OpenFlow Communications and TLS Security in Software-Defined Networks

Belema Agborubere 1st
Dept. of Computing and Technology
Anglia Ruskin University
Chelmsford, Essex
belema.agborubere@pgr.anglia.ac.uk

Erika Sanchez-Velazquez 2nd
Dept. of Computing and Technology
Anglia Ruskin University
Chelmsford, Essex
erika.sanchez@anglia.ac.uk

***Abstract* – The adoption of Software-Defined Networking (SDN), a networking approach where data traffic control and execution are made independent of each other, is an ongoing process that some companies are considering as an option but have not embraced yet due to different factors. Incorporating this new paradigm into an existing network defines a shift in networking technology with different benefits expected to derive from this implementation. These benefits include (1) the ability to use customised business specific applications, (2) reduce overhead costs on legacy network infrastructure, taking full control of network, (3) reduce network application update time, increase productivity, and (4) apply increased security among others. However, the security of SDN itself has been a subject of debate. This is mainly because, the communication standard used by SDN, known as OpenFlow, and developed by the Open Network Foundation, does not enforce the implementation of the Transport Layer Security (TLS) but defines it only as optional. This could then make the network infrastructure vulnerable and therefore affect the overall security of a company. Security plays a significant part in an organisation and it is one of the determinants of the success of SDN. OpenFlow security relies on the implementation of TLS, which has been proven vulnerable, and therefore bringing to mind the question on how secure organisation's data is when the implementation of secure data transfer is treated with laxity. This paper focuses on securing OpenFlow communication in SDN by summarising TLS security flaws and recommending ways of improving TLS security thereby securing OpenFlow communication.**

Keywords – Software Defined Network; OpenFlow; Network Security; SSL/TLS

1. INTRODUCTION

Software-Defined Networking (SDN) is a new networking paradigm that enables the separation of the intelligence (control) and packet forwarding (data) planes within a single network infrastructure. This approach differs from the traditional network infrastructure setup where both, intelligence and data forwarding functionalities, are placed together in network devices such as switches and routers. SDN's separation made it necessary to introduce a new communication protocol known as OpenFlow, which enables communication between the network devices and its controller. The acceptance of this paradigm is gradually increasing. Different factors contribute to this such as network flexibility, centralised management, reduced administrative costs and network control and among others [1]. Organisations such as Google and Amazon have implemented SDN as part of their cloud based infrastructure and academic institutions, such as Stanford University, have implemented it within their existing networks. However, an embrace of the new paradigm is still not certain [2], [3]. Figure 1 shows the SDN system setup.

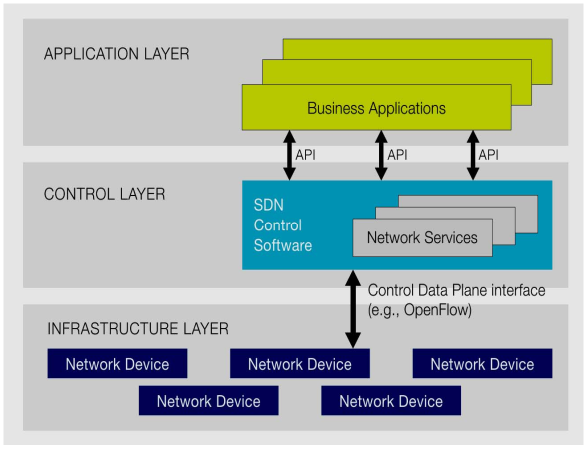

Figure 1: A sample SDN system setup with control and switching functions separate [4].

Current research explores the pros and cons of the use of SDN with a special focus on security, which remains an on-going issue that affects every organisation. The cost of data loss due to a security flaw has a heavy impact within an organisation, causing closure of businesses and financial.

Transport Layer Security (TLS) is implemented in order to provide security to data communication in OpenFlow between the data and control plane. Of interest is that the Open Network Foundation (ONF) does not recommend the implementation of TLS as mandatory. The reason could be that there are other network communication protocols such as the Border Gateway Protocol (BGP), Multiprotocol Label Switching (MPLS), NETCONF (IETF Network management protocol) [5]. This, therefore, gives OpenFlow-enabled infrastructure vendors the freedom to use what seems better for them rather than particular protocols being imposed on them.

As of 2014, and even with recent data breaches and network exposures, the implementation of secure OpenFlow communication using the Transport Layer Security (TLS) is still optional [6] and only recommended as the default security mechanism in 2015 [7]. On the other hand, the Open Network Foundation also recommended TCP as an alternative transport protocol despite the security challenges associated with it [8].

The functionality of SDN depends on the communication between the controller(s) and the switches therefore making it fundamental to secure OpenFlow. A flow is installed on the switch through instruction it receives from the controller, and also, data packets are forwarded, depending on instructions from the controller through the OpenFlow link [9].

On the other hand, communication through OpenFlow is secured using TLS, raising the question how secure is the Transport Layer Security in the face of attacks? Secondly, since the use of the TLS protocol is recommended rather than mandatory, not all SDN vendors implement it. This undermines the importance of security, which is the very focus of this paper on how to provide an extension to TLS to mitigate attacks against TLS such as Man in the Middle (MiTM) attacks.

This paper is divided into six sections. The first section gives an introductory discussion of the paper. Section two gives a brief background to Software-defined network (SDN) challenges while section three is on OpenFlow and communication security. Section four is on the challenges of the Transport Layer Security (TLS) and existing solutions to TLS. Finally, section five talks about proposed TLS extension while, section six highlights further studies.

2. SDN BACKGROUND AND CHALLENGES

SDN as a networking paradigm provides a programmable interface for network administrators to use non-legacy applications on their network, make changes on-demand and not depending on vendor-specific hardware or protocols. Customised business software can be implemented following a company's policy and bringing more control to the network administrator, unlike the conventional network setup where both the control and distribution functionalities are fused together and can communicate freely.

In SDN, the logical (control) and distribution (data) planes are different entities, one residing in the controller (control) and the other on the switches (data). Therefore, for both planes to communicate, an intermediary is needed, which the ONF proposed to be OpenFlow.

To secure communication between the controller and the switch, Transport Layer Security (TLS) should be enabled within the OpenFlow protocol, which is the transport link. However, as mentioned earlier, implementing TLS to secure OpenFlow communication has been defined as optional by the Open Network Foundation [10]. Network administrators have the choice of choosing what security protocol they wish to use depending on the type of programmable networks they want to use [11], [12], [13], [14]. This is one of the benefits of Software-Defined Networks as different organisations can implement their customised applications. On the other hand, if not considered properly, this flexibility comes with the risk of affecting the confidentiality, integrity and availability of the transmitted data.

It is possible to argue that, vulnerabilities were discovered because of the underlying hardware infrastructure and semi-independent setup of the technology, a great difference between the traditional network setup and that of the SDN. There is also the external programmable aspect [15] of it and the communication pattern between network nodes (switch and the controller). Table 1 is a list of the attack vectors and solution given.

Table 1: List of SDN security challenges and solutions

S/N	ATTACK VECTOR	POSSIBLE ATTACK MEDIUM	SOLUTION
1	Third party applications	Controller	Authentication/ authorisation of applications
2	Client	Client programmable interface	Resource isolation, define privilege level
3	Network deployment	Configuration mistakes	Avoiding previous known security weaknesses
4	Controller communication	Controller and applications	Define trust boundaries
5	Client programmatic access	Client	Resource isolation and limits
6	Multi-controller environment	Inappropriate configuration	Compatibility check and boundary definition
7	Use of legacy protocols	Controller and Switch	Compatibility check
8	Inter-domain connection with different legacy controllers	Inter-communication methods	Clear definition of security dependencies
9	OpenFlow	Deployment errors and obsolete protocol use	Use of recommended TLS v1.2

Though there is more than one point to breach Software-Defined Networks [16], most of the SDN security analysis focuses on OpenFlow. In one research [17], the security analysis of OpenFlow in SDN was looked at but not the security protocol itself. To the best of our knowledge, this is the first work focused primarily on the security of TLS protocol used in SDN with emphasis on enhancing the protocol for increased security.

With a decoupled controller, SDN technology presents a single point of failure [18]. However, there is provision for multiple controllers [19] with the requirement of a high level of administrative know-how. The need for highly skilled programming should be free of errors, which, if undiscovered, would be a point of exploiting in the network. Most organisations that have implemented the technology are cloud based [20]. This could be viewed as one of the reasons smaller organisations don't feel confident to implement or accept it yet [21]. There is a need for a holistic strategy for secure deployment because of new security.

There are other challenging factors that could impact so much on the technology. Table 2 highlights some of these challenges.

Table 2: Other SDN challenges

S/N	CHALLENGES
1	Controller placement
2	Scalability
3	Performance
4	**Security**
5	Interoperability
6	Reliability
7	Device authentication (making sure the right device is configured and can communicate)
8	Trust issues
9	Lack of good policy framework
10	Addressing dynamic changes with new devices

Resolving the challenges in SDN places the technology in a higher acceptance level. The right placement [22] of controllers increases the chances of availability of resources to users or access to the network, especially in a large network environment such as data centres. Using a single controller will not only impact negatively on the performance of the network, it also risks network failure. Also, network congestion will be experienced. This, therefore, demands multi-controller implementation to provide network availability and data packet flows.

However, the challenges are linked as one affects the other. One of the best attributes of SDN is its flexibility to expand and adapt to the different needs within the organisation. Device authentication and trust within the network devices is very important [22], [23]. Communication should be only between trusted and authenticated devices to avoid communication between untrusted devices. Also, efficient and reliable flow management [24] is important for SDN to achieve its purpose. There is a need for cost effectiveness of which interoperability and security should be guaranteed without which, the purpose of abstraction of the network devices is defeated.

Though SDN presents different challenges, this research focuses on securing the communication between the controller and the switch as it represents one of the biggest security flaws within SDN.

3. OPENFLOW AND COMMUNICATION SECURITY

OpenFlow version 1.0 was first introduced in 2009 and has gone through changes to improve on the functionalities such as running FlowVisor. As of January 2015, the current version is 1.5 [25].

One of the changes to networking as enabled by Software-Defined Networks is the communication link between the Controller and Switch via OpenFlow. The Open Network Foundation standardised the protocol as the communication interface between the switch and the controller [10] however, other communication and network configuration protocols exist [26] such as NETCONF.

The importance of the protocol cannot be overlooked because of its role as the medium of network configuration and communication; hence, it is vital to appropriately and adequately secure the protocol. This is not only for the acceptance of the technology, but also to provide confidentiality, integrity and authentication of every data packet and avoid unnecessary data exposures.

In securing data traffic or communication within OpenFlow, there is the need to look further to the base protocol that provides the security, which is the Transport Layer Security (TLS), the recommended data encryption protocol currently is version 1.2 [8]. It is also the most used protocol not only within OpenFlow but also throughout the Internet [27], [28]. TLS version 1.3 is a working draft at the moment.

The Transport Layer Security is divided into two layers (Record and Handshake). Although the record layer encapsulates the handshake layer, they perform different functions such as data link encryption, device authentication (client and server). One aspect has been silently overlooked which is the authentication [29] of network elements while more attention is given to the cryptographic part of the protocol. Figure 2 below shows a client-server handshake messages.

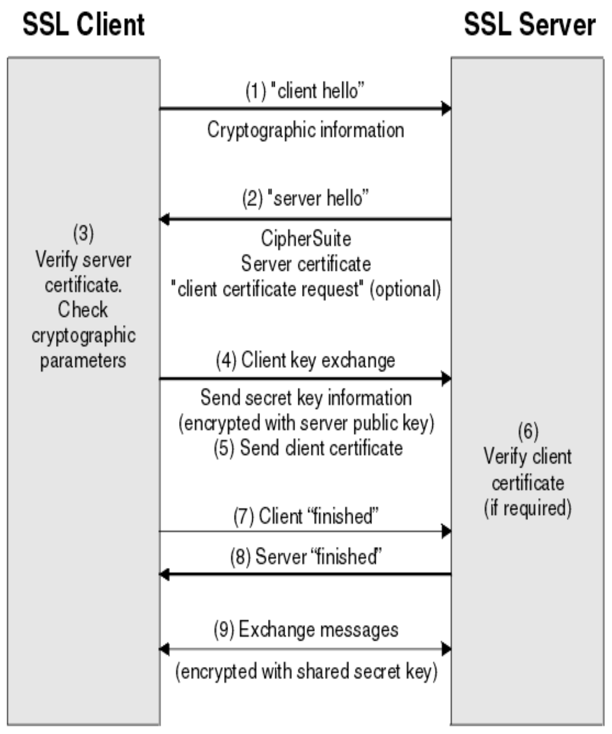
SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3) Verify server certificate. Check cryptographic parameters
(4) Client key exchange
Send secret key information (encrypted with server public key)
(5) Send client certificate
(6) Verify client certificate (if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

Figure 2: Client-Server handshake process [30]

4. CHALLENGES OF TLS

The challenges of the transport layer security dates back to the first SSL implementation in the 90s and since then subsequent versions have also been targeted of which TLS 1.2 is not an exception.

One of the requests from a server to client during an SSL/TLS handshake is the request for client certificates. Unfortunately, during the handshakes, client certificate authentications most times never occur and have remained optional.

The Transport Layer Security protocol has faced different attacks, placing it under constant threat [31], [32], [33], [34], [35]. Most of these attacks are not new [36] but rather are modified to affect newer versions of the security protocols.

Also, the improper configuration [37], [38], [39] of TLS results in the introduction of more vulnerabilities.

In an effort to provide countermeasures, several solutions have been rolled out for the different versions of the protocol, however, the attacks are modified to reveal new vulnerabilities as the cases of DROWN [40] and Transcript Collision Attacks [41].

These solutions include:

- Disabling older versions of the protocol,
- Providing no support for vulnerable versions
- Totally uninstalling such version from the server
- Use of alternate protocol

Unfortunately, these countermeasures are targeting specific attacks [42], [43], which at times open other vulnerabilities and become attack points.

5. PROPOSED TLS SUGGESTION AND EXTENSION

The Transport Layer Security is a standardized protocol. However, and for different reasons, it is still open to attacks, especially Man-in-The-Middle (MiTM). In order to secure the OpenFlow communications in Software-Defined Networks, a security extension is proposed. Figure 3 shows the steps proposed for TLS extension to mitigate man-in-the-middle attack. Table 3 shows the handshake message.

PROPOSED TLS EXTENSION

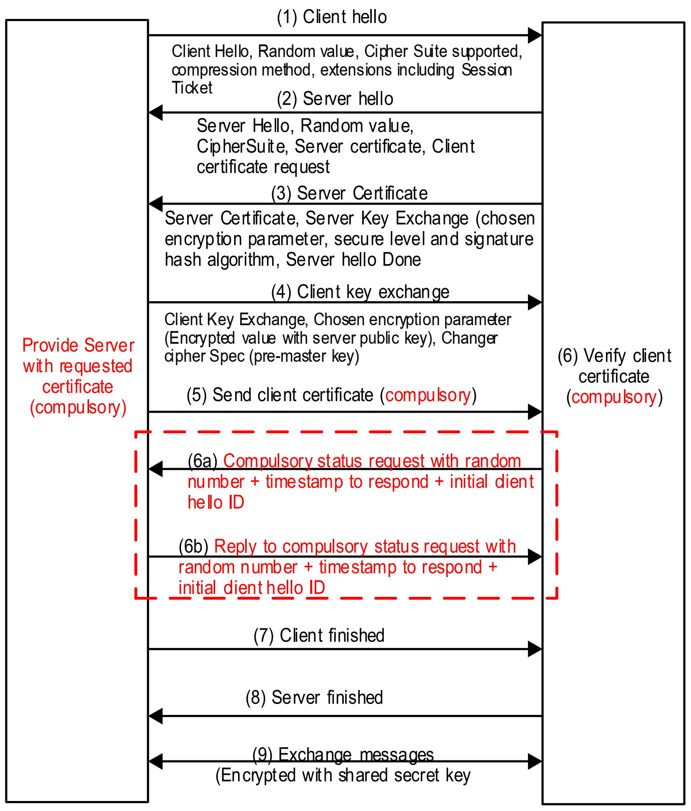
(1) Client hello
Client Hello, Random value, Cipher Suite supported, compression method, extensions including Session Ticket
(2) Server hello
Server Hello, Random value, CipherSuite, Server certificate, Client certificate request
(3) Server Certificate
Server Certificate, Server Key Exchange (chosen encryption parameter, secure level and signature hash algorithm, Server hello Done
(4) Client key exchange
Client Key Exchange, Chosen encryption parameter (Encrypted value with server public key), Changer cipher Spec (pre-master key)
(5) Send client certificate (compulsory)
Provide Server with requested certificate (compulsory)
(6) Verify client certificate (compulsory)
(6a) Compulsory status request with random number + timestamp to respond + initial client hello ID
(6b) Reply to compulsory status request with random number + timestamp to respond + initial client hello ID
(7) Client finished
(8) Server finished
(9) Exchange messages
(Encrypted with shared secret key

Figure 3: Proposed change to TLS

CHANGE TO EXISTING TLS STRUCTURE

Steps	Shared Information	Importance	Attack points and methods	Solutions given
1	Client hello message to Server Cryptographic info & version CipherSuites the Client support Compression method 32-byte Random Number Session ID (blank)	Very High (Request for secure connection)	Transcript collision attack (Bhargavan and Leurent, 2016) and Version rollback of CipherSuites to vulnerable version	Authentication every message (Bellare and Phillip, 1993) impersonation. Disable obsolete ciphersuites (Dierks, 2008 and Rescorla, et al, 2010, Bharagavan and Leurent, 2016)
2	Server hello message to Client Cryptographic info & version CipherSuites the Server supports Compression method Time and Date stamp Filled Session ID	Very High (Server choices options)	Attack point with CipherSuite rollback or Null-ciphers	Disable all obsolete ciphersuites (Dierks, 2008, Rescorla, et al, 2010)
3	Server Key Exchange Server certificate for verification	Important Server identified		
4	Server Hello Done Telling the Client of the completion of its hello message	Avoid client waiting		
5	Client Key Exchange Client sends Pre-master secret to generate session key	Certificate confirmation (Optional)	DROWN attack (Aviram, et al, 2016)	Each server should have unique key (Aviram, et al, 2016)
6	Change Cipher Spec (Client sends) client exchanges generated session key	Very important as it establishes encryption type to use	Drops message to non encrypted connection	Force all parties to send and receive before sending Finished message
			Confusion attack	Explicit message definition from both client server
6a	Server sends randomized (encrypted) status request with timestamp	If time elapses without expected response, server terminates the session. NOTE: In case of exception (network failure) try resend and terminate.		
6b	Client returns status reply within timestamp with hello message ID			
7	Finished (Client) Exchange of handshake completed	Important as Server is informed		
8	Server Finished	Important as Client is informed		
9	**Server-Client information exchanged (encrypted with secret key)**			

Table 3: Change to the existing TLS

The proposal seeks to enhance TLS through:

1. Making steps 5 and 6 of the handshake between a client and server, which requests a client to present its certificate to the server for authentication compulsory.
2. Before sending a client and server finished messages, a re-verification of client and server statuses should occur. The server sends a randomised status re-verification request with a time frame for the client to reply.
3. The verification should include the client hello message ID.
4. On a successful reply, the client and server Finished messages should be transmitted.
5. In a failed instance, because of time expires, the server should end the handshake.

These improvements will lock out an impersonator because of the timestamp to reply. The time it takes an attacker to decipher the randomized number will definitely exceed the time frame for client response to server request.

6. CONCLUSION AND FUTURE WORK

In conclusion, TLS is still used almost in every computer networks, however, considering the technique of DROWN which, is a time based Man-in-the-Middle attack, the enhancement of the TLS, would counter every of such attacks and also increase the TLS security in every network in theory.

To achieve this aim and validate the proposed solution, a virtual network will be setup and the enhanced TLS tested and a formal method technique will be used. The test will compare time difference with normal TLS, accuracy of attack mitigation, and the effectiveness to stop DROWN attacks. The formal method will be used to verify that the exchange of messages remains error prone and optimal.

This specific formal method will verify the practical security it provides in protecting against Man-in-The-Middle attacks in OpenFlow and in general TLS applications. To the best of our knowledge, there is no other OpenFlow communication security that directly focuses on the root communication security protocol.

ACKNOWLEDGMENT

I am grateful to my supervisor Dr Erika Sanchez-Velazquez, for the untiring support and advice. I also thank the VSIRG group for creating the platform for me to present my findings. Finally, the Department of Computing and Technology for providing me the place to carry out my research work.

REFERENCES

[1] Vissicchio, S., Vanbever, L. and Bonaventure, O., 2014. Opportunities and research challenges of hybrid software defined networks. *ACM SIGCOMM Computer Communication Review, 44(2)*, pp.70-75.

[2] Sezer, S., Scott-Hayward, S., Chouhan, P.K., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M. and Rao, N., 2013. Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, *51*(7), pp.36-43.

[3] Hubbard, P., 2015. *The case against SDN implementation*. [online] Available at: <http://searchsdn.techtarget.com/opinion/The-case-against-SDN-implementation> [Accessed 13 October 2016].

[4] Sdxcentral, n.d. *Understanding the SDN architecture*. [online] Available at: <https://www.sdxcentral.com/sdn/definitions/inside-sdn-architecture/> [Accessed 13 February 2017].

[5] McNickle, M., 2014. *Five SDN protocols other than OpenFlow*. [online] Available at: <http://searchsdn.techtarget.com/news/2240227714/Five-SDN-protocols-other-than-OpenFlow> [Accessed 13 October 2016].

[6] ONF, 2014. *OpenFlow Switch Specification. Version 1.5.0 (Protocol version 0x06)*. [pdf] Available at: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.pdf> [Accessed 3 May 2016].

[7] ONF, 2015. *Openflow Switch Specification. Version 1.5.1 (Protocol version 0x06)*. [pdf] Available at: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf> [Accessed 10 October 2016].

[8] Wasserman, M., Hartman, S., and Zhang, D., 2012. *Security analysis of the open networking foundation (ONF) openflow switch specification*. [online] Available at: <https://tools.ietf.org/id/draft-mrw-sdnsec-openflow-analysis-00.html> [Accessed 15 June 2016].

[9] Sdxcentral, n.d., *What is an OpenFlow Controller*? [online] Available at: <https://www.sdxcentral.com/sdn/definitions/sdn-controllers/openflow-controller/> [Accessed 14 December 2016].

[10] ONF, 2015. *OpenFlow*. [online] Available at: <https://www.opennetworking.org/sdn-resources/openflow> [Accessed 12 June 2015].

[11] TechTarget (n.d). Vendors take alternatives to OpenFlow SDN. [pdf] Available at: <http://media.techtarget.com/digitalguide/images/Misc/EA-Marketing/Eguides/Vendors_Take_Alternatives_to_OpenFlow_SDN.pdf> [Accessed 10 February 2017].

[12] McGillicuddy, S., 2012. Why Nicira abandoned OpenFlow hardware control. [online] Available at: <http://searchnetworking.techtarget.com/news/2240174517/Why-Nicira-abandoned-OpenFlow-hardware-control> [Accessed 10 February 2017].

[13] Little, R., G., 2012. Software-defined networking is not OpenFlow, companies proclaim. [online] Available at: <http://searchsdn.techtarget.com/news/2240158633/Software-defined-networking-is-not-OpenFlow-companies-proclaim> [Accessed 10 February 2017].

[14] Burt, J., 2014. Cisco unveils OpFlex as alternative to OpenFlow. [pdf] Available at: <http://www.eweek.com/print/networking/cisco-unveils-opflex-as-alternative-to-openflow.html> [Accessed 10 February 2017].

[15] Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M. and Gu, G., 2012, August. A security enforcement kernel for OpenFlow networks. In *Proceedings of the first workshop on Hot topics in software defined networks* (pp. 121-126). ACM.

[16] ONF, 2015. *Principles and practices for securing software-defined networks*. [pdf] Available at: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Principles_and_Practices_for_Securing_Software-Defined_Networks_applied_to_OFv1.3.4_V1.0.pdf> [Accessed 15 December 2016].

[17] Kloti, R., Kotronis, V. and Smith, P., 2013, October. Openflow: A security analysis. In Network Protocols (ICNP), 2013 21st IEEE International Conference on (pp. 1-6). IEEE.

[18] Dabbagh, M., Hamdaoui, B., Guizani, M. and Rayes, A., 2015. Software-defined networking security: pros and cons. *IEEE Communications Magazine*, *53*(6), pp.73-79.

[19] Phemius, K., Bouet, M. and Leguay, J., 2014, May. Disco: Distributed multi-domain sdn controllers. In *Network Operations and Management Symposium (NOMS), 2014 IEEE* (pp. 1-4). IEEE.

[20] Müller, L.F., Oliveira, R.R., Luizelli, M.C., Gaspary, L.P. and Barcellos, M.P., 2014, December. Survivor: an enhanced controller placement strategy for improving SDN survivability. In *Global Communications Conference (GLOBECOM), 2014 IEEE* (pp. 1909-1915). IEEE.

[21] Kobayashi, M., Seetharaman, S., Parulkar, G., Appenzeller, G., Little, J., Van Reijendam, J., Weissmann, P. and McKeown, N., 2014. Maturing of OpenFlow and software-defined networking through deployments. *Computer Networks*, 61, pp.151-175.

[22] Shamugam, V., Murray, I., Leong, J.A. and Sidhu, A.S., 2016, March. Software Defined Networking challenges and future direction: A case study of implementing SDN features on OpenStack private cloud. In *IOP Conference Series: Materials Science and Engineering* (Vol. 121, No. 1, p. 012003). IOP Publishing.

[23] Sdxcentral, n.d. *SDN security challenges in SDN environment*. [online] Available at: <https://www.sdxcentral.com/security/definitions/security-challenges-sdn-software-defined-networks/> [Accessed 14 February 2017].

[24] Bakalov, V., 2015. *Opportunities and challenges with SDN: performance monitoring approach must evolve to avoid visibility gaps*. [online] Available at: <http://www.networkworld.com/article/2973610/software-defined-networking/opportunities-and-challenges-with-sdn.html> [Accessed 13 February 2017].

[25] Sadasivarao, A., Syed, S., Pan, P., Liou, C., Lake, A., Guok, C. and Monga, I., 2013. *Open transport switch - a software defined networking architecture for transport networks*. [pdf] Available at: <http://conferences.sigcomm.org/sigcomm/2013/papers/hotsdn/p115.pdf> [Accessed 19 May 2015].

[26] Scott-Hayward, S., O'Callaghan, G. and Sezer, S., 2013, November. SDN security: A survey. In *Future Networks and Services (SDN4FNS), 2013 IEEE SDN For* (pp. 1-7). IEEE.

[27] Ching-Hao, C. and Lin, Y.D., 2015. *OpenFlow Version Roadmap*. [pdf] Available at: <https://pdfs.semanticscholar.org/b3cf/5442420eeadeb6bfc5558456223a69d5f5eb.pdf> [Accessed 25 October 2016].

[28] Rouse, M. and Cobb, M., 2016. *Transport Layer Security (TLS)*. [online] Available at: <http://searchsecurity.techtarget.com/definition/Transport-Layer-Security-TLS> [Accessed 19 November 2016].

[29] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J. and Wright, T., 2006. *Transport layer security (TLS) extensions* (No. RFC 4366).

[30] IBM Knowledge Center, 2017. *An overview of the SSL or TLS handshake*. [online] Available at: <http://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10660_.htm> [Accessed 7 February 2017].

[31] Thomas, S., 2000. *SSL and TLS essentials. Securing the web* [e-book] New York: Wiley Computer Publishing: Available through <https://cdn.preterhuman.net/texts/computing/security/SSL%20And%20TLS%20Essentials%20-%20Securing%20The%20Web%202000.pdf> [Accessed].

[32] Meyer, C. and Schwenk, J., 2013. Lessons Learned From Previous SSL/TLS Attacks-A Brief Chronology Of Attacks And Weaknesses. *IACR Cryptology ePrint Archive*, *2013*, p.49.

[33] Sarkar, P.G. and Fitzgerald, S., 2013. Attacks on ssl a comprehensive study of beast, crime, time, breach, lucky 13 & rc4 biases. *Internet: https://www. isecpartners. com/media/106031/ssl_attacks_survey. pdf* [June, 2014].

[34] Aviram, N., Schinzel, S., Somorovsky, J., Heninger, N., Dankel, M., Steube, J., Valenta, L., Adrian, D., Halderman, J.A., Dukhovni, V. and Käsper, E., 2016. DROWN: Breaking TLS using SSLv2. In *Proceedings of the 25th USENIX Security Symposium*, August 2016.

[35] Bhargavan, K. and Leurent, G., 2016. Transcript collision attacks: Breaking authentication in TLS, IKE, and SSH. *NDSS (Feb. 2016)*.

[36] Goodin, D., 2016. *More than 11 million HTTPS websites imperilled by new decryption attack*. [online] Available at: <http://arstechnica.com/security/2016/03/more-than-13-million-https-websites-imperiled-by-new-decryption-attack/> [Accessed 25 November 2016].

[37] Bock, H., 2016. *TLS – the most important crypto protocol*. [video online] Available at: <https://www.youtube.com/watch?v=o_kOJgvypKY> [Accessed 11 January 2017].

[38] ICO, 2014. *Protecting personal data in online services: learning from the mistakes of others*. [pdf] Available at: <https://ico.org.uk/media/for-organisations/documents/1042221/protecting-personal-data-in-online-services-learning-from-the-mistakes-of-others.pdf> [Accessed 9 January 2017].

[39] Schum, C., 2014. Correctly implementing forward secrecy. [pdf] Available at: <https://www.sans.org/reading-room/whitepapers/bestprac/correctly-implementing-secrecy-35842> [Accessed 9 February 2017].

[40] Seltzer, L., 2014. Best practices and applications of TLS/SSL. [pdf] Available at: <http://resources.idgenterprise.com/original/AST-0036092_Best_Practices_and_Applications_of_TLS_SSL.pdf> [Accessed 9 February 2017].

[41] Levillain, O., Gourdin, B. and Debar, H., 2015, April. TLS Record Protocol: Security Analysis and Defense-in-depth Countermeasures for HTTPS. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security* (pp. 225-236). ACM.

[42] Han, S.W., Kwon, H., Hahn, C., Koo, D. and Hur, J., 2016, July. A survey on MITM and its countermeasures in the TLS handshake protocol. In *Ubiquitous and Future Networks (ICUFN), 2016 Eighth International Conference on* (pp. 724-729). IEEE.

[43] Bleichenbacher, D., 1998, August. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1. In *Annual International Cryptology Conference* (pp. 1-12). Springer Berlin Heidelberg.