Implementation of Extensible Authentication Protocol in OPNET Modeller

S. Saed Rezaie
Department of Electrical Engineering
Amirkabir University
Tehran, Iran
saed.rezaei@aut.ac.ir

S. Amir Hoseini
Department of Electrical Engineering
Amirkabir University
Tehran, Iran
a.hoseini@aut.ac.ir

H. Taheri
Department of Electrical Engineering
Amirkabir University
Tehran, Iran
htaheri@aut.ac.ir

***Abstract*—As wireless technology is becoming more and more prevalent; security in such networks is becoming a challenging issue and is in great demand. One of the most powerful tools in this area which helps to simulate and evaluate the behaviour of networks and protocols is OPtimised Network Engineering Tool (OPNET). Although OPNET has covered a wide range of models and a variety of protocols, there is no implementation of security in it. In this paper we implement and simulate Extensible Authentication Protocol (EAP) using discrete event simulator OPNET modeller. EAP is an authentication framework which is compatible with IEEE 802.1x the security framework for IEEE 802 family.**

Keywords-WLAN; Security; Authentication; EAP; OPNET; Simulation;

I. INTRODUCTION

Widespread acceptance and implementation of wireless local area networks (WLANs) have also brought concerns about the security of these networks. Transmitting data via an air interface rather than a more secure physical conduit brings along with it certain inherent vulnerabilities to security [1].

Since IEEE introduced WEP [2] for the security of IEEE 802.11 [2], up to the latest release of 802.1X [3] in 2010, a multitude of methods and protocols have been proposed [4, 5, 6, 7]. But none of them achieved the success and acceptability of EAP [8]. EAP is commonly used for authentication in port-based access control, and originally developed for point-to-point protocol (PPP) [9] connections. We give a brief overview of EAP in the next section.

OPNET [10, 11] as a powerful simulation tool is applied to implement EAP and evaluate and measure parameters such as delay and overhead in a wireless test bed, as there is no evidence of consideration of security in it. With this measurement we can define how wireless networks can be affected by security consideration.

The rest of the paper is organized as follow. In next section we briefly discus EAP. In section 3 we present modular concept of OPNET. Section 4 gives details about the model we are implementing. Validation of this work is presented in section 5. Finally in section 6 the conclusion and future work is provided.

II. EAP

In this section we only focus on details which we utilized to implement EAP. For further information refer to [3]. EAP is an authentication framework which supports multiple authentication methods. EAP typically runs directly over data link layers such as Point-to-Point Protocol (PPP) or IEEE 802, without requiring IP.

A. Terminology

There are different terminologies due to different standards, in this paper we refer to RFC 3748.

Authenticator: The end of the link initiating EAP authentication. The term authenticator is used in IEEE 802.1X and has the same meaning in this paper.

Peer: The end of the link that responds to the authenticator. In IEEE 802.1X, this end is known as the Supplicant. In this paper this end of the link is called the peer.

Backend Authentication Server: A backend authentication server is an entity that provides an authentication service to an authenticator. When used, this server typically executes EAP methods for the authenticator. This terminology is also used in IEEE 802.1X.

B. EAP Authentication Exchange Process

Step1. The authenticator sends a Request to authenticate the peer. The Request has a Type field to indicate what is being requested.

Step2. The peer sends a Response packet in reply to a valid Request. As with the Request packet, the Response packet contains a Type field, which corresponds to the Type field of the Request.

Step3. The authenticator sends an additional Request packet, and the peer replies with a Response. The sequence of Requests and Responses continues as long as needed.

Step4. The conversation continues until the authenticator cannot authenticate the peer, in which case the authenticator transmits an EAP Failure (Code 4).

Alternatively, the authentication conversation can continue until the authenticator determines that successful authentication has occurred, in which case the authenticator transmits an EAP Success (Code 3). Fig. 1 demonstrates the packet exchange between different entities. Note that in this paper authenticator and server merged into one node.

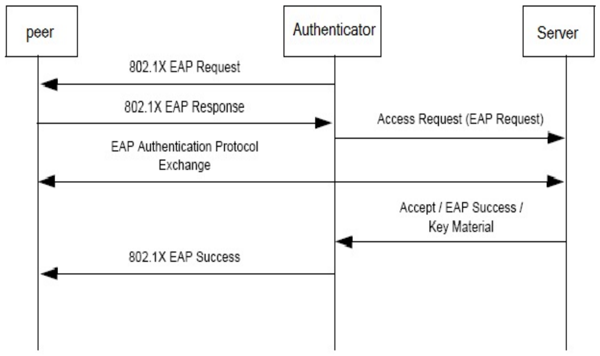
peer
Authenticator
Server
802.1X EAP Request
802.1X EAP Response
Access Request (EAP Request)
EAP Authentication Protocol Exchange
Accept / EAP Success / Key Material
802.1X EAP Success

Figure 1. Packet exchange in EAP process

C. *Encapsulation*

The encapsulation of EAP over IEEE 802 is defined in IEEE 802.1X and known as EAP over LANs or EAPOL. EAPOL was originally designed for IEEE 802.3 Ethernet in 802.1X-2001, but was clarified to suit other IEEE 802 LAN technologies such as IEEE 802.11 wireless LANs. Encapsulation of EAP over different protocols is illustrated in Fig. 2.

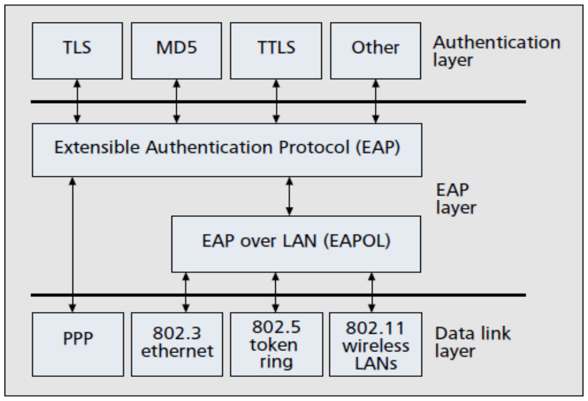

Figure 2. EAP encapsulation over different LAN technologies

As it is seen, several methods of EAP are available and encapsulate in EAP Data field. Some have been developed specifically for wireless networks in addition to EAP methods existing for wired networks. This includes a class of methods based on public key infrastructure [12] and the use of certificates as well as a class of methods that do not use certificates but passwords for their authentication. A comparative study of these methods can be found in [13].

In this paper EAP-MD5 [14] have been implemented and analysed which is based on challenge-response mechanism and due to [15] is one of the most implanting protocols in authentication mechanisms. The packet format of EAP-MD5 due to RFC 1994 consists of four simple fields and is depicted in Fig. 3. Note that this packet is encapsulated in data field of EAP packet.

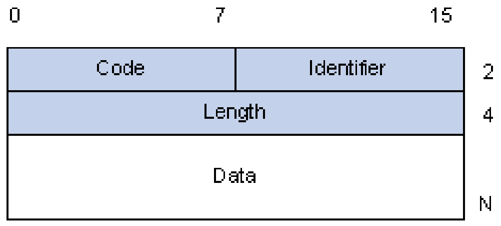

Figure 3. EAP-MD5 packet format

Different code values correspond to different EAP messages. Identifier field is the entity ID which creates the packet, length field is used to show the entire packet size. We use this format to simulate EAP in OPNET.

III. OPNET

OPNET is a vast software package with an extensive set of features designed to support general network modeling and to provide specific support for particular types of network simulation projects. OPNET provides a flexible, high-level programming language with extensive support for communications and distributed systems. This environment allows realistic modeling of all communications protocols, algorithms, and transmission technologies. OPNET supports model specification with a number of tools or editors that capture the characteristics of a modeled system's behavior. Because it is based on a suite of editors that address different aspects of a model, OPNET is able to offer specific capabilities to address the diverse issues encountered in networks and distributed systems.

A. *Hierarchical Architecture*

To present the model developer with an intuitive interface, the editors break down the required modeling information in a manner that parallels the structure of actual network systems. Thus, the model-specification editors are organized in an essentially hierarchical fashion. Model specifications performed in the Project Editor rely on elements specified in the Node Editor; in turn, when working in the Node Editor, the developer makes use of models defined in the Process Editor. The remaining editors are used to define various data models; packet format editor, link model editor, etc.

The Network, Node, and Process modeling environments are sometimes referred to as the *modeling* *domains* of OPNET. The issues addressed by each domain are summarized in TABLE I.

OPNET's Process Editor expresses process models in a language called *Proto-C*, which is specifically designed to support development of protocols and algorithms. Proto-C is based on a combination of state transition diagrams (STDs), a library of high-level commands known as Kernel Procedures, and the general facilities of the C or C++ programming language.

TABLE I: OPNET Modeling Domains

OPNET Modeling Domains		
Domain	*Editor*	*Modeling Focus*
Network	Project	Network topology described in terms of subnetworks, nodes, links, and geographical context.
Node	Node	Node internal architecture described in terms of functional elements and data flow between them.
Process	Process	Behavior of processes (protocols, algorithms, applications), specified using finite state machines and extended high-level language.

Network Domain: The Network Domain's role is to define the topology of a communication network. The communicating entities are called *nodes.* A network model may make use of any number of node models. Modelers can develop their own library of customized node models, implementing any functionality they require.

Node Domain: The Node Domain provides for the modeling of communication devices that can be deployed and interconnected at the network level. Node models are developed in the Node Editor and are expressed in terms of smaller building blocks called *modules*. Some modules offer capability that is substantially predefined and can only be configured through a set of built-in parameters. These include various transmitters and receivers allowing a node to be attached to communication links in the network domain. Other modules, called *processors*, are highly programmable, their behavior being prescribed by an assigned *process model*.

Process model: Process models are developed using the Process Editor. Processor modules are user-programmable elements that are key elements of communication nodes. Processes in OPNET are designed to respond to interrupts and/or invocations. Interrupts typically correspond to events such as messages arriving, timers expiring, resources being released, or state changes in other modules. As previously mentioned, processes are extended by a language called Proto-C. Proto-C models allow actions to be specified at various points in the finite state machine. Since Proto-C is focused on modeling protocols and algorithms, it provides an extensive library of over 300 Kernel Procedures (also known as *KP*s).

IV. Implementing EAP

Our scenario consists of two nodes in project domain as illustrated in Fig. 4; an access point and a client.

The access point (AP) plays the role of Authenticator as well as Backend Authentication Server simultaneously, as it is defined in EAP terminology, while client corresponds to peer.

A. *Project Model*

In Fig. 4, project domain and the nodes insides are presented.

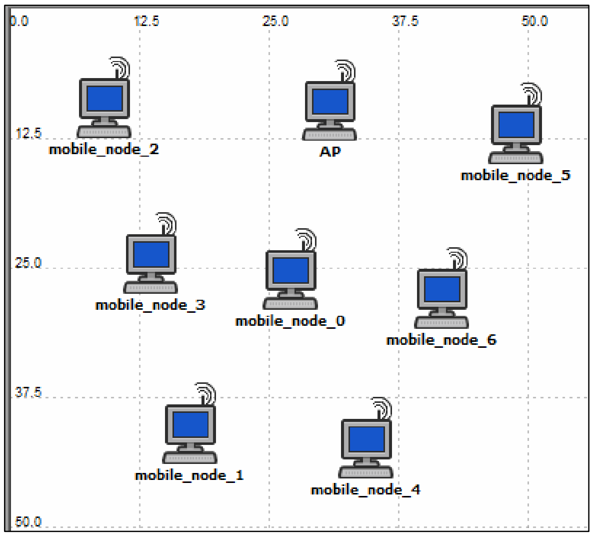

Figure 4. Scenario Topology

Since these nodes are pre-defined, they look like each other but their functionalities are different due to assigned node model. Besides, as they are wireless, there isn't any link between them, otherwise a link model should have been provided.

B. *Packet Format Model*

The packet format is designed in packet format editor. It can be seen in Fig. 5, it is exactly same as the format which is specified in the standard.

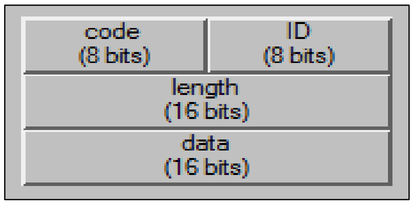

Figure 5. EAP Packet Format Designed in OPNET Packet Editor

All fields were explained earlier in section 2. This packet type will be assigned to receiver and transmitter in the node domain. Furthermore, this kind of traffic is flowed in the network.

C. Node Model

In node model, processors, transceivers and streams are applied to design the layered concept of protocols and models. In this paper we use pre-defined wireless node model to benefit from its useful mechanism and other layers which are not necessary to implement.

The raw node model of mobile *Wireless LAN station* is depicted in Fig. 6.

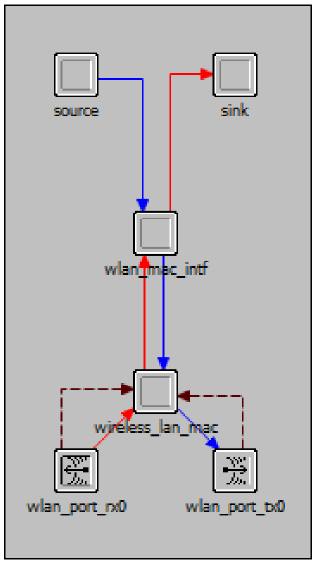

Figure 6. Pre-defined WLAN Node Model in OPNET

We modified this model as follow to encapsulate EAP in wlan_mac_intf layer:

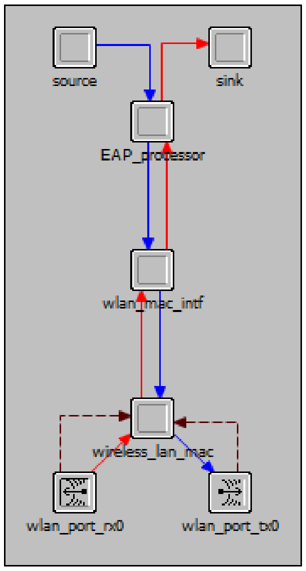

Figure 7. Modified WLAN Node Model

As it can be seen, an extra module is added to the previous model in order to complete our node model. Note that this model is applied for both AP and peer. Figs. 6 and 7 recall the OSI layering model. This model consists of a pair of transmitter-receiver and four processors.

Source processor has the role of generating packets and injecting traffic into the network. *EAP_processor* is in charge of authentication and will be explained more in the next section. *wlan_mac_intf*, encapsulates higher layer PDUs into *wireless_lan_mac* which has the layer two duties such as CSMA/CA and other MAC layer mechanisms.

The solid streams are packet streams and dashed ones are statistic streams. Wlan_port_rx0 and wlan_port_tx0 are radio receiver and transmitter respectively.

D. Process Model

In this section, we go into further details and explain how *EAP_processor* in node model works. Process domain consists of Finite State Machine(s) or FSMs; hence we should design an FSM for each party.

Peer_processor and *AP_processor* FSMs are revealed in Fig. 8 and Fig. 9 respectively. Red states are called unforced states. It means whenever simulator reaches in such states, it waits until the condition becomes true. Conditions are those in parentheses and written in capital letters. Dashed lines are conditional transitions. When the condition is true, transition from one state to another occurs and a function is executed. As previously mentioned, in OPNET, conditions correspond to interrupts. Here interrupts are of packet arrival types that mean whenever a packet is received, transition takes place and corresponding function is executed.

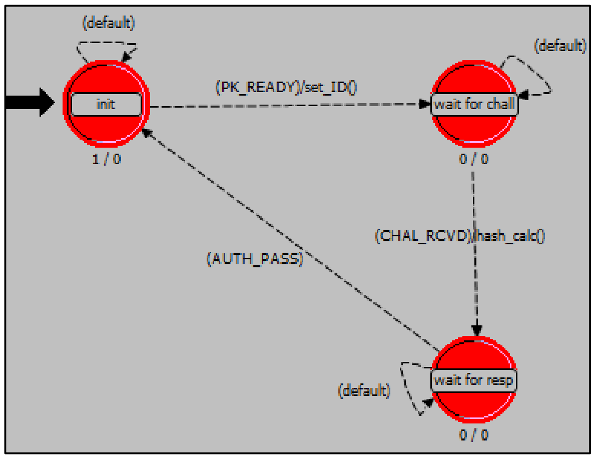

Figure 8. Peer Process Model

In our scenario, peer is the initiator of authentication process. The packet arrival from source processor in node model satisfies the condition PK_READY and starts to transit to the next state. During transition, *set_ID* function is executed. This function sets packet fields appropriately and sends it via its radio transmitter in node model.

When process enters state 2 named *wait for chall*, it pauses until an interrupt takes place. Now AP wireless receiver in node model gets the packet. (In AP processor, a structure is defined to keep information about each user such as user name, password, status of authentication and corresponding address). Through this arrival, ID_RCVD condition (see Fig. 9) becomes true and *chal_calc* procedure is executed. This procedure calculates challenge string and sends the packet to the network, then pauses in *wait for hash* state. Note that first state in this FSM is a forced state and is used to initialize some parameters. And the solid transition indicates that there is no condition for moving from one state to another.

Now CHAL_RCVD condition in peer process model is triggered, thus, process enters state *wait for resp* and *hash_calc* is performed. In this function the hash value of challenge string is calculated, all fields are filled with proper values and the provided packet is sent to the network.

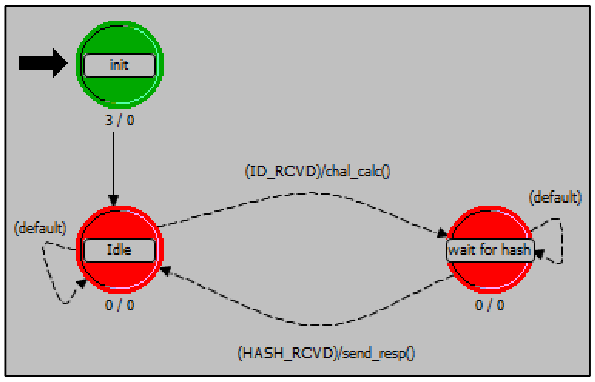

Figure 9. AP Process Model

As soon as a packet containing hash value is received in AP, HASH_RCVD condition is satisfied and *send_resp* function is executed. This function compares the newly arrived hash value with the one stored in its database, if two values match, it generates a session key for secondary communication encryption. When client gets the response, AUTH_PASS condition becomes true and FSM reaches in its final/initial state and waits for other requests.

V. Validation

Now it is time to run the simulation, after collecting pre-configured statistical data. As it is revealed from our topology (Fig. 5) there are a number of mobile nodes randomly situated around an AP. Two scenarios are considered to validate this model.

First we set wrong passwords for some clients and run the simulation and throughput of each node is measured separately, it can be seen in Fig. 10 that their connectivity is cut, after a short time, due to authentication failure. Node 0 has wrong information while node 1 is correctly authenticated.

In the second scenario we measured end-to-end delay for two cases; with and without authentication. And the results show (see Fig. 11) that a slight increase is seen in delay in initial phase in blue graph compared with the original scenario without authentication (green graph).

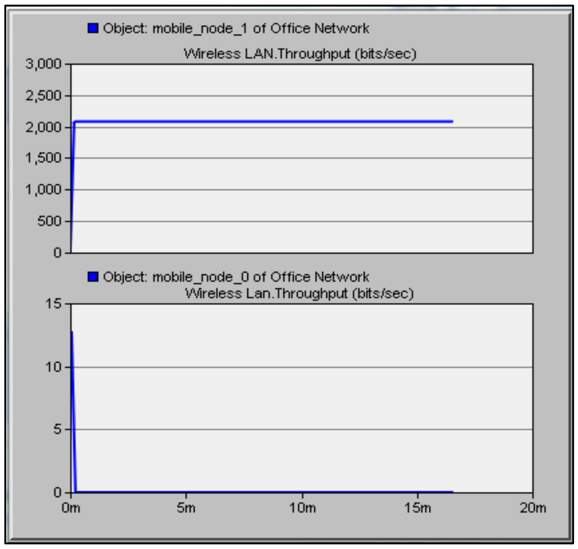

Figure 10. Authentication Failure and Pass

We explain this pattern as follows. When peers start to communicate, they need to be authenticated, and as they start to send authentication request to AP simultaneously, an excessive amount of packets are transmitted over the air and it takes more time for packets to arrive to their destination and this increase is inevitable. But after a small period of time, when the authentication process is over, the two graphs coincide.

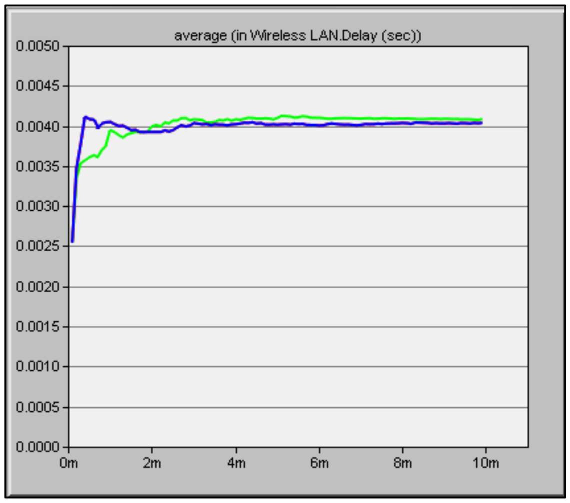

Figure 11. Delay Comparison; With and Without Authentication

VI. Conclusion and Future Work

In this work we implemented and simulated one of the most practical authentication methods in OPNET simulator and it worked properly. We saw that when a node sends wrong parameters, it is not authenticated and cannot communicate anymore. Furthermore we showed that the throughput increases in initial phase because of authentication process.

In future work other EAP methods are simulated and a comparative study would be given between these methods. We are also working on a secure and user friendly method to improve authentication in WLANs.

References

[1] R. Dantu, G. Clothier, and A. Atri, "EAP methods for wireless networks," *Computer Standards & Interfaces* Vol. 29, Issue 3, Pages 289-30, March 2007.

[2] IEEE standard 802.11. [Online]. Pages 158-161 Available: http://standards.ieee.org/getieee802/download/802.11-2007.pdf

[3] IEEE standard 802.1X. [Online]. Available: http://standards.ieee.org/getieee802/download/802.1X-2010.pdf

[4] G. Ateniese, M. Steiner, and G. Tsudik, "New Multiparty Authentication Services and Key Agreement Protocol," in IEEE Journal of Selected Areas in Communications, Vol 18, April 2000

[5] H. H. Ngo, X. Wu, P. D. Le, and B. Srinivasan, "An Authentication Model for Wireless Network Services," in *24th IEEE International Conference on Advanced Information Networking and Applications,* Pages 996-1003, April 2010.

[6] M. Badra, A. Serhrouchni, and P. Urien, "A lightweight identity authentication protocol for wireless networks," in *Computer Communications* Vol. 27, Issue 17, Pages 1738-45, November 2004.

[7] G. Yang, Q. Huang, D. S. Wong, and X. Deng, "Universal Authentication Protocols for Anonymous Wireless Communications," IEEE Trans. on wireless communication, Vol. 9, No. 1, Pages 168-74 January 2010.

[8] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlsonand, and H. Levkowetz, RFC 3748 "Extensible Authentication Protocol (EAP)," June 2004 [Online]. Available: http://tools.ietf.org/html/rfc3748

[9] W. Simpson. RFC 1661 "The Point-to-Point Protocol (PPP)," [Online]. July 1994 Available: http://tools.ietf.org/html/rfc1661

[10] OPNET tutorial.

[11] K. Molnar, and P. Kovar, "Low-Level Process Modelling in the OPNET Modeler Simulation Environment," *Electronics Sozopol, Bulgaria*, September 2007.

[12] R. Housley, and T. Polk, *Planning for PKI*, Vol. 1. New York: Wiley, 2001.

[13] J. Leia, X. Fua, D. Hogrefea, and J. Tanbrotor, "Comparative studies on authentication and key exchange methods for 802.11 wireless LAN," in *computers and security*, Pages 401-9, 2007.

[14] W. Simpson. RFC 1994 "PPP Challenge Handshake Authentication Protocol (CHAP)," [Online]. August 1996 Available: http://tools.ietf.org/html/rfc1994.

[15] S. Convery, *Network Security Architecture*, vol. I. Cisco Press, 2004, p. 112.