IOActive Security Advisory

Title	Ninebot by Segway miniPRO Vulnerabilities
Severity	Critical
Discovered by	Thomas Kilbride James Thomas and Stefan Boesen
Advisory Date	July 19, 2017

Affected Products

Ninebot by Segway miniPRO hands-free, two-wheel electric scooter

Impact

A malicious attacker could potentially perform one or more of the following behaviors:

- Malicious firmware updates
- Remote code execution/control
- Device tracking and theft of self-balancing vehicles with the potential to circumvent critical safety interlocks

Background

Ninebot Limited, which purchased Segway Inc. in 2015, sells a line of self-balancing motorized electric scooters used for transportation under 30km/h. Recently, issues regarding the safety of scooters have surfaced, primarily caused by poor manufacturing quality or a general lack of safety-centered design. In response to these issues, the Federal Trade Commission has required that any scooter imported to the US meet baseline safety requirements set by Underwriters Laboratories (UL). Current regulations require scooters to meet certain mechanical and electrical specifications with the goal of preventing battery fires and various mechanical failures; however, there are currently no regulations that ensure firmware integrity and validation, even though this is also integral to system safety.

Using reverse engineering and forensic techniques, IOActive determined that the Ninebot by Segway miniPRO had several critical vulnerabilities which were wirelessly exploitable. These vulnerabilities could be used by an attacker to bypass safety systems designed by Ninebot.

Technical Details

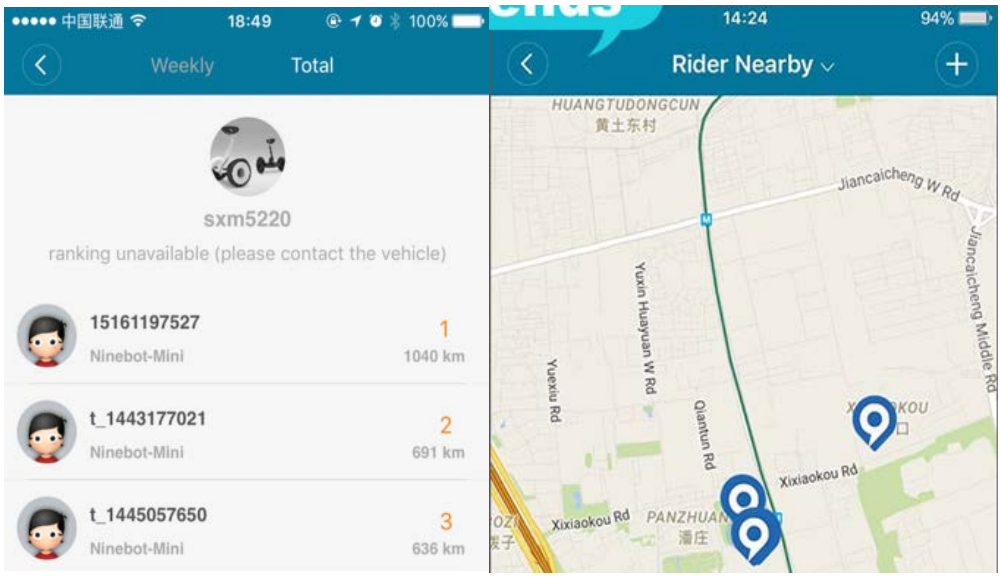
IOActive performed the following steps to compromise the miniPRO scooter:

1. By intercepting communication between the scooter and mobile application, it was determined that Personal Identification Number (PIN) authentication was not required to establish a connection.

2. After intercepting the communications, the IOActive researcher reverse-engineered the scooter's communications protocol using a Bluetooth sniffer. This is the same system used for remote control and configuration settings.
3. Using the packet structure, the IOActive researcher was able to reverse engineer the firmware update mechanism, and discovered that Ninebot did not implement any integrity checks on firmware images before accepting a firmware update to the scooter.
4. Upon further investigation of the Ninebot application, IOActive also determined that riders in the area were indexed using their smartphones' GPS; therefore, each riders' location was published and publicly available, which makes weaponization of an exploit much easier for an attacker.

Proof-of-Concept

1. Using the Ninebot application, an attacker can locate scooter riders nearby.



2. An attacker can then connect to the Ninebot using a modified version of the Nordic UART application, without being asked for a password.
3. By sending the following payload from the Nordic application, the attacker can change the Segway Ninebot MiniPro's PIN to "111111".

```
unsigned char payload[13] =

{0x55, 0xAA, 0x08, 0x0A, 0x03, 0x17, 0x31, 0x31, 0x31,
0x31, 0x31, 0x31, 0xAD, 0xFE}; // Set The Scooter Pin to
"111111"
```

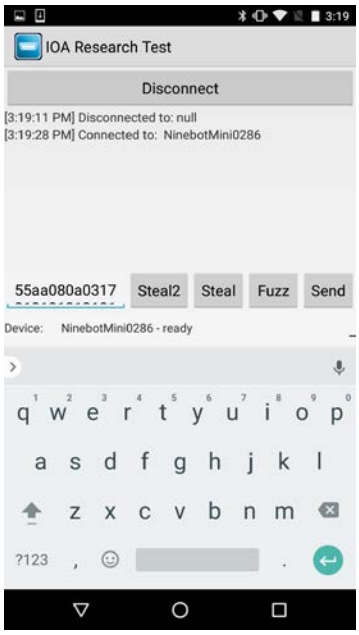



Figure 1 - Ninebot miniPRO PIN Theft

4. Using the newly changed PIN, an attacker can launch the Ninebot application and connect to the scooter. This would lock a normal user out of the Ninebot mobile application because a new PIN has been set.
5. An attacker can then upload an arbitrary firmware image to the scooter by DNS spoofing. By changing the A-Record for apptest.ninebot.cn, the attacker can direct the rider application to download any firmware image.
6. Next, The attacker can use the following process to notify the rider application that there is a firmware update available:
    a. On http://apptest.ninebot.cn, change the /appversion/appdownload/NinebotMini/version.json file to match the new firmware version and size. The example below forces the application to update the control/mainboard firmware image (aka Driver board firmware) to v1.3.3.7, which is 50212 bytes in size.

        ```
        “CtrlVersionCode":["1337","50212"]
        ```

    b. Create a matching directory and file including the malicious firmware /appversion/appdownload/NinebotMini/v1.3.3.7/Mini_Driver_v1.3.3.7.zip with the modified update file Mini_Driver_V1.3.3.7.bin compressed inside the firmware update archive.
7. When launched, the Ninebot application checks to see if the firmware version on the scooter matches the one downloaded from apptest.ninebot.cn. If there is a later version available (that is, if the version in the JSON object is newer than the version currently installed), the app triggers the firmware update process.

Analysis of Findings

Even though the Ninebot application prompted a user to enter a PIN when launched, it was not checked at a lower level before allowing the user to connect. This left the Bluetooth interface exposed to an attack at the protocol level. Additionally, since this device did not use Bluetooth encryption, communications could be wirelessly intercepted by an attacker.

Exposed management interfaces should not be available on a production device. An attacker may leverage an open management interface to execute privileged actions remotely. Due to the implementation in this scenario, IOActive was able to leverage this vulnerability and perform a firmware update of the scooter's control system without authentication.

Firmware integrity checking is imperative in embedded systems. Unverified or corrupted firmware images could permanently damage systems and may allow an attacker to cause unintended behavior. IOActive researchers were able to modify the controller firmware to remove rider detection, and may have been able to change configuration parameters in other onboard systems, such as the BMS (Battery Management System) and Bluetooth module.

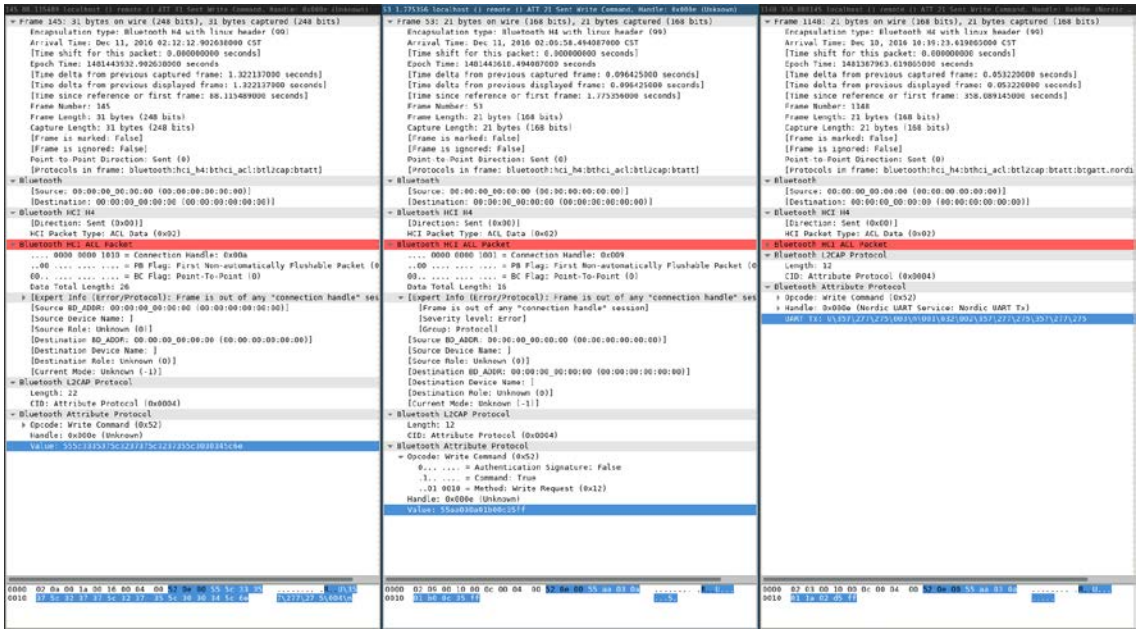


Figure 2 - Unencrypted Communications between Scooter and Rider Application

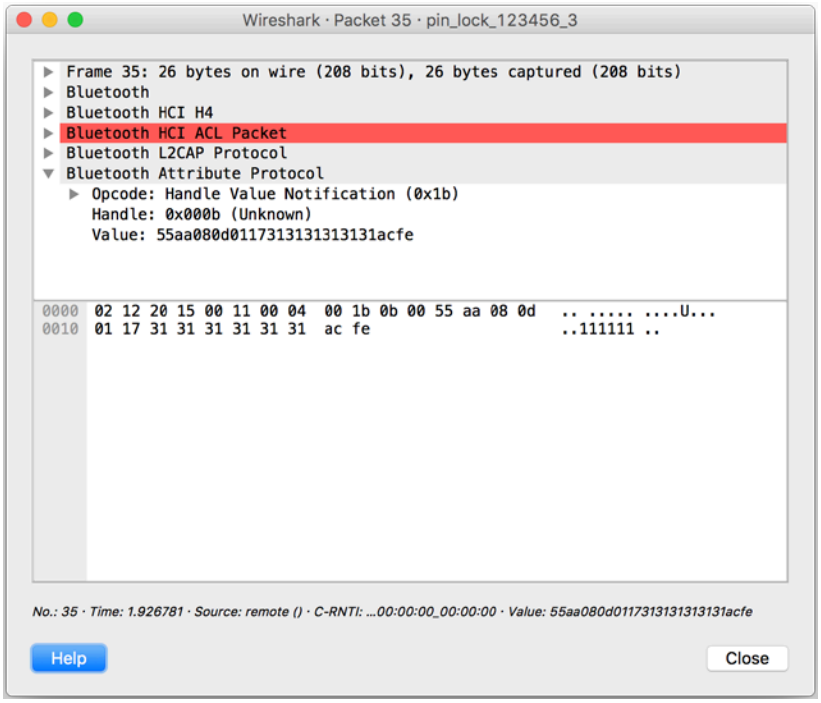


Figure 3 - Interception of the Rider Application Setting the Scooter PIN Code to "111111"

Mitigation

IOActive recommended the following mitigation for these vulnerabilities:

- Implement firmware integrity checking
- Use Bluetooth Pre-Shared Key authentication or PIN authentication
- Use strong encryption for wireless communications between the application and scooter
- Implement a "pairing mode" as the sole mode in which the scooter pairs over Bluetooth
- Protect rider privacy by not exposing rider location within the Ninebot mobile application

Fixes

After IOActive disclosed these vulnerabilities to Segway/Ninebot, the company subsequently reported remediation of the issues IOActive identified as critical.

Timeline

December 2016:	IOActive conducts testing on Segway/Ninebot MiniPRO scooter.
December 24, 2016:	IOActive contacts Segway/Ninebot via a public email address to establish a line of communication.
January 4, 2017:	Segway/Ninebot responds to IOActive.

January 27, 2017:	IOActive discloses issues to Segway/Ninebot.
April 2017:	Segway/Ninebot releases an updated application (3.20), which addresses some of IOActive's findings.
April 17, 2017:	Segway/Ninebot informs IOActive that remediation of critical issues is complete.
July 19, 2017:	Findings are published.

