Internet Engineering Task Force (IETF)
Internet-Draft
Intended Status: Standards Track
Expires: April 19, 2015

Phillip Hallam-Baker
Comodo Group Inc.

October 16, 2014

Software and Configuration Management
draft-hallambaker-securecode-00

Abstract

Use cases and requirements for secure distribution and management of code are considered. In particular constraints imposed by embedded devices that do not provide affordances for user interaction are considered.

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."

Table of Contents

1. Motivation

All computing devices operate under control of software. As the applications of computing systems diversify, the management of the software code determining their function becomes an increasingly difficult challenge.

The recognition of this problem in the dektop computing environment and the introduction of automatic update mechanisms has played a major role in reducing vulnerabilities on these platforms. But no similar platforms have emerged to support the proliferation of embedded devices currently occuring.

While there is a clear need for a software management infrastructure that meets these emerging needs, it is essential that any new security infrastructure meet all the security risks of users and device owners, including the very real possibility that the device vendors themselves might pose the threat. The only difference between a voice activated, network conected toaster oven and a covert surveillance device (aka bug) is the software it runs.

The possibility that a device might change its function through an unsolicited software is just as important a security concern as the possibility that code might contain zero day vulnerabilities.

Existing systems, including those deployed to update open source platforms have been developed to serve the proprietary interest of the software provider to update their code. The disregard for the concerns of the user/owner is made clear by a user interaction where the only choice is to update now or to be reminded in an hour's time.

Rather unusually, devices sold to the consumer market tend to be considerably worse than those sold to enterprises. Enterprises understand that automatic software updates present security risks as well as benefits. A software update mechanism that is outside their direct control may invalidate previous Quality Assurance processes causing a system to fail.

Since automatic update mechanisms rarely receive attention in product reviews, the needs of consumers have been treated with conspicuous contempt. In the majority of cases, no attempt is made to minimize the inconvenience to the user. The device determines that an update is required and refuses to perform its intended function until the user approves installation of the update, the update is downloaded and installed.

Approaches such as this are at best discourteous but can pose a serious safety risk if they interfere with the functioning of critical systems. While the manufacturer of a defibrilating device is

likely to understand that it must work immediately every time it is used, most systems become critical because people rely on them to

function rather than this being an intrinsic aspect of their purpose.

2. Definitions

2.1. Software

The term 'software' is used to describe any content that might affect the operation of a device that is not provided by its administrator and/or user(s).

Rationale: What is important from the security point of view is that the content might affect the operation of the machine rather than the classification of the content as 'code' or 'data'. A data driven code system need not be Turing complete for it present a user-access or root-access level vulnerability.

Note that for the purposes of software management, there is no useful distinction between 'software' and 'firmware'. Either may affect the operation of the machine.

2.2. Configuration

The term 'configuration' is used to describe any content that might affect the operation of a device that is provided by its administrator and/or user(s).

Rationale: Anything that is not software that might affect the operation of the device is a security concern and thus should be considered in the device management infrastructure.

3. Principles

* The integrity of the system software and configuration should always be assured.

* The operation of a system should be controlled by the owner. No modifications should be made to the operation of a device without express permission from the owner or their delegate.

* If the owner's ability to modify either software or configuration is limited in any fashion, these constrains should be clearly declared.

* The software and configuration of a system should always be known with certainty.

* Changes to software and/or configuration should be auditable and reversible.

Note that the requirement for express permission does not entail specific user interaction on the part of the owner. Since most owners have neither the means nor the inclination to decide whether an update should be performed, the ability to delegate decision making powers to the software provider or a third party is highly desirable provided that such delegation is revokable and the exercise of the delegated powers can be audited.

4. Requirements

Consumer software update systems are generally implemented as a feature of the system under management while enterprise software management systems typically observe a separation between the administration system and the systems under management.

4.1. Device Requirements

* Report the software packages installed on a system.
* Transfer a software package to a system.
* Install a software package on a system.
* Delete a software package from a system.
* Change the software package version to be executed on a system at next restart.
* Change the software package version to be executed now.
* Schedule a restart.

4.2. Administration Requirements

* Determine what vulnerabilities have been reported for a software package.
* Verify the provenance of a software package.

4.3. Interface Requirements

* Bind a device to an administration source.

5. Technical Requirements

6. Acnowledgementsts

This document was written in response to discussion on the IETF list begun by Jim Gettys.

Author's Address

Phillip Hallam-Baker
Comodo Group Inc.

philliph@comodo.com