Secure Communication Between OpenFlow Switches and Controllers

Dominik Samociuk

Silesian University of Technology
Institute of Informatics
Email: dominik.samociuk@polsl.pl

***Abstract*—We study the applicability of different protocols related to authentication and access control for secure channel between switches and controllers in Software Defined Networks communicating with OpenFlow. We firstly show possible problems with the lack of security mechanisms in OpenFlow architecture. Then we analyze the usability, advantages, drawbacks and implementation details of Transport Layer Security, Secure Shell and IPSec protocols as the secure channel medium for OpenFlow communication between switches and controllers. Finally, we discuss their possible extensions to authentication and access control mechanisms.**

***Keywords*–Network security; Secure architecture; OpenFlow, Software Defined Network.**

I. Introduction

OpenFlow protocol [1] has generated interest in academic and business society due to the features it offers to architects and developers of Software Defined Networks (SDN). By creating a standardized interface to connect switches with controllers, control-plane logic was moved to a centralized controller (or controller group). However, even when strictly adhering to the specification [2], OpenFlow does not enforce the use of secure communication channel between a switch and controller. Namely, the entry on the Transport Layer Security (TLS) usage was introduced in the OpenFlow specification and then modified. In the latest version, its usage is just a recommendation (not a "must-have" requirement). In the the OpenFlow switch specification ver. 1.4.0 sec. 6.3.3. it reads: "The switch and controller may communicate through a TLS connection". Moreover, due to evolving nature of the OpenFlow protocol, many vendors have not fully implemented this recommendation. Lack of the TLS adoption and problems with the implementation of the TLS infrastructure leave a clear path for attackers to infiltrate OpenFlow networks, using possible attacks described in the following sections. To worsen the security case, there are no authentication nor access control mechanisms (except for an ersatz of authentication in TLS).

Without forcing a secure communication channel, OpenFlow risks repeating the mistakes of other management protocols, designed basing on the assumptions that the link and infrastructure are secure (e.eg. Telnet, SNMPv2, TFTP). Of course, in production environment they must have been replaced with their safe versions (SSH, SNMPv3, SFTP, respectively). One of the proposed solutions for OpenFlow is controlling switches through the Internet in the architecture of branch-office network or offering switch management as "security-as-a-service". If the secure communication channel, authentication and access control are not enforced, OpenFlow will not be able to develop into the above-described roles and will be replaced by a secure protocol (like in the mentioned transition from Telnet to SSH) or by another, secured version of the protocol (like in transition from SNMPv2 to SNMPv3).

In this paper, we compare the possibility of implementing different authentication and access control mechanisms over a secure channel in OpenFlow communication with three popular protocols, namely Transport Layer Security, Secure Shell (SSH) and IPSec.

Transport Layer Security [3] and its predecessor, Secure Sockets Layer (SSL), are cryptographic network protocols for securing data communication. TLS provides confidentiality and integrity of data, as well as the authentication of the server, and sometimes of the client. It is based on asymmetric encryption and can be deployed in two modes, namely as:

- X.509 certificates and Public Key Infrastructure cryptosystem to provide authentication, encryption, integrity and non-repudiation using public and private key cryptography and digital certificates;
- Web of Trust architecture – decentralized authentication method, in which there is no hierarchical structure of the authenticating organizations and trust of each certificate is the sum of the signatures by the other members of the web, signed under this certificate.

SSH [4] is a common name for the whole family of protocols, not just terminal. It is also used for file transfer (SCP, SFTP), remote control of resources, tunneling and other applications. A common feature of all these protocols is identical to the SSH data encryption technology and user recognition. It is possible to configure SSH tunnels to transfer unencrypted traffic on the network through an encrypted channel.

IPSec [5] is a set of protocols for implementing secure connection and encryption exchange of keys between hosts. IPSec can be used for protecting the transmission in three modes:

- host-to-host – between pair of hosts;
- network-to-network – between pair of the security gateways;
- network-to-host – between the gateway and a host.

IPSec consists of at least two channels of communication between connected devices: (a) the exchange channel, through which data associated with authentication and encryption (keys) is transmitted and (b) the channel (one or more) that carries packets transmitted over the already secured line.

This paper is organized as follows. In Section 2, we discuss related works. Section 3 describes the secure channel and authentication problems with OpenFlow in the recent specification. It also presents some possible attacks and their effects. In Section 4, we present technical details about the proposed solutions, including Transport Layer Security with PKI architecture, Secure Shell tunneling and IPSec protocol. In Section 5, a brief security analysis of the security of the proposed solutions is carried out. Section 6 concludes the paper and presents our planned future work on the subject, including the implementations of new solutions and experiments on PL-LAB2020 - a large testbed and experimental network being built in Poland right now.

II. Related work

So far, the following works have been focused on the secure SDN architecture using OpenFlow.

FlowVisor [6] acts as a transparent proxy between controllers and switches in order to apply limitations to the rules created by controllers. It creates slices (combinations of switch ports, MAC addresses, IP addresses, port addresses or ICMP type) of network resources and delegates the control of slices to different controllers. The role of FlowVisor is to isolate the effect of rewritten rules to the specific slice of the network i.e., one slice cannot control another's traffic.

A similar concept is FortNOX [7] – a software extension developed on NOX controller to check the flow rule contradictions in real time. It uses the role-based authorization on OpenFlow applications (in this case, something that wants to modify the network traffic using the OpenFlow protocol, e.g., firewalls, intrusion prevention systems). The difference between FortNOX and FlowVisor is that FortNOX is a single controller software, that executes parallel applications, while FlowVisor runs apart from controllers (usually on a different host). Both of these solutions restrict intrusted controllers/applications from introducing security threats. However, they rely on the assumption that the OpenFlow protocol and its communication channel are secure.

Another approach is to provide the security to Software Defined Networks using OpenFlow. NICE [8] Distributed Denial of Service protection infrastructure-as-a-service is a distributed vulnerability detection tool based on attack graph-based analytical models and reconfigurable countermeasures.

Moving Target Defense [9] in an OpenFlow environment is a mechanism to change internal hosts' IP addresses frequently and to mitigate attacks and reconnaissance from external network.

All the mentioned works are meant to provide the security to an OpenFlow-based SDN architecture, but their underlying assumption is that there are no network design vulnerabilities with the protocol. A comprehensive OpenFlow vulnerability assessment was presented in [10]. The list of vulnerabilities contain the lack of TLS adoption, flow enforcement, denial of service risk and controller vulnerabilities. The first three of them can be fully mitigated (or significantly weakened at least) by using authentication and access control mechanisms, which are examined in this paper.

III. OpenFlow secure channel problems

The OpenFlow specification in version 1.0 contains the requirement about the use of TLS [11]. However, the next version changed this requirement from 'must' to 'should'. This is also the case of the current version (1.4). There is a noticeable lack of support for TLS in current SDN switches and controllers. Table I shows TLS support offered by OpenFlow equipment vendors [10][12].

TABLE I. TLS Support in OpenFlow by vendors.

VENDOR	*TLS Support*
HP switch	No
Brocade switch	Controller port only
Dell switch	No
NEC switch	Partial
Indigo switch	No
Pica8 switch	Only new versions
Open vSwitch	Yes
NOX controller	No
Brocade Vyatta controller	Yes
POX controller	No
Beacon controller	No
Floodlight controller	No
MuL controller	No
FlowVisor	No
Big Network controller	Yes
Open Source controllers (f.e. Ryu, OpenDaylight)	Yes

The usage of Transport Layer Security has also its impact on the preparation and maintenance of Software Defined Networks based on the OpenFlow protocol. In particular, this includes generating controller and switch certificates, signing certificates with private keys, installing correct keys and certificates on devices. Assuming a topology distributed in different locations, though not connected to the Internet, technicians must prepare all components of Public Key Infrastructure and provide its security.

While the lack of TLS support is feasible in secured networks (such as data centers) where the access to physical devices is difficult, it becomes a serious security vulnerability in architectures similar to campus-style or branch-offices deployments, in which access to the network is less restricted. In the role of the management protocol in the "security-as-a-service", it can be the case when OpenFlow is transmitted by an untrusted ISP (e.g., the client is in the country interested in intercepting transmission). In such scenarios we cannot neglect the possibility of an attacker placing a device on communication path between the switch and the controller, or simply copy the flow to his/her machine (Figure 1). He/she is able then to get the configuration, insert or delete rules to modify/record sensitive data flow (configured using the OpenFlow device). Additionally, an attack can be performed without any observable differences from normal transmission, i.e., the attacker acts as a transparent proxy (compare this with the FlowVisor proxy modification of the rules through the process of forwarding messages). This type of attacks (so called 'man-in-the-middle' attacks) were very popular and successful before, [13][14]. The Software Defined Networking may decrease the difficulty of a full exploitation and may allow the attacker to automate the process. In the SDN, the man-in-the-middle attacks are arguably worse than in non-software defined networks, due to the lack of necessity to the sniff traffic to obtain plain-text credentials or the possibility to reconfigure all groups of devices in a single attack.

It is important to note that assuring a secure communication between the switch and controller is not enough – we should also authenticate all devices connected to the controller or switch. For instance, for the following two reasons the authentication and access control may be needed:

- to limit the possibility of adding a bogus device (e.g.,

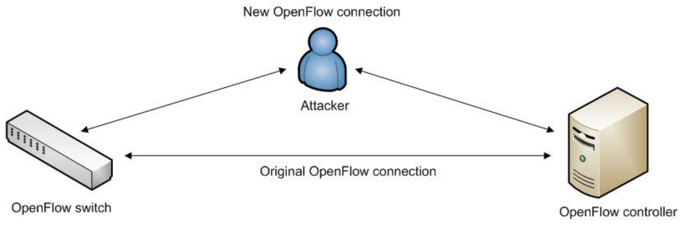

Figure 1. Man-in-the-middle attack scenario.

adding a new switch or controller to the OpenFlow domain);

- to group devices in federation (compare e.g., the Ofelia project, [15]), when different groups should be configured with different rules.

With full TLS implementation and authentication mechanism we can assure safety of the hosts and messages in transmission. We are unable, however, to detect switches that operate erroneously on rules in databases. The only solution to maintain the same view of traffic that flows through the network on the controller and switch seems to be dumping regularly and inspecting the flow tables on all hosts in the OpenFlow architecture. A potential solution described in [10] is based on generating keep-alive messages with checksums of flow tables that switches sent to controller.

IV. Secure channel creation and authentication

In this section, we describe three possibilities of creating the secure channel for communication between switches and controllers using OpenFlow. We also investigate how to further secure the OpenFlow architecture with an authentication mechanism that reduces the possibility of spoofing a device with a rogue switch or controller instead of any mitigation of eavesdropping.

A. Transport Layer Security

Deploying Transport Layer Security as an authentication and access protocol in the OpenFlow-based architectures was done by vendors using the public key infrastructure (PKI). The Web of trust architecture was neglected due to the small amount of devices in the authentication domain which could result in the vulnerabilities described in [16].

The proposed solution is an example of the peer encryption. The number of exchanges of the keys in such systems is proportional to the square of the number of users of the system (precisely, proportional to $\frac{n(n-1)}{2}$, where n is the number of users of the system). As already discussed, the reliability of the key distribution is essential for the credibility of the system. The solution to the number of the needed exchanges of keys and the need to ensure their authenticity was an application of the principle of the implied trust.

In cryptographic systems, there might be available some certification institutions, to which we have trust. This trust is supported by, for example, their protection level, their regular auditing, etc. If all users of the system have trust in the certification center, it is assumed that they also have trust in each other. In practice, this means that a user should know the address of the certification center and have its public key. In this way the user can reliably verify the authenticity of the received documents or exchange encrypted information with a reliable key from the communication partners, published by the certification center. In the PKI infrastructure, the trusted certification center is known as the Certificate Authority (CA).

To provide an opportunity to exchange keys between different systems, the cryptographic standard X.509 has been introduced also for certification. A certificate contains not only the owner of the public key signed by a certification authority, but also the information about the owner and other fields predicted by the standard X.509. Not only must CA be able to issue the certificate, but also to public and cancel the certificate, for example due to a theft or other security breach.

Equally important, as the implementation of the infrastructure, is developing and implementing the procedures for handling of the certificates. These procedures are called Certificate Practice Statement and include, among others:

- rules for issuing, verification and control the distribution of certificates;
- rules for cancellation of certificates;
- ability to recover private keys;
- methods for securing the infrastructure.

Especially in the case of PKI systems, the procedures are pillar of security.

In order to implement a TLS based secure channel and authentication mechanism, the administrator must build all the infrastructure to support it, i.e. the PKI (Figure 2). It will consist of elements such as:

- Each user of the system (in the case of an OpenFlow architecture, each switch and controller) posses key pair of private and public, stored in a secure space.
- Users of the system adopt a standard for certificate to the exchange keys.
- The existence of the certifying authority (internal or external, when developing OpenFlow in security-as-a-service manner) providing the issuance, invalidation and publication of certificates.
- Implementation of procedures to ensure the safety of the system.

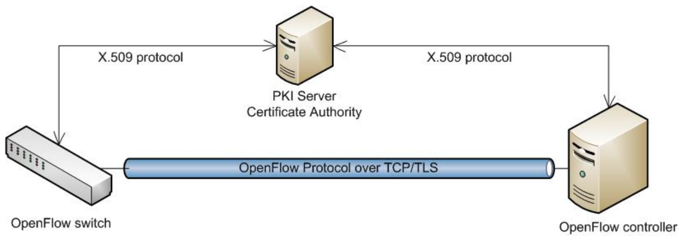

Figure 2. Architecture of the secured channel based on TLS protocol.

A simple connection between the switch and the controller illustrating the handshake with authentication by means of the Transport Layer Security mechanism consists of the following steps (by C we denote the controller; by S – the OpenFlow switch):

- S to C: sends ClientHello

The switch sends to the controller a message containing, inter alia, supported version of TLS protocol, supported methods of data encryption and compression, and session ID. This message also contains a random number which later will be used for key generation.

- C to S: sends ServerHello
  The controller responds with a similar message, in which it returns to the switch the selected parameters of the connection: TLS protocol version, supported types of encryption and compression, and a random number.
- C to S: sends Certificate
  The controller sends its certificate allowing the switch to verify its identity.
- C to S: sends ServerKeyExchange
  The controller sends its public key. The type and length of the key is determined by the type of algorithm in the previously sent message.
- C to S: sends ServerHelloDone
  The controller notifies that the switch can move to the next phase of the secure channel creation setup.
- S to C: sends ClientKeyExchange
  The switch sends to the controller an initial session key encrypted with the public key of the controller. Using the previous messages, the two random numbers (one for switch and another for controller) as well as pre-determined by the switch session key, both sides generate a session key used for the actual data exchange. The key is generated using a symmetric algorithm (typically DES). However, it is set in a safe way and known only for communicating parties.
- S to C: sends ChangeCipherSpec
  The switch informs the controller that it can switch to encrypted communication.
- S to C: sends Finished
  The switch sends this messages to report readiness to receive encrypted messages.
- C to S: sends ChangeCipherSpec
  The controller notifies that it obeyed the request - from now on the controller will only send encrypted information.
- C to S: sends Finished
  The message is sent over the secure channel to check reliability of the used mechanism.

As shown in the steps in the previous section, the default TLS mechanism provides only server authentication, resulting in authenticating the OpenFlow controller and leaving the possibility of spoofing the OpenFlow switches (grabbing configuration or modifying rules and sending them not so secured switch if possible). However, there are methods to authenticate the client switch. For this purpose, three additional messages can be used:

- C to S: sends CertificateRequest
  After submitting controller's certificate server notifies the client that it would like to receive a certificate from the OpenFlow switch
- S to C: sends Certificate
  After receiving the message ServerHelloDone the switch sends its certificate
- S to C: sends CertificateVerify
  The switch must confirm that it actually has the private key corresponding to the transmitted certificate. To prove this, the switch signs with its private key digest of all previously established connection parameters and sends it using this message.

B. *Secure Shell*

Another approach to encryption and authentication can be using the automatically generated keys and trust-of-first-use method as in the Secure Shell protocol. All switches are treated as SSH clients and the controller is treated as their server. The client connects to the server, authenticates with the key. In the process, it authenticates also the server key. Then the secure tunnel is created between OpenFlow's communication port on the server side (the controller) and configured port on switch. The switch sends unencrypted traffic to a local port (it is assumed that insecure architecture starts when a packet is leaving the network card and the device itself is trusted), and then the traffic is transmitted to the right port on the controller using the encrypted and secure tunnel (Figure 3). In this way, we limit the possibility of eavesdropping messages directly to the locally bugging switch and controller. We also use the authentication against the possibility of spoofing devices.

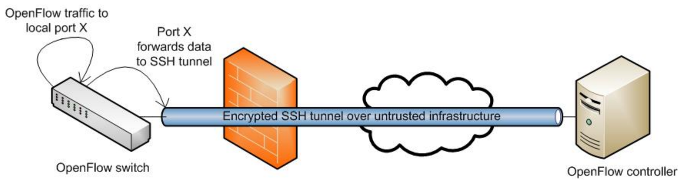
OpenFlow traffic to local port X
Port X forwards data to SSH tunnel
Encrypted SSH tunnel over untrusted infrastructure
OpenFlow switch
OpenFlow controller

Figure 3. Secure SSH tunnel transporting the OpenFlow data.

We propose two similar authentication-related solutions, but with different level of security and configuration required. The first one is the usage of automatically generated keys and automatic acceptation of the connection by a client before the typical SSH authentication process. This will reduce the possibility of performing and attack on the transmission to the small window of the first communication between the switch and controller, but it does not require any additional configuration. Alternatively, the more secure (but requiring the involvement of the administrator) solution would be to verify the public key thumbprint of each device when connecting to the server for the first time. Additional overhead of the administrative work when adding a new switch to the OpenFlow topology will completely reduce the possibility of eavesdropping transmission of the OpenFlow messages tunneled via SSH.

C. *IPSec*

The last analyzed option of authentication and creation of a secure channel between the switch and controller over the OpenFlow communication is the IPSec protocol, in particular the host-to-host architecture. IPSec in the host-to-host configuration (which connects two hosts without the need of additional devices, see Figure 4), creates the secure channel between them and allows to authenticate each other, while the only needs are connections dedicated to the other side. The process can be summarized in the following five steps:

1) Configuring switches and controllers.
   In this phase the network administrator must prepare the information (IP addresses of both hosts, encryption key generator, pre-shared key used to initiate the connection and exchange generated keys during secure transmission) and then use them during configuration of each device. This configuration generally will be prepared once and then copied between hosts.
2) IKE phase one.
   The purpose of this phase is to authenticate hosts and set up the first secure channel for IKE exchanges. Following functions are performed: the authentication and protection of the hosts identities, the negotiation of the IKE policy to protect the exchange, the usage of Diffie-Hellman authenticated exchange to obtain matching shared keys, the setting up of the first secure channel for IKE phase two. Phase one has two operational modes: main and aggressive. The main mode consists of three exchanges (agreement of the algorithm and hashes used to secure communication, generation of secret keys components used to passing random numbers - nonces to prove identities, last exchange is the verification of the other side's identity). The aggressive mode has fewer exchanges (we obtain second IKE phase quicker) but some of the information must be exchanged before there is a secure channel available.
   Due to the security aspect of the work, we prefer the main mode in the planned implementations.
3) IKE phase two.
   In this step the IPSec tunnel is created. The following functions are performed: the negotiation of protected parameters, establishing of security associations, renegotiation IPSec SA to ensure security (additional authenticated Diffie-Hellman exchanges can be performed).
4) Data transfer.
   After the IKE phase two, the IPSec tunnel is created and OpenFlow packets can be encrypted and decrypted using the encryption mechanism specified in the configuration, resulting in authenticated secure channel transmission.
5) IPSec tunnel termination.
   After the successful transmission, the tunnel can be terminated by deletion or by timing-out.

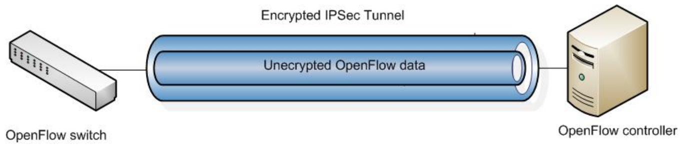

Figure 4. Secure IPSec tunnel transporting the OpenFlow data over an encrypted channel.

While TLS and SSH operate in the application layer, IPSec is a scheme operating in the network layer. Hence, only IPSec protocol protects any application traffic over an IP network. Therefore, an implementation for the OpenFlow protocol can be relatively easily migrated to another protocol or (due to evolving nature of OpenFlow) next specifications requirements.

V. SECURITY ANALYSIS

We analyzed benefits and drawbacks in the area of security of the proposed solutions. TLS, SSH and IPSec reduce or prevent the possibility of performing the following attacks:

- sniffing (lack of confidentiality),
- data modification,
- identity spoofing, password-based and application-layer attacks,
- man-in-the-middle attacks,
- denial-of-service attacks.

However, there are a few concerns regarding each of the solutions.

Transport Layer Security relies on secrecy of private keys and Certificate Authority trust, so assuring the security of these parts is the mission-critical aspect of maintaining this type of architecture. Some significant attacks against TLS include FREAK [17], BEAST [18] and CRIME and BREACH [19] attacks. However, assuming a proper implementation of TLS and its newest version, it is regarded as safe. Theoretically TLS can be compromised using SSL-Striping and SSL-Spitting attack, [20], but due to the newness of software-defined networking with TLS-secured OpenFlow, the attacks has not been confirmed and sufficiently researched yet.

As mentioned above, Secure Shell can be deployed using public and private keys or pre-shared key. Regardless of the option used, secrecy of keys (and CA if used) is the crucial security aspect. As regards possible attacks, using the revised version SSH-2 is assumed to be secure, however, theoretical vulnerability was discovered in [21] for the default encryption mode CBC. Therefore, we recommend the usage of CTR mode in the implementation. Another security concerns are that some institutions are able to decrypt the SSH traffic (see [22]). The details associated with such attacks were not released.

IPSec, similar to TLS and SSH relies on the secrecy of pre-shared key. However, the critical aspects are the randomness of the encryption key generator and security of the encryption and hash algorithm. As for this work, from available in IPSec implementation algoritms we assume sha1 as hash algorithm and 3DES, AES as encryption algorithms are secure. There are known attacks on IPSec when other than recommended solutions were applied [23]. Similar to SSH, there are allegations that some institutions have been working actively to insert vulnerabilities into IPSec implementations, [24].

VI. CONCLUSION AND FUTURE WORK

In this paper, some possible solutions to the lack of authentication, access control and creation of the secure channel over the OpenFlow protocol were investigated. As argued, implementing TLS (as in the OpenFlow specification recommendations) does not address configuration problems for network operators. With the core idea of increased security of the OpenFlow transmission, a novel utilization of known Internet security systems was proposed.

By comparing TLS, SSH and IPSec, it was demonstrated that, in relation to the OpenFlow architecture usage, each of the proposed protocols has its own strengths (i.e. the ease of implementation in IPSec, or conformation with the specification in TLS) and weaknesses (possible attacks). The paper showed that implementing any of the proposed solutions will result in increased security and reduction or prevention against numerous attacks on the OpenFlow protocol.

As for the future work, we are planning to utilize the PL-LAB2020 laboratory, [25], to implement each solution and analyze the security concerns mentioned in the article while verifying the performance of the protocols. The PL-LAB2020 laboratory, which is under construction now, will consist of six geographically dispersed nodes associated with leading Polish research and academic centers:

- National Institute of Telecommunication (NIT),
- Warsaw University of Technology (WUT),
- Poznan Supercomputing and Networking Center (PSNC),
- Silesian University of Technology (SUT),
- Gdansk University of Technology (GUT),
- Wroclaw University of Technology (WrUT),

connected via dedicated 2*10Gb/s fiber links. The nodes will be equipped with specialized devices for carrying out research in several directions, including the Software Defined Networking. In particular, there will be over 30 OpenFlow switches from at least 3 different vendors and a few OpenFlow controllers creating different technology domains distributed over 5 locations of PL-LAB2020 infrastructure (Figure 5). To validate the performance and analyze security of the studied solutions, PL-LAB2020 will also consist of several servers with Data Plane Development Kit and three network traffic generators and analyzers with 10Gb/s interfaces, placed in different locations.

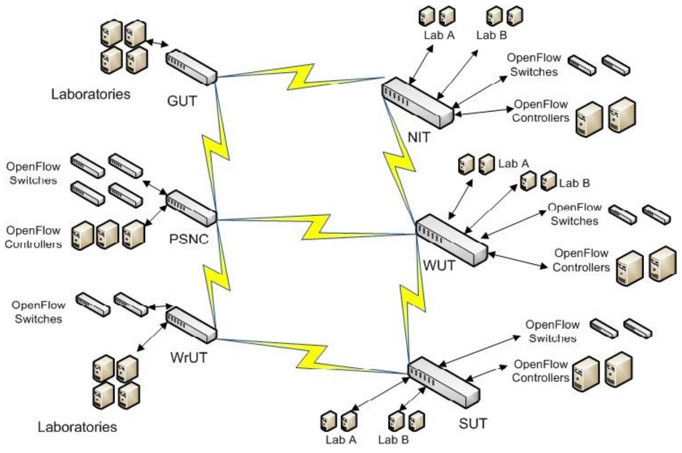

Figure 5. Architecture of PL-LAB2020.

Acknowledgment

This work has been supported by the National Centre for Research and Development under the European Regional Development Fund, Grant No. POIG.02.03.01-00-104/13, PL-LAB2020 project.

References

[1] N. McKeown and et al., "Openflow: enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, 2008, pp. 69–74.

[2] "Openflow switch specification: Version 1.4.0," URL: https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.4.0.pdf [accessed: 2015-02-28].

[3] T. Dierks, "The transport layer security (tls) protocol version 1.2," 2008, pp. 4–68.

[4] T. Ylonen and C. Lonvick, "The secure shell (ssh) protocol architecture," 2006, pp. 4–26.

[5] N. Doraswamy and D. Harkins, IPSec: the new security standard for the Internet, intranets, and virtual private networks. Prentice Hall Professional, 2003.

[6] R. Sherwood and et al., "Flowvisor: A network virtualization layer," OpenFlow Switch Consortium, Tech. Rep, 2009, pp. 1–13.

[7] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for openflow networks," in Proceedings of the first workshop on Hot topics in software defined networks. New York: ACM, 2012, pp. 121–126.

[8] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "Nice: Network intrusion detection and countermeasure selection in virtual network systems," IEEE transactions on dependable and secure computing, no. 4, 2013, pp. 198–211.

[9] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: transparent moving target defense using software defined networking," in Proceedings of the first workshop on Hot topics in software defined networks. New York: ACM, 2012, pp. 127–132.

[10] K. Benton, L. J. Camp, and C. Small, "Openflow vulnerability assessment," in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. New York: ACM, 2013, pp. 151–152.

[11] "Openflow switch specification: Version 1.0.0," URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf [accessed: 2015-02-28].

[12] Y. Patil, "Vulnerability analysis of openflow control channel," 2014, pp. 1–2.

[13] F. Callegati, W. Cerroni, and M. Ramilli, "Man-in-the-middle attack to the https protocol," IEEE Security and Privacy, vol. 7, no. 1, 2009, pp. 78–81.

[14] K. Ouafi, R. Overbeck, and S. Vaudenay, "On the security of hb# against a man-in-the-middle attack," in Advances in Cryptology-ASIACRYPT 2008. Springer, 2008, pp. 108–124.

[15] "OFELIA PROJECT homepage," URL: http://www.fp7-ofelia.eu/ [accessed: 2015-02-28].

[16] N. Ferguson and B. Schneier, Practical cryptography. Wiley New York, 2003, vol. 141.

[17] B. Beurdouche and et al., "A messy state of the union: Taming the composite state machines of tls," in IEEE Symposium on Security and Privacy. IEEE, San Jose, 2015, pp. 1–16.

[18] T. Duong and J. Rizzo, "Here come the ninjas," Unpublished manuscript, 2011, p. 4.

[19] D. Goodin, "Crack in internets foundation of trust allows https session hijacking," Ars Technica, 2012, pp. 1–2.

[20] M. Marlinspike, "New tricks for defeating ssl in practice," BlackHat DC, February, 2009, pp. 1–114.

[21] "SSH Vuln Note," URL: http://www.kb.cert.org/vuls/id/958563 [accessed: 2015-02-28].

[22] "Prying Eyes: Inside the NSA's War on Internet Security," URL: http://www.spiegel.de/international/germany/inside-the-nsa-s-war-on-internet-security-a-1010361.html [accessed: 2015-02-28].

[23] J. P. Degabriele and K. G. Paterson, "Attacking the ipsec standards in encryption-only configurations." in IEEE Symposium on Security and Privacy, vol. 161, Oakland, 2007, pp. 335–349.

[24] "Secret Documents Reveal N.S.A. Campaign Against Encryption," URL: http://www.nytimes.com/interactive/2013/09/05/us/documents-reveal-nsa-campaign-against-encryption.html [accessed: 2015-02-28].

[25] URL: http://http://www.pllab.pl/ [accessed: 2015-02-28].