Technical Disclosure Commons

Defensive Publications Series

January 2020

TRUSTWORTHINESS AMONG CONTROLLERS AND SWITCHES IN MULTI-PROVIDER SOFTWARE DEFINED NETWORK DEPLOYMENTS USING A TRUSTED PLATFORM MODULE (TPM) AND SECURE LEDGER

Niranjan M. M

Nagaraj Kenchaiah

TRUSTWORTHINESS AMONG CONTROLLERS AND SWITCHES IN MULTI-PROVIDER SOFTWARE DEFINED NETWORK DEPLOYMENTS USING A TRUSTED PLATFORM MODULE (TPM) AND SECURE LEDGER

AUTHORS:

Niranjan M M
Nagaraj Kenchaiah

ABSTRACT

The OpenFlow® protocol especially OpenFlow® Discovery Protocol (OFDP) utilizes clear text Link Layer Discovery Protocol (LLDP) message exchanges to discover network topology. Such exchanges lack security and may lead to network attacks such as LLDP flooding, link fabrication, etc. Currently, the OpenFlow® protocol both in the case of discovery (OFDP) as well during subsequent communication between a controller and a switch (even with Transport Layer Security (TLS)) does not offer a way to understand whether or not a discovered controller or switch is a trustworthy device. Presented herein are techniques that provide Trusted Platform Module (TPM) and blockchain-based trust establishment for OpenFlow® protocol communications that may be utilized between controllers and switches in multi-provider software defined network (SDN) deployments.

DETAILED DESCRIPTION

The OpenFlow® protocol is often used for communication between controllers and switches in SDN deployments. For Topology Discovery, the OpenFlow® defined OpenFlow Discovery Protocol (OFDP) is used by controllers to discover the underlying network topology using clear, non-authenticated Link Layer Discovery Protocol (LLDP) packets. LLDP is an open and extendable part of the Internet protocol suite used in IEEE 802 to advertise the identity and abilities of the devices, as well as other devices connected within the same network.

The OpenFlow® defined OFDP, however, is not a secure protocol. While LLDP packets do carry information such as the chassis-id or system name, the protocol currently does not offer a mechanism to determine whether or not discovered devices are trustworthy. Further, the use of clear, non-authenticated LLDP packets to detect the links between switches makes OFDP vulnerable to a number of attacks such as switch spoofing, link fabrication (e.g., LLDP duplication and LLDP injection), controller fingerprinting, LLDP flood, etc.

One form of link fabrication may include packet duplication in which an attacker can gain control over a host connected to a switch, determine the Datapath ID (DPID) of the switch, and can fabricate a link by injecting fake LLDP packets into another switch. Another form of link fabrication may include LLDP injection in which an attacker, by monitoring the traffic, can obtain the LLDP content used by the controller, and can inject the same LLDP packets into the network thereby creating bogus links between switches or between the malicious hosts and switches.

For controller fingerprinting, consider that LLDP content is different from one controller to another, which allows fingerprinting attacks on SDN controllers. For example, an attacker that has control over a host can match the LLDP content received from a switch (LLDP packets originate from the controller) against a controller signature database to detect which controller is managing the network. Such information can then be used to launch specific and more efficient attacks on the controller.

For switch spoofing, consider that each LLDP packet contains a version field, flags, Time-to-Live (TTL) information, and Type-Length-Value objects (TLVs) for information advertisement. Mandatory TLVs in OFDP include ChassisSubtype (which may be the Media Access Control (MAC) address of the local port of the switch) and PortSubtype, which can be used to track packets by a controller. By intercepting clear LLDP packets containing MAC addresses, a malicious switch can spoof other switches to falsify the topology graph of the controller.

LLDP flooding is a form of Denial-of-Service (DoS) attack in which an attacker generates enough fake LLDP packets to exhaust the link connecting a switch to a controller as well as the controller resources.

Some current solutions for mitigating such attacks may include countermeasure methods to avoid these attacks, however they do not consider potential side effects on legitimate traffic flows. For example, for link fabrication, one solution may include authenticating the LLDP packets by adding a key-Hash Message Authentication Code (HMAC) as an optional TLV in the packets. However, this technique only works against fake LLDP injection but not against link fabrication by packet duplication. For LLDP flood attacks, basic countermeasure methods like port blocking or packet filtering may not be effective, especially in the case of very dynamic environments (e.g., multi-tenant cloud) since connected hosts and switches change frequently, which may result in preventing legitimate LLDP packets from reaching the controller.

Also in SDN deployments, OpenFlow® communications are provided over TLS between controllers and switches. When a controller and a switch establish a communication channel using TLS, they perform mutual authentication across a network, typically based on certificates or a public key infrastructure. However, this may be insufficient for cases in which a controller or switch may become compromised such that they are no longer a trusted entity. As the OpenFlow® protocol is used during topology discovery as well as for configuration of policies etc., it is important to know that a device is trusted before involving the device in protocol flows.

Currently, the OpenFlow® protocol, both in the case of discovery (OFDP) as well during subsequent communication between a controller and switch (even with TLS), does not offer a mechanism to determine whether or not a discovered controller or switch is a trustworthy device.

However, trustworthiness of a controller and switch should be verified during discovery and subsequent communication between a controller and switch in SDN deployments. This includes an integrity check for both hardware as well as software for all the devices participating in OpenFlow® communications.

Secure computing environments often provide that connectivity is only to be established with a trustworthy device. Thus, devices that are not trustworthy are to be excluded from network operations as early as possible in the overall operational process. In one implementation, trustworthiness of a device can be achieved through Attestation (Stamping).

Attestation is a trusted computing technology which can be applied in hardware, software (applications), and/or protocols, especially in networking, and can broadly be divided into two Attestation methods including A) Bi-directional Attestation, and B) Uni-directional Attestation.

Trusted Platform Module (TPM) functionality can be embedded in a wide variety of devices including Mobile phones, PCs and Routers. TPM functionality, also known as also known as ISO/IEC 11889, may be a dedicated cryptographic device that supports secure key generation and remote system attestation. Attestation, as defined by the Trusted Computing Group (TCG), describes that the TPM functionality can be used as a hardware root of trust and offer Proof of Integrity of a node, in which integrity may include hardware integrity, software integrity, and/or runtime integrity.

Neither blockchain nor trusted computing (TPM) alone may provide a sufficient level of network security for SDN deployments. Blockchain is not an authentication technology (i.e., there is no authentication in blockchain), rather, blockchain is typically used for encrypting messages. In a blockchain, data on the chain is unchangeable/immutable such that the ledger never changes (i.e., it is known exactly who sends data, but the data itself is not known). Further, once a block is recorded on a block chain, it remains the same throughout time and is protected by mathematical algorithms.

Blockchain is an append-only distributed database technology, also known as a distributed ledger. It allows a group of peers to maintain a database while guaranteeing its integrity and assuring that all peers have equal rights as far as owning, accessing, and managing the database are concerned. From a data structure perspective, the blockchain is a singly linked list composed of structures called blocks. Each block, apart from the first block (the genesis block), points to the previous block in the chain. If any of the earlier blocks in the chain is tampered with, this change is propagated to every subsequent block, thus assuring detection. Each block on the chain has a unique address, timestamp, and relation with the previous block. This chaining mechanism also deters any adversary from changing a target (historical) block as then it has to modify all the blocks that were appended to the chain after the target block.

It is important to understand that it cannot be proved that data that is written on the chain is the data that was intended to be written. Thus, it is cannot be determined what a device may write to the chain. Stated differently from a security perspective, there is no evidence to identify how a private key may be protected on the chain. If all that is present is the chain itself, it is difficult to identify the identity of a device that performed a transaction, whether it was performed by a trusted device or was performed by a device that stole or borrowed the private key and performed the transaction. All that is known that a device having the key performed the transaction.

TPM provides Proof of Integrity measurement and assurance. Proof of Integrity measurement is not historical but related to the current state of either data or hardware/software. Hence, along with Proof of Integrity, it is good to have freshness of Proof of Integrity by maintaining even historical integrity values that can easily be verified. For this, consider the following two conditions: 1) No entity should be able to change the historical integrity and records, and 2) all the information is going to be in the public domain. These two conditions can be satisfied with blockchain. Hence, what is needed is trusted computing (TPM) along with Blockchain to increase the level of network security.

This proposal provides techniques to enhance security in multi-provider SDN deployments by integrating TPM and blockchain technologies into the OpenFlow® protocol for both topology discovery processes and also for subsequent communications between controllers and switches over TLS in SDN deployments in order to incorporate trustworthiness among participating devices.

The techniques of this proposal can be divided into two parts, remote attestation and trustworthiness. For the techniques discussed herein, the term 'device' may generally be used to refer to any of an SDN controller or switch.

Figure 1, below, illustrates example details associated with remote attestation provided in accordance with the techniques of this proposal.

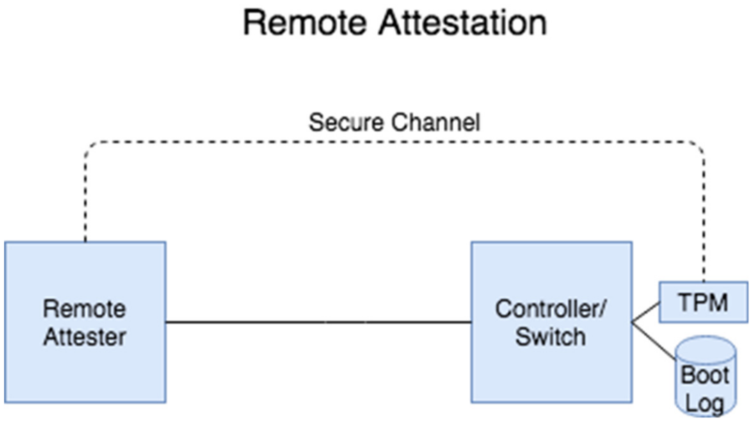

Figure 1

To facilitate remote attestation, various operations may be performed as follows:

1. At boot time, a device may compute a Measurement List (ML), which may include a hardware signature, a sequence of hashes of the software involved in the boot sequence, primarily the BIOS, the boot-loader, kernel, software implementing the platform, etc. The ML may also be used to capture the device system states.
2. The ML (containing sequence of measurements) can be stored in a set of registers called Platform Configuration Registers (PCRs). In particular, the ML can be securely stored in the PCRs inside the local TPM of the device.
3. To attest the device, a remote Attester challenges the device with a nonce "nU." The device queries the local TPM to create a message containing both the ML and the “nU” and sign the message with a private key Attestation Identity Key (AIK) of local the TPM. Generally, the AIK is a signing key provided and certified by the TPM owner that can be used to sign PCR quotes and certify other keys loaded into the local TPM.
4. The device sends this message to the remote Attester, which can verify the message using a corresponding public key with respect to the private key (AIK), thereby authenticating the device. By checking that the nonces match and the ML corresponds to a configuration that is deemed trusted, a remote Attester can reliably identify the device as a trusted device.
    a. Verifying that a message has a correct signature guarantees that it was produced by the TPM at some point in the past. Verifying that a correctly signed message includes the nonce guarantees that the message was produced by the TPM at some point after the attester/verifier generated the nonce, which can prevents replay attacks.
5. The remote Attester can communicate with the device over secure channel with Public Key Cryptography.
6. The remote Attester can also read the boot event log and fresh PCR values in the ML. The integrity of event logs can be validated by comparing the actual PCR values to expected values from the log. The integrity state of the device can then be evaluated (e.g., whether trusted firmware, boot loader,

and/or OS kernel are running on the target device). The remote Attester can also set a policy around these measurements, and enforce, for instance, that an up-to-date kernel is running. With these measurements, any compromised device can be detected. Thus, the remote Attester can verify that a received ML is fresh, genuine, and has not been tampered with.

Once the device is verified as trusted, it can be added to the blockchain of trusted devices and thus, added to the distributed secure ledger. With the distributed secure ledger, all the blockchain enabled devices would know the trustworthiness of all other peer devices before connecting to them. Here Remote Attester would do the functionality of Miner when compared with Blockchain.

To provide trustworthiness for an SDN deployment, SDN controllers can be enabled with blockchain functionality and acts as a blockchain node. The SDN Controllers may be provided with information regarding the trustworthiness of switches connected in the SDN deployment and/or the trustworthiness of other SDN Controllers in multiple provider deployments. If SDN switches (or controllers) do not supports blockchain functionality, then a bidirectional Attestation method can be used to verify the trustworthiness of devices in an SDN deployment.

Techniques herein provide for integrating the process of remote attestation into the OpenFlow® TLS handshake protocol, which can be achieved by modifying the client key exchange messages in the OpenFlow® TLS handshake protocol using keys and signatures generated by a local TPM so as to provide the required level of security and trust. In one example, TLS 1.3 can be extended with a new ExtensionType as defined in https://tools.ietf.org/html/rfc8446#section-4.2.

Figure 2, below illustrates example details associated with trustworthiness provided in accordance with the techniques of this proposal.

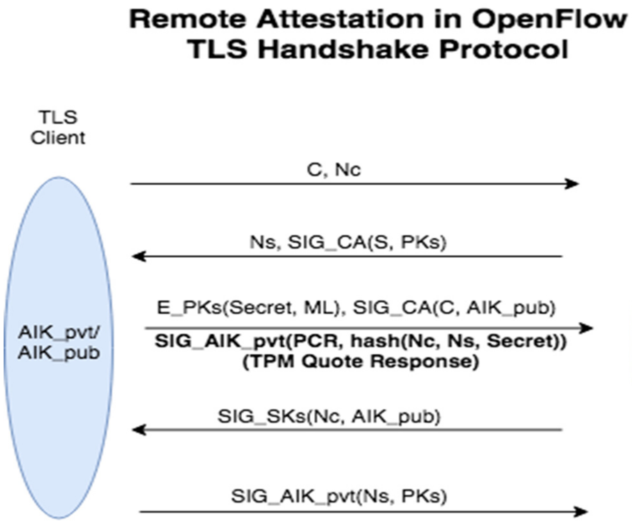

Figure 2

To provide trustworthiness in accordance with techniques of this proposal, various operations may be performed as follows:

1. An OpenFlow® TLS client (Attestator) "C" creates a non-predictable nonce "Nc" and sends it along with its identity to a server (challenger) "S". This message may be a client hello message as prescribed for the OpenFlow® TLS handshake.
2. The server responds with a server hello message, which contains a non-predictable nonce "Ns" generated by the server and the certificate of server signed by a trusted certificate authority (CA) (e.g., SIG_CA(S, PKs)). This message may be a server hello message as prescribed for the OpenFlow® TLS handshake.
3. The message that the client sends back to server may similar to the client key exchange message in OpenFlow® TLS handshake with the following modifications (as illustrated in Figure 2, above):
    a. The client sends a Measurement List (ML) along with the session secret encrypted with server's public key "PKs".
    b. The client owns a pair of public/private Rivest-Shamir-Adelman (RSA) keys, referred to as AIK keys, generated by the local TPM. The client also obtains an AIK certificate which contains the AIK

public key signed by a trusted CA. The client sends this AIK certificate to the server to authenticate itself.

c. The client sends a TPM Quote response to the server. In order to obtain a TPM Quote from the local TPM, the client sends a hash of the two nonces and the session secret to the local TPM and requests a quote signed by the AIK. The local TPM returns the signature over PCR values and the given hash by AIK private key.

d. The server validates whether the AIK certificate of the client was signed by a trusted CA and belongs to a genuine TPM. The server then verifies the freshness of Quote response by comparing a hash of the nonces and the secret with the signed hash. Next, the server validates the integrity of ML by verifying the hash of ML against the PCR value in the signature. Finally, the server validates individual entries in the ML by comparing the hashes against acceptable values.

4. If the integrity of the client platform is trusted by server in the above step, then the server and client continue to exchange messages according to the OpenFlow® TLS handshake protocol to establish a secure session.

In summary, techniques of this proposal provide TPM and blockchain-based trust establishment for OpenFlow® protocol communications that may be utilized between controllers and switches in multi-provider SDN deployments. For example, trustworthiness can be provided between controllers and switches during network topology discovery and/or between controllers when communicating in multi-provider SDN deployments.