INTRUSION PREVENTION/INTRUSION DETECTION SYSTEM (IPS/IDS) FOR WIFI NETWORKS

Michal Korčák[1] and Jaroslav Lámer[2] and František Jakab[3]

[1,2,3]Department of Computer and Informatics, Technical University of Košice, TUKE Košice, Slovakia

ABSTRACT

The nature of wireless networks itself created new vulnerabilities that in the classical wired networks do not exist. This results in an evolutional requirement to implement new sophisticated security mechanism in form of Intrusion Detection and Prevention Systems. This paper deals with security issues of small office and home office wireless networks. The goal of our work is to design and evaluate wireless IDPS with use of packet injection method. Decrease of attacker's traffic by 95% was observed when compared to attacker's traffic without deployment of proposed IDPS system.

KEYWORDS

Deauthentification, Intrusion detection, Intrusion prevention, Packet injection, WiFi

1. INTRODUCTION TO WI-FI NETWORKS SECURITY

Danger of violating the security of wireless network includes both small office and home networks. Authors in [1] conclude fact that in present day almost every person has a wireless router at home and in most cases this person is not an IT specialist or a skilled network administrator. Bearing this in mind, the probability of poorly configured router is pretty high. Modern Wi-Fi routers can be configured with standardized authentication mechanisms. In [2] authors demonstrate that the most commonly used mechanisms are WEP or WPA however even these have certain flaws. Such flaws can be further misused by determined attacker for his malicious plans that can endanger the security of the network and enable access to sensitive information. Even unskilled person may be able to find open source utilities on the internet – to crack Wi-Fi passwords, WEP or WPA encryption and subsequently gain the access to the network. WEP/WPA cracking demonstration was presented by [3]. On account of above stated, even in case of small home wireless networks it is appropriate to deploy intrusion prevention systems except for the regular authentication and security mechanisms.

Network intrusion detection is described in [4], as the process of monitoring the network for the activity that may compromise the security of the area that is under surveillance, and analysing events that may indicate possible incidents. The very same source also presents the Network Intrusion Detection System (NIDS) used as a tool that provides the intrusion detection functionality by sniffing the network traffic in real-time. Such event is then logged and/or the administrator of the system is automatically notified. Except for the detection, Intrusion Detection and Prevention System (IDPS) also executes automated responses to the detected malicious behaviour. This is useful in cases when the attack against the network is carried out very quickly. Thus the IDPS has the ability to take immediate action based on a set of rules, as configured by the network administrator. These rules can be based on IP address matching, TCP

port matching or traffic anomaly detection. Then the response carried out by IDPS could drop the suspicious traffic and further block the traffic based on IP address or port [5].

DOI : 10.5121/ijcnc.2014.6407

The principal goal of this paper is to design and verify (evaluate) the system that would utilize passive monitoring of wireless network traffic in small home network. In case of attack, detection system should actively respond to the particular event by injecting the packets on the wireless medium and disrupt detected attack.

2. ANALYSIS OF IDS/IPS TECHNIQUES

Intrusion Detection and Prevention Systems are primarily used to identify possible threats, log information about them, attempt to stop them and report the information to security administrators. Many systems are able to respond to a detected threat and attempt to prevent it from succeeding. The IDPS can stop the attack itself, change the security environment by reconfiguring the firewall or change the nature of attack [5]. Response to intrusion in the mobile wireless network depends on specific type or nature of an intrusion and used protocols. When compared to the IDS technologies, the IDPS technologies not only detect suspicious activity but also attempt to prevent it from succeeding by the automated responses. These responses usually result in actions taken to supress detected attack or some kind of dynamic reconfiguration of the surrounding equipment based on preconfigured dynamic rules. Automated responses do not necessarily intercept the suspicious traffic directly but can assist the security administrator when handling with the incidents [4].

IDPS technologies are distinguished by the types of events that may be recognized and by the methods that are used to identify incidents. Except for monitoring and analysis of abnormal activity, all types of IDPS technologies can perform the following three functions [5]:

- **recording the information related to detected events** - information about events is recorded locally, afterwards it may be sent to other systems, e.g. logging servers, security information and event management solutions, and enterprise management systems,
- **security administrator notification of c detected critical events** – IDPS uses several methods to send notifications to administrator, such as e-mails, messages on the IDPS user interface, Simple Network Management Protocol (SNMP) traps, syslog messages and user-defined programs or scripts. The message about event includes only basic information. Administrator needs to access the IDPS for additional information,
- **reporting** - the information about events is summarized and can provide additional details.

As presented in [9] Network Intrusion Detection Systems can be categorized based on the detection techniques that are used for detection of abnormal traffic, these are signature-based and statistical anomaly-based detection. Occasionally in order to achieve more accurate detection many IDPS technologies use multiple methods for incident detection.

2.1. Types of IPS/IDS response

Several response techniques are utilized in IDPS [5]:

session termination - terminates the network connection or user session also can block access to the destination or blocks entire communication with the destination host, service, application, or other resource,

- **dynamic reconfiguration** - IDPS can change its configuration to disrupt an attack, this can be done by reconfiguration of a network device such as firewall or switch to block access from the attacker or to the destination,

- **attack content manipulation** - some IDPS systems are able to remove infected or suspicious parts of an attack and thus make it harmless, e.g. IDPS can remove an infected attached file from an e-mail and then send clean email to its recipient.

2.2. Session sniping technique

This technique directly interrupts the traffic between the victim and the attacker and results in disrupted communication. Typically it is done by injecting the packets containing the message to disassociate and terminate current established session and it is particularly useful in case of stateful protocols such is Transmission Control Protocol (TCP) [4].

2.3. ICMP Messaging

In case of TCP protocol its connection establishment method can be used to disrupt the malicious traffic however User Datagram Protocol (UDP) cannot be disrupted like TCP. Since UDP is connectionless and does not support flags to establish and close connection, the UDP traffic cannot be interrupted without involving other connection control mechanisms. In this research, Internet Control Message Protocol (ICMP) may be used – this is one of the main protocols of the TCP/IP suite and is used by network interfaces to test the availability of requested service or host. [10]

IDPS allows to forge packets containing ICMP error message utilizing ICMP protocol. This message is then sent to the attacker, tricking him into thinking that the victim is not available what is to result in termination of connection with victim. In theory the message should be received by attacker's interface and TCP/IP stack should evaluate that the victim is unreachable, regardless of other traffic received. However the chances the message will be followed as it was intended are actually really low. Sophisticated attack tools do not even use TCP/IP stack or have embedded their own rules of accepting the traffic. This implies that this method can be effective only in case of amateur or very simple attack from regular personal computer [6].

3. NEW IDS/IPS SOLUTION

Following section presents designed solution and results as measured for each experiment and subsequently the solution that is the most suitable for described problem.

3.1. The method of deauthentication

This method is used by attackers to gain information needed to crack WPA-PSK key, it is rather new approach of using the attacker's weapon against him. We consider this approach to be novel as a tool for intrusion prevention. Fundamental in this method is to make an attempt to kick the user out of the network by sending repeated disassociation frames which prevents the user from sending any data within the protected network. The disassociation frame is part of the set of management frames used in the wireless networks. The access-point send this type of the frame to the user to terminate the connection [7]. Tool for packet injection utilized during development was *Aireplay-ng* tool, it was used to generate and inject disassociation packets to the wireless medium.

The primary advantage of this method is its independence from the type of attack – basically it is applicable to any attack originating from within the network, i.e. from the associated station. Specific pros and cons of this method are described in the end of this paper.

3.2. System components and architecture

The solution consists of two main components. The intrusion detection subsystem and packet injection component. As an IDS was utilized *Snort*, mainly due to its popularity amongst networking specialists and its rich documentation. As mentioned, for packet injection *Aireplay-ng* tool was used, it is member of *Aireplay-ng* wireless auditing suite. In Figure 1 is depicted the basic system architecture.

Snort is used to monitor the traffic in the wireless medium and report alerts when some malicious behaviour is detected. In order to archive such capability the configuration file and signature database is required to be functional. In the configuration file are written important information, e.g. which Snort rules should be used, which pre-processors should be enabled or what is the IP prefix of protected network.

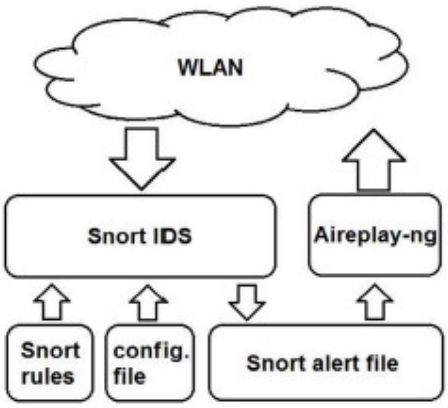

Figure 1. System architecture and its main components

When Snort is properly configured and signature database is loaded, it can be run in IDS mode. Snort triggers the alert and writes it to the alert file once any malicious behaviour is detected. Proper functionality of this file is crucial for our system, it represents the entity through which the main two components are connected. In this file is written the main information about detected malicious behaviour, e.g. type of attack, protocol, IP address or MAC address of the attacker etc. This information is then parsed and passed to the packet injection software which then takes proper countermeasures.

System repeatedly checks the current size of the alert file and compares it with the size of the file recorded during last iteration of the endless loop. If the two sizes match, meaning that alert file did not change its content, program loops again. If the size of the file has grown, meaning new information has been written to the file, program reads last record and parses information needed for packet injection, i.e. MAC address of the attacker and other additional information. Subsequently the system calls Aireplay-ng software with appropriate arguments to start the packet injection process. When this ends, system returns back to looping through the alert file. It is important to set some delay in the loop to avoid excessive exhaustion of processor and memory resources (we use 0.05s delay). The shell command *wc* (short for word count) is used to check the size of the alert file and command *grep* is used to read and parse out information needed from the file.

3.3. Experimental test scenarios

Running Snort on client station – at first we run Snort on the wireless interface of the computer that is connected to the wireless network. Simple wireless network was created, it contained one access point and two clients. The first client acts as an attacker and on the second is running Snort IDS. In Figure 2 is shown just presented topology.

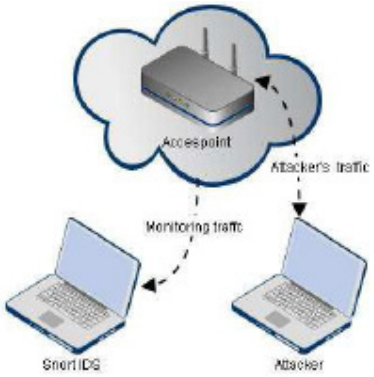

Figure 2. Scenario: Running Snort on client station - system topology

Once Snort was initialized we used another client to generate some traffic, then we were pinging and flooding the access point with ICMP messages. The results were unsatisfying because no traffic generated by the attacker's station was captured by Snort's interface. The reason is the Snort's optimization for the Ethernet interfaces. Snort is not able to capture wireless traffic if it is running on the wireless interface of client. Snort puts the interface to the promiscuous mode instead of monitor mode. Promiscuous mode works only for Ethernet interfaces and is not able to decode wireless 802.11 frames.

Snort was also used on the wireless interface that was set to monitor mode before running the Snort. However this scenario has not been successful, due to the main problem – Snort cannot decode data link type 127, because current implementation of Snort does not provide a packed decoder for the type 127 packets.

Running Snort with Kismet – the same topology as shown in Figure 2 was used again. This time was used the Kismet to sniff the traffic on the monitor interface and then was created so-called *tuntap* interface. This virtual interface was used to pass the captured traffic on the monitor interface. Snort was used on this *tuntap* interface and its ability to decode the captured packets was tested.

Table 1. Snort packet analysis on tuntap interface (capt. time: 20 s, includes rebuilt packets).

Protocol	Packet count	% of traffic type
Eth	0	0
VLAN	0	0
IP4	0	0
Frag	0	0
ICMP	0	0
Other	*2031*	*59.735*
Management	190	5.588
Control	1134	33.353
Data	45	1.324

Recorded statistics were not promising. Snort was able to capture traffic tunnelled from monitor interface to virtual interface however it was not able to decode the protocols. Snort recognized the wireless nature of the captured traffic but it was not able to break down packets to protocols, thus did not decapsulate captured frames. Snort classified these frames as „Other".

Running Snort on wireless router - Snort was installed directly on wireless router but introduced memory issues in router. Due to this constraint it was decided to create virtual access point on computer using *Hostapd* tool (tool is described in [8]). In combination with this tool we finally got satisfactory results.

Table 2. Snort packet analysis on virtual access point (capturing time: 17 s, includes rebuilt packets).

Protocol	Packet count	% of traffic type
Eth	27	100
VLAN	0	0
IP4	22	81.481
Frag	0	0
ICMP	20	74.074
UDP	2	7.407

In Figure 3 is shown the final topology, where one computer with two wireless interfaces is used – one for access point and the other one for packet injection. On this machine was running our system and another machine was acting as an attacker. Computer on the left has installed Kali Linux and Snort IDS. Interface *wlan2* acts as an access point, creating wireless network. Interface *wlan1* is used by *Aireplay-ng* tool for packet injection and deauthentication burst sending, to both access point and the attacker.

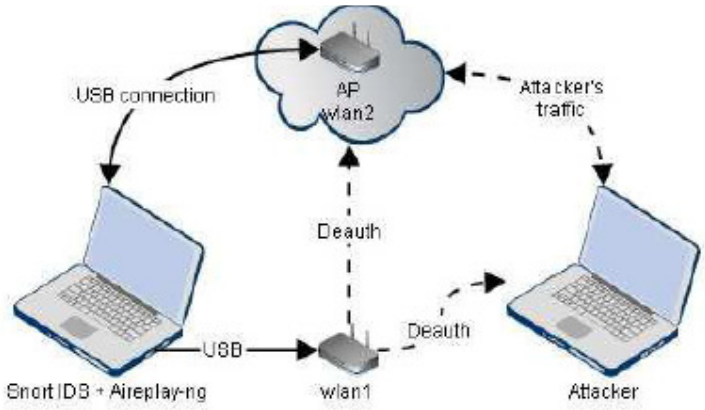

Figure 3. Final topology – Scenario: Running Snort on wireless router

4. MEASURED RESULTS

System was tested on the wireless router in laboratory conditions, while attempting ICMP flooding (attack was realized using *hping3* tool). At first the ICMP flood was running without the system deployed. The experiment procedure can be divided into the following essential steps:

Setup the access point using *Hostapd* software; Run *Snort* as IDS in console; Start a *script* to emulate interface which will be used to inject the packets to the network in second console; Start *Wireshark* listening on the same interface as Snort is running; Attempt ICMP attack using *hping3* on another computer associated with our access-point (wait for unreachable message); Stop Wireshark sniffer, save the dump to the file; Terminate Snort and emulation script; Record Snort statistics as well as script output. In Figure 4 is shown the output from Wireshark statistics of captured attack.

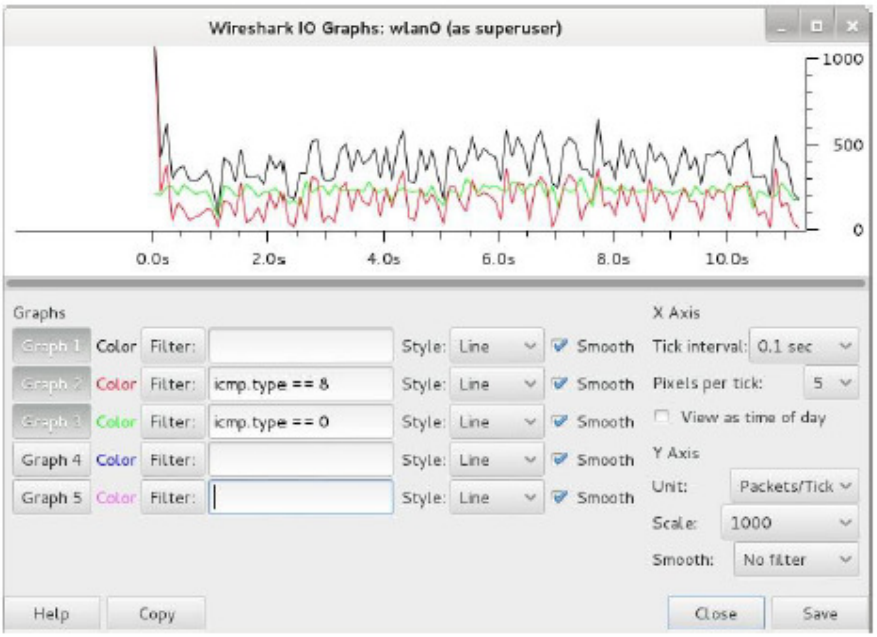

Figure 4. Wireshark statistics of ICMP flood attack

Graph (Figure 4) shows packet rate measured in time domain. This experiment was performed in sterile environment, i.e. all the traffic represents ICMP flood. The filters are set to ICMP types of 8 and 0, which means the request and reply, respectively. Black line represents all ICMP request traffic plus the traffic generated by the access-point. This ICMP stream can be decomposed to the ICMP request traffic and ICMP reply traffic. The red line represents ICMP request flood. Green line represents the ICMP reply traffic generated by the access-point. **31053 packets** were send, after running ICMP flood attack for approximately **10.5 seconds**, this averages to **3100 packets** per second. On the other hand, Figure 5 shows the output from Wireshark statistics of an attack blocked by the IDPS system.

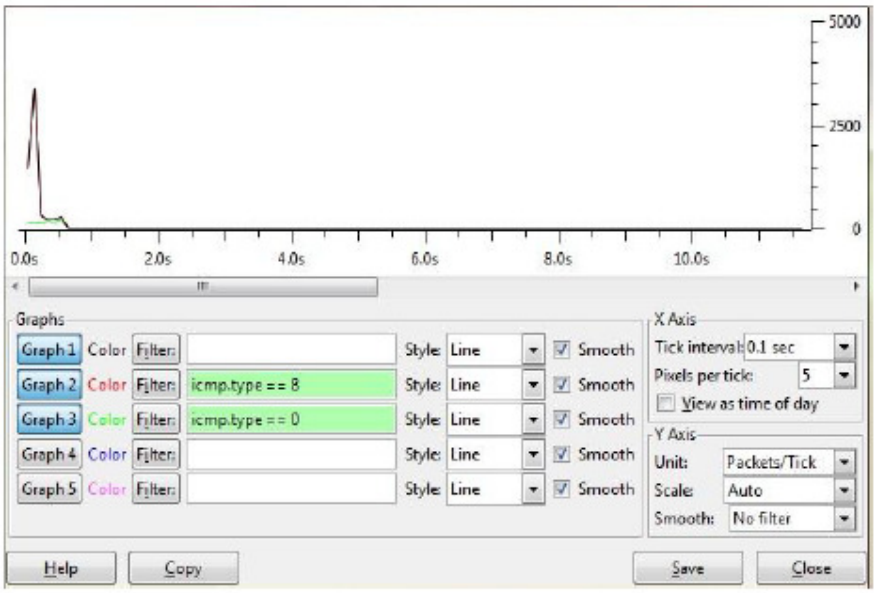

Figure 5. Wireshark statistics of blocked attack

Graph (Figure 5) from Wireshark statistics shows how an attack started and immediately was stopped by our system. In first half second of the attack the big burst of packets occurred and then suddenly dropped down. Comparison of statistical information between Figure 4 and Figure 5 from the ICMP flood attempt is self-explanatory. Notice time around 0.2 second were is visible when IDPS system responded to detected attack. This measurements were repeated for 5 times and in Figure 6 are represented results.

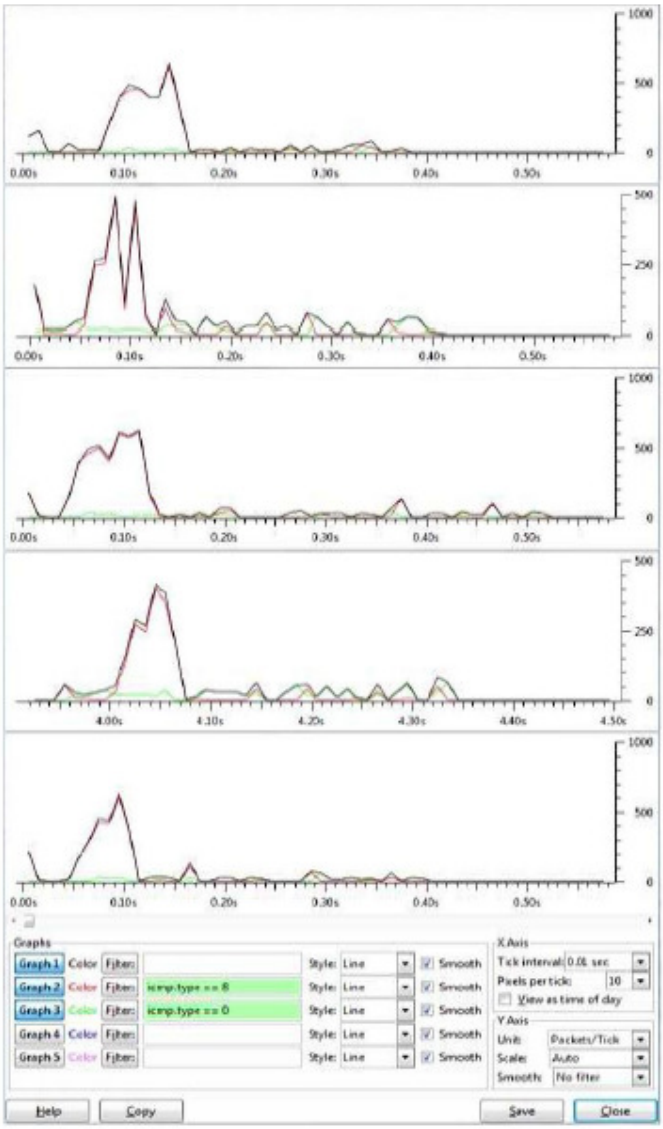

Figure 6. Wireshark statistics of blocked attacks in 5 attempts (zoomed)

As a demonstration the snort packet analysis from the first attempt is presented. Snort has been running for 35 seconds and during this period it captured and analysed totally 8089 packets. 8051 packets belong to the generated ICMP flood. In our previous experiments (Figure 4), the total amount of packets generated as ICMP traffic was 31053 in just 10.5 seconds. That averages to approximately 3100 packets per second. With our system deployed, the number of 8051 ICMP packets generated during the 35 second interval averages to 230 packets per second. That is almost 14 times less than before which results in approximately 92% decrease in traffic.

Table 3. Snort packet analysis in measured results (capt. time: 35 s, includes rebuilt packets).

Protocol	Packet count	% of traffic type
Eth	8089	100
VLAN	0	0
IP4	8054	99.567
Frag	0	0
ICMP	8051	99.530

The reaction time of the system, for five attempts of an attack, was recorded and average values has been calculated. Reaction time for all 5 attempts was measured and recorded based on the timestamps from alert file and from script output. In order to make it easier the timestamps are stripped to seconds. From the calculated differences and after subtracting 0.15 second delay was calculated the average reaction time of 19 milliseconds as is shown in the Table 1. All values in table are in seconds.

Table 4. Average system reaction time is seconds.

Attempt#	**1**	**2**	**3**	**4**	**5**
Alert file timestamp	9.018415	8.082391	4.682434	1.506548	0.397884
Script timestamp	9.196891	8.255933	4.833462	1.679955	0.567699
Difference	0.17848	0.17354	0.15103	0.17341	0.16982
Difference – delay 0.15s	0.2848	0.2354	0.00103	0.02341	0.01981
Average reaction time	**0.0192536**				

During the attack attempts, deauthentication statistics of each attempt were recorded. In Table 1 are computed statistics from the standard output of the Aireplay-ng tool. The statistics were calculated from acknowledgements received from the access-point as well as from the attacker's wireless interface. Table 2 shows the average number of received acknowledgements for all attempts. From this table can be concluded that the access-point had much more stable connection than the attacker. This is directly affected by the strength and quality of the signal since the attacker was physically further from the injecting interface, when compared to the access-point it had less stable acknowledgment response rate.

Table 5. Average number of received acknowledgments.

Attempt#	**Attacker**	**Access-point**
1	24	66
2	32	66
3	28	67
4	26	65
5	26	66
Average	**27**	**66**

In Figure 6 is shown a graph containing all received acknowledgements from the access-point and the attacker. Acknowledgements from the access-point range between 60 and 70 per one deauthentication burst. This averages to 66 packets received corresponding to 64 packets injected. Implying that one hundred percent of injected deauthentication packets were acknowledged by the access-point.

Acknowledgements received from the attacker had very unstable nature, ranging from 9 up to 67. This averages to 27 packets received corresponding to 64 packets injected. That means that 42% of injected deauthentication packets were acknowledged by the attacker's interface.

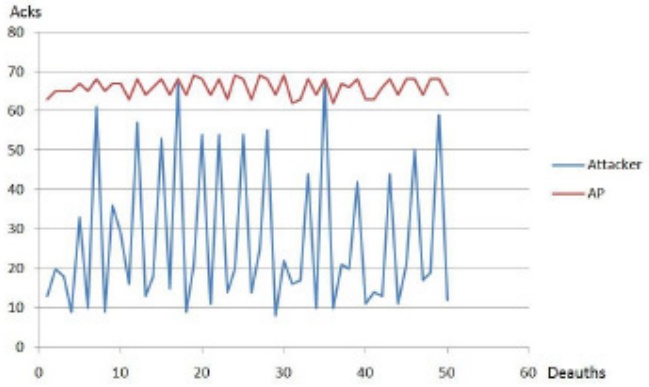

Figure 7. Graphical representation of received acknowledgements

Finally, the ratio of the Snort average packet per second for all 5 attempts is presented. Table 6 contains information about Snort run time and the number of captured ICMP packets. The average packet per second ratio is 162, when compared to previous ICMP flood statistics (Figure 4 - 3100 packets per second) it is almost 19 times greater than with our IDPS system deployed. Approximately difference of **95%** in the amount of captured traffic is achieved.

Table 6. Average packet per second ratio for all 5 attempts (IDPS system activated).

Attempt#	**1**	**2**	**3**	**4**	**5**
Snort run time [s]	35	37	34	32	38
Captured packets	8051	3743	5674	4121	6976
Packets per second	230	101	167	129	184
Average [packets/s]	**162**				

5. CONCLUSIONS

The goal of this paper was to design and evaluate the system that would passively monitor wireless network traffic of small home network. Such system, in case of attack detection, attempts to disrupt detected attack using packet injection method. During experiments was attempt to carry out DoS by ICMP flood on the access-point. This type of attack was chosen because of its simple detection and execution. From the output of Snort statistics was shown decrease of attacker's traffic by 95% with proposed IDPS system deployed – when compared to the amount of attacker's traffic without deployment of IDPS. From the results we conclude that the deauthentication of the attacker successfully disassociates the attacker from the access point, and thus prohibits the attack. Since the initiation of the attack the system was able to react in 0.2 seconds, as is concluded from graphs generated by Wireshark statistics. Aireplay-ng standard output was used to calculate statistics of deauthentication. From these statistics is assumed that deauthentication of the attacker was in all cases successful, nevertheless the percentage of received deauthentication acknowledgements from the attacker was only 42% when compared to the 100% of received acknowledgements from the access point.

Presented method provides several advantages – it has been proven effective in case of attacks originating from the authenticated user, moreover was tested in laboratory network environment

and was successful for all cases tested. The primary advantage of this method is protocol independency.
There are also some disadvantages to consider – this method is useless in case of attack when an attacker is not associated with an access-point. There is a need to cover this area of wireless attacks and perhaps use the combination of deauthentication and other methods.

Future research is to develop techniques which will actively respond to the other types of attacks like sniffing, spoofing, WPA cracking etc. The new security mechanisms are required to compete with new attack methods and algorithms which are quickly getting more and more sophisticated. Bearing this in mind there is an evolutional need to develop more sophisticated and intelligent IDPS solutions for both wired and wireless networks

ACKNOWLEDGEMENTS

Paper is the result of the Project implementation: University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology, ITMS: 26220220182, supported by the Research & Development Operational Programme funded by the ERDF.

REFERENCES

[1] Henry, Paul & Luo, Hui, (2002) “WiFi: what's next?”. Communications Magazine, IEEE, 40.12: 66-72.

[2] Tews, Erik & Beck, Martin, (2009) “Practical attacks against WEP and WPA” In: Proceedings of the second ACM conference on Wireless network security. ACM, p. 79-86.

[3] Gounaris, Georgios, (2014) “WiFi security and testbed implementation for WEP/WPA cracking demonstration”.

[4] L. T. Heberlein & K. N. Levitt & B. Mukherjee, (1991) “A Method To Detect Intrusive Activity in a Networked Environment”. In: 14th National Computer Security Conference. Washington, D.C.: National Institute of Standards and Technology, National Computer Security Center, pp. 362-371

[5] Karen, Scarfone & Peter Mell, (2007) “Guide To Intrusion Detection And Prevention Systems (IDPS)”. Washington, D.C.: National Institute of Standards and Technology, Special Publication 800-94, 128 p.

[6] Michael Rash, (2007) “Linux Firewalls - Attack Detection And Response With Iptables”, Psad And Fwsnort. San Francisco: No Starch Press, 388 p.

[7] Allen, Lee (2012) “Advanced Penetration Testing for Highly--Secured Environments: The Ultimate Security Guide”. Birmingham: Packt Publishing Ltd., 414p.

[8] “Linux Wireless - Hostapd Linux Documentation Page”. [online]. [cit. 14. April. 2014]. Available online: <http://wireless.kernel.org/en/users/Documentation/hostapd>.

[9] KAZIENKO, Przemyslaw; DOROSZ, Piotr. Intrusion detection systems (IDS) Part 2-Classification; methods; techniques. WindowsSecurity. com, 2004.

[10] CARL, Glenn, et al. Denial-of-service attack-detection techniques. Internet Computing, IEEE, 2006, 10.1: 82-89.

Authors

Michal KORČÁK, born 1990 in Prešov graduated from Electrical engineering on Secondary School of Electrical Engineering in Prešov. Later on continued his study on Technical University of Košice and got his Master's degree in Applied Informatics and Computer Science. During the study he worked on summer jobs as a Machine operator and Electrical line installer. After the graduation his primary interests are software development, web development, programming in PHP and Python, networking and network security.

Jaroslav LÁMER, born 1987 in Brezno graduated from the electrotechnical High School Liptovský Hrádok after which he continued his studies on the Bachelor level at the University of Cyril and Methodius in Trnava. He got his Master's degree from the Technical University Kosice, where he is currently enrolled in the second year of his PhD study. He is a member of the Computer Networks Laboratory, participating on broad range of research activities. In the past he worked as a professional software developer. His primary interests are video processing, computer security, optimization of all kinds and development of devices for aviation

František JAKAB, born in 1959. Graduated from St. Petersburg Electro technical University (MSc. In System Engineering, Russia, 1984), PhD degree from Technical university of Kosice (Slovakia, 2005) & associated professor since 2008. He has extensive experience in area of computer networks and utilization of ICT in education (more than 25 years teaching experience). He has been established Computer Networks Laboratory (www.cnl.sk, in 1995). He has been a coordinator of several large international projects financed by EC, coordinator of national wide ICT projects and research grants. He act as an expert in areas: projecting of computer networks, new form of multimedia based communication (videoconferences, IP streaming, IPTV), QoS and intelligent tutoring systems. Published more than 200 papers, 5 books (author, co-author.); Chair of many International Symposiums and conferences, editor of conference Proceedings. Since 1999 until 2014 involved into Cisco Networking academy program in Slovakia as an instructor (CCAI certification) and since 2001 in position of coordinator of the Program in Slovakia (network of 70 universities and secondary schools}, since 2008 until 2014 Regional Lead for Russia, Ukraine and CIS. He is a head of the Application Section of the Communication Technology Forum Association in Slovakia and head of Committee on business – academic cooperation, American Chamber of Commerce in the Slovak Republic, awarded as an "IT person of the year 2006" in Slovakia. Since 2014 director of University Centre for Innovation, Technology Transfer and Intellectual Property Protection at Technical University of Kosice.