FOG BASED ATTESTATION - INLINE DATAPLANE BASED INTEGRITY VALIDATION AND EVENT BASED ATTESTATION QUERY

An IP.com Prior Art Database Technical Disclosure

Authors et. al.: Jeff Apcar
Carlos M. Pignataro
Nagendra Kumar Nainar
Omar Santos

IP.com Number: IPCOM000245102D
IP.com Electronic Publication Date: February 08, 2016

FOG BASED ATTESTATION - INLINE DATAPLANE BASED INTEGRITY VALIDATION AND EVENT BASED ATTESTATION QUERY

AUTHORS:

Carlos M. Pignataro
Nagendra Kumar Nainar
Omar Santos
Jeff Apcar

CISCO SYSTEMS, INC.

ABSTRACT

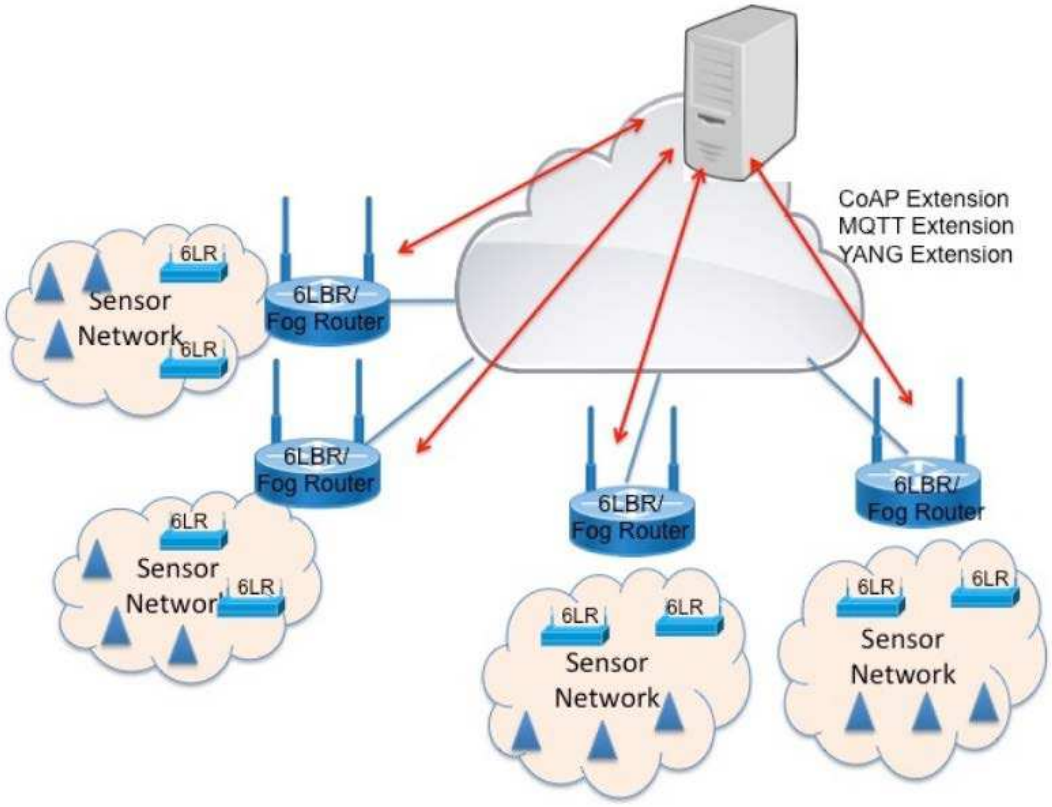
A process is presented that uses extensions to offload a hashing value for sensors to edge nodes. The hashing value can be included in an IPv6 Extension Header (in all packets, few packets, on-demand/request based). Any deviation in hashing is detected, and an instruction may be sent to a server using various extensions to trigger integrity/attestation validation. This provides an event-based integrity validation trigger from server to sensor. Until the integrity validation is completed, all the packets from the sensors may be dropped. Thus, a dataplane-based integrity validation process is provided that achieves faster abnormality detection and prevents the falsified data from affecting the service.

DETAILED DESCRIPTION

Currently, there are different ways of checking the integrity of the IoT sensors. Examples include using a Trusted Platform Module (TPM), which is a hardware-based approach, and a memory filling technique, which is a software-based approach. Another approach uses any network protocol and any transport technology for interval-based polling, such as an out-of-band Low Power Wide Area Network (LPWAN) or LoRa™ techniques. In each of those approaches, the validation is based generally on a periodic timer based query. This may lead to a time interval where the data can be forged/compromised until the next query is triggered for validation.

In the above-mentioned prior approaches, it is observed that they are based on a periodic OOB mechanism that is triggered to validate the integrity of the device and mark it as untrusted if required. In such scenarios, if a sensor is compromised and loaded with malicious code, it might affect other sensors or will provide falsified data until the next iteration of integrity validation triggered from the server. Presented herein are techniques to avoid this potential intrusion gap by offloading a hashing value and sensor information to the edge nodes and performing a dataplane-based validation to detect any anomaly. That is, the sensors are given / calculate the hashing value as well.

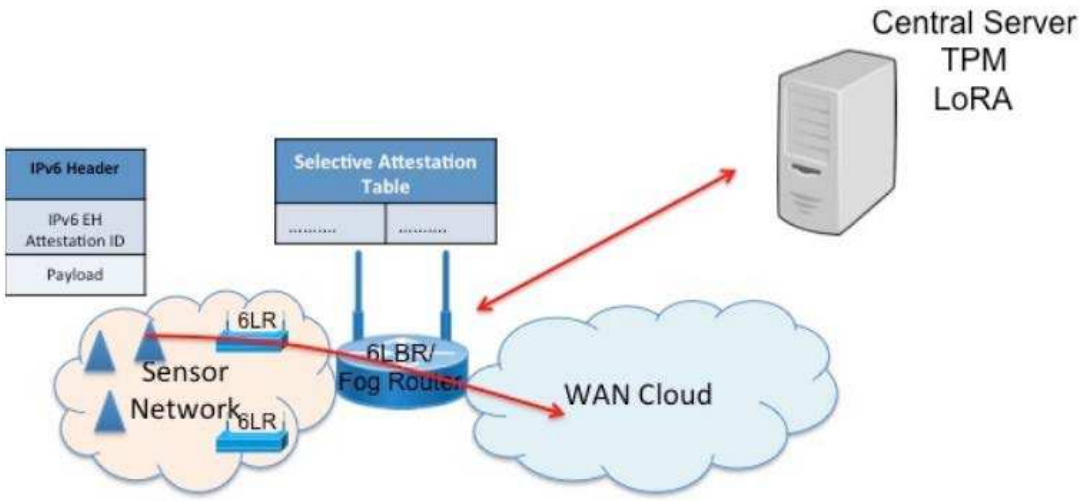
FIG. 1



In the topology shown in FIG. 1, assume the sensors use any of those OOB mechanism defined in the prior approaches. Once the integrity validation on the hash value (from the program memory, MMU) is done, the hash value will be downloaded to the border nodes (Fog routers, 6LBR) using, for example, protocols such as Constrained Application Protocol (CoAP), MQ Telemetry Transport (MQTT), the YANG extension

(RFC 6020) and the Routing Protocol for Low-Power and Lossy Networks (RPL). The data offload is done selectively so that information relevant only to those set of PAN/Mesh ID is downloaded to the Fog routers. In general, distribution of the hash value can be constrained to a specific "network domain." The above can be further downloaded from the 6LBR to 6LR depending on the node capability.

FIG. 2



As shown in FIG. 2 above, any sensor forwarding data to remote sensor or cloud outside the Fog network will include the hash value as an Attestation ID in the IPv6 Extension Header of the data traffic. For scale, different options may be provided on when to include the same in IPv6 Extension Header:

- This could be done on all packets sent from the sensor.
- Selectively on different packets to reduce the power consumption.
- On demand request (using IPv6 NS extension) from Fog router to sensor.

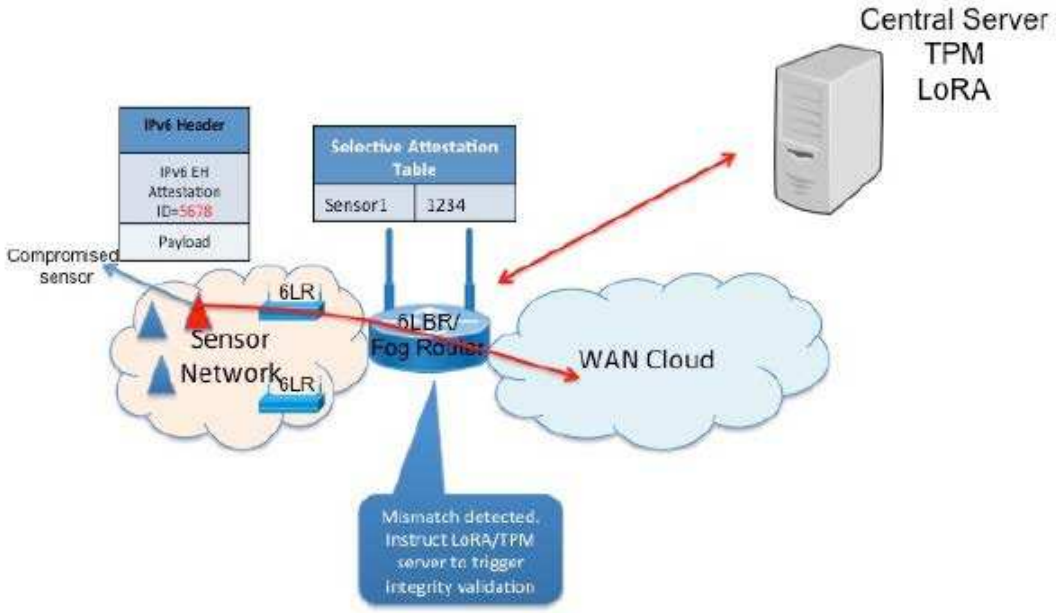
In any of the above situations, when there is any malicious code download to the sensor, the hashing value will change resulting in a mismatch between the value received in IPv6 data packet and the cached value in edge nodes.

LoRa defines the radio-PHY and is low bandwidth. There are already IP devices using LoRa as the transport radio through the use the 6LowPAN adaption layer which

offers significant IPv6 header compression. Since 6LowPAN is agnostic to the layers below it, it can be used with LoRa, IEEE802.15.4 or any other technology.

It is also noted that the operation of a LoRa network is not dependent upon IP protocols. The most common way to deploy a LoRa network is using LORAWAN, which defines the MAC layer, device registration, key exchange for encryption etc. LORAWAN defines a series of "MAC Commands" such as Link Check, Link ADR (Adaptive Data Rate), End Device Status. The device attestation mechanism presented herein could be incorporated in the LORAWAN frame payload or as an unsolicited MAC command sent from the device. The attestation can include, but it is not limited to, the ability to analyze memory and configuration changes for integrity validation of an IoT device.

FIG. 3



As shown in FIG. 3, if an edge node on detects such anomaly it will trigger the Server (TPM, LoRA etc.) to trigger an integrity validation. Until the sensor validity is confirmed by the server, none of the traffic from the sensor will be forwarded to any nodes and will be dropped by the 6LR/6LBR.

The approach presented herein integrates the integrity validation inline with data traffic and detects any compromised sensors quickly and helps avoid falsified data or affecting other sensors in a scalable manner.

The basic idea leverages the use of checksum value computed by different algorithms (like a noise filling technique that computes with input including the program memory) and using the checksum value as an Attestation ID as part of the initial attestation verification. A Fog router (6LBR, for example) will store it in local database.

Any further data packets from the sensor will carry the Attestation ID in the IPv6 extension header of Network Service Header (NSH) Metadata (MD) from Sensor to Fog Router, which will be used by the Fog router as a lightweight verification on a per-packet basis. Any change in program memory (like installing malicious code in compromised sensor) will result in hashing a different Attestation ID which can be detected by Fog router, which in turn will trigger the Attestation verification procedure before forwarding any more data traffic from the compromised sensor.

Sensor-----Fog-Router---Cloud

In above topology, the sensor uses a hardware- or software-based attestation approach. In either case, the sensor will hash a checksum based on the program memory space and include the same in attestation protocol (like a Platform Trust Service Protocol) as part of the initial validation (Integrity verification) exchanged with the fog router. This Attestation ID will be mapped to the sensor in Fog router and will be used for lightweight validation.

Any data traffic originated by the sensor will include the Attestation ID in the data packets (as IPv6 extension header, NSH Metadata etc.). Any local event change (like memory consumption - specifically in process memory area), it runs the local hashing to derive the Attestation ID. Alternately (to avoid having the sensor always include NSH MD), it could intermittently include the same in some data packets.

The Fog node, on receiving the data packet with "Attestation ID" as IPv6 extension header or NSH MD will use the Attestation ID for a simple validation.

Any Malicious user loading any code will result in a change of the Attestation ID which can be immediately detected by the Fog node on the next received packet and will trigger attestation query before forwarding any more data traffic.

Time/Seasonal Based Hash

Since an IPv6 extension header (EH) is used as the mechanism to carry the Attestation ID it is possible for a malicious process elsewhere in the network to snoop on the EH to retrieve the hash ID. Once this is obtained, malware could insert itself into the sensor and replay the hash to remain undetected. To avoid this situation, the hash computation should be based on a key that periodically changes. The edge devices could send an encrypted key in a control frame periodically based on time (one a day, week) or based on the number of frame received (change key every 100 packets). Each time the sensor sent a frame the hash could be calculated with the received key, and then the key should be incremented by 1 to ensure a different hash the next transmission. The receiving edge/Fog device should increment the key to remain in sync. There could also be a simple protocol exchange to keep other edge devices in sync with the current key. All sensors in the same domain could use the same initial key although the hash generated may be different.

Thus, the Fog node can share a key that changes (increment by 1 or 2) conditionally (such as every 10 seconds/minutes or every 10 frames sent by the end node, etc.). A new hashing value will be done by the sensor and the same included in the extension header. Since the Fog router is aware of the details, it can hash the same value to identify if the value received in data packet is legitimate or compromised by any malware.

Even if malware captures the packet to get the last used value, it cannot be used more than the predetermined timeframe or number of packets (as the sensor will hash it again and include the same in extension header). This re-hashing is an incremental operation performed as needed one bucket at a time. Adding rehashing is straightforward. In a dynamic array, resizing by a factor of x implies that only n/xi keys are inserted i or more times. This is feasible even in low compute environments.

Thus, core to this solution is to catch problems in between integrity challenges. In addition, each challenge uses up bandwidth and energy. It is not feasible to constantly repeatedly challenging. If a sensor does get infected and keeps quiet, then this is not a concern. But as soon as the sensor decides to transmit a malicious packet and the hash does not correlate, then it can be immediately determined that the sensor's integrity has been compromised.

In summary, the process presented herein uses extensions (e.g., MQTT, CoAP, YANG, etc.) to offload the hashing value for sensors to edge nodes. The hashing value can be included in an IPv6 Extension Header (in all packets, few packets, on-demand/request based). When an on-demand option is used, IPv6 NS may be employed to request a sensor to include the hash in next data packet or in IPv6 NA. Any deviation in hashing is detected, and an instruction may be sent to the (TPM, LoRA) server using MQTT, CoAP or YANG extensions to trigger integrity/attestation validation. This provides an event-based integrity validation trigger from server to sensor. Until the integrity validation is completed, all the packets from the sensors may be dropped. Thus, a dataplane-based integrity validation process is provided that achieves faster abnormality detection and prevents the falsified data from affecting the service.