MouseJack

Injecting Keystrokes into Wireless Mice

Marc Newlin
Bastille Threat Research Team
Version 1.1

Abstract

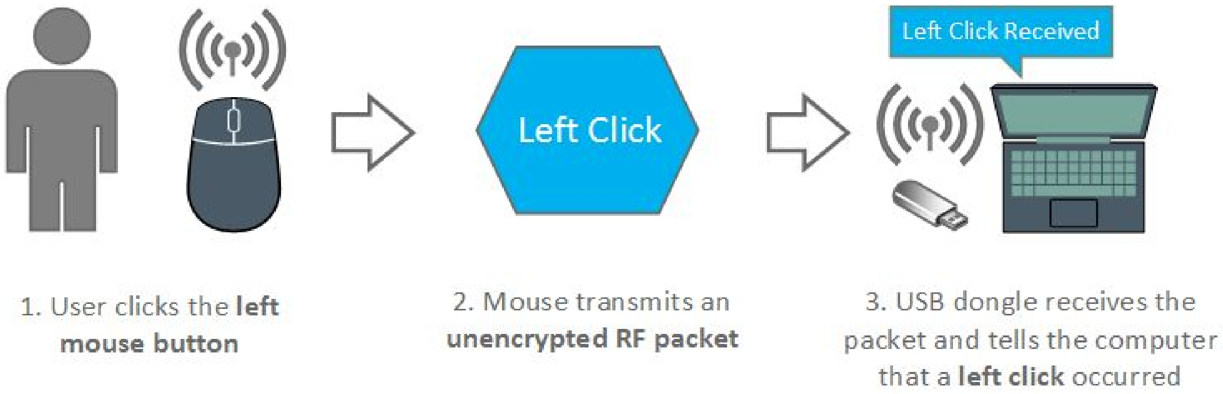
MouseJack is a collection of security vulnerabilities affecting non-Bluetooth wireless mice and keyboards. Spanning seven vendors, these vulnerabilities enable an attacker to type arbitrary commands into a victim's computer from up to 100 meters away using a $15 USB dongle.

Overview

Wireless mice and keyboards commonly communicate using proprietary protocols operating in the 2.4GHz ISM band. In contrast to Bluetooth, there is no industry standard to follow, leaving each vendor to implement their own security scheme.

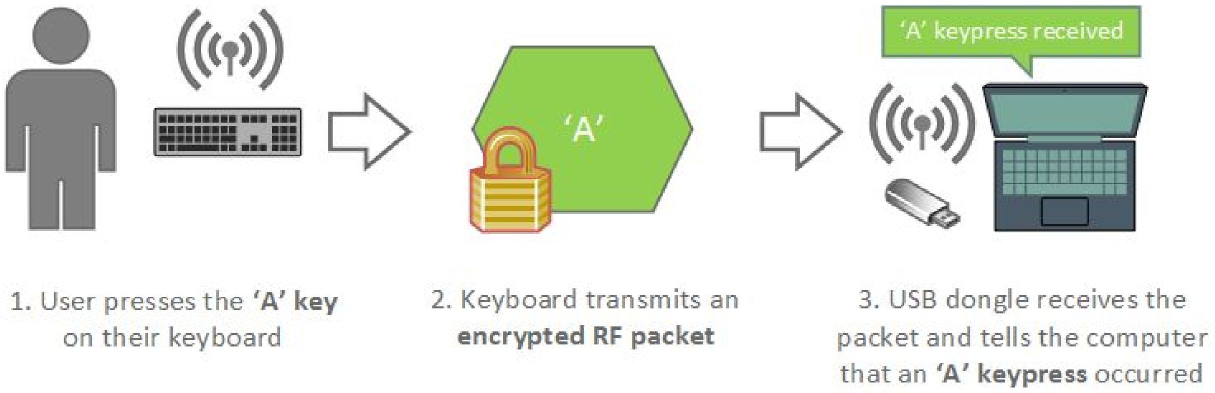
Wireless mice and keyboards work by transmitting radio frequency packets to a USB dongle plugged into a user's computer. When a user presses a key on their keyboard or moves their mouse, information describing the actions are sent wirelessly to the USB dongle.

The dongle listens for radio frequency packets sent by the mouse or keyboard, and notifies the computer whenever the user moves their mouse or types on their keyboard.


Left Click
Left Click Received
1. User clicks the **left mouse button**
2. Mouse transmits an **unencrypted RF packet**
3. USB dongle receives the packet and tells the computer that a **left click** occurred

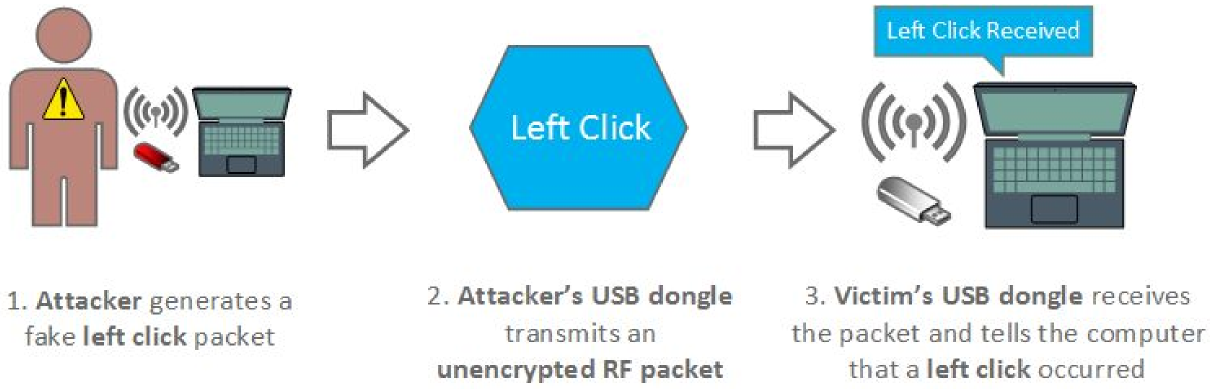
Unencrypted Mouse Packet

In order to prevent eavesdropping, most vendors encrypt the data being transmitted by wireless keyboards. The dongle knows the encryption key being used by the keyboard, so it is able to decrypt the data and see what key was pressed. Without knowing the encryption key, an attacker is unable to decrypt the data, so they are unable to see what is being typed.


'A' keypress received
'A'
1. User presses the 'A' key on their keyboard
2. Keyboard transmits an encrypted RF packet
3. USB dongle receives the packet and tells the computer that an 'A' keypress occurred

Encrypted Keyboard Packet

Conversely, none of the mice that were tested encrypt their wireless communications. This means that there is no authentication mechanism, and the dongle is unable to distinguish between packets transmitted by a mouse, and those transmitted by an attacker. As a result, an attacker is able to pretend to be a mouse and transmit their own movement/click packets to a dongle.


Left Click Received
Left Click
1. Attacker generates a fake left click packet
2. Attacker's USB dongle transmits an unencrypted RF packet
3. Victim's USB dongle receives the packet and tells the computer that a left click occurred

Spoofed Unencrypted Keyboard Packet

Problems in the way the dongles process received packets make it possible for an attacker to transmit specially crafted packets which generate keypresses instead of mouse movement/clicks.

Common Transceivers

Nordic Semiconductor makes the popular nRF24L series of transceivers used in most of the devices vulnerable to MouseJack. The nRF24L transceivers provide a mechanism to wirelessly transmit data between two devices, but the functionality that turns mouse clicks and keypresses into bytes sent over the air is implemented by each vendor. From a research standpoint, this means that the same tools and procedures can be used to evaluate products from different vendors.

Software Defined Radio Decoder

The nRF24L transceivers support multiple data rates, address lengths, packet formats, and checksums. To accommodate this, the initial research was performed using a USRP B210 Software Defined Radio, coupled with a custom GNU Radio block designed to decode all of the possible packet configurations. This proved fruitful, but there were drawbacks to using an SDR.

None of the tested devices employ frequency hopping in the traditional sense, but they all change channels to avoid interference from other 2.4GHz devices (Bluetooth, Wi-Fi, etc). The channel hopping is generally unpredictable, and Software Defined Radios are slower to retune than the nRF24L radios. This makes it difficult for an SDR based decoder to observe all of the transmitted packets.

When a mouse transmits a movement packet to a dongle, the dongle replies with an acknowledgement packet telling the mouse that the movement packet was received. The mouse waits for a short period before determining that the packet it transmitted was lost, which can be as short as 250 microseconds. Due to USB latency and processing overhead, the SDR-based decoder is unable to transmit ACKs within the narrow timeout window, so two way communication between an SDR and dongle/mouse is not a viable option.

Enter the NES Controller

The SDR decoder made it possible to figure out the formats of the data being transmitted over the air, but reliable two way communication would be necessary to start evaluating the devices for potential weaknesses.

Parallel to the MouseJack research, an Arduino/nRF24L-based NES controller was being built as part of a Burning Man project. The nRF24L was chosen because they are inexpensive and easy to use, but the connection with the MouseJack project quickly became apparent and it was decided to build an NES controller that could spoof wireless mice.

MouseJack NES Controller

The nRF24L chips do not officially support packet sniffing, but Travis Goodspeed documented[1] a pseudo-promiscuous mode in 2011 which makes it possible to sniff a subset of packets being transmitted by other devices. This enables the NES controller to passively identify wireless mice and keyboards without the need for an SDR.

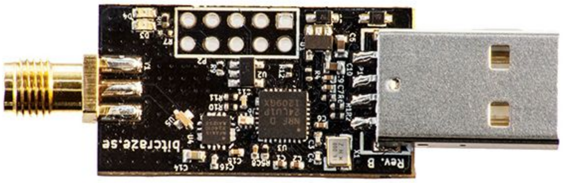
The NES controller proved to be an excellent platform for learning about the behavior of mouse communication protocols. As opposed to passively collecting data, the NES controller translates d-pad arrows into mouse movement packets, and A/B buttons into left and right clicks. In order to achieve a smooth user experience, it was necessary to create a model of the packet timing and specific behavior expected by the dongle.

The concept worked well, and the NES controller was presented at ToorCon in 2015[2], which demonstrated the viability of controlling previously unseen wireless mice at will.

Despite being a marked improvement over the SDR decoder, the NES controller was not without problems. Running off of battery power made it impractical to use amplified transceivers, limiting the practical range to around 10 meters.

Crazyradio PA Dongles

The Crazyflie[3] is an open source drone which is controlled with an amplified nRF24L-based USB dongle called the Crazyradio PA[4]. This is equivalent to an amplified version of the USB dongles commonly used with wireless mice and keyboards. Modifying the Crazyradio PA firmware to include support for pseudo-promiscuous mode made it possible to distill the packet sniffing and injection functionality down to a minimal amount of Python code.



Crazyradio PA USB Dongle

Fuzzing

The Crazyradio PA dongles made it possible to implement an efficient and effective fuzzer. Mouse and keyboard USB dongles communicate user actions to the operating system in the form of USB HID packets, which can be sniffed by enabling the usbmon kernel module on Linux.

The implemented fuzzer takes advantage of this by transmitting RF packets to a mouse/keyboard dongle attached to the same computer, and monitoring USB traffic for generated USB HID packets. Anytime mouse movement or keypresses are sent to the operating system, the recently transmitted RF packets are recorded for analysis. Fuzzing variants of observed packet formats and behaviors yielded the best results.

Vulnerabilities

Specifics of the discovered vulnerabilities vary from vendor to vendor, but they generally fall into one of three categories:

1. Keystroke injection, spoofing a mouse

When processing received RF packets, some dongles do not verify that the type of packet received matches the type of device that transmitted it. Under normal circumstances, a mouse will only transmit movement/clicks to the dongle, and a keyboard will only transmit keypresses. If the dongle does not verify that the packet type and transmitting device type match, it is possible for an attacker to pretend to be a mouse, but transmit a keypress packet. The dongle does not expect packets coming from a mouse to be encrypted, so it accepts the keypress packet, allowing the attacker to type arbitrary commands on the victim's computer.

2. Keystroke injection, spoofing a keyboard

Most of the tested keyboards encrypt data before transmitting it wirelessly to the dongle, but not all of the dongles require that encryption is used. This makes it possible for an attacker to pretend to be a keyboard, and transmit unencrypted keyboard packets to the dongle. This bypasses the encryption normally used by the keyboard, and allows an attacker to type arbitrary commands on the victim's computer.

3. Forced pairing

Before a wireless keyboard or mouse leaves the factory, it is paired with a dongle. This means that it knows the wireless address of the dongle, and in the case of a keyboard, the secret encryption key.

Some vendors include the ability to pair new devices with a dongle, or pair an existing keyboard or mouse with a new dongle. If a dongle is lost, for example, this means that the user only needs to purchase a new dongle, instead of an entirely new set.

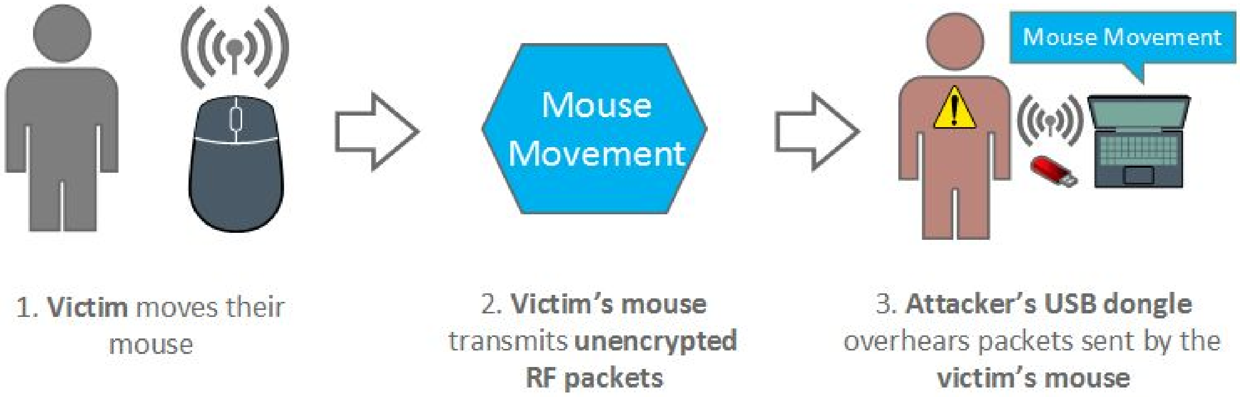
To prevent unauthorized devices from pairing with a dongle, the dongle will only accept new devices when it has been placed into a special "pairing mode" by the user, which lasts for 30-60 seconds.

It is possible to bypass this pairing mode on some dongles and pair a new device without any user interaction. In the case where a victim only has a mouse, but is using a dongle vulnerable to keystroke injection by spoofing a keyboard, an attacker can pair a fake keyboard with the dongle, and use it to type arbitrary commands on the victim's computer.

Anatomy of an Attack

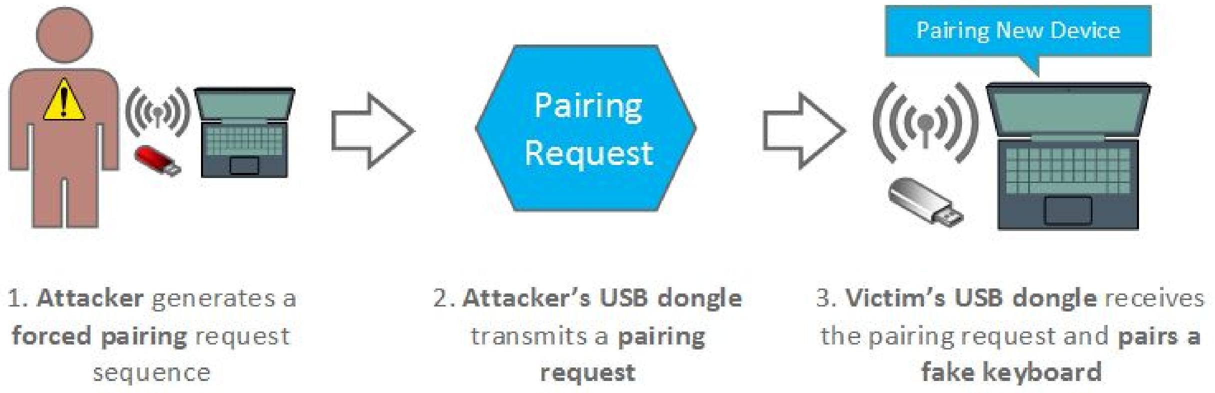
Carrying out a MouseJack attack does not require specialized or expensive equipment, and can be done with a $15 USB dongle.

First, the attacker identifies a target wireless mouse or keyboard by listening for RF packets transmitted when a user is moving/clicking the mouse or typing on the keyboard.


Mouse Movement
Mouse Movement
1. Victim moves their mouse
2. Victim's mouse transmits unencrypted RF packets
3. Attacker's USB dongle overhears packets sent by the victim's mouse

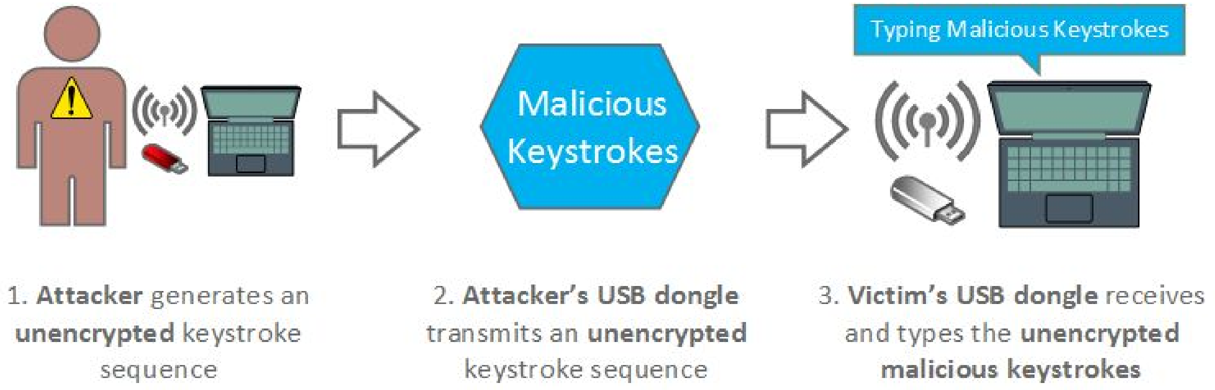
Attacker Identifying a Victim's Mouse or Keyboard

If necessary, the attacker now force-pairs a fake keyboard with the victim's dongle.


Pairing Request
Pairing New Device
1. Attacker generates a forced pairing request sequence
2. Attacker's USB dongle transmits a pairing request
3. Victim's USB dongle receives the pairing request and pairs a fake keyboard

Attacker Force-Pairing a Fake Keyboard with the Victim's Dongle

Finally, the attacker transmits keypress packets to type a series of commands into the victim's computer. This can include downloading a virus or rootkit, transferring files off of the victim's computer, or anything else the attacker could do if they were physically typing on the computer's keyboard.


Malicious Keystrokes
Typing Malicious Keystrokes
1. Attacker generates an unencrypted keystroke sequence
2. Attacker's USB dongle transmits an unencrypted keystroke sequence
3. Victim's USB dongle receives and types the unencrypted malicious keystrokes

Attacker Injecting Keystrokes into the Victim's Dongle

Mitigation

There are two basic types of nRF24L chips used by keyboards, mice, and dongles: one-time-programmable, and flash memory. One-time-programmable devices cannot be updated once they leave the factory, but flash memory devices can.

For non-updateable devices, which represent the majority of those tested, there is no mechanism to secure a vulnerable device short of unplugging the USB dongle from the computer.

For devices with updated firmware available from the manufacturer, it is recommended to install the update before continuing to use the affected mouse or keyboard.

References

1. Goodspeed, Travis. (February 7, 2011). Promiscuity is the nRF24L01+'s Duty. Retrieved from http://travisgoodspeed.blogspot.com/2011/02/promiscuity-is-nrf24l01s-duty.html
2. Newlin, Marc. (October 24, 2015). Hacking Wireless Mice with an NES Controller. Presented at ToorCon 17, San Diego, CA.
3. Bitcraze AB. (2016). Crazyflie 2.0. Retrieved from https://www.bitcraze.io/crazyflie-2/
4. Bitcraze AB. (2016). Crazyradio PA. Retrieved from https://www.bitcraze.io/crazyradio-pa/

More Information

Please refer to: https://www.mousejack.com/

Document History

Version	*Date*	*Comment*
1.0	Feb 12, 2016	Original version.
1.1	Feb 22, 2016	Edits & ‘More Information’.