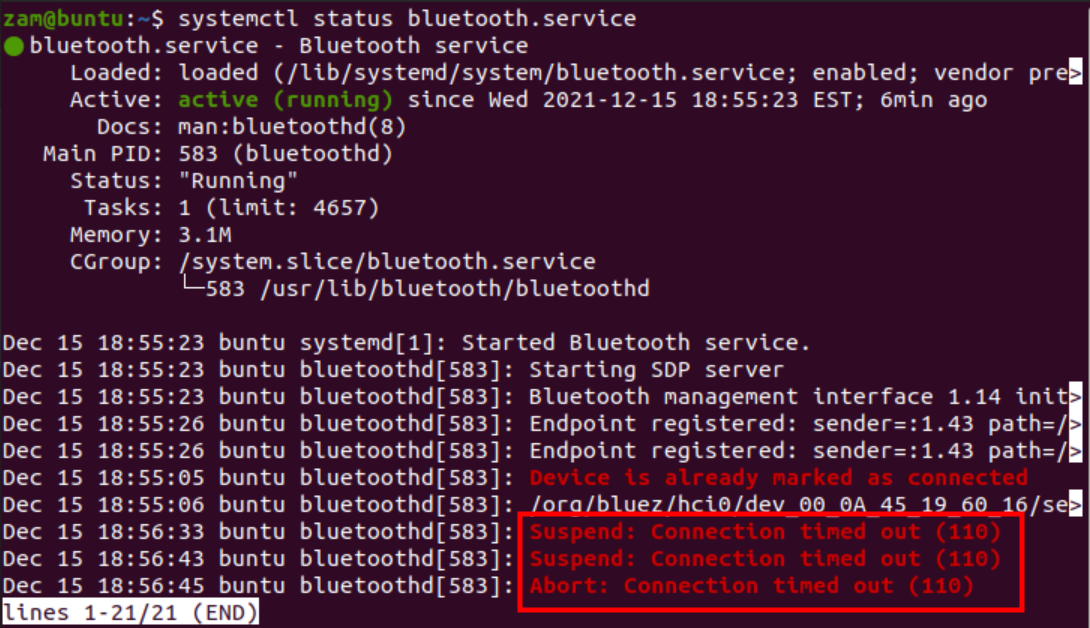
1. **After the attack, Gnome-settings will crash:**



1. **Attempting to power off Bluetooth from the command line results in a dbus error:**



1. **systemctl shows that our headphone’s connection timed out and was aborted:**



# DEMONSTRATION

The following video provides a brief live demonstration of the Bad Karma Bleedingtooth attack. As you can observe, the denial of service is successful when the connection with the Bluetooth headphones drops and the music stops playing.

[Bad Karma Bleedingtooth Exploit](https://youtu.be/Lr6ytrEpW-g):



# 

# SOURCE CODE

The source code used to execute the vulnerability was included in the [GitHub repository](https://github.com/google/security-research/tree/master/pocs/linux/bleedingtooth) created by Andy Nguyen. The repository included three documents[, exploit.c](https://github.com/google/security-research/blob/master/pocs/linux/bleedingtooth/exploit.c), [readme.md](https://github.com/google/security-research/blob/master/pocs/linux/bleedingtooth/readme.md) and [writeup.md](https://github.com/google/security-research/blob/master/pocs/linux/bleedingtooth/writeup.md). The exploit.c file contains the proof-of-concept that shows the demonstration of the attack. After the code is compiled and executed it takes advantage of the A2MP higher level protocol and by-pass authentication to gain kernel privilege. The second document, readme.md included the instructions and commands on how to perform the attack, the computer it was tested on was the (DELL XP 15 with Ubuntu 20.04.1 LTS), and the success rate of the attack is (80%). With the privilege an attacker can open applications, the example showed opened the gnome calculator. Lastly, the writeup.md file included additional information on the exploit. It includes an introduction and explanation of how the BlueZ stack is vulnerable, the process of how the vulnerability was discovered by looking at the methods used to create the stack, an explanation of how the code exploits the vulnerability, the timeline of discovery, and credits.