Technical Disclosure Commons

Defensive Publications Series

December 21, 2018

ADVERTISING SOFTWARE/SECURITY USAGE DESCRIPTIONS WITH POLICY RESPONSE

Eliot Lear

Brian Weis

Chris Steck

Nancy Cam-Winget

ADVERTISING SOFTWARE/SECURITY USAGE DESCRIPTIONS WITH POLICY RESPONSE

AUTHORS:
Eliot Lear
Brian Weis
Chris Steck
Nancy Cam-Winget

ABSTRACT

Techniques are provided for an organization-maintained server which takes three inputs: (1) a set of Uniform Resource Identifiers (URIs) from Internet of Things (IoT) devices, each of which point to a manifest; (2) a set of manifests resolved from the URIs; and (3) a set of threat feeds. The server periodically compares the vulnerabilities in the threat feeds to the manifests. When a vulnerability is found, steps are taken to protect the rest of the network from the vulnerable devices until they can be remediated.

DETAILED DESCRIPTION

Vulnerabilities in software are typically associated with a software package (e.g., OpenSSL) and the version of that package. It is important for enterprise administrators to know which packages are installed on which devices for purposes of remediating risk tied to those vulnerabilities. Given a particular version, an action or response may be required when a vulnerability is discovered.

Methods exist for maintaining software manifests for complex network devices, such as Personal Computers (PCs). But no such method has been defined for Internet of Things (IoT) devices, where the focus has been on reporting a level of firmware running on those devices. The reality is that many IoT devices will have third-party software packages included as part of their firmware or software. Some IoT devices are based on a minimal Linux implementation that includes a number of open source packages.

While complex network devices such as PCs may have the resources available to store and/or maintain a manifest, the IoT device probably does not. Likewise, even though PCs report to a posture assessment server what it contains, an IoT device probably does not have that ability. So traditional software inventory management and posture assessment methods are not generally applicable to IoT devices.

Published by Technical Disclosure Commons, 2018

An IoT device generally will not have the capability of responding to a posture assessment server. Still, it is valuable for software inventory management and posture assessment servers to be able to understand what source packages are used on it in order to determine if an IoT device is at risk.

Additionally, there may be some security-related attributes about the device that would be valuable for a network administrator to know (e.g., expected security strength and capabilities of the IoT device such that finer access controls can be enabled on them).

It is envisioned that end users will not have the ability to install random software on IoT devices, and that in general they will treat whatever software is installed as a single monolithic system. The entity with the best knowledge of the installed software is the manufacturer. The manufacturer would create and make available a manifest that describes the version of the current software running on the IoT device, but more importantly lists the software packages and their versions that would be used. It would also list other security related information such as what kinds of security measures are provided (e.g., does it have a Trusted Platform Module (TPM) / Trusted Execution Environment (TEE), does it have secure boot, etc.).

Figure 1 below illustrates a method that can be used by the owner of a device to mitigate known software vulnerabilities in an IoT device. The server managing manifests may be a posture assessment server, or software inventory management or other server that is monitoring a network and implementing a policy.

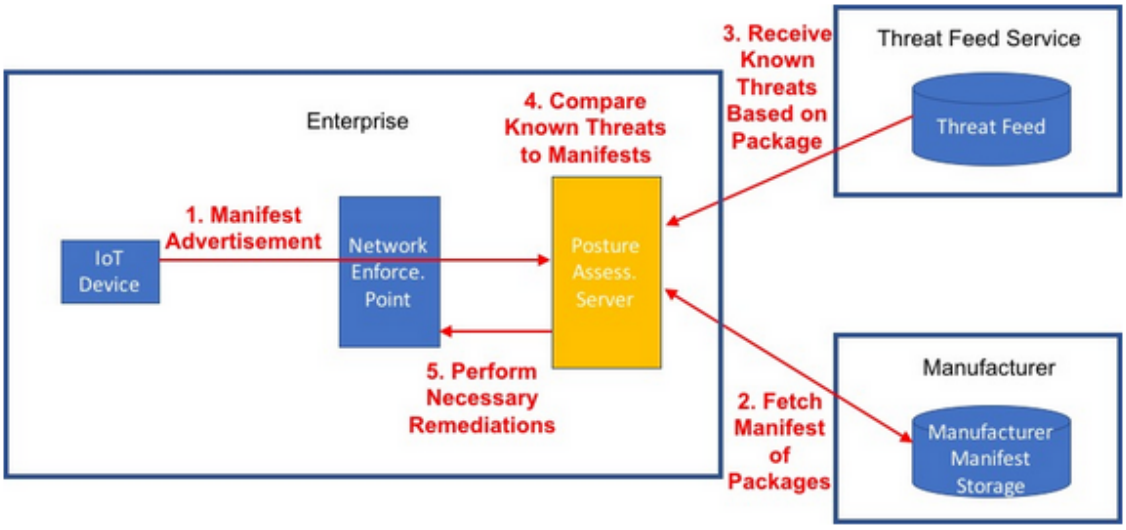

Figure 1

At 1, the device manufacturer advertises a capability (e.g., in the form of a Uniform Resource Identifiers (URI)) to an interface to retrieve a manifest. The location of the manifest could be included as a data element in a Manufacturer Usage Description (MUD) file. The URI could point to a static file on a server, or it could refer to an Open Mobile Alliance (OMA) Lightweight Machine-to-Machine (LwM2M) Server or Security Automation and Continuous Monitoring (SACM) protocol. A device manufacturer that wishes to restrict the information could require authorization of the posture assessment server based on configuration information received via a certificate extension that identifies a certificate for these purposes. The URI could even point to an interface available on the device itself which would return the information (e.g., the Common Industrial Protocol (CIP) getIdentity() message).

At 2, a device owner receives knowledge of an IoT device manifest (e.g., by observing it in a MUD file to which an IoT device has advertised). It fetches the manifest, verifies that the manufacturer signed it, and stores it in its posture assessment server.

At 3, the posture assessment server receives feeds that indicate new vulnerabilities in various open software packages (e.g., from the National Vulnerability Database at https://nvd.nist.gov/vuln/data-feeds). In an alternate manifestation, a MUD extension might point to one or more vendor feeds of the same information via Managed Incident Lightweight Exchange (MILE) or some other protocol.

At 4, the posture assessment server compares the IoT device manifest that it has fetched to the information in the known vulnerability feeds, and determines a policy based on the criticality of the device, the number of impacted devices, the age of the vulnerability, and the criticality of the vulnerability severity scores.

At 5, the posture assessment server implements the policy. This may be anything from ignoring the problem to shutting a system down to waking someone up (e.g., via ServiceNow support queues). In some cases, such as devices that cannot be upgraded, the remediation taken may become a permanent policy to provide additional monitoring or security to compensate for the package vulnerability.

The manifest should be tied to a particular version of firmware that the manufacturer releases for the device, or else include a cache validity field as part of the

manifest to alert the posture assessment server to periodically re-validate the contents of the manifest.

The use of a URI allows for the manifest to be fetched in a number of ways. In one example, it could be a Uniform Resource Locator (URL) to a manufacture-provided server. In another example, it could be "posture.sacm:" where the local controller knows how to get to the local SACM server (Software Inventory Message and Attributes Posture Collector (SWIMA-PV)) to ask it for a manifest received from the device through the Network Access Control (NAC) / SACM protocol. In yet another example, it could be "posture.cip" that refers to the device itself, where the posture server or MUD manager would understand that the CIP protocol can be used to obtain a manifest from the device itself.

The device manufacturer may advertise a URI by which a network device can retrieve a manifest. Automatic vulnerability analysis of a new IoT device on the network may be performed by a posture assessment server. The analysis is based on the manifest of software packages and security attributes that the device manufacturer makes available in one of several ways. When a concern is found, a network administrator can immediately put remediations into effect. Automatic identification of vendor vulnerability information is also possible (when the vendor wants to offer it).

IoT is the typical case where the manufacturer is able to provide reliable information about the software and security attributes of a device. For IoT devices, the software is relatively static and is generally sourced from the manufacturer. However, the solution described herein may also apply to other limited function devices such as network switches and routers (including ruggedized routers used for IoT) that have a limited set of software packages.

The manifest advertisement may be trusted (e.g., not provided by an attacker) based on two levels of protection. For the first level, the advertisement is expected to be a URI included in a MUD file, signed by the manufacturer. The second level depends on the URI itself. If the URI is "https:" (i.e., as a web page), then the manifest itself is expected to be signed by the manufacturer. If the URI describes a local controller or the device itself (e.g., "posture.sacm" or "posture.cip") then the posture protocol itself would include an integrity method for its data.

A URI that could point to one of many useful methods of describing information about the device may be embedded. The URI approach allows for a wider variety of information to the considered, and for a wider variety of methods by which device information may be obtained. Furthermore, the process described herein is intended to be an iterative posture assessment process used over the lifetime of the device, which reacts to newly announced vulnerabilities. This is a significant improvement to statically examining a list of network services available on the device.

In summary, techniques are provided for an organization-maintained server which takes three inputs: (1) a set of URIs from IoT devices, each of which point to a manifest; (2) a set of manifests resolved from the URIs; and (3) a set of threat feeds. The server periodically compares the vulnerabilities in the threat feeds to the manifests. When a vulnerability is found, steps are taken to protect the rest of the network from the vulnerable devices until they can be remediated.