Input to the Commission on Enhancing National Cybersecurity On behalf of the Free Software Foundation

Read online: https://www.fsf.org/news/free-software-foundation-stresses

-necessity-of-full-user-control-over-internet-connected-devices

Free Software Foundation stresses necessity of full user control over Internet-connected devices

by Matthew Garrett, FSF Board of Directors

The Internet of Things (IoT) refers to the integration of Internet technology into a wider range of home devices than previously envisaged by most users. Early adopters of IoT may now have homes with Internet- connected lightbulbs, alarm systems, baby monitors and even coffee machines. Internet integration allows owners to have greater flexibility over their devices, making it possible to turn on their air conditioning as they leave work to cool the house before they return, to have curtains that automatically close based on sunset time, or lights that automatically turn off after the owner has left the house.

Each individual benefit may seem marginal, but overall they add significant benefit to the owners.

Most IoT systems consist of three components:

1) The "smart" device itself, capable of communicating via a protocol such as Z-Wave, Zigbee, Bluetooth or IEEE 802.11, running either a full operating system (commonly based on the kernel Linux) or an embedded OS designed for this purpose.

2) A remote service provided by the device manufacturer. The smart device communicates with this service in order to provide information about its current state and in order to provide an interface for users to control the device.

3) An application designed for mobile platforms which interacts with the remote service and allows control of the smart device regardless of whether the user is currently located near the device or not.

Devices that use the Zigbee or Z-Wave protocols also typically require a local "hub," a device running interface software that bridges the devices to the remote service.

There are multiple significant security concerns around this design pattern. The first is that either the smart devices themselves or the hub that they communicate with require Internet access. Depending on local network configuration, this may result in the devices being visible to the public Internet. These devices inherently provide a service of some description in order to permit their integration with the remote services, but frequently also provide additional services for directly local communication and often include further unnecessary services used for diagnostics during the design and production stage (such as MicroCell -- the same backdoor was present on a series of baby monitors shipped by a major manufacturer).

These devices are often locked down in such a way that it is impossible for the user to replace the software that they run. These devices are also often abandoned by their manufacturers after a short space of time due to them being either discontinued or replaced by newer devices.

Users who continue using these devices are thus at significant risk, without any real chance of security updates being made available and frequently without any notification that any security issues have been identified. If any issues are identified, then without the permission of the manufacturer it is impossible for any third party to provide aid to said users.

This concern is frequently mitigated by typical home network setups that restrict external access to internal devices. But smart devices inherently require external access to be possible, and this functionality is provided by the remote service. The smart device connects to the remote service and awaits commands -- users in turn connect to the remote service and send commands.

These remote services are themselves frequently insecure.

Authentication details are often sent in plaintext, allowing anyone who can observe network traffic to obtain credentials. Some systems involve no authentication at all (for instance). This makes it possible for a malicious individual to gain control over home devices, in some cases potentially even being able to execute arbitrary code on said devices and gain access to the internal network.

If vendors are unwilling or unable to fix these security issues, users are left in an unfortunate position. They can either retain the convenience provided by the smart devices they paid for, or they can remove them and attempt to obtain a refund. The worst case scenario is perhaps when the vendor unilaterally decides to shut down the remote service, rendering the devices useless.

Another consideration is the behavior of the manufacturer itself.

Manufacturers may not always act in the interests of their customers, doing things ranging from invasive collection of personal data to intrusive advertising or even disabling device functionality remotely.

Even if ostensibly permitted by terms of service, users should be able to protect themselves against such scenarios.

There is an alternative. Third-party free software alternatives to the pre-installed software are common in certain market segments, such as home routers (libreCMC, OpenWrt and DD-WRT, for instance). Security vulnerabilities can be mitigated by replacing the original software with a functional equivalent provided by a third party. Unfortunately, many IoT devices are designed such that the software can only be replaced by the manufacturer. The software will only communicate with the manufacturer's remote service -- no third party can provide a functional equivalent.

To ensure that users do not end up in a situation where they are left choosing between security and convenience, or left with no ability whatsoever to use the devices they bought, it is vital that these devices be ultimately under the control of the user. The user should be able to replace the software on the device in order to fix security vulnerabilities. The user should be able to modify the software on the device such that it communicates with a different remote service that provides strong security guarantees. The user should not be left with no option other than to discard the device and replace it with a new version.

In order for this to be possible, it is necessary to know how the devices communicate with the remote server. Unfortunately this is frequently in the form of a proprietary protocol that lacks any public documentation, and as such it is a significant engineering effort for anyone to implement a replacement service. Several well-known protocols exist for controlling remote devices (such as MQTT) and re-using these rather than proprietary protocols makes it easier to both identify whether any security issues exist (being forced to reverse engineer a protocol may result in missing subtle aspects that cause security issues) and provide alternative implementations in the event of significant security flaws being discovered or the vendor choosing to cease support of the remote services.

To that end, we encourage the adoption of practices that:

a) Ensure that documented and freely-implementable (rather than patent-encumbered) protocols be used for communication between smart devices and remote services, and

b) Ensure that owners of smart devices are able to replace their software with implementations provided by either themselves or third parties in order to prevent the vendor being a single point of failure in either service

c) Strongly encourage the use of free "as in freedom" software throughout the entire stack, making it easier for security researchers to identify issues, third parties to provide alternative implementations and users to retain as much control as possible over devices that will become increasingly integrated into their homes and lives.

--

stephen mahood

outreach & communication

free software foundation

http://fsf.org http://gnu.org

GPG Key: 7CF9305D