**Crackle 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](#_Toc263237908)

[Virtual Machine Setup 4](#_Toc1675826707)

[Bluetooth Setup 4](#_Toc1195646275)

[ATTACKING ENVIRONMENT SETUP 5](#_Toc620700195)

[EXPLOITATION GUIDE 6](#_Toc1478047747)

[Steps for Exploitation 7](#_Toc625769525)

[RESULT 10](#_Toc1247506904)

[DEMONSTRATION 11](#_Toc393090009)

[SOURCE CODE 12](#_Toc1980596216)

# 

# TARGET ENVIRONMENT SETUP

In order for Wireshark to be able to get the packets from pairing, a user and a Bluetooth device must be within range and pairing for the first time. When a device is pairing for the first time, a long term key is established. This key can be found through the packets once decrypted. If the long term key is found, a malicious user could take advantage of this key and use it to connect to the device without needing to pair.

## 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).

## 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 Kali Linux system on a virtual machine. The steps for setting up the virtual machine are outlined above. On the Linux machine you will need to install Python, [Pyserial](https://pyserial.readthedocs.io/en/latest/pyserial.html), [Wireshark](https://www.wireshark.org/download.html), and the Crackle repository from GitHub which can be found [here](https://github.com/mikeryan/crackle).

# 

# EXPLOITATION GUIDE

Crackle relies on Bluetooth Low Energy Legacy pairing modes in order to decrypt the captured packets. To decrypt packets using crackle, you first need a pcapng file containing the captured Low Energy packets. The packets required for crackle are: CONNECT\_REQ establishing the connection, Pairing Request and Pairing Response, two random values, two confirms, LL\_ENC\_REQ, LL\_ENC\_RSP, and LL\_START\_ENC\_REQ. Crackle also has a brute force method for which you will need: two random values, LL\_ENC\_REQ, LL\_ENC\_RSP, and LL\_START\_ENC\_REQ. To use the brute force method will take more time as the attack is slower, but the results should still be the same. To capture the packets you will need a bluetooth sniffer tool like Adafruit BLE or Ubertooth One, Wireshark, and Crackle installed on a Linux system.

## Steps for Exploitation

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

1. **Install Python and Pyserial on the Linux machine.**

This can be done using the command:

$ sudo apt-get install python3

1. **Install Crackle and its dependencies.**

To install Crackle we need the latest version which can be found on the GitHub here: <https://github.com/mikeryan/crackle.> Then we will use the following commands to install Crackle on our machine:

$ make

$ make install

After these two commands are run, Crackle is installed onto our Linux machine and is ready to be used. The GitHub also helpfully provides example files which you can run to ensure that Crackle was installed properly.

1. **Install and set up Wireshark.**

You will also need Wireshark for this attack to work, which you can install by using:

$ sudo apt update

$ sudo apt-get install wireshark

$wireshark -h (to verify Wireshark has successfully installed)

Before running the scan you need to prepare your Wireshark environment. You will need \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(stuff here about the parameters, image as well if possible)

1. **Scan for Bluetooth Low Energy packets during pairing.**

With the Wireshark environment set up, the sniffer tool plugged in, and Crackle installed you can scan the network for any BLE pairing. If there is successful pairing you will see a number of packets pass through Wireshark including a connect request, a pairing request and response, an encryption request and response, as well as potentially empty packets which is normal when using sniffers. Once the pairing process is complete, the scan can be stopped and saved as a .pcapng file.

1. **Using Crackle to decrypt the packets.**

With the pcapng file from the previous step, we first check to ensure that the pncap file contains the proper packets needed for decryption. We can do this by running the following command in our terminal:

$ crackle -i <file.pcapng>

This should give an output that either says that it found the amount of encrypted packets and proceeds to find the temporary key, or it will state that there are not a sufficient amount of packets. If the latter option happens, you will need to try to re-sniff the pairing process to ensure that you get all of the required packets.

If you wish to decrypt all of the packets, you simply need to add a -o as well as an <output.pcap> file name for the decryption to save to.

# 

# RESULT

A successful execution of Crackle will result in decrypted Bluetooth Low Energy packets. Crackle will display a screen showing how many of each required packet was found in the pcap file as well as the TK that was found. If the pairing happens between a user and a Bluetooth Low Energy Secure Connection device, Crackle will not be able to decrypt the packets and will alert the user that the pairing mode uses Secure Connection.

**The following screenshots display after-effects of using Crackle**

1. **Using the commands as outlined above, we should get the following result:**



# DEMONSTRATION

Currently, the area we were working and the devices we had to exploit both played a part in us being unable to fully capture all of the packets needed for Crackle to successfully run. We are looking into more devices that are susceptible to this sniffing attack as well as areas that have less people which will have less Bluetooth interference. This sniffing attack is quick to run and execute so it will be no issue fully completing this exploit once those two issues are sorted.

# 

# SOURCE CODE

The source code used in the packet decryption can be found in the [GitHub repository](https://github.com/mikeryan/crackle) created by user mikeryan. The repository contains information about how to use Crackle for cracking a TK or found decrypting packets using an LTK. There are also many troubleshooting tips discussing the various ways that a user may not get the intended results. The most common issue was that all of the required packets were not being captured, in which case the process needs to be run again until the packets are completely captured. The repository also contains sample files that allow you to make sure the program is running as expected and to help the user practice the commands. There is also an addition to the original Crackle program found [here](https://github.com/mikeryan/crackle/pull/36/commits/f9c1fe7c5185735aa6accdc53213db3b33f6ce3d) created by user pccr10001 which specifically handle packets captured by a Nordic brand sniffer. Previously, Crackle only worked with specific sniffers, namely Ubertooth One, but this update allows for compatibility with more affordable sniffers.