Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: January 4, 2018

K. Moore
Network Heretics
R. Barnes
Mozilla
H. Tschofenig
ARM Limited

July 3, 2017

Best Current Practices for Securing Internet of Things (IoT) Devices
draft-moore-iot-security-bcp-01

Abstract

In recent years, embedded computing devices have increasingly been provided with Internet interfaces, and the typically-weak network security of such devices has become a challenge for the Internet infrastructure. This document lists a number of minimum requirements that vendors of Internet of Things (IoT) devices need to take into account during development and when producing firmware updates, in order to reduce the frequency and severity of security incidents in which such devices are implicated.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

Table of Contents

1. Introduction 3
1.1. Terminology 4
1.2. Note about version -01 of this document 5
2. Design Considerations 5
2.1. General security design considerations 5
2.1.1. Threat analysis 6
2.1.2. Use of Standard Cryptographic Algorithms 6
2.1.3. Use of Standard Security Protocols 7
2.1.4. Security protocols should support algorithm agility . 7
2.2. Authentication requirements 7
2.2.1. Resistance to keyspace-searching attacks 7
2.2.2. Protection of authentication credentials 8
2.2.3. Resistance to authentication DoS attacks 8
2.2.4. Unauthenticated device use disabled by default . . . 8
2.2.5. Per-device unique authentication credentials 8
2.3. Encryption Requirements 9
2.3.1. Encryption should be supported 9
2.3.2. Encryption of traffic should be the default 9
2.3.3. Encryption algorithm strength 9
2.3.4. Man in the middle attack 9
2.4. Firmware Updates 9
2.4.1. Automatic update capability 9
2.4.2. Enable automatic firmware update by default 10
2.4.3. Backward compatibility of firmware updates 10
2.4.4. Automatic updates should be phased in 10
2.4.5. Authentication of firmware updates 10
2.5. Private key management 10
2.6. Operating system features 11
2.6.1. Use of memory compartmentalization 11
2.6.2. Privilege minimization 11
2.7. Miscellaneous 11
3. Implementation Considerations 11
3.1. Randomness 11
4. Firmware Development Practices 12
5. Documentation and Support Practices 12
5.1. Support Commitment 12
5.2. Bug Reporting 13
5.3. Labeling 13
5.4. Documentation 13

6. Security Considerations 13
7. IANA Considerations 13
8. Acknowledgements 14
9. References 14
9.1. Normative References 14
9.2. Informative References 14
Authors' Addresses 15

1. Introduction

The weak security of Internet of Things devices has resulted in many well-publicized security incidents over the last few years. Unfortunately, it appears that very few lessons have been learned from those incidents. The rate at which IoT devices are compromised via network-based attacks appears to be increasing. The effect of such security breaches goes far beyond the immediate effect on the compromised devices and their users. A compromised device may, for example, expose to an attacker secrets (such as passwords) stored in the device. A compromised device also may be used to attack other computers on the same local network as the device, or elsewhere on the Internet. Attackers have constructed application networks of compromised devices which have then been used for the purpose of attacking other network hosts and services, for example distributed password guessing attacks and distributed denial of service (DDoS) attacks. [SNMP-DDOS][DDOS-KREBS] This document recommends a small number of minimum security requirements to reduce some of the more easily prevented security problems.

The scope of these recommendations is as follows:

- These measures described in this document are intended to impede network-based attacks. These measures are not intended to impede other kinds of attacks, e.g. those requiring physical access to the device, though following these requirements may help reduce the effectiveness of some such attacks. This document does not address physical attacks because thwarting such attacks is generally outside of IETF's expertise, and because it is understood that the physical security requirements of Internet-connected devices vary widely from one application to another. However, because a device compromised by physical means may be used to attack other devices or to obtain information that useful in attacking other devices, it is strongly recommended that vendors of Internet-connected devices carefully consider physical security requirements when designing their products.

- In principle these requirements apply to all hosts that connect to the Internet, but this list of requirements is specifically targeted at devices that are constrained in their capabilities,

more than general-purpose programmable hosts (PCs, servers, laptops, tablets, etc.), routers, middleboxes, etc. While this is a fuzzy boundary, it reflects the current understanding of IoT. A more detailed treatment of some of the constraints of IoT devices can be found in [RFC7228].

- These are MINIMUM requirements that apply to all devices. They are unlikely to be sufficient by themselves, to ensure security of hosts from attack. Because IoT devices are used in a large number of different domains with different needs, each device will have its own unique security considerations. It is not feasible to completely list all security requirements in a document such as this. Vendors should conduct threat assessments of each device they produce, to determine which additional security considerations are applicable for use in a given application domain.

- It is expected that this list of requirements will be revised from time to time, as new threats are identified, and/or new security techniques become feasible.

- This document makes broad recommendations, but avoids recommending specific technological solutions to security issues. This is because there is a wide variety of IoT devices with a wide variety of use cases and threat scenarios, so there are few one-size-fits-all technological solutions. A companion document may be produced with suggestions for design choices and implementations that may aid in meeting these requirements.

We expect that many of the requirements can easily be met by most vendors, but may require additional documentation and transparency of a vendor’s development practices to improve credibility of their security practices in the marketplace.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119]. These key words describe normative requirements of this specification. This specification also contains non-normative recommendations that do not use these key words.

This document uses the term "firmware" to refer to the executable code and associated data that, in combination with device hardware, implements the functionality of an Internet- connected device. Traditionally the term "firmware" refers to code and data stored in non-volatile memory as distinguished from "software" which presumably

refers to code stored in read/write or erasable memory, or code that can be loaded from other devices. For the purpose of this document, "firmware" applies to any kind of code or data that implements the functions that the device provides. Both software and firmware present similar issues regarding device security, and it is easier to use "firmware" consistently than to write "software and firmware".

1.2. Note about version -01 of this document

The goal for the initial versions of this document is to invite discussion about what minimum security standards for Internet-connected devices are appropriate. Consequently, this draft suggests a wide range of potential measures. The authors, however, understand that imposing too many barriers to adoption might discourage device manufacturers from attempting to comply with this standard. We seek to find the right balance that helps improve the security of the Internet. We understand that some of the requirements in this draft may need to be removed or relaxed, at least in an initial version of a BCP document, and that other requirements may require additional refinement and justification.

2. Design Considerations

This section lists requirements and considerations that should affect the design of an Internet-connected device. Broadly speaking, such considerations include device architecture, hardware and firmware component choices, partitioning of function, design and/or choice of protocols used to communicate with the device.

2.1. General security design considerations

In general an Internet connected device should:

- Protect itself from attacks that impair its function or allow it to be used for unintended purposes, without authorization;

- Protect its private authentication credentials and key material from being compromised resulting in disclosure to unauthorized parties;

- Protect the information received from the device, transmitted from the device, or stored on the device, from inappropriate disclosure to unauthorized parties; and

- Protect itself from being used as a vector to attack other devices or hosts on the Internet.

- When appropriate, protect itself from communications with unauthorized/unauthenticated parties or devices.

Each device is responsible for its own security and for ensuring that it is not used as a vector for attack on other Internet hosts. The design of a device MUST NOT assume that a firewall or other perimeter security measure will protect the device from attack. While useful as part of a layered defense strategy, perimeter security has consistently been demonstrated to be insufficient to thwart attacks by itself. There are nearly always mechanisms by which one or more hosts on the local network can be compromised, and which in turn can serve as a means to attack other hosts. Perimeter security mechanisms also cannot distinguish hostile traffic from safe traffic with 100% reliability. And even devices on "air gapped" networks have been compromised by portable storage devices or software updates.

For some kinds of attack, there is a limited amount that a device can do to prevent the attack. For instance, any device can fall victim to certain kinds of denial-of-service attack caused by receiving more traffic in a given amount of time than the device can process. A device SHOULD be designed to gracefully tolerate some amount of excessive traffic without failing entirely, but at some point the device receives so much traffic that it cannot distinguish valid requests from invalid ones.

2.1.1. Threat analysis

The design for a device MUST enumerate specific security threats considered in its design, and the specific measures taken (if any) to remedy or limit the effect of each threat. This requirement encourages making deliberate, explicit choices about security measures at design time rather than leaving security as an afterthought. This document is also useful later in the life cycle of a device if it becomes necessary to improve security; for instance it can help identify whether the original design choices fulfilled their intended function or failed to do so, or whether a newly discovered threat was not anticipated in the original design.

2.1.2. Use of Standard Cryptographic Algorithms

Standard or well-established, mature algorithms for cryptographic functions (such as symmetric encryption, public-key encryption, digital signatures, cryptographic hash / message integrity check) MUST be used.

Explanation: A tremendous amount of subtlety must be understood in order to construct cryptographic algorithms that are resistant to

attack. A very few people in the world have the knowledge required to construct or analyze robust new cryptographic algorithms, and even then, many knowledgeable people have constructed algorithms that were found to be flawed within a short time.

2.1.3. Use of Standard Security Protocols

Standard protocols for authentication, encryption, and other means of assuring security SHOULD be used whenever apparently-robust, applicable protocols exist.

Explanation: The amount of expertise required to design robust security protocols is comparable to that required to design robust cryptographic algorithms. However, there are sometimes use cases for which no existing standard protocol may be suitable. In these cases it may be necessary to adapt an existing protocol for a new use case, or even to design a new security protocol.

2.1.4. Security protocols should support algorithm agility

The security protocols chosen for a device design, and the implementations of those protocols, SHOULD support the ability to choose between multiple cryptographic algorithms and/or to negotiate minimum key sizes.

Explanation: This way, if a flaw in one algorithm is discovered that weakens its security, updated devices or their application peers with which they communicate, may refuse to use that algorithm, or permit its use only with a longer key than originally required. This allows devices and protocol implementations to continue providing adequate security even after weaknesses in algorithms are discovered.

The concept of crypto agility is further described in [RFC7696].

2.2. Authentication requirements

The vast majority of Internet-connected devices will require authentication for some purposes, whether to protect the device from unauthorized use or reconfiguration, and to protect information stored within the device from disclosure or modification. This section details authentication requirements for devices that require authentication.

2.2.1. Resistance to keyspace-searching attacks

A device that requires authentication MUST be designed to make brute-force authentication attacks, dictionary attacks, or other attacks

that involve exhaustive searching of the device's key or password space, infeasible.

2.2.2. Protection of authentication credentials

A device MUST be designed to protect any secrets used to authenticate to the device (such as passwords or private keys) from disclosure via monitoring of network traffic to or from the device. For example, if a password is used to authenticate a client to the device, that password must not appear "in the clear", or in any form via which extraction of the password from network traffic is computationally feasible.

2.2.3. Resistance to authentication DoS attacks

A device SHOULD be designed to gracefully tolerate excessive numbers of authentication attempts, for instance by giving CPU priority to existing protocol sessions that have already successfully authenticated, limiting the number of concurrent new sessions in the process of authenticating, and randomly discarding attempts to establish new sessions beyond that limit. The specific mechanism is a design choice to be made in light of the specific function of the device and the protocols used by the device. What's important for this requirement is that this be an explicit choice.

2.2.4. Unauthenticated device use disabled by default

A device that supports authentication SHOULD NOT be shipped in a condition that allows an unauthenticated client to use any function of the device that requires authentication, or to change that device's authentication credentials.

Explanation: Most devices that can be used in an unauthenticated state will never be configured to require authentication. These devices are attractive targets for attack and compromise, especially by botnets. This is very similar to the problems caused by shipping devices with default passwords.

2.2.5. Per-device unique authentication credentials

Many devices that require authentication will be shipped with default authentication credentials, so that the customer can authenticate to the device using those credentials until they are changed. Each device that requires authentication SHOULD be instantiated either prior to shipping, or on initial configuration by the user, with credentials unique to that device. If a device is not instantiated with device-unique credentials, that device MUST NOT permit normal

operation until those credentials have been changed to something other than the default credentials.

Explanation: devices that were shipped with default passwords have been implicated in several serious denial-of-service attacks on widely-used Internet services.

2.3. Encryption Requirements

2.3.1. Encryption should be supported

Internet-connected devices SHOULD support the capability to encrypt traffic sent to or from the device. Any information transmitted over a network is potentially sensitive to some customers. For example, even a home temperature monitoring sensor may reveal information about when occupants are away from home, when they wake up and when they go to bed, when and how often they cook meals - all of which are useful to, say, a thief.

Note: This requirement is separate from the requirement to protect authentication secrets from disclosure. Authentication secrets MUST be protected from disclosure even if a general encryption capability is not supported, or if the capability is optional and a particular client or user doesn't use it.

2.3.2. Encryption of traffic should be the default

If a device supports encryption and use of encryption is optional, the device SHOULD be configurable to require encryption, and this SHOULD be the default.

2.3.3. Encryption algorithm strength

Encryption algorithms and minimum key lengths SHOULD be chosen to make brute-force attack infeasible.

2.3.4. Man in the middle attack

Encryption protocols SHOULD be resistant to man-in-the-middle attack.

2.4. Firmware Updates

2.4.1. Automatic update capability

Vendors MUST offer an automatic firmware update mechanism. A discussion about the firmware update mechanisms can be found in [I-D.iab-iotsu-workshop].

Devices SHOULD be configured to check for the existence of firmware updates at frequent but irregular intervals.

2.4.2. Enable automatic firmware update by default

Automatic firmware updates SHOULD be enabled by default. A device MAY offer an option to disable automatic firmware updates.

Especially for any device for which a firmware update would disrupt operation, the device SHOULD be configurable to allow the operator to control the timing of firmware updates.

If enabling or disabling or changing the timing of the automatic update feature is controlled by a network protocol, the device MUST require authentication of any request to control those features.

2.4.3. Backward compatibility of firmware updates

Automatic firmware updates SHOULD NOT change network protocol interfaces in any way that is incompatible with previous versions. A vendor MAY offer firmware updates which add new features as long as those updates are not automatically initiated.

2.4.4. Automatic updates should be phased in

To prevent widespread simultaneous failure of all instances of a particular kind of device due to a bug in a new firmware release, automatic firmware updates SHOULD be phased-in over a short time interval rather than updating all devices at once.

2.4.5. Authentication of firmware updates

Firmware updates MUST be authenticated and the integrity of such updates assured before the update is installed. Unauthenticated updates or updates where the authentication or integrity checking fails MUST be rejected.

Firmware updates SHOULD be authenticated using digital signature items that use public key cryptography to verify the authenticity of the signer. Ordinary checksums or hash algorithms are insufficient by themselves, and keyed hashes that use shared secrets are generally discoverable by a determined attacker.

2.5. Private key management

If public key cryptography is used by the device to authenticate itself to other devices or parties, each device MUST be instantiated with its own unique private key or keys. In many cases it will be

necessary for the vendor to sign such keys or arrange for them to be signed by a trusted party, prior to shipping the device.

Per-device private keys SHOULD be generated on the device and never exposed outside the device.

2.6. Operating system features

2.6.1. Use of memory compartmentalization

Device firmware SHOULD be designed to use hardware and operating systems that implement memory compartmentalization techniques, in order to prevent read, write, and/or execute access to areas of memory by processes not authorized to use those areas for those purposes.

Vendors that do not make use of such features MUST document their design rationale.

Explanation: Such mechanisms, when properly used, reduce the impact of a firmware bug, such as a buffer overflow vulnerability. Operating systems, or even firmware running on "bare metal", that do not provide such a separation allow an attacker to gain access to the complete address space. While these concepts have been available in hardware for a long time already, they often are not utilized by real-time operating systems.

2.6.2. Privilege minimization

Device firmware SHOULD be designed to isolate privileged code and data from portions of the firmware that do not need to access them, in order to minimize the potential for compromised code to access those code and/or data.

2.7. Miscellaneous

3. Implementation Considerations

This section lists requirements for implementation that broadly affect security of a device.

3.1. Randomness

Vendors MUST include a solution for generating cryptographic quality random numbers in their products. Randomness is an important component in security protocols and without such randomness many of today’s security protocols offer weak or no security protection.

Hardware random-number generators, when feasible, SHOULD be utilized, but MAY be combined with other sources of randomness.

A discussion about randomness can be found in [RFC4086].

4. Firmware Development Practices

This section outlines requirements for development of firmware that is employed on Internet-connected devices.

Vendors SHOULD use modern firmware development practices, including:

- Source code change control systems, which record all changes made to source code along with the identity of the person who committed the change. Such systems help to identify which versions of code contain a particular bug, as well as protect against insertion of malicious code.

- Bug tracking systems.

- Automated testing of a set of pre-defined test conditions, including tests for all security vulnerabilities identified to date via either analysis or experience.

- Periodic checking of bug databases for reported security issues associated with the product itself, and with all components (for example: kernel, libraries, and protocol servers) used in the product.

- Whenever feasible, checking externally-provided source code and object code for authenticity.

- Periodic checking of externally-provided source code and object code for known security bugs, or updates intended to thwart security bugs.

All known security bugs for which fixes or workarounds are known MUST be addressed prior to shipping a new product or or a code update.

5. Documentation and Support Practices

5.1. Support Commitment

Vendors MUST be transparent about their commitment to supply devices with updates before selling products to their customers and what happens with those devices after the support period finishes.

Within the support period, vendors SHOULD provide firmware updates whenever new security risks associated with their products are identified. Such firmware updates SHOULD NOT change the protocol interfaces to those products, except as necessary to address security issues, so that they can be deployed without disruption to customers' networks. Firmware updates MAY introduce new features which change protocol interfaces if those features are optional and disabled by default.

5.2. Bug Reporting

Vendors MUST provide an easy to find way for reporting of security bugs, which is free of charge.

5.3. Labeling

Vendors MUST have a manufacturer, model number and hardware revision number legibly printed on the device. This attempts to help customers with bug reporting.

There SHOULD be a documented means of querying a device for its model number, hardware revision number, and firmware revision number via its network interface and/or via any manual input and display. This interface MAY require authentication.

5.4. Documentation

Vendors MUST offer documentation about their products so that security experts are able to assess the design choices. While such a document will not directly help end customers since they will almost always lack the expertise to judge these design decisions but they help security experts to assess liability and independent third parties to compare products without spending an disproportional amount of time.

This form of public documentation will help transparency similar to other documentation requirements found in other industries. It will also help to evolve the best practices described in this document.

6. Security Considerations

This entire document is about security.

7. IANA Considerations

This document does not contain any requests to IANA.

8. Acknowledgements

Add acknowledgments here.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

9.2. Informative References

[DDOS-KREBS] Goodin, D., "Record-breaking DDoS reportedly delivered by >145k hacked cameras", September 2016, <http://arstechnica.com/security/2016/09/botnet-of-145k-cameras-reportedly-deliver-internets-biggest-ddos-ever/>.

[I-D.iab-iotsu-workshop] Tschofenig, H. and S. Farrell, "Report from the Internet of Things (IoT) Software Update (IoTSU) Workshop 2016", draft-iab-iotsu-workshop-00 (work in progress), October 2016.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <http://www.rfc-editor.org/info/rfc7228>.

[RFC7696] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, DOI 10.17487/RFC7696, November 2015, <http://www.rfc-editor.org/info/rfc7696>.

[SNMP-DDOS] BITAG, "SNMP Reflected Amplification DDoS Attack Mitigation", August 2012, <https://www.bitag.org/documents/SNMP-Reflected-Amplification-DDoS-Attack-Mitigation.pdf>.

Authors' Addresses

Keith Moore
Network Heretics
PO Box 1934
Knoxville, TN 37901
United States

EMail: moore@network-heretics.com

Richard Barnes
Mozilla

EMail: rbarnes@mozilla.com

Hannes Tschofenig
ARM Limited

EMail: hannes.tschofenig@gmx.net